Journal of Global Optimization (2006) 34: 467–474
DOI 10.1007/s10898-005-6470-0

Constrained Inverse Minimum Spanning Tree Problems under the Bottleneck-Type Hamming Distance*

BINWU ZHANG[1,3], JIANZHONG ZHANG[2] and YONG HE[3]
[1]*Department of Mathematics and Physics, Hohai University, Changzhou Campus, Changzhou, China*
[2]*Department of Mathematics, City University of Hong Kong, Hong Kong, China; (E-mail: mazhang@cityu.edu.hk)*
[3]*Department of Mathematics, Zhejiang University, Hangzhou, China*

(Received 25 March 2005; accepted in revised form 22 April 2005)

Abstract. In this paper, we consider the inverse minimum spanning tree problem under the bottleneck-type Hamming distance, where the weights of edges can be modified only within given intervals. We further consider the constrained case in which the total modification cost cannot exceed a given upper bound. It is shown that these inverse problems can be transformed into a minimum node cover problem on a bipartite graph, and we give a strongly polynomial time algorithm to solve this type of node cover problems.

Key words: Inverse optimization problem, Hamming distance, minimum spanning tree

1. Introduction

There are several papers discussing inverse minimum spanning tree problems under l_1 and l_∞ norms (distances), see the recent survey paper [2] and the papers cited therein, and recently [1] proposed to consider inverse minimum spanning tree problems under Hamming distance. This paper is a continuation in this direction. The considered problem can be described as follows:

Let $G=(V,E)$ be a connected undirected network consisting of the node set $V=\{1,2,\dots,n\}$ and the edge set $E=\{e_1,e_2,\dots,e_m\}$. Each edge e_i is associated with a weight q_i and a cost $c_i \geqslant 0$ for modifying the weight. Let $q=(q_1,q_2,\dots,q_m)$ denote the weight vector and $c=(c_1,c_2,\dots,c_m)$ denote the cost vector. Let T^0 be spanning tree of G. We look for a new edge weight vector $w=(w_1,w_2,\dots,w_m)$ such that

*This work is supported by The National Natural Science Foundation of China (60021201), The Hong Kong Research Grant Council under the grant CERG 9040883 (CITYU 103003), and the Doctoral Foundation of Hohai University (2005-02).

(a) T^0 is the minimum weight spanning tree with respect to the weight vector w,
(b) for each $i=1,2,\dots,m$, $-l_i \leqslant w_i - q_i \leqslant u_i$, where $l_i, u_i \geqslant 0$ are given lower and upper bounds for reducing and increasing the weight q_i of e_i,
(c) the total modification cost for changing weights of all edges cannot exceed a given upper bound $M>0$, i,e., $\sum_{i=1}^{m} c_i H(w_i, q_i) \leqslant M$, where $H(w_i, q_i)=0$ if $q_i = w_i$ and 1 otherwise,
(d) the maximum modification cost among all edges, i.e., $\max\{c_i H(w_i, q_i) | i=1,\dots,m\}$, is minimized.

This model can be formulated as follows:

$$\begin{aligned}
&\min && \max_{i=1,\dots,m} c_i H(w_i, q_i) \\
&\text{s.t.} && \sum_{e_i \in T^0} w_i \leqslant \sum_{e_j \in T} w_j, \quad \text{for any spanning tree } T \text{ of } G; \\
& && -l_i \leqslant w_i - q_i \leqslant u_i, \quad 1 \leqslant i \leqslant m; \\
& && \sum_{i=1}^{m} c_i H(w_i, q_i) \leqslant M.
\end{aligned} \tag{1}$$

which is different from the problem considered in [1], where we look for a new edge weight vector w such that $\sum_{i=1}^{m} c_i H(w_i, q_i)$ is minimized with the above constraints (a) and (b).

In Section 2, we will consider the *standard case* of problem (1) in which all l_i and u_i are nonnegative and finite, but $M=+\infty$. In Section 3, we will consider the *constrained case* in which in addition to l_i and u_i M is also nonnegative and finite. We will show that the two cases can both be solved by strongly polynomial time algorithms.

Before we discuss the two cases, we introduce some useful notations. For a given spanning tree T^0, we refer to the edge set consisting of the edges in T^0 as the set of *tree edges*, and the set of other edges as the set of *non-tree edges*, which are denoted by E^0 and $\overline{E^0}$, respectively. For each $e_j \in \overline{E^0}$, $E^0 \cup \{e_j\}$ contains a unique cycle, and we denote by P_j all edges in this cycle except e_j. For a given vector $\alpha=(\alpha_1, \alpha_2, \dots \alpha_m)$ define $\phi(\alpha) = \max_{i=1,\dots,m} c_i H(\alpha_i, 0)$ and $\eta(\alpha) = \sum_{i=1}^{m} c_i H(\alpha_i, 0)$.

2. The Standard Case

By setting $\alpha_r = |w_r - q_r|, r=1,2,\dots,m$, the problem considered in [1] is equivalent to

$$\begin{aligned} &\min_{\alpha} \eta(\alpha) \\ &\text{s.t. } q_i - \alpha_i \leqslant q_j + \alpha_j, && \text{for each } e_j \in \overline{E^0} \text{ and } e_i \in P'_j; \\ &0 \leqslant \alpha_i \leqslant l_i, && e_i \in E^0; \\ &0 \leqslant \alpha_j \leqslant u_j, && e_j \in \overline{E^0}, \end{aligned} \tag{2}$$

where $P'_j = \{e_i | e_i \in P_j \text{ and } q_i > q_j\}$. Hence the standard case of our problem is equivalent to

$$\begin{aligned} &\min_{\alpha} \phi(\alpha) \\ &\text{s.t. } q_i - \alpha_i \leqslant q_j + \alpha_j, && \text{for each } e_j \in \overline{E^0} \text{ and } e_i \in P'_j; \\ &0 \leqslant \alpha_i \leqslant l_i, && e_i \in E^0; \\ &0 \leqslant \alpha_j \leqslant u_j, && e_j \in \overline{E^0}. \end{aligned} \tag{3}$$

Similarly, to obtain an algorithm for solving (3) in polynomial time, we still construct a bipartite graph $G' = (N, A) = (E^0 \cup \overline{E^0}, A)$ with respect to the tree T^0 as follows: The node set $N = E^0 \cup \overline{E^0}$, i.e., each edge of E corresponds to a node of G', either on the 'left' side of G' if the edge is in E^0, or on the 'right' side otherwise, and the edge set $A = \{(e_i, e_j) | e_j \in \overline{E^0} \text{and } e_i \in P'_j\}$. We further define the weight of each node e_i of G'as c_i.

Noting that problems (2) and (3) have the same constraints, and as shown in [1], we conclude that they have a feasible solution if and only if for each $(e_i, e_j) \in A$, we have $q_i - q_j \leqslant l_i + u_j$.

In order to solve (3), we first need to find the nodes in G' whose weights must be changed in *every* feasible solution. Define $Z = \{e_i | \alpha_i \neq 0$ in every feasible solution $\alpha = (\alpha_1, \alpha_2, \cdots, \alpha_m)\}$.

LEMMA 2.1. [1] *Suppose problem (3) is feasible. (i) For each $e_i \in E^0, e_i \in Z$ if and only if there exists $e_j \in \overline{E^0}$ such that $(e_i, e_j) \in A$ and $q_i - q_j > u_j$. (ii) For each $e_j \in \overline{E^0}, e_j \in Z$ if and only if there exists $e_i \in E^0$ such that $(e_i, e_j) \in A$ and $q_i - q_j > l_i$.*

Lemma 2.1 tells us how to determine Z. With the above analysis, we further *normalize* the bipartite graph G' in the following way: We start with $Z = \emptyset$. Check every edge (e_i, e_j) in G' to see whether the condition in one of the above Cases 1 and 2 satisfies. If yes, modify Z (add e_i or e_j or both to Z) and delete the edge (e_i, e_j) from G'. After this process, for each $e_i \in Z$, delete all edges in G' which are incident to the node e_i as well as the node e_i itself. In this way, we reduce G' to G'', where $G'' = ((E^0 \backslash Z) \cup (\overline{E_0} \backslash Z), A')$ and $A' = \{(e_i, e_j) | (e_i, e_j) \in A, e_i \notin Z \text{ and } e_j \notin Z\}$.

THEOREM 2.1. *Suppose problem (3) has a feasible solution. Let C^* be a minimum bottleneck-weight node cover of G'', i.e., C^* is a node cover of G''*

such that $c^b(C^*) = \min\{c^b(C) | C$ *is a node cover of* $G''\}$, *where* $c^b(C)$ *is the maximum weight of the elements in* C. *Define* $\alpha' = (\alpha'_1, \alpha'_2, \dots, \alpha'_m)$ *as*

$$\alpha'_i = \begin{cases} l_i, & if\ e_i \in (C^* \cup Z) \cap E^0, \\ u_i, & if\ e_i \in (C^* \cup Z) \cap \overline{E^0}, \\ 0, & if\ e_i \notin (C^* \cup Z), \end{cases}$$

then α' *is an optimal solution of problem (3) with the minimum objective value* $c^b(C^* \cup Z)$.

Proof. As shown in [1], if C^* is a minimum sum-weight node cover of G'', then α' defined in the theorem is the optimal solution of problem (2). Noting again that problems (2) and (3) have the same feasible solutions, hence by a similar argument we can obtain the result. □

We now describe an approach for obtaining C^*. As we do not see any reference giving explicitly an algorithm for the purpose, here we present an algorithm in detail.

ALGORITHM A$_1$. Step 1. Let $C = \emptyset$, sort the nodes of the current graph (it is G'' initially) according to the non-decreasing order of node weights.

Step 2. Find the node e_i with the minimum weight and a positive degree in the current graph, set $C := C \cup \{e_i\}$, and delete the node e_i and all edges incident to it in the current graph.

Step 3. Repeat the process in Step 2 until the edge set of the current graph is empty. Take $C^* = C$ and stop.

Next we show that the set C^* resulted from algorithm A_1 is indeed a minimum bottleneck-weight node cover of G''. In fact, it is easy to know that C^* is a node cover of G''. We further show that C^* has the minimum bottleneck-weight as follows. Denote $c_{i_k} = \max\{c_i | e_i \in C^*\}$, and let C' be another node set of G'' such that $\max\{c_i | e_i \in C'\} < c_{i_k}$. Then we know that $e_{i_k} \notin C'$. On the other hand, according to algorithm A_1, there is a node e_{i_r} which is adjacent to e_{i_k} such that $c_{i_r} \geqslant c_{i_k}$ (otherwise, it is easy to get that $e_{i_k} \notin C^*$, a contradiction). $c_{i_r} \geqslant c_{i_k}$ implies that $e_{i_r} \notin C'$ and thus both end nodes of the edge (e_{i_k}, e_{i_r}) are not covered by C'. So, C' cannot be a node cover of G'' and we have finished the proof.

It is clear that Step 1 takes $O(m \log m)$ time and Steps 2 and 3 take $O(mn)$ time. Hence the algorithm runs in $O(m \log m + mn) = O(mn)$ time. Now we are able to present the following algorithm for solving problem (3) with a time complexity $O(mn)$.

ALGORITHM A$_2$. Step 1. For each $e_j \in \overline{E^0}$, determine first P_j and then P'_j.

Step 2. Then construct a bipartite graph $G' = (E^0 \cup \overline{E^0}, A)$, where $A = \{(e_i, e_j) | e_j \in \overline{E^0}$ and $e_i \in P'_j\}$, and define the weight of each node e_i as c_i. If there exists some $(e_i, e_j) \in A$ such that $q_i - q_j > l_i + u_j$, then the problem is infeasible, stop. Otherwise go to step 3.

Step 3. Obtain Z by Lemma 2.1 and G'' by normalizing G'.

Step 4. Run algorithm A_1 to get a minimum bottleneck-weight node cover C^* of G''.

Step 5. Output an optimal solution $\alpha' = (\alpha'_1, \alpha'_2, \ldots, \alpha'_m)$, where α' is specified in Theorem 2.1.

3. The Constrained Case

By the same notations as what used in Section 2, problem (1) is clearly equivalent to

$$\begin{aligned}
&\min_{\alpha} \phi(\alpha) \\
\text{s.t. } & q_i - \alpha_i \leqslant q_j + \alpha_j, && \text{for each } e_j \in \overline{E_0} \text{ and } e_i \in P'_j; \\
& 0 \leqslant \alpha_i \leqslant l_i, && e_i \in E^0; \\
& 0 \leqslant \alpha_j \leqslant u_j, && e_j \in \overline{E^0}; \\
& \sum_{i=1}^{m} c_i H(\alpha_i, 0) \leqslant M.
\end{aligned} \tag{4}$$

It is trivial that problem (4) has a feasible solution if and only if problem (2) has a feasible solution with the objective function value not greater than M. The main idea of our algorithm for solving problem (4) is as follows: First, note that problems (3) and (4) have the same objective function and the first three groups of constraints, whereas problem (4) has one more constraint. Hence if we denote by α' and α^* the optimal solutions of problems (3) and (4), respectively, it must have $\phi(\alpha') \leqslant \phi(\alpha^*)$. In other words, $\phi(\alpha')$ can be taken as a lower bound for the optimal value $\phi(\alpha^*)$. Clearly, the values $\phi(\alpha')$ and $\phi(\alpha^*)$ are actually costs of two edges, and hence can be expressed as c_i and c_j for two indexes i and j.

Second, if $\bar{\alpha}$ is a minimum solution of problem (2) with the objective value $\eta(\bar{\alpha}) \leqslant M$, then as this $\bar{\alpha}$ is feasible to problem (4), $\phi(\bar{\alpha})$ must be an upper bound for the minimum value $\phi(\alpha^*)$ of problem (4). Let $\phi(\bar{\alpha}) = c_\ell$ for some ℓ between 1 and m. Then $\phi(\alpha') \leqslant \phi(\alpha^*) \leqslant \phi(\bar{\alpha})$. In other words, $\phi(\alpha^*)$ must be one of the several cost values in the interval $[c_i, c_\ell]$.

Third, in order to determine the minimum cost of problem (4) from the interval $[c_i, c_\ell]$ quickly, we may use the bisection method. Take a cost value, say c_k, which is in the quite middle of the above interval. Then we ask: if the minimum value of problem (4) is not greater than c_k? If yes, the search interval can be reduced to $[c_i, c_k]$, otherwise to $[c_k, c_\ell]$. And this question can be answered by computing problem (2) with the revision that if the cost c_τ of edge e_τ is greater than c_k, then α_τ must be 0 (i.e., let the corresponding l_τ or u_τ in the last two groups of constraints in (2) be 0), and then checking if the minimum value of (2) is not greater than M. Repeating the process several times, the interval is reduced to a single cost value which is the optimal value of (4).

ALGORITHM A$_3$. Step 1. Let $c_0 = -1$, rearrange the costs $c_0, c_1, c_2, \dots, c_m$ in an increasing order. Then we express their different values as: $-1 = c_{j_1} < c_{j_2} < \cdots < c_{j_k}$.

Step 2. Call algorithm A_2 to solve problem (3). If the algorithm finds problem (3) infeasible, then output that problem (4) is infeasible, stop; otherwise let the optimal solution of (3) be α' and find the index $\underline{t}$ such that $c_{j_{\underline{t}+1}} = \phi(\alpha')$.

Step 3. Solve problem (2) and let the optimal solution be $\bar{\alpha}$ with objective value $\eta(\bar{\alpha})$. If $\eta(\bar{\alpha}) > M$, then output that problem (4) is infeasible, stop; otherwise $\bar{\alpha}$ is a feasible solution of problem (4) with the objective value $\phi(\bar{\alpha})$. Find the index t such that $c_{j_t} = \phi(\bar{\alpha})$.

Step 4. If $t - \underline{t} = 1$, then output that $\bar{\alpha}$ is an optimal solution of problem (4) with minimum cost $\phi(\bar{\alpha})$, stop; otherwise, we have $t - \underline{t} > 1$, and go to Step 5.

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

$$l_i = 0, \quad \text{if } e_i \in E^0 \quad \text{and } c_i > c_{j_{t'}}, \tag{5}$$

$$u_i = 0, \quad \text{if } e_i \in \overline{E^0} \quad \text{and } c_i > c_{j_{t'}}. \tag{6}$$

If the restricted problem (2) is infeasible, set $\underline{t} \leftarrow t'$ and return to Step 4; otherwise let the optimal solution of the restricted problem (2) be $\hat{\alpha}$ and the minimum value be $\eta(\hat{\alpha})$, then go to Step 6.

Step 6. If $\eta(\hat{\alpha}) \leqslant M$, then $t \leftarrow t', \bar{\alpha} \leftarrow \hat{\alpha}, \phi(\bar{\alpha}) \leftarrow \phi(\hat{\alpha})$; otherwise $\underline{t} \leftarrow t'$. Return to Step 4.

THEOREM 3.1. *Algorithm A$_3$ solves problem (4) with a time complexity $O(n^3 m \log m)$.*

Proof. If the algorithm Stops at Step 2 or 3, then clearly problem (4) is infeasible. We next consider the case that problem (4) 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 (4) by at most $\lceil \log m \rceil$ iterations.

By the introduction before proposing the algorithm, we know that the optimal value of problem (4) is one of the $t-\underline{t}$ distinct cost values in the initial interval $(c_{j_{\underline{t}}}, c_{j_t}]$, where $\underline{t}$ and 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 any $c_{j_{\underline{t}}}$ and c_{j_t} obtained in any iteration. To prove this fact we should consider the following three cases in which the search interval $(c_{j_{\underline{t}}}, c_{j_t}]$ is reduced. □

Case 1. The algorithm finds in Step 5 that problem (2) with the restrictions (7) and (8) is infeasible. We show that in this case for every feasible solution $\gamma=(\gamma_1,\gamma_2,\ldots,\gamma_m)$ of (2), $\phi(\gamma)>c_{j_{t'}}$. In fact, if there exists a feasible solution γ satisfying $\phi(\gamma)\leqslant c_{j_{t'}}$, that is, for each $1\leqslant i\leqslant m$, $c_i H(\gamma_i,0)\leqslant c_{j_{t'}}$, which implies that if $c_i>c_{j_{t'}}$, then $\gamma_i=0$. Thus γ is a feasible solution of the restricted problem (2), a contradiction. Hence for every feasible solution γ of (2), $\phi(\gamma)>c_{j_{t'}}$. It means that the optimal value of problem (4) is greater than $c_{j_{t'}}$, and hence must be in the interval $(c_{j_{t'}}, c_{j_t}]$. As in this case by Step 5, we let the next $\underline{t}$ equal t', it guarantees that the optimal value of (4) is in the next search interval $(c_{j_{\underline{t}}}, c_{j_t}]$.

Case 2. The restricted problem (2) has an optimal solution $\hat{\alpha}$ and $\eta(\hat{\alpha})>M$. This means that for any α satisfying the constraints of problem (2), if it also makes

$$\alpha_i=0 \quad \text{whenever } c_i>c_{j_{t'}}, \tag{7}$$

then we must have $\sum c_i H(\alpha_i,0)>M$. As (9) is equivalent to $\phi(\alpha)\leqslant c_{j_{t'}}$, the above conclusion implies that for every feasible solution γ to problem (4), it must have $\phi(\gamma)>c_{j_{t'}}$. Therefore, the optimal value of (4) must be in the interval $(c_{j_{t'}}, c_{j_t}]$, i.e., in the next interval $(c_{j_{\underline{t}}}, c_{j_t}]$ as in this case, we let $\underline{t}=t'$ (see Step 6).

Case 3. The restricted problem (2) has an optimal solution $\hat{\alpha}$ and $\eta(\hat{\alpha})\leqslant M$. This means that $\hat{\alpha}$ satisfies all constraints of problem (4), and $\phi(\hat{\alpha})\leqslant c_{j_{t'}}$. So, the optimal value of (4) must be in the interval $(c_{j_{\underline{t}}}, c_{j_{t'}}]$, i.e., in the next interval $(c_{j_{\underline{t}}}, c_{j_t}]$ as in this case by Step 6, we let $t=t'$.

Combining the above three cases, we conclude that in each iteration the optimal value of problem (4) is one of the $t-\underline{t}$ distinct cost values in the interval $(c_{j_{\underline{t}}}, c_{j_t}]$. As we know, the bisection method guarantees that after at most $\lceil \log m \rceil$ iterations, the search interval $(c_{j_{\underline{t}}}, c_{j_t}]$ must satisfy $t-\underline{t}=1$, i.e., there is only one cost value in the interval $(c_{j_{\underline{t}}}, c_{j_t}]$, which is just c_{j_t}. The corresponding weight adjustment vector is $\bar{\alpha}$ which is feasible. So, $\bar{\alpha}$ is an optimal solution of problem (4), and the optimal value is $\phi(\bar{\alpha}) = c_{j_t}$. So the validity of the algorithm is proved.

It is clear that Step 1 takes $O(m \log m)$ time, Step 2 takes $O(mn)$ time, Step 3 takes $O(n^3m)$ time, and Steps 4–6 take $O(n^3m)$ time. As the algorithm iterates for at most $O(\log m)$ times, it runs in $O(n^3m \log m)$ time in the worst-case and hence is a strongly polynomial time algorithm. □

References

1. He, Y., Zhang, B. and Yao, E. (2005), Weighted inverse minimum spanning tree problems under Hamming distance, *Journal of Combinatorial Optimization*, 9, 91–100.
2. Heuberger, C. (2004), Inverse optimization: A survey on problems, methods, and results, *Journal of Combinatorial Optimization*, 8, 329–361.