ANZIAM J. 58(2017), 342–349
doi:10.1017/S1446181117000116

NETWORK DESIGN FOR MINIMUM SPANNING TREES UNDER HAMMING DISTANCE

QIN WANG[1] and LONGSHU WU✉[1]

(Received 17 July, 2016; accepted 16 October, 2016; first published online 26 April 2017)

Abstract

We consider a class of network-design problems with minimum sum of modification and network costs for minimum spanning trees under Hamming distance. By constructing three auxiliary networks, we present a strongly polynomial-time algorithm for this problem. The method can be applied to solve many network-design problems. And, we show that a variation model of this problem is NP-hard, even when the underlying network is a tree, by transforming the 0–1 knapsack problem to this model.

2010 *Mathematics subject classification*: 90C27.

Keywords and phrases: network design, spanning tree, Hamming distance, polynomial algorithm, NP-hard.

1. Introduction

Network design plays an important role in practice and has been intensively investigated in the literature. It has many potential applications, such as telecommunication, performance evaluation, geophysical exploration and so on [4, 5, 7]. Since Burton and Toint [2] published their paper on an instance of the inverse shortest paths problem [1, 2], there has been a number of papers concerning this kind of optimization problem. In this paper, we consider a class of network-design problems with minimum sum of modification cost and network cost. This kind of problem has some practical background. For example, in a transportation or communication network, the weight of a link may represent the cost of sending one unit of commodity from one end to the other. So, the weight of a path is the unit traveling cost from the source to the terminal and often has a lower bound. In order to improve the transmission efficiency of the network, one may want to reduce traveling costs between some pairs of sources and terminals by decreasing weights of links. It is natural that reducing traveling cost in a link needs some cost. We hope that the sum of the modification cost and the traveling cost of the shortest paths under the new weights is as small as possible. This problem

[1]College of Sciences, China Jiliang University, Hangzhou, China; e-mail: wls@cjlu.edu.cn.

can be solved in polynomial time by using a similar method as Algorithm 1 proposed in this paper.

For a weighted graph $G = (V, E, w)$, its vertex set and edge set are denoted by $V = V(G)$ and $E = E(G)$, respectively. We use $w = \{w(e) \in R \mid e \in E(G)\}$ to denote the real weight set defined on $E(G)$. We call graph G a *tree* if it is connected and acyclic. The spanning subgraph of G is a subgraph which contains all the vertices of G. If the spanning subgraph of G is a tree, we call it a *spanning tree* of G. For a subgraph T of G, the weight of T, denoted by $w(T)$, is defined by $w(T) = \sum_{e\in E(T)} w(e)$. We say that T is a *minimum spanning tree* of G if $w(T)$ is minimum among all the spanning trees of G. Throughout this paper, for a set F, we use $|F|$ to represent its cardinality. Let $\mathcal{T}$ be the family of all spanning trees of graph G and let $c = \{c(e) \in R_+ \mid e \in E(G)\}$ be the given cost set. For each $e \in E$, $c(e)$ stands for the fixed cost of modifying $w(e)$. Let $b \in R_+^E$ be a bounding vector of maximum allowable modifications. Let $x \in R^E$ be such that $w(e) - b(e) \le x(e) \le w(e) + b(e)$ for all $e \in E$. We call x an adjusted weight vector. The Hamming distance is defined as

$$H(w(e), x(e)) = \begin{cases} 0 & \text{if } w(e) = x(e), \\ 1 & \text{if } w(e) \neq x(e) \end{cases}$$

for $e \in E$. The problem considered in this paper can be described as follows.

Problem 1.1. Find an adjusted weight vector x such that:

(a) $0 \le |w(e) - x(e)| \le b(e)$ for each $e \in E$; and
(b) the total cost $C(x) = \sum_{e\in E} c(e)H(w(e), x(e)) + \min_{T\in\mathcal{T}} \sum_{e\in T} x(e)$ is minimum.

We show that Problem 1.1 is equivalent to that of weight reduction, that is, there is no weight increase when modifying the edge weights. As a result, we develop a strongly polynomial-time algorithm for solving this problem. We also consider the following optimization problem and give an overall budget $B > 0$ on the total modification cost.

Problem 1.2. Find an adjusted weight vector x such that:

(a) $0 \le |w(e) - x(e)| \le b(e)$ for each $e \in E$;
(b) $\sum_{e\in E}(c(e)H(w(e), x(e))) \le B$; and
(c) the weight of the minimum spanning tree $\min_{T\in\mathcal{T}} \sum_{e\in T} x(e)$ is minimum.

This paper is organized as follows. In Section 2, we present the strongly polynomial-time algorithm for Problem 1.1. In Section 3, we show that Problem 1.2 is NP-hard, even when the underlying network is a tree, by transforming the 0–1 knapsack problem [6] into this model. Some concluding remarks are given in Section 4.

https://doi.org/10.1017/S1446181117000116 Published online by Cambridge University Press

2. Algorithm for network design under Hamming distance

First, we show that Problem 1.1 is equivalent to that of weight reduction, that is, there is no weight increase when modifying the edge weights.

THEOREM 2.1. *Suppose that $\bar{w}$ is an optimal solution of Problem 1.1; then $\bar{w}(e) \le w(e)$ for each $e \in E$.*

PROOF. Suppose, to the contrary, that there is an edge $g \in E$ such that $\bar{w}(g) > w(g)$. Then

$$w'(e) = \begin{cases} w(e) & \text{if } e = g, \\ \bar{w}(e) & \text{otherwise} \end{cases}$$

is a feasible solution of Problem 1.1. Note that

$$\sum_{e\in E} c(e)H(w(e), w'(e)) = \sum_{e\in E} c(e)H(w(e), \bar{w}(e)) - c(g)$$

and, since $c(g) > 0$,

$$\sum_{e\in E} c(e)H(w(e), w'(e)) < \sum_{e\in E} c(e)H(w(e), \bar{w}(e)).$$

Moreover, $\min_{T\in\mathcal{T}} \sum_{e\in T} w'(e) \le \min_{T\in\mathcal{T}} \sum_{e\in T} \bar{w}(e)$. So, $f(w') < f(\bar{w})$, contradicting the assumption that $\bar{w}$ is an optimal solution of Problem 1.1. Hence, we have $\bar{w}(e) \le w(e)$ for each $e \in E$. This completes the proof. □

Based on Theorem 2.1, we need only deal with the weight-reduction case of Problem 1.1; we call it Problem 2.2, which is formally described as follows.

PROBLEM 2.2. Find an adjusted weight vector x such that:

(a) $0 \le w(e) - x(e) \le b(e)$ for each $e \in E$; and
(b) the total cost $C(x) = \sum_{e\in E} c(e)H(w(e), x(e)) + \min_{T\in\mathcal{T}} \sum_{e\in T} x(e)$ is minimum.

For convenience, we call a weight vector x satisfying the condition (a) in Problem 2.2 a *feasible weight solution*.

THEOREM 2.3. *Suppose that $\bar{w}$ is an optimal solution of Problem 2.2 and $\bar{T}$ is a minimum spanning tree under weight vector $\bar{w}$. Then*

$$w'(e) = \begin{cases} \bar{w}(e) & \textit{for } e \in \bar{T}, \\ w(e) & \textit{otherwise} \end{cases}$$

is also an optimal solution of Problem 2.2.

PROOF. Note that w' satisfies condition (a) in Problem 2.2. Since

$$w'(e) = \begin{cases} \bar{w}(e) & \text{for } e \in \bar{T}, \\ w(e) \ge \bar{w}(e) & \text{for } e \in E \setminus \bar{T}, \end{cases}$$

it follows that $\bar{T}$ is also a minimum spanning tree under weight vector w'.

https://doi.org/10.1017/S1446181117000116 Published online by Cambridge University Press

Now we show that the total cost under w' is not greater than the total cost under $\bar{w}$. We have

$$\begin{aligned} C(w') &= \sum_{e\in E} c(e)H(w(e), w'(e)) + \min_{T\in\mathcal{T}} \sum_{e\in T} w'(e) \\ &= \sum_{e\in E} c(e)H(w(e), w'(e)) + \min_{T\in\mathcal{T}} \sum_{e\in T} \bar{w}(e) \\ &\le \sum_{e\in E} c(e)H(w(e), \bar{w}(e)) + \min_{T\in\mathcal{T}} \sum_{e\in T} \bar{w}(e) \\ &= C(\bar{w}). \end{aligned}$$

The proof is now complete. □

We define

$$w^*(e) = \begin{cases} w(e) - b(e) & \text{for } c(e) < b(e), \\ w(e) & \text{for } c(e) \ge b(e) \end{cases}$$

for each $e \in E$ and construct an auxiliary network as follows.

Auxiliary network 1. Let $G^1 = (V, E, w_1)$, where V and E are the vertex set and edge set in G, respectively, and $w_1(e) = c(e)H(w(e), w^*(e)) + w^*(e)$ for each $e \in E$.

Under weight vector w_1, we denote the minimum spanning tree by T^1 and construct another auxiliary network as follows.

Auxiliary network 2. Let $G^2 = (V, E, w_2)$, where V and E are the vertex set and edge set in G, respectively, and

$$w_2(e) = \begin{cases} w^*(e) & \text{for } e \in T^1, \\ w(e) & \text{otherwise.} \end{cases}$$

THEOREM 2.4. *The weight solution w_2 is a feasible weight solution of Problem 2.2.*

PROOF. By the definition of w_2 and w^*, it is easy to see that $0 \le w(e) - w_2(e) \le b(e)$ for each $e \in E$. So, w_2 satisfies condition (a) and, therefore, is a feasible weight solution of Problem 2.2. □

Next we show that w_2 is also an optimal solution of Problem 2.2. First, we have the following theorem.

THEOREM 2.5. *Every feasible weight solution x satisfies that the total cost*

$$C(x) \ge C = \sum_{e\in E} c(e)H(w(e), w_2(e)) + \sum_{e\in T^1} w_2(e).$$

PROOF. Suppose that $\bar{w}$ is any feasible weight solution of Problem 2.2 and $T^2 \in \mathcal{T}$ is a minimum spanning tree under $\bar{w}$. By Theorem 2.3, without loss of generality, we assume that

$$w'(e) = \begin{cases} \bar{w}(e) & \text{for } e \in T^2, \\ w(e) & \text{otherwise.} \end{cases}$$

https://doi.org/10.1017/S1446181117000116 Published online by Cambridge University Press

Denote $C(w') = \sum_{e\in E} c(e)H(w(e), w'(e)) + \sum_{e\in T^2} w'(e)$. From the definition of F^1,

$$\begin{aligned} C &= \sum_{e\in E} c(e)H(w(e), w_2(e)) + \sum_{e\in T^1} w_2(e) \\ &= \sum_{e\in T^1} c(e)H(w(e), w^*(e)) + \sum_{e\in T^1} w^*(e) \\ &\le \sum_{e\in T^2} c(e)H(w(e), w^*(e)) + \sum_{e\in T^2} w^*(e) \\ &\le \sum_{e\in T^2} c(e)H(w(e), w'(e)) + \sum_{e\in T^2} w'(e) \\ &= C(w'). \end{aligned}$$

This completes the proof. □

THEOREM 2.6. *T^1 is a minimum spanning tree of Problem 2.2 under weight vector w_2.*

PROOF. We prove this by contradiction. Suppose that T^1 is not the minimum spanning tree and the the minimum spanning tree is $\bar{T} \in \mathcal{T}$ such that $\sum_{e\in T^1} w_2(e) > \sum_{e\in\bar{T}} w_2(e)$. We construct the third auxiliary network as follows.

Auxiliary network 3. This network is constructed as $G^3 = (V, E, w_3)$, where V and E are the vertex set and edge set in G, respectively, and

$$w_3(e) = \begin{cases} w_2(e) & \text{for } e \in \bar{T}, \\ w(e) & \text{otherwise.} \end{cases}$$

But now the total cost of G^3 is

$$\begin{aligned} C(w_3) &= \sum_{e\in T^1\cap\bar{T}} c(e)H(w(e), w_2(e)) + \sum_{e\in\bar{T}} w_2(e) \\ &< \sum_{e\in T^1} c(e)H(w(e), w_2(e)) + \sum_{e\in T^1} w_2(e) = C, \end{aligned}$$

which is a contradiction to Theorem 2.5. The proof is now complete. □

Now we outline some polynomial solution procedures for solving Problem 2.2 in the following algorithm.

THEOREM 2.7. *Problem 2.2 is polynomially solvable and Algorithm 1 correctly solves this problem in $O(|E|\log|E|)$ time.*

PROOF. Combining Theorem 2.6 with Theorem 2.5, we can see that the weight vector w_2 is an optimal solution of Problem 2.2. The running time used in step 1 for computing the new weights $w^*(e)$, $e \in E$, is $O(|E|)$. Solving the minimum spanning tree problem in step 2 takes $O(|E|\log|E|)$ time. The running time used in step 3 for computing the adjusted weights $w_2(e)$, $e \in E$, is also $O(|E|)$. So, the time complexity of Algorithm 1 is given by $O(|E|\log|E|)$. □

https://doi.org/10.1017/S1446181117000116 Published online by Cambridge University Press

Algorithm 1:

Step 1. For each $e \in E$, define the revised weight vector w^* by

$$w^*(e) = \begin{cases} w(e) - b(e) & \text{if } c(e) < b(e), \\ w(e) & \text{if } c(e) \geq b(e). \end{cases}$$

Step 2. Construct Auxiliary network 1.

Step 3. Find the minimum spanning tree under weight vector w_1 and denote the tree by T^1.

Step 4. Construct Auxiliary network 2. The weight vector w_2 is the optimal solution and the optimal value is

$$C(w_2) = \sum_{e \in E} c(e)H(w(e), w_2(e)) + \sum_{e \in T^1} w_2(e).$$

3. NP-hardness for tree networks

In this section we will show that Problem 1.2 is nondeterministic polynomial-time (NP)-hard, even when the underlying network is a tree. We will use the NP-hard 0–1 knapsack problem [3, 6] for the reduction.

The knapsack problem is a problem in combinatorial optimization. Given a set of n items, each with a weight $a_j > 0$ and a value $p_j > 0$, $j = 1, 2, \ldots, n$, the problem is to determine the number of each item to be included in a collection, so that the total weight is less than or equal to a given limit $L > 0$ and the total value is as large as possible. The integer programming of a 0–1 knapsack problem can be described as follows:

$$\text{maximize} \quad \sum_{j=1}^{n} p_j y_j,$$

subject to

$$\sum_{j=1}^{n} a_j y_j \leq L,$$

$$y_j = 0, 1 \quad \text{and } j = 1, 2, \ldots, n.$$

Theorem 3.1. *Problem 1.2 is NP-hard even if the underlying network G is a tree.*

Proof. Suppose that the graph G is a tree; we call it T. Then T is the minimum spanning tree of Problem 1.2. And, it is obvious that there will be no weight increase when modifying the edge weights. So, in this case we need only deal with the weight-reduction case of Problem 1.2; we call it Problem 3.2, which can be described formally as follows.

https://doi.org/10.1017/S1446181117000116 Published online by Cambridge University Press

PROBLEM 3.2. Find an adjusted weight vector x such that:

(a) $0 \le w(e) - x(e) \le b(e)$ for each $e \in E$;
(b) $\sum_{e\in E} c(e)H(w(e), x(e)) \le B$; and
(c) $\sum_{e\in T} x(e)$ is minimum.

Assume that $|E(T)| = n$. By assigning an index j, $j = 1, 2, \ldots, n$, to each edge of T, we write $w(e), x(e), b(e)$ and $c(e)$ for w_j, x_j, b_j and c_j, respectively, and we transform Problem 3.2 to Problem 3.3 as follows.

PROBLEM 3.3. Find an adjusted weight vector x such that:

(a) $0 \le w_j - x_j \le b_j$ for $j = 1, 2, \ldots, n$;
(b) $\sum_{j=1}^{n} c_j H(w_j, x_j) \le B$; and
(c) $\sum_{j=1}^{n} x_j$ is minimum.

Since the given limit is under Hamming distance, we observe that if w_j is reduced, then $x_j = w_j - b_j$; otherwise, w_j remains unchanged. So, let $b_j = p_j$, $c_j = a_j$ for $j = 1, 2, \ldots, n$ and let $B = L$. Furthermore, let

$$y_j = H(w_j, x_j) = \begin{cases} 0 & \text{if } w_j = x_j, \\ 1 & \text{if } w_j \ne x_j \end{cases}$$

for $j = 1, 2, \ldots, n$. Then Problem 3.3 can be described as

$$\begin{aligned} &\text{minimize} && w(T) - \sum_{j=1}^{n} p_j y_j, \\ &\text{subject to} \\ &&& \sum_{j=1}^{n} a_j y_j \le L, \\ &&& y_j = 0, 1 \quad \text{and } j = 1, 2, \ldots, n, \end{aligned}$$

which is equivalent to solving the 0–1 knapsack problem defined above. Hence, we have proved that Problem 1.2 is NP-hard even if the underlying network G is a tree. The result follows. □

4. Concluding remarks

In this paper, we considered a class of network-design problems with minimum sum of modification and network costs. We presented a strongly polynomial-time algorithm for Problem 1.1, and proved that Problem 1.2 is NP-hard even when the underlying network is a tree by transforming the 0–1 knapsack problem to this model. Also, we noted that if $c_j = 1$ for each $j = 1, 2, \ldots, n$, Problem 3.3 can be solved in polynomial time. Due to the practical potential of the network-improvement problems, it is also meaningful to consider whether other relevant models under different norms are polynomially solvable.

https://doi.org/10.1017/S1446181117000116 Published online by Cambridge University Press

Acknowledgement

This research was supported by the National Natural Science Foundation of China under Grant No. 11171316.

References

[1] D. Burton, W. R. Pulleyblank and Ph. L. Toint, "The inverse shortest paths problem with upper bounds on shortest paths costs", in: *Network optimization*, Volume 450 of *Lecture Notes in Economics and Mathematical Systems* (Springer, Berlin–Heidelberg, 1997) 156–171.

[2] D. Burton and Ph. L. Toint, "On an instance of the inverse shortest paths problem", *Math. Program.* **53** (1992) 45–61; doi:10.1007/BF01585693.

[3] R. Diestel, *Graph theory*, 3rd edn (Springer, Heidelberg, 2005).

[4] S. P. Fekete, W. Hochstattler, St. Kromberg and Ch. Moll, "The complexity of an inverse shortest paths problem", in: *Contemporary trends in discrete mathematics,* Volume 49 (American Mathematical Society, Providence, RI, 1999) 113–127.

[5] C. Heuberger, "Inverse combinatorial optimization: a survey on problems, methods, and results", *J. Comb. Optim.* **8** (2004) 329–361; doi:10.1023/B:JOCO.0000038914.26975.9b.

[6] B. Korte and J. Vygen, *Combinatorial optimization, theory and algorithms*, 4th edn (Springer, Heidelberg, 2007).

[7] Q. Wang, J. J. Yuan and J. Z. Zhang, "An inverse model for the most uniform problem", *Oper. Res. Lett.* **36** (2008) 26–30; doi:10.1016/j.or1.2007.03.006.