Maximum shortest path interdiction problem by upgrading edges on trees under hamming distance

Article *in* Optimization Letters · November 2021

DOI: 10.1007/s11590-020-01687-9

CITATIONS
13

READS
190

4 authors, including:

Zhang Qiao
Southeast University
12 PUBLICATIONS **67** CITATIONS
SEE PROFILE

Xiucui Guan
Southeast University
41 PUBLICATIONS **235** CITATIONS
SEE PROFILE

Panos Pardalos
University of Florida
1,729 PUBLICATIONS **48,483** CITATIONS
SEE PROFILE

Optimization Letters (2021) 15:2661–2680
https://doi.org/10.1007/s11590-020-01687-9

ORIGINAL PAPER

Maximum shortest path interdiction problem by upgrading edges on trees under hamming distance

Qiao Zhang[1] · Xiucui Guan[1] · Hui Wang[1] · Panos M. Pardalos[2,3]

Received: 10 September 2020 / Accepted: 8 December 2020 / Published online: 6 January 2021

Abstract

We consider the maximum shortest path interdiction problem by upgrading edges on trees under Hamming distance (denoted by **(MSPITH)**), which has wide applications in transportation network, network war and terrorist network. The problem **(MSPITH)** aims to maximize the length of the shortest path from the root of a tree to all its leaves by upgrading edge weights such that the upgrade cost under sum-Hamming distance is upper-bounded by a given value. We show that the problem **(MSPITH)** under weighted sum-Hamming distance is NP-hard. We consider two cases of the problem **(MSPITH)** under unit sum-Hamming distance based on the number K of critical edges. We propose a greedy algorithm within $O(n + l \log l)$ time when $K = 1$ and a dynamic programming algorithm within $O(n(\log n + K^3))$ time when $K > 1$, where n and l are the numbers of nodes and leaves in a tree, respectively. Furthermore, we consider a minimum cost shortest path interdiction problem by upgrading edges on trees under unit Hamming distance, denoted by **(MCSPITUH)** and propose a binary search algorithm within $O(n^4 \log n)$ time, where a dynamic programming algorithm is executed in each iteration to solve its corresponding problem **(MSPITH)**. Finally, we design numerical experiments to show the effectiveness of the algorithms.

Keywords Network interdiction problem · Upgrading critical edges · Shortest path · Tree · Hamming distance · Dynamic programming algorithm

Research is supported by National Natural Science Foundation of China (11471073) and the Basic Research Program at the National Research University Higher School of Economics (HSE) for P. M. Pardalos.

✉ Xiucui Guan
xcguan@163.com

1 School of Mathematics, Southeast University, Nanjing 210096, China

2 Center for Applied Optimization, Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL, USA

3 LATNA, Higher School of Economics, Moscow, Russia

1 Introduction

Network interdiction problems by deleting critical edges (denoted by **(NIP-DE)**) aim to delete K edges of a network to make some network performance worse. They have wide applications in transportation network [1], terrorist network [2,3] and network war [3]. They have been widely studied on shortest path [3–6], spanning tree [7–11], maximum matching [12–15], maximum flow [16,17] and center (median) location [18] problems.

The problem **(NIP-DE)** was first applied to shortest path problem by Corley and Sha in [4]. For any K, Bar-Noy et al. [5] showed that it is NP-hard. Khachiyan et al. [3] showed that it is not approximable within ratio 2. When $K = 1$ Nardelli et al. [6] proposed an $O(m\alpha(m, n))$ algorithm for the problem on undirected networks, where α is the inverse Ackermann function, m, n are the numbers of edges and nodes in the network. Bazgan et al. [19] gave an $O(mn)$ algorithm for the shortest path interdiction problem when the increment $b = 1$ of the length of the path, while the problem is much harder when $b \geq 2$. Zhang et al. [20] studied the optimal shortest path set problem in an undirected graph. The goal is to find a minimum collection of paths for the vehicles before they start off to assure the fastest arrival of at least one vehicle, no matter which K edges are blocked. They proposed an $O(n^2)$ algorithm when $K = 1$ and a strong polynomial time algorithm when $K > 1$.

Almost all the network interdiction problems are to delete some critical edges. However, in some practical applications, it is extremely difficult to delete edges in a network, but to modify the weights of some edges to prolong service since there are always some emergence or alternative schemes available. In terrorist networks given in Fig.1 [21], under limited interdiction budget, once a terrorist attacks node k_{10} which will cause fire, terrorists want to maximize the shortest path from node k_1 to k_{10} of fire trucks by interdicting some arcs. Correspondingly, defenders should determine in advance the risky arc(s) that will be interdicted and present a relatively safety paths as emergence schemes for the fire trucks. In this case, terrorists can only increase the lengths of some arcs, but can not increase its lengths to $+\infty$, which corresponds to deleting those arcs. Therefore, consider the shortest path interdiction problems on trees by upgrading critical edges.

The maximum shortest path interdiction problem by upgrading edges on trees (denoted by **(MSPIT)**) can be defined as follows. Let $T = (V, E, w)$ be an edge–weighed tree rooted at s, where $V = \{s, v_1, v_2, \cdots, v_n\}$ and $E = \{e_1, e_2, \cdots, e_n\}$ are the sets of vertices and edges, respectively. Note that the edge $e_j = (v_i, v_j)$ is labelled by the subscript of the endpoint v_j which is further to the root s than v_i. Let $L = \{t_1, t_2, \cdots, t_l\}$ be the set of leaves. Let $w(e)$ and $u(e)$ be the original and upgrade length of edge $e \in E$, respectively, where $w(e) \leq u(e)$. Let $\Delta w(e) = u(e) - w(e)$. Let $c(e)$ be a cost to upgrade the edge e. Denote by P_{s,t_k} the unique path from s to t_k in T. Define $d_w(s, t_k) = \sum_{e \in P_{s,t_k}} w(e)$ as the length of path P_{s,t_k} under the vector w. Define "the shortest root–leaf distance" $d_w(s, L) = \min_{t_k \in L} d_w(s, t_k)$ as the shortest distance from the root s to all the leaves.

The problem **(MSPIT)** aims to find an upgrade scheme $\bar{w}$ to maximize the shortest root–leaf distance under $\bar{w}$ on the premise that the total upgrade cost under some

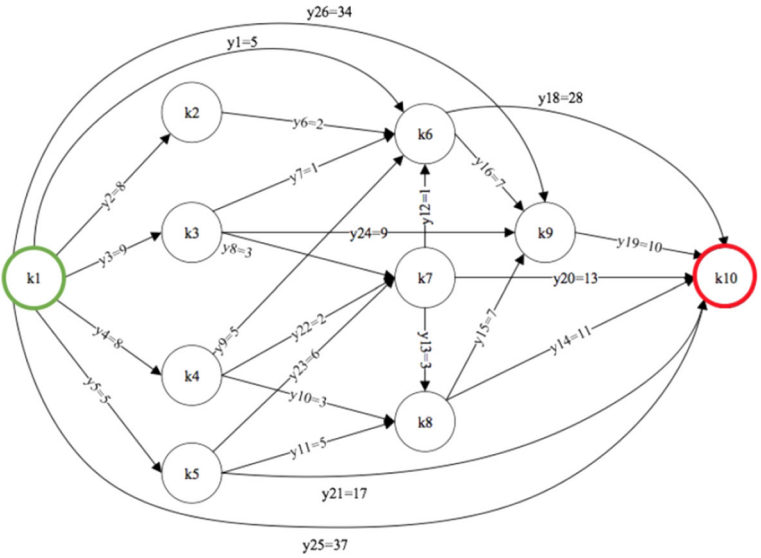


Fig. 1 A terrorist network consisting of 10 nodes and 26 arcs [21]. The interdiction costs are shown on the relevant arcs

norm is upper bounded by a given value K. The mathematical model can be stated as follows.

$$\begin{aligned} &\max \min_{t\in L} d_{\bar{w}}(s,t) \\ &s.t. \sum_{e\in E} \|\bar{w}(e) - w(e)\| \le K, \\ &\quad w(e) \le \bar{w}(e) \le u(e), e \in E. \end{aligned}$$

When the weighted l_1 norm is applied to the upgrade cost, Zhang, Guan and Pardalos [22] proposed a linear time algorithm for the problem (**MSPIT**$_1$) under unit l_1 norm, and a primal dual algorithm within $O(n^2)$ time for the problem (**MSPIT**$_1$) under weighted l_1 norm. They also devised an $O(n^2)$ time algorithm to solve the problem of minimizing the cost by upgrading edges such that the length of the shortest path is lower bounded by a value.

When the Hamming distance is applied to the upgrade cost, the problem **(MSPIT)** under Hamming distance (denoted by **(MSPITH)**) can be formulated as the following form.

$$\begin{aligned} &\max \min_{t\in L} d_{\bar{w}}(s,t) \\ &s.t. \sum_{e\in E} c(e)H(\bar{w}(e), w(e)) \le K, \\ &\quad w(e) \le \bar{w}(e) \le u(e), e \in E, \end{aligned} \tag{1}$$

where the Hamming distance between $\bar{w}(e)$ and $w(e)$ is defined as

$$H(\bar{w}(e), w(e)) = \begin{cases} 0, & \text{if } \bar{w}(e) = w(e), \\ 1, & \text{if } \bar{w}(e) \ne w(e). \end{cases}$$

Table 1 The relationship between the previous research and our research

Type	Graph	Problem	$K/b/c$		Complexity	Reference
Deleting edges	General graph	**NIP on shortest path**	any K		NP-hard	[4]
					Not approximable within ratio 2	[14]
	Undirected networks		$K=1$		$O(m\alpha(m,n))$	[17]
			$b=1$		$O(mn)$	[5]
			$K=1$		$O(n^2)$	[25]
			any K		Strongly polynomial time	
Updating edges	General graph	**MCPIP**	any K	any c	Strongly polynomial time	[16]
	Tree	**MCSPIT$_1$**		$c=1$	$O(n)$	[26]
				any c	$O(n^2)$	
		MSPIT$_1$		$c=1$	$O(n)$	
				any c	$O(n^2)$	
		SPITUH		$c=1$	Exponential time	[24]
		MSPITH		any c	NP-hard	Section 2
			$K=1$	$c=1$	$O(n+l\log l)$	Alg. 1
			any K	$c=1$	$O(n(\log n+K^3))$	Alg. 2
		MCPITUH	any K	$c=1$	$O(n^4\log n)$	Alg. 3

Mohammadi and Tayyebi [23] solved the maximum capacity path interdiction problem **(MCPIP)** on general graphs with fixed costs by binary search in strongly polynomial time. The main difference between the problem **(MCPIP)** and **(MSPITH)** is the definition of the length of a root-leaf path, which is the minimum edge weight of a path in the problem **(MCPIP)** and the sum of edge weights of a path in the problem **(MSPITH)**. Zhang, Guan etc. [24] considered the shortest path improvement problem on rooted trees under unit Hamming distance, denoted by **(SPITUH)**, whose aim is to minimize the number of modified edges satisfying $l(e) \le \bar{w}(e) \le w(e)$ and the modified shortest distance $d_{\bar{w}}(s, t_k)$ is upper bounded by a given value $d_k(t_k \in L)$. They proposed a dynamic programming algorithm to solve the problem, whose time complexity is exponential time in the worst case. The relationship between the previous research and our research can be shown in Table 1.

The paper is organized as follows. In Sect. 2, we first proved the problem **(MSPITH)** is NP–hard as its subproblem can be transformed into a 0–1 knapsack problem. In Sect. 3 and 4, we proposed a greedy algorithm and a dynamic programming algorithm to solve the problem **(MSPITH)** under unit Hamming distance when $K = 1$ and $K > 1$ within time complexities $O(n + l\log l)$ and $O(n(\log n + K^3))$, respectively. In Sect. 5, we considered a minimum cost shortest path interdiction problem by upgrading edges on trees under unit Hamming distance and proposed an $O(n^4 \log n)$ algorithm based on a binary search technique. In Sect. 6, computational experiments were given to show the effectiveness of the algorithms. In Sect. 7, we drew a conclusion and put forward future research.

2 The time complexity of the problem (MSPITH)

In this section, we will discuss the time complexity of the problem **(MSPITH)** under the weighted Hamming distance. We first prove that the problem **(MSPITH)** defined on a chain is NP–hard by transforming it into a 0–1 knapsack problem. Then we extend the NP–hardness to the problem **(MSPITH)** defined on a chain tree, whose degrees

(except the root and the leaves) are all 2. Finally, we can conclude that the problem **(MSPITH)** defined on general rooted trees is still NP–hard.

Theorem 1 *The problem **(MSPITH)** defined on a chain is NP–hard.*

Proof Suppose a tree T degenerates to a chain $P_{s,t}$, where s is the root and t is the only leaf node in L. Then the problem (1) can be transformed as follows.

$$\begin{aligned} \max & \sum_{e\in P_{s,t}} \bar{w}(e) \\ s.t. & \sum_{e\in P_{s,t}} c(e)H(\bar{w}(e), w(e)) \le K, \\ & w(e) \le \bar{w}(e) \le u(e), e \in P_{s,t}. \end{aligned} \tag{2}$$

For convenience, we substitute $H(\bar{w}(e), w(e))$ by $x(e)$, where

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

The objective function can be calculated below.

$$\begin{aligned} \max \sum_{e\in P_{s,t}} \bar{w}(e) &= \max \sum_{e\in P_{s,t}} (w(e) + x(e)(\bar{w}(e) - w(e)) \\ &= \max \sum_{e\in P_{s,t}} (w(e) + x(e) \cdot \Delta w(e)) \\ &= \sum_{e\in P_{s,t}} w(e) + \max \sum_{e\in P_{s,t}} \Delta w(e) \cdot x(e) \end{aligned}$$

Hence, the problem (2) is equivalent to the following problem.

$$\begin{aligned} \max & \sum_{e\in P_{s,t}} \Delta w(e) \cdot x(e) \\ s.t. & \sum_{e\in P_{s,t}} c(e)x(e) \le K, \\ & x(e) = \begin{cases} 0, \text{ if } \Delta w(e) = 0, \\ 1, \text{ if } \Delta w(e) \neq 0. \end{cases} \end{aligned} \tag{3}$$

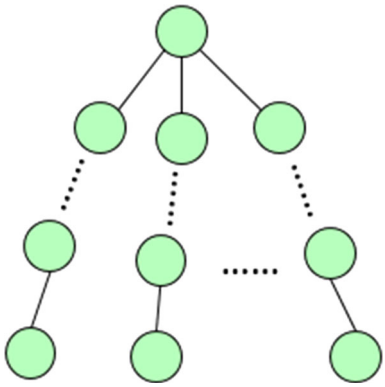
The problem (3) is just a 0–1 knapsack problem which is NP–hard [25]. □

Definition 1 A tree is a **chain tree** if the degrees of its vertices (except the root and the leaves) are all 2.

The tree as shown in Fig. 2 is a chain tree.

Notice that a chain tree is a combination of chains rooted at the same root. Then we can obviously conclude that

Fig. 2 A chain tree



Corollary 2 *The problem* ***(MSPITH)*** *defined on a chain tree is NP–hard.*

Without loss of generality, a general rooted tree is combined with chains and chain trees. Hence we can conclude that

Corollary 3 *The problem* ***(MSPITH)*** *defined on rooted trees is NP–hard.*

Notice that the problem **(MSPITH)** only concerns whether an edge is upgraded or not. The larger the upgrade weight is, the better the objective value of the problem (1) is. Hence, the following theorem holds.

Theorem 4 *If the edge length vector* $\bar{w}$ *is an optimal solution to the problem* ***(MSPITH)****, so is* w^* *defined below.*

$$w^*(e) = \begin{cases} w(e), & \text{if } \bar{w}(e) = w(e), \\ u(e), & \text{if } \bar{w}(e) \neq w(e). \end{cases}$$

Proof Obviously, $\sum_{e\in E} H(w^*(e), w(e)) = \sum_{e\in E} H(\bar{w}(e), w(e))$ and $w(e) \leq w^*(e) \leq u(e)$, thus, w^* is a feasible solution of the problem **(MSPITH)**. We next show that w^* is an optimal solution of the problem **(MSPITH)**.

For convenience, denote by $P_{s,t'}$ the shortest path under $\bar{w}$ and let e_i be the any one upgrade edge with $\bar{w}(e_i) \neq w(e_i)$. Then the relationship between e_i and $P_{s,t'}$ contains the following three cases.

(1) If $e_i \notin P_{s,t'}$, then $d_{w^*}(s, L) = d_{\bar{w}}(s, L) = d_{\bar{w}}(s, t')$.

(2) If $e_i \in P_{s,t'}$ and there is another one path $P_{s,t''}$ that is also the shortest path under $\bar{w}$, we have $d_{w^*}(s, L) = d_{\bar{w}}(s, L) = d_{\bar{w}}(s, t'')$.

(3) If $e_i \in P_{s,t'}$ and $P_{s,t'}$ is the only shortest path under $\bar{w}$, it follows that $w^*(e_i) = u(e_i) > \bar{w}(e_i)$, then

$$\begin{aligned} d_{w^*}(s, L) \geq d_{w^*}(s, t') &= d_{w^*}(P_{s,t'}\backslash\{e_i\}) + w^*(e_i) \\ &= d_{\bar{w}}(P_{s,t'}\backslash\{e_i\}) + u(e_i) \\ &> d_{\bar{w}}(P_{s,t'}\backslash\{e_i\}) + \bar{w}(e_i) \\ &= d_{\bar{w}}(s, t') = d_{\bar{w}}(s, L). \end{aligned}$$

Thus, we have $d_{w^*}(s, L) > d_{\bar{w}}(s, L)$, which contradicts with that $\bar{w}$ is an optimal solution of the problem **(MSPITH)**.

Hence, w^* is also an optimal solution of the problem **(MSPITH)**. □

Since the problem **(MSPITH)** on rooted trees is NP–hard, we will focus on the problem **(MSPIT)** under unit Hamming distance when $c = 1$, which is denoted by **(MSPITUH)** and can be formulated as follows. We can only upgrade at most K critical edges in a rooted tree to maximize the shortest root-leaf distance of the tree in the problem **(MSPITUH)**.

$$\begin{aligned} &\max \min_{t\in L} d_{\bar{w}}(s,t) \\ &s.t. \sum_{e\in E} H(\bar{w}(e), w(e)) \le K \\ &\quad w(e) \le \bar{w}(e) \le u(e), e \in E. \end{aligned} \tag{4}$$

In the next two sections, we will design a greedy algorithm and a dynamic programming algorithm to solve the problem **(MSPITUH)** when $K = 1$ and $K > 1$, respectively.

3 Solve the problem (MSPITUH) when $K = 1$

When $K = 1$, the problem **(MSPITUH)** aims to upgrade one critical edge to maximize the shortest root–leaf distance of a rooted tree. We first analyze some properties of the problem, then present a greedy algorithm with time complexity $O(n + l \log l)$.

Definition 2 Define $L(e) = \{t_k | e \in P_{s,t_k}, k = 1, 2, \cdots, l\}$ as the set of leaves t_k to which P_{s,t_k} passes through e. If $t_k \in L(e)$, we say the edge e **control** the leaf t_k, that is, t_k is a leaf **controlled by** edge e.

Firstly, sort the values of $d_w(s, t_k)(k = 1, 2, \cdots, l)$ in an ascending order and rearrange the label of the leaves, such that

$$d_w(s, t_1^*) \le d_w(s, t_2^*) \le \cdots \le d_w(s, t_l^*).$$

Then we have $d_w(s, t_i^*) \le d_w(s, t_j^*)$ when $i < j$.

In order to maximize the shortest root–leaf distance by updating only one edge, this edge must locate on the path P_{s,t_1^*}, which is the shortest path among all the root–leaf paths. To determine which edge to be upgraded on the path P_{s,t_1^*}, for any edge e, we introduce two leaves with minimum labels that controlled and not controlled by the edge e, respectively.

$$t'(e) = t_{k'}^*, k' = \min\{k | t_k^* \in L(e), k = 1, 2, \cdots, l\}, \tag{5}$$

$$t''(e) = t_{k''}^*, k'' = \min\{k | t_k^* \in L \backslash L(e), k = 1, 2, \cdots, l\}. \tag{6}$$

For any $e_i \in P_{s,t_1^*}$, let

$$\bar{w}^i(e) = \begin{cases} u(e), & e = e_i, \\ w(e), & e \neq e_i. \end{cases} \tag{7}$$

Then $t'(e_i) = t_1^*$ and

$$d_{\bar{w}^i}(s, t_1^*) = d_{\bar{w}^i}(s, t'(e_i)) = \min_{t_k \in L(e_i)} d_{\bar{w}^i}(s, t_k),$$

since the lengths of the paths $P_{s,t_k}(t_k \in L(e_i))$ increase by $\Delta w(e_i)$ simultaneously. On the other hand, the lengths of the paths $P_{s,t_k}(t_k \in L \backslash L(e_i))$ do not change since such leaves t_k are not controlled by e_i, then we have

$$d_{\bar{w}^i}(s, t''(e_i)) = d_w(s, t''(e_i)) = \min_{t_k \in L \backslash L(e_i)} d_w(s, t_k).$$

Thus, the minimum value between $d_{\bar{w}^i}(s, t_1^*)$ and $d_{\bar{w}^i}(s, t''(e_i))$, which is denoted by $\bar{d}(e_i)$, will be the shortest distance when the edge $e_i \in P_{s,t_1^*}$ is upgraded.

To maximize the shortest path, for all edges e_i on path P_{s,t_1^*}, we only need to calculate the relevant value $\bar{d}(e_i)$, then upgrade the edge e_θ with the maximum value of $\bar{d}(e_\theta) = \max_{e_i \in P_{s,t_1^*}} \bar{d}(e_i)$.

Theorem 5 *For all $e_i \in P_{s,t_1^*}$, let*

$$\bar{d}(e_i) = \min\{d_{\bar{w}^i}(s, t_1^*), d_{\bar{w}^i}(s, t''(e_i))\} \tag{8}$$

*and $\bar{d}(e_\theta) = \max_{e_i \in P_{s,t_1^*}} \bar{d}(e_i)$. Then $\bar{w}^\theta$ is an optimal solution of the problem **(MSPITUH)** when $K = 1$.*

Proof We prove it by contradiction. Suppose $\bar{w}^\theta$ is not an optimal solution, but $\bar{w}^\tau$ is, where $e_\tau \in P_{s,t_1^*}$ and $\bar{w}^\tau(e) = \begin{cases} u(e), & e = e_\tau, \\ w(e), & e \neq e_\tau. \end{cases}$ Then we have $\min_{t \in L} d_{\bar{w}^\tau}(s, t) > \min_{t \in L} d_{\bar{w}^\theta}(s, t)$ and

$$\begin{aligned} \min_{t \in L} d_{\bar{w}^\tau}(s, t) &= \min\{\min_{t \in L(e_\tau)} d_{\bar{w}^\tau}(s, t), \min_{t \in L \backslash L(e_\tau)} d_{\bar{w}^\tau}(s, t)\} \\ &= \min\{d_{\bar{w}^\tau}(s, t_1^*), d_{\bar{w}^\tau}(s, t''(e_\tau))\} = \bar{d}(e_\tau). \end{aligned}$$

Similarly, we have

$$\min_{t \in L} d_{\bar{w}^\theta}(s, t) = \bar{d}(e_\theta) < \bar{d}(e_\tau),$$

which contradicts to the condition $\bar{d}(e_\theta) = \max_{e_i \in P_{s,t_1^*}} \bar{d}(e_i)$. Thus, $\bar{w}^\theta$ is an optimal solution. □

Notice that formula (8) can also be presented as:

$$\begin{aligned}\bar{d}(e_i) &= \min\{d_{\bar{w}^i}(s, t_1^*), d_{\bar{w}^i}(s, t''(e_i))\} \\ &= \min\{d_w(s, t_1^*) + \Delta w(e_i), d_w(s, t''(e_i))\}. \end{aligned} \tag{9}$$

According to the analysis above, we propose the following greedy algorithm to solve the problem **(MSPITUH)** when $K = 1$.

Algorithm 1 A greedy algorithm to solve the problem **(MSPITUH)** when $K = 1$.

Require: A tree T rooted at s, the set L of leaves and two edge weight vectors w and u.
Ensure: The upgraded edge e_θ, the optimal value $\bar{d}(e_\theta)$ and the optimal solution $\bar{w}^\theta$.
1: Rearrange the value of $d_w(s, t_k)$ for all $k = 1, 2, \cdots, l$ in an ascending order such that

$$d_w(s, t_1^*) \le d_w(s, t_2^*) \le \cdots \le d_w(s, t_l^*).$$

2: **for** any edge $e_i \in P_{s,t_1^*}$ **do**
3: Calculate the value of $\bar{d}(e_i)$ according to (9), where $t''(e_i)$ is defined by (6).
4: **end for**
5: The upgraded edge is $e_\theta = \arg\max_{e_i \in P_{s,t_1^*}} \bar{d}(e_i)$, the optimal value is $\bar{d}(e_\theta)$ and the optimal solution is $\bar{w}^\theta(e) = \begin{cases} u(e), & e = e_\theta, \\ w(e), & e \neq e_\theta. \end{cases}$

Theorem 6 *When $K = 1$, the problem **(MSPITUH)** can be solved by Algorithm 1 in $O(n + l \log l)$ time.*

Proof In Algorithm 1, sorting the values $d_w(s, t_k), k = 1, 2, \cdots, l$ in Line 1 can be done in $O(l \log l)$ time, where $d_w(s, t_k)$ can be calculated in $O(n)$ time. In Lines 2-4, $\bar{d}(e_i)$ for any $e_i \in P_{s,t_1^*}$ can be obtained by (9) in a constant time. Since $|P_{s,t_1^*}| = O(n)$, then lines 2-4 can be finished in $O(n)$ time. Then the maximal value $\bar{d}(e_\theta)$ can be obtained in Line 5 in $O(n)$ time. Hence, Algorithm 1 runs in $O(n + l \log l)$ time. □

4 Solve the problem (MSPITUH) when $K > 1$

In this section, we first introduce several important definitions and a special data structure of left-subtrees. Then a dynamic programming algorithm with time complexity $O(n(\log n + K^3))$ is proposed. Finally we give an example to execute the algorithm and some numerical experiments to test the effectiveness of the algorithm in Sect. 6.

We firstly introduce a concept of $Tab(v)$ given in [24], but we use a different definition as in [22].

Definition 3 For edge $e_j = (v_i, v_j)$, where v_i is closer to the root s, we call v_i is the **father** of v_j. Define $Tab(s) := 1$ and the Tab of any other vertex $v \in V \setminus \{s\}$ by

$$Tab(v) := \begin{cases} Tab(father(v)), & \text{if } deg(v) \le 2, \\ Tab(father(v)) + 1, & \text{if } deg(v) > 2. \end{cases}$$

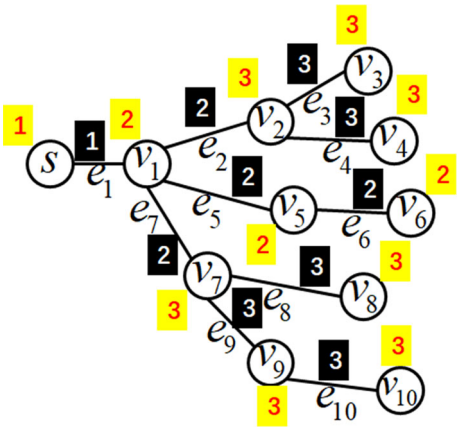


Fig. 3 A tree with $Tab(v)$ in red on node v and layer number $LN(e)$ in white on edge e

Definition 4 [24] For each edge $e = (u, v) \in E$, if $Tab(u) \leq Tab(v)$, we define $LN(e) := Tab(u)$ as the **layer number** of the edge e.

As is shown in Fig. 3, $Tab(s) = 1$, $degree(v_1) = 4 > 2$, $father(v_1) = s$, thus, $Tab(v_1) = 1 + 1 = 2$; $degree(v_5) = 2$, $father(v_5) = v_1$, thus, $Tab(v_5) = 2$; $degree(v_7) = 3$, $father(v_7) = v_1$, then $Tab(v_7) = 2 + 1 = 3$. For edge $e_5 = (v_1, v_5)$, $Tab(v_1) \leq Tab(v_5)$, so $LN(e_5) = Tab(v_1) = 2$.

For convenience, denote by $V^* = \{v \in V \backslash \{s\} | degree(v) > 2\}$ the set of nodes whose degrees are more than 2.

Definition 5 [24] For the any node $v \in V^* \cup \{s\}$, we define a set of **critical children** (denoted by $CC(v)$) as follows. Let v be in the path from s to u. If $degree(u) > 2$ and $Tab(u) = Tab(v) + 1$, then $u \in CC(v)$; if u is a leaf node with $Tab(u) = Tab(v)$, then $u \in CC(v)$.

Now we find the set of critical children of $v_1 \in V^*$ in Fig. 3. Notice that $degree(v_2) > 2$ and $Tab(v_2) = 3 = Tab(v_1) + 1$. Thus, v_2 as well as v_7, is a critical child of the node v_1. Notice that v_6 is a leaf node with $Tab(v_6) = Tab(v_1)$. Then $v_6 \in CC(v_1)$. Hence, $CC(v_1) = \{v_2, v_6, v_7\}$.

In the following parts of this paper, for any $v \in V^* \cup \{s\}$, let $CC(v) = \{v_{h_1}, v_{h_2}, \cdots, v_{h_p}\}$ be the set of critical children of the node v, where

$$p = \begin{cases} degree(v), & v = s, \\ degree(v) - 1, & v \in V^*. \end{cases} \tag{10}$$

Lemma 7 *Suppose w' is an optimal solution of the problem (4). If $\Delta w(e_i) < \Delta w(e_j)$, $w'(e_i) > w(e_i)$, $w'(e_j) = w(e_j)$, $LN(e_i) = LN(e_j)$ and $L(e_i) = L(e_j)$, then w^* is also an optimal solution of the problem (4), where* $w^*(e) = \begin{cases} w(e), & e = e_i, \\ u(e), & e = e_j, \\ w'(e), & otherwise. \end{cases}$

Proof Obviously, $\sum_{e\in E} H(w^*(e), w(e)) = \sum_{e\in E} H(w'(e), w(e))$ and $w(e) \le w^*(e) \le u(e)$, thus, w^* is a feasible solution of the problem (4). We next show that w^* is an optimal solution of (4).

For convenience, denote by $P_{s,t'}$ the shortest path under w'. Then the relationships among the two edges e_i, e_j and $P_{s,t'}$ contain the following three cases.

(1) If $e_i \notin P_{s,t'}$ and $e_j \notin P_{s,t'}$, then $d_{w^*}(s, L) = d_{w'}(s, L) = d_{w'}(s, t')$.
(2) If $e_i \in P_{s,t'}$ and $e_j \in P_{s,t'}$ and there is another one path $P_{s,t''}$ that is also the shortest path under w', we have $d_{w^*}(s, L) = d_{w'}(s, L) = d_{w'}(s, t'')$.
(3) If $e_i \in P_{s,t'}$ and $e_j \in P_{s,t'}$ and $P_{s,t'}$ is the only shortest path under w', it follows from $\Delta w(e_i) < \Delta w(e_j)$, which is equivalent to $u(e_i) - w(e_i) < u(e_j) - w(e_j)$, that $u(e_j) + w(e_i) > u(e_i) + w(e_j)$. Since $w'(e_j) = w(e_j)$, then

$$\begin{aligned} d_{w^*}(s, L) \ge d_{w^*}(s, t') &= d_{w^*}(P_{s,t'}\backslash\{e_i, e_j\}) + w^*(e_i) + w^*(e_j) \\ &= d_{w'}(P_{s,t'}\backslash\{e_i, e_j\}) + w(e_i) + u(e_j) \\ &> d_{w'}(P_{s,t'}\backslash\{e_i, e_j\}) + u(e_i) + w(e_j) \\ &\ge d_{w'}(P_{s,t'}\backslash\{e_i, e_j\}) + w'(e_i) + w'(e_j) \\ &= d_{w'}(s, t') = d_{w'}(s, L). \end{aligned}$$

Thus, we have $d_{w^*}(s, L) > d_{w'}(s, L)$, which contradicts with that w' is an optimal solution of (4).

Hence, w^* is also an optimal solution of the problem (4). □

Based on the lemma above, without loss of generality, we can rearrange the edges with the same layer number on the same path such that their values of $\Delta w(e)$ are sorted non-increasingly. Notice that if the K upgraded edges have the same layer number on the same path, we can update the first K edges in this layer on this path.

For any $v \in V^* \cup \{s\}$, $CC(v) = \{v_{h_1}, v_{h_2}, \cdots, v_{h_p}\}$ is the set of critical children, where p is defined as in (10), we define the left $q-$subtree as follows.

Definition 6 Define the left $q-$subtree rooted at v as $T_v^{1:q} := P_{v,v_{h_1}} \cup T_{v_{h_1}} \cup P_{v,v_{h_2}} \cup T_{v_{h_2}} \cup \cdots \cup P_{v,v_{h_q}} \cup T_{v_{h_q}}$, $q = 1, 2, \cdots, p$, where $T_{v_{h_i}}$ is the subtree rooted at v_{h_i} and $T_{v_{h_i}} := \emptyset$ if v_{h_i} is a leaf. Specially, $T_v^{1:p}$ can be denoted by T_v and $T_v^{1:0} := \emptyset$.

As shown in Fig. 4, the area labeled in red and black are the left 2-subtree $T_v^{1:2}$ and the left $q-$subtree $T_v^{1:q}$ of $T_v = T_v^{1:p}$, respectively.

Now we define the following two functions $g(P_{v,v_h}, k)$ and $f(T_v^{1:q}, k)$ to obtain a dynamic programming algorithm for the problem **(MSPITUH)** when $K > 1$.

1. The function $g(P_{v,v_h}, k)$ defined on the chain P_{v,v_h}.

For any $v \in V^* \cup \{s\}$ and any $v_h \in CC(v)$, let $P_{v,v_h} := \{e_{i_1}, e_{i_2}, \cdots, e_{i_\beta}\}$ with $\Delta w(e_{i_1}) \ge \Delta w(e_{i_2}) \ge \cdots \ge \Delta w(e_{i_\beta})$, where $\beta = |E(P_{v,v_h})|$. Define $g(P_{v,v_h}, k)$ as the sum of the edge lengths when the first k edges are upgraded on the chain P_{v,v_h}, where $0 \le k \le \beta$. Let $E(P_{v,v_h}, k)$ be the set of upgraded edges on the chain P_{v,v_h}.

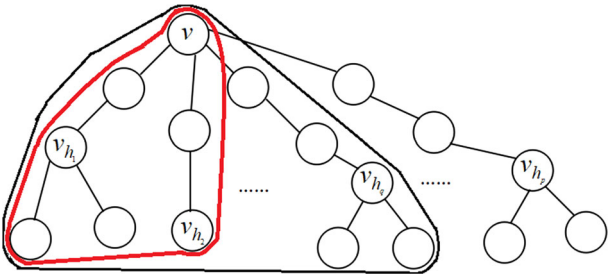


Fig. 4 The subtree $T_v = T_v^{1:p}$ is shown. The area labeled in red and black are the subtrees $T_v^{1:2}$ and $T_v^{1:q}$, respectively

We can calculate all the values $g(P_{v,v_h}, k)$ for all the chains in the tree T according to the formula below.

$$g(P_{v,v_h},k) = \begin{cases} \sum_{j=1}^{\beta} w(e_{i_j}), & \text{if } k = 0, \\ \sum_{j=1}^{\beta} w(e_{i_j}) + \sum_{j=1}^{k} \Delta w(e_{i_j}), & \text{if } 1 \le k \le \beta. \end{cases} \tag{11}$$

$$E(P_{v,v_h},k) = \bigcup_{j=1}^{k} \{e_{i_j}\}. \tag{12}$$

2. The function $f(T_v^{1:q}, k)$ defined on the non-chain $T_v^{1:q}$.

For all $v \in V^* \cup \{s\}$, define $f(T_v^{1:q}, k) := \max \min_{t \in T_v^{1:q} \cap L} d_{\tilde{w}}(s,t)$ as the optimal value of $T_v^{1:q}$ when k edges are upgraded, where $0 \le k \le |E(T_v^{1:q})|$ and let $E_v^{1:q}(k)$ be the corresponding set of k upgraded edges. Specially, denote $E_v^{1:p}(k)$ by $E_v(k)$ for simplicity.

We consider two cases to calculate the optimal value $f(T_v^{1:q}, k)$.

Case 1. when $q = 1$.

We have $T_v^{1:1} = P_{v,v_{h_1}} \cup T_{v_{h_1}}$ as shown in Fig. 4.

$$\begin{aligned} f(T_v^{1:1},k) &= \max\{g(P_{v,v_{h_1}},k_1) + f(T_{v_{h_1}},k_2)\} \\ s.t. \quad & k_1 + k_2 = k, \\ & k = 0,1,2,\cdots,|E(T_v^{1:1})|, \\ & k_1 = 0,1,2,\cdots,\min\{k,|E(P_{v,v_{h_1}})|\}, \\ & k_2 = 0,1,2,\cdots,\min\{k,|E(T_{v_{h_1}})|\}. \end{aligned} \tag{13}$$

If the optimal value $f(T_v^{1:1}, k) = g(P_{v,v_{h_1}}, k_1^*) + f(T_{v_{h_1}}, k - k_1^*)$ is obtained when $k_1 = k_1^*$, then the set of upgraded edges is

$$E_v^{1:1}(k) = E(P_{v,v_{h_1}}, k_1^*) \cup E_{v_{h_1}}(k - k_1^*). \tag{14}$$

Specially, when v_{h_1} is a leaf, $E(T_{v_{h_1}}) = \varnothing$ and $|E(T_{v_{h_1}})| = 0$, thus, $k_2 = 0$, $f(T_{v_{h_1}}, k_2) = f(T_{v_{h_1}}, 0) = 0$,

$$f(T_v^{1:1}, k) = g(P_{v,v_{h_1}}, k). \tag{15}$$

$$E_v^{1:1}(k) = E(P_{v,v_{h_1}}, k). \tag{16}$$

Case 2: when $q \geq 2$.

For any $v \in V^* \cup \{s\}$, $CC(v) = \{v_{h_1}, v_{h_2}, \cdots, v_{h_p}\}$ is the set of critical children. Then $T_v^{1:q} = T_v^{1:(q-1)} \cup P_{v,v_{h_q}} \cup T_{v_{h_q}}$, $q = 2, \cdots, p$.

$$\begin{aligned} f(T_v^{1:q}, k) = \max \min\{&g(P_{v,v_{h_q}}, k_1) + f(T_{v_{h_q}}, k_2),\ f(T_v^{1:(q-1)}, k_3)\}, \\ s.t. \quad & k_1 + k_2 + k_3 = k, \\ & k = 0, 1, \cdots, |E(T_v^{1:q})|, \\ & k_1 = 0, 1, \cdots, \min\{k, |E(P_{v,v_{h_q}})|\}, \\ & k_2 = 0, 1, \cdots, \min\{k, |E(T_{v_{h_q}})|\}, \\ & k_3 = 0, 1, \cdots, \min\{k, |E(T_v^{1:(q-1)})|\}. \end{aligned} \tag{17}$$

If the optimal value

$$f(T_v^{1:q}, k) = \max\{g(P_{v,v_{h_q}}, k_1^*) + f(T_{v_{h_q}}, k_2^*),\ f(T_v^{1:(q-1)}, k - k_1^* - k_2^*)\}$$

is obtained when $k_1 = k_1^*$, $k_2 = k_2^*$, then its set of upgraded edges is

$$E_v^{1:q}(k) = E(P_{v,v_{h_q}}, k_1^*) \cup E_{v_{h_q}}(k_2^*) \cup E_v^{1:(q-1)}(k - k_1^* - k_2^*). \tag{18}$$

Specially, when v_{h_q} is a leaf, $E(T_{v_{h_q}}) = \varnothing$ and $|E(T_{v_{h_q}})| = 0$, thus, $k_2 = 0$, $f(T_{v_{h_q}}, k_2) = f(T_{v_{h_q}}, 0) = 0$, $T_v^{1:q} = T_v^{1:(q-1)} \cup P_{v,v_{h_q}}$,

$$f(T_v^{1:q}, k) = \max \min\{g(P_{v,v_{h_q}}, k_1),\ f(T_v^{1:(q-1)}, k - k_1)\}, \tag{19}$$

If the optimal value $f(T_v^{1:q}, k) = \max\{g(P_{v,v_{h_q}}, k_1^*),\ f(T_v^{1:(q-1)}, k - k_1^*)\}$ is obtained when $k_1 = k_1^*$, then its set of upgraded edges is

$$E_v^{1:q}(k) = E(P_{v,v_{h_q}}, k_1^*) \cup E_v^{1:(q-1)}(k - k_1^*). \tag{20}$$

To sum up, traverse the rooted tree T from leaves to the root s to calculate all the function values $g(P_{v,v_h}, k)$ and $f(T_v^{1:q}, k)$. Then $f(T_s, K)$ is the optimal value of the problem **(MSPITUH)**, $E_s(K)$ is the set of upgraded edges and an optimal solution is

$$\bar{w}(e) = \begin{cases} u(e), & e \in E_s(K), \\ w(e), & e \notin E_s(K). \end{cases} \tag{21}$$

According to the analysis above, we have the following dynamic programming algorithm for the problem **(MSPITUH)** when $K > 1$.

Algorithm 2 A dynamic programming algorithm $[E_s(K), \bar{w}, f(T_s, K)] = MSPITUH(T, L, w, u, K)$

Require: A tree T rooted at s, the set L of leaves, two edge weight vectors w and u and the number K of upgraded edges.

Ensure: The set $E_s(K)$ of upgraded edges, the upgraded length vector $\bar{w}$ and the optimal value $f(T_s, K)$.

1: **(Breath-First Search (BFS).)** Let $V^* = \{v \in V \backslash \{s\} | degree(v) > 2\}$. Start from the root s and label each node v by $Lab(v)$ when using the breath-first search strategy on T, where $Lab(s) = 1$. While executing the BFS, calculate $Tab(v)$ for each node v and $LN(e)$ for each edge e. Find the sets $CC(v)$ of critical children for each $v \in V^* \cup \{s\}$.

2: For each $v \in V^* \cup \{s\}$ and each $v_{h_j} \in CC(v)$, find all the chains $P_{v,v_{h_j}}$, and rearrange the edges in each chain $P_{v,v_{h_j}}$ such that the values $\Delta w(e) := u(e) - w(e)$ are in non-increasing order, where $j = 1, \cdots, p$, and p is as defined in (10).

3: **for** any $v \in V^* \cup \{s\}$ **do**

4: **for** $v_h \in CC(v)$ **do**

5: Calculate the value $g(P_{v,v_h}, k)$ and the set $E(P_{v,v_h}, k)$ of upgraded edges by (11) and (12), for each $k = 0, 1, \cdots, \min\{K, |E(P_{v,v_h})|\}$.

6: **end for**

7: **end for**

8: **for** any $v \in V^* \cup \{s\}$ in descending order of the labels $Lab(v)$ of nodes **do**

9: **for** $q = 1 : p$ **do**

10: **for** $k = 0 : \min\{K, |E(T_v^{1:q})|\}$ **do**

11: Calculate the value $f(T_v^{1:q}, k)$ by (13), (15), (17) and (19), and obtain the set $E_v^{1:q}(k)$ of upgraded edges by (14), (16), (18) and (20).

12: **end for**

13: **end for**

14: **end for**

15: For the root s, calculate the optimal value $f(T_s, K)$, the set $E_s(K)$ of upgraded edges and the optimal solution $\bar{w}$ given by (21).

Theorem 8 *The dynamic programming Algorithm 2 can solve the problem* ***(MSPITUH)*** *when $K > 1$ in $O(n(\log n + K^3))$ time, where n is the number of nodes in a given tree.*

Proof In Line 1, the BFS can be executed in $O(n)$ time. Rearranging the edges in each chain can be finished in $O(n \log n)$ time in Line 2.

In Lines 3-7, for a given chain P_{v,v_h}, the value $g(P_{v,v_h}, 0)$ can be obtained in $O(|P_{v,v_h}|)$ and $g(P_{v,v_h}, k+1) = g(P_{v,v_h}, k) + \Delta w(e_{i_{k+1}})$ holds. Thus, all the values $g(P_{v,v_h}, k)$ for $k = 0, 1, \cdots, \min\{K, |E(P_{v,v_h})|\}$ can be calculated in $O(|P_{v,v_h}|)$. Therefore for any $v \in V^* \cup \{s\}$ and any $v_h \in CC(v)$, calculating all the K values $g(P_{v,v_h}, k)$ are just traversing the edges in every chain. Hence, the total time of Lines 3-7 is $\sum_{v \in V^* \cup \{s\}, v_h \in CC(v)} O(|P_{v,v_h}|) = O(n)$.

In Lines 8-14, for a given subtree $T_v^{1:q}$ and a value k, $f(T_v^{1:q}, k)$ can be solved in $O(K^2)$ by (17), since $k_3 = k - k_1 - k_2$, k_1 and k_2 have $O(K^2)$ kinds of possible combinations and there is only one addictive operation and one comparison for each pair of combination. Thus, all the K values $f(T_v^{1:q}, k)$ can be completed in $O(K^3)$ time

in Lines 10-12. For any $v \in V^* \cup \{s\}$, $q = 1, 2 \cdots, p$,where p is defined as in (10). in Lines 8-14 we need to solve $\sum_{v \in V^* \cup \{s\}} (deg(v) - 1) \leq 2(n-1) - |V^*| = O(n)$ subproblems defined on the subtrees $T_v^{1:q}$. Therefore, the total time complexity of Lines 8-14 is $O(nK^3)$.

As a conclusion, the time complexity of Algorithm 2 is $O(n \log n + n) + O(nK^3) = O(n(\log n + K^3))$. □

5 Solve the problem (MCPITUH)

Now we consider a minimum cost shortest path interdiction problem by upgrading edges on trees under unit Hamming distance, which is similarly denoted by **(MCPITUH)**. We aim to minimize the total number of upgrade edges on the premise that the shortest root-leaf distance of the tree is lower bounded by a given value D.

$$\begin{aligned} & \min \sum_{e \in E} H(\bar{w}(e), w(e)) \\ \textbf{(MCSPITUH)} \quad & s.t. \ \min_{t \in L} d_{\bar{w}}(s,t) \geq D, \\ & w(e) \leq \bar{w}(e) \leq u(e), e \in E. \end{aligned} \tag{22}$$

For convenience, denote by **(MSPITUH**(K)**)** and **(MCSPITUH**(D)**)** the problem **(MSPITUH)** with a given K and the problem **(MCSPITUH)** with a given D, respectively.

The problem **(MSPITUH**(K)**)** can be solved by Algorithm 2 for a given K in Sect. 4. In the problem **(MCSPITUH**(D)**)**, we are searching for the smallest K^* such that the problem **(MSPITUH**(K^*)**)** generates an upgrade vector w^* with $\min_{t \in L} d_{w^*}(s,t) \geq D$. Furthermore, we can obviously observe that for any D' and D'' with $D' < D''$, the number of upgrade edges for the problem **(MCPITUH**(D')**)** is not larger than that for **(MCPITUH**(D'')**)**.

To solve the problem **(MCSPITUH)**, we aim to find the optimal K^* among the values $\{1, 2, \cdots, n\}$ by the binary search method, and in each iteration we solve a problem **(MSPITUH**(k)**)** by Algorithm 2, in which k is the median of the current interval $[k_1, k_2] \subseteq [1, n]$. Hence, the problem **(MCSPITUH)** can be solved in $O(n \log n(\log n + k^3)) = O(n \log n(\log n + n^3)) = O(n^4 \log n)$, as shown in Algorithm 3.

6 Computational experiments

6.1 An example to show the process of Algorithm 2

For the better understanding of Algorithm 2, Example 1 is given to show the detailed computing process.

Algorithm 3 Solve the problem **(MCSPITUH)**.

Require: A tree T rooted at s, the set L of leaves, two edge weight vectors w and u and the lower bound D of the length of the shortest path.
Ensure: The optimal objective value K^*, the set E^* of upgrade edges and an optimal solution w^*.
1: Initialize $k_1 := 1, k_2 := n, i := 1$.
2: **while** $k_2 \neq k_1 + 1$ **do**
3: let $k := \lceil \frac{k_1+k_2}{2} \rceil$,
4: Call $[E^*, w^*, D^*(i)] = MSPITUH(T, L, w, u, k)$.
5: **if** $D^*(i) < D$ **then**
6: let $k_1 := k$,
7: **else if** $D^*(i) > D$ **then**
8: let $k_2 := k$,
9: **else**
10: let $K^* := k$,
11: **return** (K^*, E^*, w^*).
12: **end if**
13: Update $i := i + 1$.
14: **end while**
15: Update $K^* := k_2$.
16: Call $[E^*, w^*, D^*(i)] = MSPITUH(T, L, w, u, K^*)$.
17: **return** (K^*, E^*, w^*).

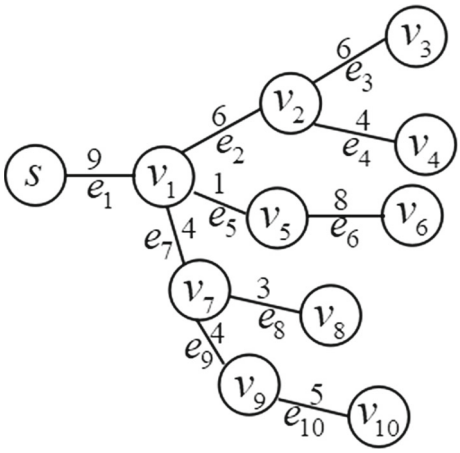


Fig. 5 A tree with rearranged edges

Example 1 As shown in Fig. 5, let $V := \{s, v_1, \cdots, v_{10}\}$, $E := \{e_1, \cdots, e_{10}\}$, $t_1 := v_3$, $t_2 := v_4$, $t_3 := v_6$, $t_4 := v_8$, $t_5 := v_{10}$, $u(e_j) := 10$ for all $j := 1, 2, \cdots, 10$ and $K := 5$. The length vector $w := (9, 6, 6, 4, 1, 8, 4, 3, 4, 5)$ has been rearranged in the non-increasing order of $\Delta w(e)$ for each chain.

1. Note that $V^* \cup \{s\} := \{s, v_1, v_2, v_7\}$, and the sets of critical children are $CC(s) := v_1$, $CC(v_1) := \{v_2, v_6, v_7\}$, $CC(v_2) = \{v_3, v_4\}$, $CC(v_7) := \{v_8, v_{10}\}$.

2. For every $v \in V^* \cup \{s\}$ and $v_h \in CC(v)$, calculate the values $g(P_{v,v_h}, k)$ and the set $E(P_{v,v_h}, k)$ of upgraded edges by (11) and (12) for all $k = 0, 1, \cdots, |E(P_{v,v_h})|$. We take the chain P_{v_1,t_3} as an example. Other values $g(P_{v,v_h}, k)$ can be found on the first three columns in Table 2.

Table 2 The detailed results of Example 1

P	$g(P,k)$		T_v	$f(T_v,k)$		T_{v_1}	$f(T_{v_1},k)$		$f(T_s,K)$ $E_s(K)$
	k	$g(P,k)$ $E(P,k)$		k	$f(T_v,k)$ $E_v(k)$		k	$f(T_{v_1},k)$ $E_{v_1}(k)$	
P_{v_7,t_5}	0	9 $(\emptyset)$	T_{v_7}	0	3 $\emptyset$	T_{v_1}			$f(T_s,5)=25$ $\{e_2,e_4,e_5,e_7,e_8\}$
	1	15 $\{e_9\}$		1	9 $(\{e_8\})$				
	2	20 $\{e_9,e_{10}\}$		2	10 $\{e_8,e_9\}$				
P_{v_7,t_4}	0	3 $\emptyset$		3	10 $\{e_8,e_9,e_{10}\}$				
	1	10 $\{e_8\}$					0	7 $\emptyset$	
P_{v_1,v_7}	0	4 $\emptyset$					1	9 $\{e_7\}$	
	1	10 $\{e_7\}$					2	10 $\{e_5,e_7\}$	
P_{v_1,t_3}	0	9 $\emptyset$					3	13 $\{e_2,e_5,e_7\}$	
	1	18 $\{e_5\}$					4	14 $\{e_2,e_5,e_7,e_8\}$	
	2	20 $\{e_5,e_6\}$					5	16 $\{e_2,e_4,e_5,e_7,e_8\}$	
P_{v_2,t_2}	0	4 $\emptyset$	T_{v_2}	0	4 $\emptyset$				
	1	10 $\{e_4\}$		1	6 $\{e_4\}$				
P_{v_2,t_1}	0	6 $\emptyset$		2	10 $\{e_3,e_4\}$				
	1	10 $\{e_3\}$							
P_{v_1,v_2}	0	6 $\emptyset$							
	1	10 $\{e_2\}$							
P_{s,v_1}	0	9 $\emptyset$							
	1	10 $\{e_1\}$							

$g(P_{v_1,t_3},0) := 9$, $E(P_{v_1,t_3},0) := \varnothing$; $g(P_{v_1,t_3},1) := 18$, $E(P_{v_1,t_3},1) := \{e_5\}$; $g(P_{v_1,t_3},2) := 20$, $E(P_{v_1,t_3},2) := \{e_5,e_6\}$.

3. For any $v \in V^* \cup \{s\} := \{s,v_1,v_2,v_7\}$ and $k = 0,\cdots,\min\{K,|E(T_v^{1:q})|\}$, calculate all the values $f(T_v^{1:q},k)$ by (13), (15), (17) and (19).

(1) For v_7, $CC(v_7) := \{t_5,t_4\}$.

Note that $T_{v_7}^{1:1} := P_{v_7,t_5}$, it follows from (15) and (16) that

$f(T_{v_7}^{1:1}, 0) := g(P_{v_7,t_5}, 0) = 9$, $E_{v_7}^{1:1}(0) := E(P_{v_7,t_5}, 0) = \varnothing$;
$f(T_{v_7}^{1:1}, 1) := g(P_{v_7,t_5}, 1) = 10$, $E_{v_7}^{1:1}(1) := E(P_{v_7,t_5}, 1) = \{e_9\}$;
$f(T_{v_7}^{1:1}, 2) := g(P_{v_7,t_5}, 2) = 20$, $E_{v_7}^{1:1}(2) := E(P_{v_7,t_5}, 2) = \{e_9, e_{10}\}$.

Note that $T_{v_7} := T_{v_7}^{1:1} \cup P_{v_7,t_4}$, it follows from (19) and (20) that

$f(T_{v_7}, 0) := \min\{g(P_{v_7,t_4}, 0), f(T_{v_7}^{1:1}, 0)\} = \min\{3, 9\} = 3$, $E_{v_7}(0) = \varnothing$;

$f(T_{v_7}, 1) := \max\{\min\{g(P_{v_7,t_4}, 0), f(T_{v_7}^{1:1}, 1)\}, \min\{g(P_{v_7,t_4}, 1), f(T_{v_7}^{1:1}, 0)\}\} = \max\{3, 9\} = 9$, $E_{v_7}^{1:2}(1) := E(P_{v_7,t_4}, 1) \cup E_{v_7}^{1:1}(0) = \{e_8\}$;

$f(T_{v_7}, 2) := \max\{\min\{g(P_{v_7,t_4}, 0), f(T_{v_7}^{1:1}, 2)\}, \min\{g(P_{v_7,t_4}, 1), f(T_{v_7}^{1:1}, 1)\}\} = \max\{3, 10\} = 10$, $E_{v_7}(2) := E(P_{v_7,t_4}, 1) \cup E_{v_7}^{1:1}(1) = \{e_8, e_9\}$;

$f(T_{v_7}, 3) := \min\{g(P_{v_7,t_4}, 1), f(T_{v_7}^{1:1}, 2)\} = \min\{10, 20\} = 10$, $E_{v_7}(3) := E_{v_7}^{1:1}(2) \cup E(P_{v_7,t_4}, 1) = \{e_8, e_9, e_{10}\}$.

We omit the calculation process of $f(T_{v_2}, k)(k := 0, 1, 2)$ and $f(T_{v_1}, k)(k := 0, 1, 2, 3, 4)$, which can be found in Table 2. We take the calculation of $f(T_{v_1}, 5)$ as an example.

$f(T_{v_1}, 5) := \max\{\min\{g(P_{v_1,v_2}, 0) + f(T_{v_2}, 0), f(T_{v_1}^{1:2}, 5)\}, \min\{g(P_{v_1,v_2}, 0) + f(T_{v_2}, 1), f(T_{v_1}^{1:2}, 4)\}, \min\{g(P_{v_1,v_2}, 0) + f(T_{v_2}, 2), f(T_{v_1}^{1:2}, 3)\}, \min\{g(P_{v_1,v_2}, 1) + f(T_{v_2}, 0), f(T_{v_1}^{1:2}, 4)\}, \min\{g(P_{v_1,v_2}, 1) + f(T_{v_2}, 1), f(T_{v_1}^{1:2}, 3)\}, \min\{g(P_{v_1,v_2}, 1) + f(T_{v_2}, 2), f(T_{v_1}^{1:2}, 2)\}\} = \max\{10, 12, 16, 14, 16, 13\} = 16$,

$E_{v_1}(5) := E(P_{v_1,v_2}, 1) \cup E_{v_2}(1) \cup E_{v_1}^{1:2}(3) = \{e_2, e_4, e_5, e_7, e_8\}$;

For s, $CC(s) := \{v_1\}$, $T_s := P_{s,v_1} \cup T_{v_1}$, it follows from (13) and (14)that $f(T_s, 5) := \max\{g(P_{s,v_1}, 0) + f(T_{v_1}, 5), g(P_{s,v_1}, 1) + f(T_{v_1}, 4)\} = \max\{25, 24\} = 25$,$E_s(5) := E(P_{s,v_1}, 0) \cup E_s(5) = \{e_2, e_4, e_5, e_7, e_8\}$.

Thus, when $K := 5$, the set of upgraded edges is $E_s(5) := \{e_2, e_4, e_6, e_7, e_8\}$ and an optimal solution is $\bar{w} := (9, 10, 6, 10, 8, 10, 10, 10, 5, 4)$ with the optimal value $f(T_s, 5) := 25$.

6.2 Computational experiments

Now we present computational experiments of Algorithms 1, 2 and 3. The programs were coded in Matlab 7.0 and run on a PC Intel(R), Core(TM)i7-8565U CPU @ 1.8 GHz 1.99 GHz under Windows 10. We have tested the algorithms on 8 classes of random trees, with the number n of vertices varying from 100 to 10000. For each class, we randomly generate 30 instances. For each instance, we randomly generated a tree such that the length of each root-leaf path is between $Pathmin$ and $Pathmax$, where $Pathmin$ is a random integer in the range $(\lfloor n/100 \rfloor, \lfloor n/30 \rfloor)$ and $Pathmax$ is the sum of $Pathmin$ and a random integer in the range $(\lfloor n/50 \rfloor, \lfloor n/30 \rfloor)$. For each tree, we use the maximum distance λ of the longest root-leaf path to describe the depth of the tree and the number l of leaves to denote the width of the tree. Let λ_{ave} and l_{ave} be the average values of λ and l, respectively. We randomly generated two vectors w, u satisfying $w < u$ which means that $\Delta w = u - w > 0$. For each randomly generated tree, we first solve the problem **(MCSPITUH)** with $K = 1$ by Algorithms 1 and 2 for comparison, respectively. Then solve the problem **(MSPITUH)** by Algorithm 2 with randomly generated $K > 1$. Let T_1, T_2^{K1}, T_2 be the average CPU time of Algorithm 1, Algorithm 2 for $K = 1$ and $K > 1$ respectively. Let T_3 be that of Algorithm

Table 3 Performances of Algorithms 1, 2 and 3

n	100	200	500	1000	3000	5000	7000	10000
λ_{ave}	5.73	12.90	34.40	64.43	214.33	335.90	452.20	682.78
l_{ave}	55.07	81.23	91.27	115.90	304.73	302.87	297.95	262.22
T_1	0.02	0.04	0.09	0.18	0.63	1.34	3.15	6.72
T_2^{K1}	0.23	0.41	0.57	0.83	3.27	4.40	8.43	12.55
T_2	**0.39**	**1.05**	**3.87**	**22.77**	**181.44**	**772.52**	**3533**	**15975**
K_{ave}	27.77	52.13	101.17	238.03	658.93	1172.30	1987.40	3027.30
T_{2min}	0.16	0.50	0.38	1.96	3.53	5.40	35	4320
T_{2max}	0.62	1.79	9.18	63.07	688.18	2255.90	10073	40035
T_3	**2.27**	**5.63**	**21.29**	**95.01**	**939.65**	**4195.30**	**13646**	**44108**
L_{ave}	47.90	166.77	1392.50	4702.70	20923	66987	124780	266950
T_{3min}	0.49	3.85	15.05	55.96	631.60	2142	6742	30729
T_{3max}	3.35	9.15	27.02	167.63	833.60	7678	24387	71509

3. Let K_{ave} be the average value of 30 randomly generated integers K in the range $(1, \lfloor n/2 \rfloor)$. Let L_{ave} be the average value of 30 randomly generated integers L in the range $[L_{min}, L_{max}]$, where $L_{min} = \min_{t \in Y} d_w(s,t)$ and $L_{max} = \max_{t \in Y} d_u(s,t)$, respectively. For T_2 and T_3, the relevant minimum and maximum running time, denoted by T_{min}, T_{max}, respectively, are recorded as well.

Compared the time T_1 with T_2^{K1} in Table 3, it can be seen that Algorithm 1 for the problem **(MSPITUH)**, which has time complexity $O(n + l \log l)$, is really more efficient than Algorithm 2 that runs in $O(n(\log n + K^3))$ time. For general K, the running time mainly depends on the number K. If $K = O(n)$, then the time complexity is $O(n^4)$. Thus we can see from Table 3 that the running time increases dramatically as K increases.

7 Conclusion and further research

We consider the problem **(MSPITH)** for $c(e) > 0$ and $c(e) = 1$, $e \in E$. We prove the problem **(MSPITH)** is NP-hard by transforming the problem on a chain into a 0-1 knapsack problem. For the problem **(MSPITUH)** under unit Hamming distance, we propose a greedy algorithm and a dynamic programming algorithm when $K = 1$ and $K > 1$ with time complexity $O(n + l \log l)$ and $O(n(\log n + K^3))$, respectively, where l is the number of leaves. Furthermore, we solve the problem **(MCSPITUH)** in $O(n^4 \log n)$ time by using the binary search method, and in each iteration we call the algorithm to solve the problem **(MSPITUH)**.

For further research, we can devise approximation algorithm for the problem **(MSPITH)** under weighted Hamming distance. We can also consider the shortest path interdiction problem on a general graph. Moreover, we can consider some other network interdiction problems by upgrading critical edges, such as minimum spanning tree interdiction problems.

References

1. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows. Prentice-Hall, Englewood Cliffs, NJ (1993)

2. Albert, R., Jeong, H., Barabasi, A.: Error and attack tolerance of complex networks. Nature **406**(6794), 378–382 (2000)
3. Khachiyan, L., Boros, E., Borys, K., Elbassioni, K., Gurvich, V., Rudolf, G., Zhao, J.: On short paths interdiction problems: total and node-wise limited interdiction. Theory Comput. Syst. **43**(2), 204–233 (2008)
4. Corley, H.W., Sha, D.Y.: Most vital links and nodes in weighted networks. Oper. Res. Lett. **1**, 157–161 (1982)
5. Bar-Noy, A., Khuller, S., Schieber, B.: The complexity of finding most vital arcs¡¡ and nodes, Technical Report CS-TR-3539. University of Maryland, Department of Computer Science (1995)
6. Nardelli, E., Proietti, G., Widmyer, P.: A faster computation of the most vital edge of a shortest path between two nodes. Inf. Process. Lett. **79**(2), 81–85 (2001)
7. Frederickson, G.N., Solis-Oba, R.: Increasing the weight of minimum spanning trees. In: Proceedings of the 7th ACM–SIAM Symposium on Discrete Algorithms (SODA 1996), 539–546, (1996)
8. Bazgan, C., Toubaline, S., Vanderpooten, D.: Efficient determination of the k most vital edges for the minimum spanning tree problem. Comput. Oper. Res. **39**(11), 2888–2898 (2012)
9. Pettie, S.: Sensitivity analysis of minimum spanning tree in sub-inverse- Ackermann time. In: Proceedings of 16th international symposium on algorithms and computation (ISAAC 2005), Lecture notes in computer science, 3827, 964–73, (2005)
10. Iwano, K., Katoh, N.: Efficient algorithms for finding the most vital edge of a minimum spanning tree. Inf. Process. Lett. **48**(5), 211–213 (1993)
11. Liang, W.: Finding the k most vital edges with respect to minimum spanning trees for fixed k. Dis. Appl. Math. **113**(2–3), 319–327 (2001)
12. Zenklusen, R., Ries, B., Picouleau, C., Werra, D., Bentz, C., de Costa, M.: Blockers and transversals. Dis. Math. **309**(13), 4306–4314 (2009)
13. Zenklusen, R.: Matching interdiction. Dis. Appl. Math. **158**(15), 1676–1690 (2010)
14. Bazgan, C., Toubaline, S., Vanderpooten, D.: Critical edges for the assignment problem: complexity and exact resolution. Oper. Res. Lett. **41**, 685–689 (2013)
15. Ries, B., Bentz, C., Picouleau, C., Werra, D., Zenklusen, R., de Costa, M.: Blockers and transversals in some subclasses of bipartite graphs: when caterpillars are dancing on a grid. Dis. Math. **310**(1), 132–146 (2010)
16. Zenklusen, R.: Network flow interdiction on planar graphs. Dis. Appl. Math. **158**(13), 1441–1455 (2010)
17. Altner, D.S., Ergun, Z., Uhan, N.A.: The maximum flow network interdiction problem: valid inequalities, integrality gaps and approximability. Oper. Res. Lett. **38**, 33–38 (2010)
18. Bazgan, C., Toubaline, S., Vanderpooten, D.: Complexity of determining the most vital elements for the p-median and p-center location problems. J. Combin. Optim. **25**(2), 191–207 (2013)
19. Bazgan, C., Nichterlein, A., et al.: A refined complexity analysis of finding the most vital edges for undirected shortest paths: algorithms and complexity. Lecture Notes Comput. Sci. **9079**, 47–60 (2015)
20. Zhang, H.L., Xu, Y.F., Wen, X.G.: Optimal shortest path set problem in undirected graphs. J. Combin. Optim. **29**(3), 511–530 (2015)
21. Ertugrl, A., Gokhan, O., Cevriye, T.G.: Determining the most vital arcs on the shortest path for fire trucks in terrorist actions that will cause fire. Commun. Fac. Sci. Univ. Ank. Ser. A1 Math. Stat. **68**(1), 441–450 (2019)
22. Zhang, Q., Guan, X.C., Pardalos, P.M.: Maximum shortest path interdiction problem by upgrading edges on trees under weighted l_1 norm. J. Global Optim. (2020). https://doi.org/10.1007/s10898-020-00958-0
23. Mohammadi, A., Tayyebi, J.: Maximum capacity path interdiction problem with fixed costs. Asia Pacific J. Oper. Res. **36**(4), 1950018 (2019)
24. Zhang, B.W., Guan, X.C., Pardalos, P.M., et al.: An algorithm for solving the shortest path improvement problem on rooted trees under unit hamming distance. J. Optim. Theory Appl. **178**, 538–559 (2018)
25. Papadimitriou, C.H., Steiglitz, K.: Combinatorial optimization: algorithms and complexity, Dover Publications, the second edition, (1988)