Journal of Combinatorial Optimization (2022) 44:74–93
https://doi.org/10.1007/s10878-021-00819-w

The sum of root-leaf distance interdiction problem by upgrading edges/nodes on trees

Qiao Zhang[1] · Xiucui Guan[1] · Junhua Jia[1] · Xinqiang Qian[1]

Accepted: 24 September 2021 / Published online: 9 October 2021

Abstract

Network interdiction problems by upgading critical edges/nodes have important applications to reduce the infectivity of the COVID-19. A network of confirmed cases can be described as a rooted tree that has a weight of infectious intensity for each edge. Upgrading edges (nodes) can reduce the infectious intensity with contacts by taking prevention measures such as disinfection (treating the confirmed cases, isolating their close contacts or vaccinating the uninfected people). We take the sum of root-leaf distance on a rooted tree as the whole infectious intensity of the tree. Hence, we consider the sum of root-leaf distance interdiction problem by upgrading edges/nodes on trees (SDIPT-UE/N). The problem (SDIPT-UE) aims to minimize the sum of root-leaf distance by reducing the weights of some critical edges such that the upgrade cost under some measurement is upper-bounded by a given value. Different from the problem (SDIPT-UE), the problem (SDIPT-UN) aims to upgrade a set of critical nodes to reduce the weights of the edges adjacent to the nodes. The relevant minimum cost problem (MCSDIPT-UE/N) aims to minimize the upgrade cost on the premise that the sum of root-leaf distance is upper-bounded by a given value. We develop different norms to measure the upgrade cost. Under weighted Hamming distance, we show the problems (SDIPT-UE/N) and (MCSDIPT-UE/N) are NP-hard by showing the equivalence of the two problems and the 0–1 knapsack problem. Under weighted l_1 norm, we solve the problems (SDIPT-UE) and (MCSDIPT-UE) in $O(n)$ time by transforimg them into continuous knapsack problems. We propose two linear time greedy algorithms to solve the problem (SDIPT-UE) under unit Hamming distance and the problem (SDIPT-UN) with unit cost, respectively. Furthermore, for the the minimum cost problem (MCSDIPT-UE) under unit Hamming distance and the problem (MCSDIPT-UN) with unit cost, we provide two $O(n \log n)$ time algorithms by the

Research is supported by National Natural Science Foundation of China (11471073).

✉ Xiucui Guan
xcguan@163.com

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

binary search methods. Finally, we perform some numerical experiments to compare the results obtained by these algorithms.

Keywords Network interdiction problem · Upgrading critical edges · Upgrading critical nodes · Tree · Knapsack problem · Greedy algorithm

1 Introduction

The coronavirus disease-2019 (COVID-19) has been a global pandemic with nearly 230 million confirmed cases and more than 4.6 million deaths since December 2019 (COVID-19 Global Outbreak Live 2021). The pressing task is to control and isolate the sources of infection and treat the infected cases so as to reduce their infectivity. In response to the need of treating a huge number of cases with limited medical and epidemic prevention materials, we propose the infectious intensity interdiction problem by upgrading critical edges/nodes on a transmission network of confirmed cases. Such a network can be regarded as a rooted tree assumed that the confirmed cases will not be infected again (Will COVID-19 2020), just as the transmission tree infected by the "super 31" in South Korea in February 2020 shown in Fig. 1 (Italy, Iran in the Middle East 2020). We study on determining which critical edges/nodes to be upgraded so as to make the infectious intensity of the transmission tree as small as possible. Upgrading critical edges means taking prevention measures such as disinfection while upgrading critical nodes means treating the confirmed cases, isolating their close contacts or vaccinating the uninfected people. In a word, upgrading edges/nodes can reduce the infectious intensity with contacts although it can not completely stop the transmission of viruses. The upgrade cost for upgrading critical edges/nodes may be overall different. However, the cost may be the same in some special cases where we aim to determine the number of upgraded edges /treated nodes in the above problems. The infectious intensity of an infectious disease is related to the type, quantity and virulence of viruses and immune status of susceptible people (Infectious Diseases 2021). We describe the infectious intensity of an edge as a weight $w(e)$.

More generally, networks with one-direction link can be described as tree networks. For example, in the transmission network of confirmed cases, the epidemic can only spread from the confirmed cases to susceptible population. The relevant problems can be described as follows.

Let $T = (V, E, w)$ be an edge-weighed tree rooted at s, where $V = \{s, v_1, v_2, \ldots, v_n\}$ and $E = \{e_1, e_2, \ldots, e_n\}$ are the sets of nodes and edges, respectively. Let $Y = \{t_1, t_2, \ldots, t_r\}$ be the set of leaves. Let $w(e)$ and $l(e)$ be the original weight and the lower bound of the upgrade weight of the edge $e \in E$, respectively, where $w(e) \geq l(e)$. Let $\Delta w(e) = w(e) - l(e)$. Let $c(e)$ be the cost to upgrade the edge e. Denoted by P_{s,t_k} the unique root-leaf path from s to t_k on T. Denote the length of path P_{s,t_k} under the weight w by $d_w(s, t_k) = \sum_{e \in P_{s,t_k}} w(e)$. Define "the sum of root-leaf distance" under the weight w as $d_w(T) = \sum_{t \in Y} d_w(s, t)$. The sum of root-leaf distance interdiction problem by *upgrading edges* on trees, denoted by **(SDIPT-UE)**, aims to find an upgrade scheme $\bar{w}$ to minimize the distance $d_{\bar{w}}(T)$ under $\bar{w}$ on the

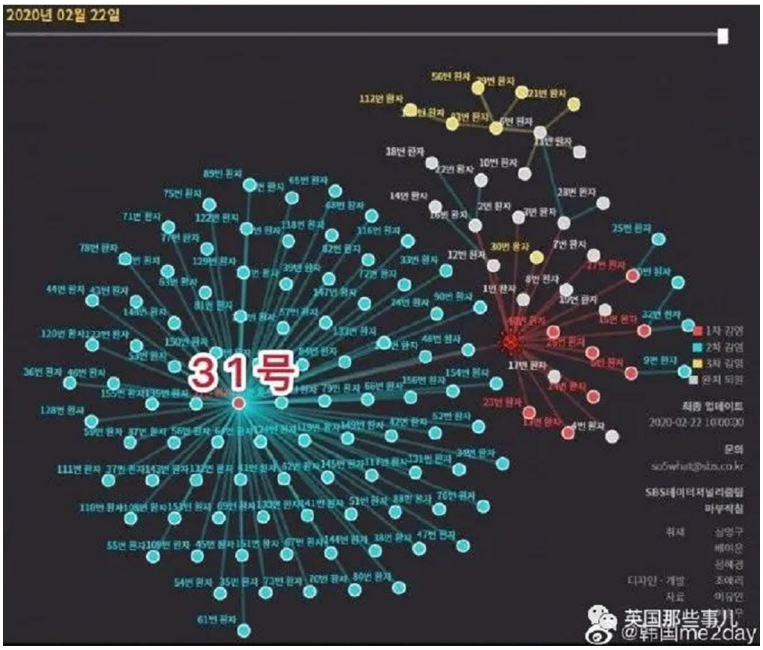

Fig. 1 The transmission tree infected by the "super 31" in South Korea in February 2020 (Italy, Iran in the Middle East 2020)

premise that the total upgrade cost under some norm is upper bounded by a given value K. Its mathematical model can be stated as follows.

$$\textbf{(SDIPT-UE)} \quad \begin{aligned} \min_{\bar{w}} \ & \sum_{t\in Y} d_{\bar{w}}(s,t) \\ s.t. \ & \|\bar{w}-w\| \le K, \\ & l(e) \le \bar{w}(e) \le w(e), e \in E. \end{aligned}$$

The relevant minimum cost problem **(SDIPT-UE)**, denoted by **(MCSDIPT-UE)**, aims to find an upgrade scheme $\bar{w}$ to minimize the total upgrade cost such that the distance $d_{\bar{w}}(T)$ under $\bar{w}$ is upper bounded by a given value D. Its mathematical model can be stated as follows.

$$\textbf{(MCSDIPT-UE)} \quad \begin{aligned} \min_{\bar{w}} \ & \|\bar{w}-w\| \\ s.t. \ & \sum_{t\in Y} d_{\bar{w}}(s,t) \le D, \\ & l(e) \le \bar{w}(e) \le w(e), e \in E. \end{aligned}$$

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 $A(v_i) = \{e_j = (v_i, v_j) | e_j \in E\}$ be the set of edges adjacent to v_i. Let $\beta_1(e_j)$ and $\beta_0(e_j)$ be the weight when the node v_i is upgraded or not, respectively. Let $c(v_i)$ be the upgrade cost of the node v_i . The sum of root-leaf distance interdiction problem by *upgrading nodes* on trees, denoted by **(SDIPT-UN)**,

aims to upgrade a subset $S \subseteq V$ of nodes to minimize the sum of root-leaf distance such that the total upgrade cost under some norm is upper bounded by a given value K. Its mathematical model can be stated as follows.

$$\textbf{(SDIPT-UN)} \quad \begin{array}{ll} \min_{S\subseteq V} & \sum\limits_{t\in Y} d_{\bar{w}}(s,t) \\ s.t. & \sum\limits_{v\in S} c(v) \le K, \\ & \bar{w}(e) = \begin{cases} \beta_1(e), & v \in S, e \in A(v), \\ \beta_0(e), & otherwise. \end{cases} \end{array} \tag{1.1}$$

The relevant minimum cost problem **(SDIPT-UN)**, denoted by **(MCSDIPT-UN)**, aims to upgrade a subset $S \subseteq V$ of nodes to minimize the total upgrade cost such that the sum of root-leaf distance is upper bounded by a given value D. Its mathematical model can be stated as follows.

$$\textbf{(MCSDIPT-UN)} \quad \begin{array}{ll} \min_{S\subseteq V} & \sum\limits_{v\in S} c(v) \\ s.t. & \sum\limits_{t\in Y} d_{\bar{w}}(s,t) \le D. \\ & \bar{w}(e) = \begin{cases} \beta_1(e), & v \in S, e \in A(v), \\ \beta_0(e), & otherwise. \end{cases} \end{array} \tag{1.2}$$

Notice that the upgrade cost for upgrading critical edges/nodes are different in general. However, the cost may be the same in some special cases where we consider the number of the upgrade edges/nodes instead of the upgrade cost in the cost constraint/objective of the above problems.

Most network interdiction problems aim to delete some critical edges/nodes to make some network performance worse. They have wide applications in drug trafficking network (Albert et al. 2000), terrorist network (Ayyldz et al. 2019) and network war (Albert et al. 2000; Khachiyan et al. 2008). Magnouche and Martin (2020) from Huawei Technologies in France studied how to delete the least number of critical nodes so that the length of s–t path in the remained graph was at least d. They analyzed the NP-hardeness of the problem, presented an integer linear programming model with multiple exponential constraints and designed a branch-and-bound algorithm to solve it.

The network interdiction problems by deleting critical edges was first applied to shortest path problem by Corley and Sha (1982), where K edges were deleted to maximize the length of the shortest path of the network. Ball et al. (1989) showed that the problem is NP-hard. Khachiyan et al. (2008) showed that it has no approximation algorithm with ratio 2. Bazgan et al. (2015) provided an $O(mn)$ algorithm for the shortest path interdiction problem when the increment $b = 1$ of the length of the path and they Bazgan et al. (2019) showed that the problem is NP-hard with $b \ge 2$.

In some practical applications, it is extremely difficult to delete edges/nodes in a network and we can only modify the weights of some edges since there are always some emergency or alternative schemes available. Zhang et al. (2021a, b) proposed the maximum shortest path interdiction problem by upgrading edges on trees **(MSPIT)** and its relevant minimum cost problem **(MCSPIT)**, respectively. Under weighted

l_1 norm, they provided two $O(n^2)$ time primal dual algorithms, respectively. Under unit l_1 norm, they designed two linear time algorithms, respectively. Under weighted Hamming distance they Zhang et al. (2021b) showed the problem **(MSPIT)** is NP-hard. Under unit Hamming distance, they proposed an $O(n(\log n + K^3))$ time algorithm by dynamic programming for the problem **(MSPIT)** and an $O(n^4 \log n)$ time algorithm by binary search for the problem **(MCSPIT)**.

In this paper, we also consider the problems **(SDIPT-UE/N)** and **(MCSDIPT-UE/N)** using different norms to measure the upgrade cost. We list our research results in Table 1 compared with results in the previous research, where subscripts 1, H, uH and u denote the weighted l_1 norm, the weighted Hamming distance, the unit Hamming distance and the unit node cost, respectively.

The paper is organized as follows. In Sect. 2, we proved the problems **(SDIPT-UE)** and **(MCSDIPT-UE)** under weighted Hamming distance are NP-hard. In Sects. 3 and 4, we proposed two algorithms in $O(n)$ and $O(n \log n)$ time for the problem **(SDIPT-UE)** and **(MCSDIPT-UE)** under unit Hamming distance , respectively. In Sect. 5, we showed the problems **(SDIPT-UE)** and **(MCSDIPT-UE)** under l_1 norm are equivalent to the continuous knapsack problems, and hence they can be solved in $O(n)$ time. In Sect. 6, we proved the equivalence of the problems **(SDIPT-UN)**, **(MCSDIPT-UN)** and the 0–1 knapsack problems. In Sects. 7 and 8, we developed two algorithms with time complexities $O(n)$ and $O(n \log n)$ for the problems **(SDIPT-UN)** and **(MCSDIPT-UN)** with unit node cost, respectively. In Sect. 9, computational experiments were given to show the effectiveness of all these polynomial time algorithms. In Sect. 10, we drew a conclusion and put forward our future research.

2 The NP-hardness of the problems (SDIPT-UE) and (MCSDIPT-UE) under weighted Hamming distance

The weighted Hamming distance is defiend as

$$\|\bar{w} - w\|_H = \sum_{e \in E} c(e) H\big(\bar{w}(e), w(e)\big), \tag{2.1}$$

$$H\big(\bar{w}(e), w(e)\big) = \begin{cases} 0, & if\ \bar{w}(e) = w(e), \\ 1, & if\ \bar{w}(e) \neq w(e). \end{cases} \tag{2.2}$$

The problems **(SDIPT-UE)** and **(MCSDIPT-UE)** under weighted Hamming distance, denoted by **(SDIPT-UE$_H$)** and **(MCSDIPT-UE$_H$)**, can be formulated as the following models (2.3) and (2.4), respectively.

$$\textbf{(SDIPT-UE}_H\textbf{)} \quad \begin{array}{l} \min_{\bar{w}} \ \sum\limits_{t \in Y} d_{\bar{w}}(s,t) \\ s.t. \ \ \sum\limits_{e \in E} c(e) H\big(\bar{w}(e), w(e)\big) \leq K, \\ \quad\ l(e) \leq \bar{w}(e) \leq w(e), e \in E. \end{array} \tag{2.3}$$

Table 1 The relationship between the previous research and our research

Type	Graph	Problem	$K/b/c$		Complexity	Reference
Deleting nodes	General graph	**Network interdiction problem on shortest path**	any K		NP-hard	Magnouche and Martin (2020)
Deleting edges					NP-hard	Ball et al. (1989)
					Not approximable within ratio 2	Khachiyan et al. (2008)
	Undirected network		$b=1$		$O(mn)$	Bazgan et al. (2015)
			$b\geq 2$		NP-hard	Bazgan et al. (2019)
Upgrading edges	Tree	**MCSPIT**$_1$	any K	$c=1$	$O(n)$	Zhang et al. (2021a)
				any c	$O(n^2)$	
		MSPIT$_1$		$c=1$	$O(n)$	
				any c	$O(n^2)$	
		MSPIT$_H$		any c	NP-hard	Zhang et al. (2021b)
			$K=1$	$c=1$	$O(n+l\log l)$	
			any K	$c=1$	$O(n(\log n+K^3))$	
		MCSPIT$_{uH}$			$O(n^4\log n)$	
		SDIPT-UE$_H$	any K	any c	NP-hard	Section 2
		MCSDIPT-UE$_H$				
		SDIPT-UE$_{uH}$	any K	$c=1$	$O(n)$	Section 3
		MCSDIPT-UE$_{uH}$			$O(n\log n)$	Section 4
		SDIPT-UE$_1$	any K	any c	$O(n)$	Section 5
		MCSDIPT-UE$_1$				
Upgrading nodes		**SDIPT-UN**	any K	any c	NP-hard	Section 6
		MCSDIPT-UN				
		SDIPT-UN$_u$		$c=1$	$O(n)$	Section 7
		MCSDIPT-UN$_u$			$O(n\log n)$	Section 8

Description: MSPIT: MCSPIT: Minimum cost SPIT; SDIPT: The sum of root-leaf distance interdiction problem; MCSDIPT: Minimum cost SDIPT; UE: Upgrading edges; UN: Upgrading nodes; The subscripts $1, H, uH, u$ represent weighted l_1 norm, weighted Hamming distance, unit Hamming distance, unit node cost, respectively.

$$\textbf{(MCSDIPT-UE}_H\textbf{)} \quad \begin{array}{l} \min_{\bar{w}} \sum\limits_{e\in E} c(e)H\big(\bar{w}(e), w(e)\big) \\ s.t. \quad \sum\limits_{t\in Y} d_{\bar{w}}(s,t) \le D, \\ \quad\; l(e) \le \bar{w}(e) \le w(e), e \in E. \end{array} \tag{2.4}$$

In this section, we first prove a property of the optimal solution of the problems **(SDIPT-UE$_H$)** and **(MCSDIPT-UE$_H$)**. Then show their NP-hardness.

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

Theorem 2 *If $\bar{w}$ is an optimal solution of the problem* **(MCSDIPT-UE$_H$)** *or* **(SDIPT-UE$_H$)**, *so is w^* defined below.*

$$w^*(e) = \begin{cases} w(e), & if \ \bar{w}(e) = w(e), \\ l(e), & if \ \bar{w}(e) \neq w(e). \end{cases} \tag{2.5}$$

Proof (1) We first show that Theorem 2 holds for the problem **(SDIPT-UE$_H$)**.

Obviously, w^* is a feasible solution of the problem **(SDIPT-UE$_H$)**, since $l(e) \le w^*(e) \le w(e)$ and $\sum_{e\in E} c(e)H\big(w^*(e), w(e)\big) = \sum_{e\in E} c(e)H\big(\bar{w}(e), w(e)\big) \le K$.

Now we show that w^* is an optimal solution of the problem **(SDIPT-UE$_H$)**. Notice that $l(e) \le w^*(e) \le \bar{w}(e) \le w(e)$ for all edges $e \in E$. Suppose there exists an edge $e_i \in E$ satisfying $w^*(e_i) = l(e_i) < \bar{w}(e_i) < w(e_i)$. Then $\sum_{t\in Y\setminus L(e_i)} d_{w^*}(s,t) \le \sum_{t\in Y\setminus L(e_i)} d_{\bar{w}}(s,t)$ and $\sum_{t\in L(e_i)} d_{w^*}(s,t) < \sum_{t\in L(e_i)} d_{\bar{w}}(s,t)$ follows. Hence,

$$\begin{aligned} \sum_{t\in Y} d_{w^*}(s,t) &= \sum_{t\in Y\setminus L(e_i)} d_{w^*}(s,t) + \sum_{t\in L(e_i)} d_{w^*}(s,t) \\ &< \sum_{t\in Y\setminus L(e_i)} d_{\bar{w}}(s,t) + \sum_{t\in L(e_i)} d_{\bar{w}}(s,t) = \sum_{t\in Y} d_{\bar{w}}(s,t), \end{aligned} \tag{2.6}$$

which contracts that $\bar{w}$ is an optimal solution.

(2) We then show that Theorem 2 holds for the problem **(MCSDIPT-UE$_H$)**. The formula (2.6) also holds for the problem **(MCSDIPT-UE$_H$)** and thus we have $\sum_{t\in Y} d_{w^*}(s,t) < \sum_{t\in Y} d_{\bar{w}}(s,t) \le D$ with $l(e) \le w^*(e) \le w(e)$. Hence, w^* is a feasible solution of the problem **(MCSDIPT-UE$_H$)**. Furthermore, it is obvious that $\sum_{e\in E} c(e)H\big(w^*(e), w(e)\big) = \sum_{e\in E} c(e)H\big(\bar{w}(e), w(e)\big)$ and thus w^* is also an optimal solution of the problem **(MCSDIPT-UE$_H$)**. □

Based on Theorem 2, we pursue an optimal solution w^* defined as in (2.5). If an edge e is upgraded, then the length of each path $P_{s,t_k}(t_k \in L(e))$ will decrease by $\Delta w(e)$. Next we define the *total reduction amount* of an *edge e* to describe the decreasing amount of the sum of root-leaf distance as the weight reduction of the edge e.

Definition 3 For any $e \in E$, let $Q(e) = |L(e)| \cdot \Delta w(e)$ be the *total reduction amount* of the edge e, which is the product of the upgrade amount of the edge e and the number of leaf nodes controlled by the edge e.

Next, we prove NP-hardness of the problems **(SDIPT-UE$_H$)** and **(MCSDIPT-UE$_H$)** by showing the equivalence of the problems and the 0–1 knapsack problem. For convenience, we substitute $H\big(\bar{w}(e), w(e)\big)$ by $x(e)$, where

$$x(e) = \begin{cases} 0, & if\ \ \bar{w}(e) = w(e), \\ 1, & if\ \ \bar{w}(e) \neq w(e). \end{cases} \tag{2.7}$$

Theorem 4 *The problem* **(SDIPT-UE$_H$)** *is NP-hard.*

Proof The objective function can be calculated as follows.

$$\begin{aligned}
\min_{\bar{w}} \sum_{t\in Y} d_{\bar{w}}(s,t) &= \min_{\bar{w}} \sum_{t\in Y} \sum_{e\in P_{s,t}} \bar{w}(e) \\
&= \min_{x} \sum_{t\in Y} \sum_{e\in P_{s,t}} \Big(w(e) - x(e)\cdot \Delta w(e)\Big) \\
&= \min_{x} \left\{ \sum_{t\in Y} \sum_{e\in P_{s,t}} w(e) - \sum_{t\in Y} \sum_{e\in P_{s,t}} \Delta w(e)\cdot x(e) \right\} \\
&= \sum_{t\in Y} \sum_{e\in P_{s,t}} w(e) - \max_{x} \sum_{t\in Y} \sum_{e\in P_{s,t}} \Delta w(e)\cdot x(e) \\
&= \sum_{t\in Y} d_w(s,t) - \max_{x} \sum_{e\in E} \sum_{\substack{e\in P_{s,t} \\ t\in Y}} \Delta w(e)\cdot x(e) \\
&= d_w(T) - \max_{x} \sum_{e\in E} |L(e)| \Delta w(e)\cdot x(e) \\
&= d_w(T) - \max_{x} \sum_{e\in E} Q(e)\cdot x(e).
\end{aligned} \tag{2.8}$$

Hence, the problem (2.3) is equivalent to the following 0–1 knapsack problem.

$$\begin{aligned}
&\max\nolimits_x \sum_{e\in E} Q(e)x(e) \\
&s.t. \sum_{e\in E} c(e)x(e) \le K, \\
&\qquad x(e) = \begin{cases} 0, & if\ \ \bar{w}(e) = w(e), \\ 1, & if\ \ \bar{w}(e) \neq w(e). \end{cases}
\end{aligned} \tag{2.9}$$

The problem **(SDIPT-UE$_H$)** is NP-hard by showing the equivalence of the problem and the 0–1 knapsack problem which is NP-hard (Martello and Toth 1990). □

For the 0–1 knapsack problem, there is a pseudo-polynomial time algorithm with time complexity $O(nK)$ (Martello and Toth 1990) and several approximation

algorithms. In 1975, Ibarra and Kim (1975) proposed a fully polynomial-time approximation scheme (FPTAS) with approximation factor $1+\varepsilon$ in $O(n^3(1+1/\varepsilon))$ time. Very recently in 2019, Jin (2019) provided an improved FPTAS with approximation factor $1+\varepsilon$ in $\tilde{O}(n+(1/\varepsilon)^{9/4})$ time, where $\tilde{O}$ hides polylogarithmic factors.

In a similar way, the problem **(MCSDIPT-UE$_H$)** is also equivalent to a 0–1 minimization knapsack problem.

Theorem 5 *The problem* **(MCSDIPT-UE$_H$)** *is NP-hard.*

Proof As shown in the derivation process (2.8), the constraint condition

$$\sum_{t\in Y} d_{\bar{w}}(s,t) = d_w(T) - \sum_{e\in E} Q(e)\cdot x(e) \le D, \tag{2.10}$$

is equivalent to

$$\sum_{e\in E} Q(e)\cdot x(e) \ge D', \quad D' = d_w(T) - D. \tag{2.11}$$

Then the problem **(MCSDIPT-UE$_H$)** is equivalent to the following problem.

$$\begin{aligned} &\min_x \sum_{e\in E} c(e)x(e) \\ &s.t. \sum_{e\in E} Q(e)\cdot x(e) \ge D', \\ &\quad x(e) = \begin{cases} 0, & if\ \bar{w}(e) = w(e), \\ 1, & if\ \bar{w}(e) \ne w(e). \end{cases} \end{aligned} \tag{2.12}$$

The problem (2.12) is just a 0–1 minimization knapsack problem which is also NP-hard (Martello and Toth 1990). □

3 A linear time algorithm to solve the problem (SDIPT-UE$_H$) under unit Hamming distance

The **(SDIPT-UE$_H$)** problem under unit Hamming distance, denoted by **(SDIPT-UE$_{uH}$)**, can be formulated from the models (2.9) and (2.3) as the following form.

$$\begin{aligned} &\max_{\bar{w}} \sum_{e\in E} Q(e)H\big(\bar{w}(e), w(e)\big) \\ \textbf{(SDIPT-UE}_{uH}\textbf{)}\ \ &s.t. \sum_{e\in E} H\big(\bar{w}(e), w(e)\big) \le K, \\ &\quad l(e) \le \bar{w}(e) \le w(e), e\in E. \end{aligned} \tag{3.1}$$

We can conclude from the model (3.1) that the problem **(SDIPT-UE$_{uH}$)** aims to upgrade K edges to be upgraded so that the sum of the total reduction amount is maximized. Thus we can sort the edges by the values of $Q(e)$ in non-increasing order and upgrade the first K largest $Q(e)$-value edges.

Sort the edges by the values of $Q(e)$ in non-increasing order as follows.

$$Q(e_{i_1}) \geq Q(e_{i_2}) \geq \cdots \geq Q(e_{i_K}) \geq \cdots \geq Q(e_{i_n}).$$

Theorem 6 *Let $\bar{E}_K = \{e_{i_\tau} | \tau = 1, 2, \ldots, K\}$ be the set of the first K largest $Q(e)$-value edges in E. Then $\bar{w}(e) = \begin{cases} l(e), & e \in \bar{E}_K \\ w(e), & e \notin \bar{E}_K \end{cases}$ is an optimal solution of the problem* **(SDIPT-UE$_{uH}$)**.

Proof Suppose $\bar{w}$ is not an optimal solution of the problem **(SDIPT-UE$_{uH}$)**, but $\hat{w}(e) = \begin{cases} l(e), & e \in \hat{E}_K, \\ w(e), & e \notin \hat{E}_K. \end{cases}$ is, where $\hat{E}_K = \{e_{j_\tau} | \tau = 1, 2, \ldots, K\}$ is the set of K edges different from $\bar{E}$. Then, $\sum_{e \in \hat{E}} Q(e) \geq \sum_{e \in \bar{E}} Q(e)$ and $|\hat{E}_K| = |\bar{E}_K| = K$. If $\sum_{e \in \hat{E}_K} Q(e) = \sum_{e \in \bar{E}_K} Q(e)$, then $\hat{E}_K$ is also the first K largest $Q(e)$-value edges and the theorem holds. If $\sum_{e \in \hat{E}_K} Q(e) > \sum_{e \in \bar{E}_K} Q(e)$, then it contradicts that $\bar{E}_K$ is the first K largest $Q(e)$-value edges and $\bar{w}$ is an optimal solution of the problem **(SDIPT-UE$_{uH}$)**. □

Next we present a linear time Algorithm 1 to search for the first K largest values in an array with duplicate elements. We first find the K-th largest element q of an array Q by the selection algorithm $q := Selection(Q, K)$ in Thoms et al. (2009, pp 220–222). Different from the partition algorithm in Thoms et al. (2009, pp 170–173), we may have elements with equal values in the array Q. To ensure we find the exact K elements, we determine the sets E_1 and E_2 of edges whose value is larger than and equal to q, respectively. Finally, $\bar{E}_K := E_1 \cup E_2(1 : K - |E_1|)$ is the set of the first K largest $Q(e)$-value edges in the array Q. The above two steps can both be completed in $O(n)$ time.

Algorithm 1 A greedy algorithm to solve the problem **(SDIPT-UE$_{uH}$)**.

Require: A tree $T(V, E)$ rooted at s, the number n of edges, the set Y of leaf nodes, two edge weight vectors w and l and the number K of upgrade edges.

Ensure: The set $\bar{E}_K$ of upgrade edges, the optimal solution $\bar{w}$ and the relevant value $d_{\bar{w}}(T)$.

1: **if** $K > n$ **then**
2: **return** "The problem has no feasible solution!"
3: **end if**
4: For any $e \in E$, calculate the set $L(e)$ of leaf nodes controlled by the edge e, compute $\Delta w(e) := w(e) - l(e)$ and $Q(e) := |L(e)| \cdot \Delta w(e)$.
5: Call $q := Selection(Q, K)$.
6: Let $E_1 := \{e_i | Q(e_i) > q\}$, and $E_2 =: \{e_i | Q(e_i) = q\}$ Then $\bar{E}_K := E_1 \cup E_2(1 : K - |E_1|)$.
7: The optimal solution is $\bar{w}(e) := \begin{cases} l(e), & e \in \bar{E}_K, \\ w(e), & e \notin \bar{E}_K. \end{cases}$ and the objective value is $d_{\bar{w}}(T) := d_w(T) - \sum_{e \in \bar{E}_K} Q(e)$.

Theorem 7 *Algorithm 1 can solve the problem* **(SDIPT-UE$_{uH}$)** *in $O(n)$ time.*

4 An $O(n \log n)$ time algorithm to solve the problem (MCSDIPT-UE$_H$) under unit Hamming distance

We consider the problem (**MCSDIPT-UE**$_{uH}$) under unit Hamming distance, which can be formulated from the models (2.12) and (2.4) as follows.

$$(\textbf{MCSDIPT-UE}_{uH}) \quad \begin{array}{l} \min_{\bar{w}} \sum\limits_{e\in E} H\big(\bar{w}(e), w(e)\big) \\ s.t. \quad \sum\limits_{e\in E} Q(e) H\big(\bar{w}(e), w(e)\big) \geq D', \\ \quad\quad l(e) \leq \bar{w}(e) \leq w(e), e \in E. \end{array} \tag{4.1}$$

The problem (**MCSDIPT-UE**$_{uH}$) aims to upgrade the least number of edges such that the sum of the total reduction amount is no less than D'. Obviously, we can first sort the edges by the $Q(e)$-values in non-increasing order and find the minimum number of edges to be upgraded by a binary search such that the sum of the total reduction amount achieves the lower bound D'.

Theorem 8 *Let $\bar{E}_k = \{e_{i_\tau} | \tau = 1, \ldots, k\}$ be the set with minimum number of edges satisfying $\sum_{\tau=1}^{k} Q(e_{i_\tau}) \geq D'$. Then $\tilde{w}(e) = \begin{cases} l(e), & e \in \tilde{E}_k, \\ w(e), & e \notin \tilde{E}_k. \end{cases}$ is an optimal solution of* (**MCSDIPT-UE**$_{uH}$) *for any $\tilde{E}_k = \{e_{\alpha_\tau} | \tau = 1, \ldots, k\}$ with $\sum_{\tau=1}^{k} Q(e_{\alpha_\tau}) \geq D'$.*

Proof Suppose $\tilde{w}$ is not an optimal solution of the problem (**MCSDIPT-UE**$_{uH}$), but $\hat{w}(e) = \begin{cases} l(e), & e \in \hat{E}_{k'}, \\ w(e), & e \notin \hat{E}_{k'}. \end{cases}$ is, where $\hat{E}_{k'} = \{e_{j_\tau} | \tau = 1, 2, \ldots, k'\}$ with $\sum_{\tau=1}^{k'} Q(e_{j_\tau}) \geq D'$. Then we have $k' < k$, which contradicts that $\bar{E}_k$ is the set with minimum number of edges satisfying $\sum_{\tau=1}^{k} Q(e_{i_\tau}) \geq D'$. Hence, $\tilde{w}$ is an optimal solution. □

To find the set $\bar{E}_k = \{e_{i_\tau} | \tau = 1, 2, \ldots, k\}$ with the minimum number of edges satisfying $\sum_{\tau=1}^{k} Q(e_{i_\tau}) \geq D'$, we perform a method in two steps. In the first step, we sort the edges $e_{i_1}, \ldots, e_{i_n}$ by the values of $Q(e)$ non-increasingly. In the second step, we run a binary search algorithm to determine the minimum number k^* satisfying $\sum_{\tau=1}^{k^*} Q(e_{i_\tau}) \geq D'$ and $\sum_{\tau=1}^{k^*-1} Q(e_{i_\tau}) < D'$. Finally, we upgrade the first k^* largest $Q(e)$-value edges, which is just the set $\bar{E}_{k^*}$.

Theorem 9 *The problem* (**MCSDIPT-UE**$_{uH}$) *can be solved in $O(n \log n)$ time by Algorithm 2.*

Proof The calculation process in Line 1 can be completed in $O(n)$ time. Sorting edges by the values of $Q(e)$ spends $O(n \log n)$ time in Line 2. It takes $O(\log n)$ iterations to determine k^* by the binary search in Line 4–13 and in each iteration the time complexity is $O(n)$. Hence the problem (**MCSDIPT-UE**$_{uH}$) can be solved in $O(n \log n)$ time by Algorithm 2. □

Algorithm 2 A greedy algorithm to solve the problem (**MCSDIPT-UE**$_{uH}$).

Require: A tree $T(V, E)$ rooted at s, two edge weight vectors w and l and the value D.
Ensure: The set $\bar{E}_{k^*}$ of upgrade edges, the optimal solution $\bar{w}$ and the objective value k^*.
1: Calculate the set $L(e)$ of leaf nodes controlled by the edge $e \in E$. Calculate $\Delta w(e) := w(e) - l(e)$ and $Q(e) := |L(e)| \cdot \Delta w(e)$ for any $e \in E$. Calculate $D' := d_w(T) - D$.
2: **if** $D' > \sum_{e\in E} Q(e)$ **then**
3: **return** "The problem has no feasible solution!"
4: **end if**
5: Sort and relabel the edges in non-increasing order by their values of $Q(e)$ as follows.

$$Q(e_{i_1}) \geq Q(e_{i_2}) \geq \cdots \geq Q(e_{i_n}).$$

6: Initialization: $a := 1, b := n, k := \lfloor \frac{a+b}{2} \rfloor$, OPT='NO'.
7: **while** OPT='NO' **do**
8: **if** $\sum_{j=1}^{k} Q(e_{i_j}) < D'$ **then**
9: update $a := k$.
10: **else if** $\sum_{j=1}^{k-1} Q(e_{i_j}) < D'$ **then**
11: OPT='YES'.
12: **return** $k^* := k$.
13: **else**
14: update $b := k$.
15: **end if**
16: update $k := \lfloor \frac{a+b}{2} \rfloor$.
17: **end while**
18: The optimal solution is $\bar{w}(e) := \begin{cases} l(e), & e \in \bar{E}_{k^*}, \\ w(e), & e \notin \bar{E}_{k^*}. \end{cases}$ where $\bar{E}_{k^*} := \{e_{i_\tau} | \tau = 1, 2, \ldots, k^*\}$.

5 Solve the problems (SDIPT-UE) and (MCSDIPT-UE) under weighted l_1 norm

When the weighted l_1 norm is applied to the upgrade cost, the problems (**SDIPT-UE**) and (**MCSDIPT-UE**) under weighted l_1 norm, denoted by (**SDIPT-UE**$_1$) and (**MCSDIPT-UE**$_1$), can be formulated as the following models (5.1) and (5.2), respectively.

$$\textbf{(SDIPT-UE}_1\textbf{)} \quad \begin{array}{l} \min_{\bar{w}} \sum_{t\in Y} d_{\bar{w}}(s,t) \\ s.t. \quad \sum_{e\in E} c(e)|\bar{w}(e) - w(e)| \leq K, \\ l(e) \leq \bar{w}(e) \leq w(e), e \in E. \end{array} \tag{5.1}$$

$$\textbf{(MCSDIPT-UE}_1\textbf{)} \quad \begin{array}{l} \min_{\bar{w}} \sum_{e\in E} c(e)|\bar{w}(e) - w(e)| \\ s.t. \quad \sum_{t\in Y} d_{\bar{w}}(s,t) \leq D, \\ l(e) \leq \bar{w}(e) \leq w(e), e \in E. \end{array} \tag{5.2}$$

Next we will transform the models (5.1) and (5.2) into the continuous knapsack problem, respectively, so that the problems (**SDIPT-UE**$_1$) and (**MCSDIPT-UE**$_1$) can

be solved in $O(n)$ time (Martello and Toth 1990). For convenience, let $|\bar{w}(e)-w(e)| = w(e) - \bar{w}(e) = \Delta w(e)x(e), 0 \le x(e) \le 1$.

Theorem 10 *The problem* **(SDIPT-UE$_1$)** *can be transformed into a continuous knapsack problem.*

Proof Similar to the derivation process in (2.8), the objective function of the problem **(SDIPT-UE$_1$)** can be transformed into

$$\min_{\bar{w}} \sum_{t\in Y} d_{\bar{w}}(s,t) = d_w(T) - \max_x \sum_{e\in E} Q(e)x(e).$$

Then the model (5.1) is equivalent to the following problem,

$$\begin{aligned} &\max_x \sum_{e\in E} Q(e)\cdot x(e) \\ &s.t. \sum_{e\in E} c(e)\Delta w(e)\cdot x(e) \le K, \\ &0 \le x(e) \le 1. \end{aligned}$$

which is just a continuous knapsack problem. □

Similarly, the problem **(MCSDIPT-UE$_1$)** can also be transformed into a continuous knapsack problem.

Theorem 11 *The problem* **(MCSDIPT-UE$_1$)** *can be transformed into a continuous knapsack problem.*

Proof Let $y(e) = 1 - x(e)$ and then $x(e) = 1 - y(e), 0 \le y(e) \le 1$. Hence, the objective function

$$\begin{aligned} \min_{\bar{w}} \sum_{e\in E} c(e)|\bar{w}(e) - w(e)| &= \min_x \sum_{e\in E} c(e)\Delta w(e)x(e) \\ &= \min_y \sum_{e\in E} c(e)\Delta w(e)\Big(1 - y(e)\Big) \\ &= \min_y \left\{ \sum_{e\in E} c(e)\Delta w(e) - \sum_{e\in E} c(e)\Delta w(e)y(e) \right\} \\ &= \sum_{e\in E} c(e)\Delta w(e) - \max_y \sum_{e\in E} c(e)\Delta w(e)y(e). \end{aligned}$$

Moreover, similar to the formulas (2.10)–(2.11), the constraint in model (5.2) can be transformed into $\sum_{e\in E} Q(e)x(e) \ge d_w(T) - D$. By substituting $x(e) = 1 - y(e)$ into the constraint, we have

$$\sum_{e\in E} Q(e)x(e) = \sum_{e\in E} Q(e)\Big(1 - y(e)\Big) = \sum_{e\in E} Q(e) - \sum_{e\in E} Q(e)y(e) \ge d_w(T) - D,$$

which is equivalent to

$$\sum_{e\in E} Q(e)y(e) \le D'', \quad D'' = \sum_{e\in E} Q(e) - d_w(T) + D.$$

Thus, the model (5.2) is equivalent to the following problem.

$$\begin{aligned} \max_y \ & \sum_{e\in E} c(e)\Delta w(e)y(e) \\ s.t. \ & \sum_{e\in E} Q(e)y(e) \le D'', \\ & 0 \le y(e) \le 1. \end{aligned} \tag{5.3}$$

The problem (5.3) is also a continuous knapsack problem. □

Corollary 12 *The problems* **(SDIPT-UE$_1$)** *and* **(MCSDIPT-UE$_1$)** *can both be solved in $O(n)$ time.*

6 The NP-hardness of the problems (SDIPT-UN) and (MCSDIPT-UN)

In this section, we prove that the problem **(SDIPT-UN)** is NP-hard by transforming it into a 0–1 knapsack problem, so is the problem **(MCSDIPT-UN)**.

If a node v is upgraded, then the lengths of the paths $P_{s,t_k}(t_k \in L(e), e \in A(v))$ decrease by $\Delta w(e) = \beta_0(e) - \beta_1(e)$. For convenience, relevant to Definition 3, we introduce the following definition of the *total reduction amount* of a *node* v, which describes the decreasing amount of the sum of root-leaf distance as the node v is upgraded.

Definition 13 For any $v \in V$, let $B(v) = \sum_{e\in A(v)} Q(e)$ be the *total reduction amount* of the *node* v, which is the sum of the total reduction amount of the edges adjacent to the node v.

Theorem 14 *The problem* **(SDIPT-UN)** *is NP-hard.*

Proof Let S be the set of upgraded nodes and define $z(v) = \begin{cases} 1, \ if \ v \in S, \\ 0, \ if \ v \notin S. \end{cases}$ and then the objective function can be calculated as follows.

$$\begin{aligned} \min_{S\subseteq V} \sum_{t\in Y} d_{\bar{w}}(s,t) &= \min_{S\subseteq V} \sum_{t\in Y} \sum_{e\in P_{s,t}} \bar{w}(e) \\ &= \min_{S\subseteq V} \sum_{t\in Y} \left(\sum_{v\in S\cap P_{s,t}} \sum_{e\in A(v)\cap P_{s,t}} \beta_1(e) + \sum_{v\in P_{s,t}\setminus S} \sum_{e\in A(v)\cap P_{s,t}} \beta_0(e) \right) \\ &= \min_{z} \sum_{t\in Y} \sum_{v\in P_{s,t}} \sum_{e\in A(v)\cap P_{s,t}} \Big(z(v)\beta_1(e) + \big(1 - z(v)\big)\beta_0(e) \Big) \end{aligned}$$

$$
\begin{aligned}
&= \min_z \sum_{t\in Y}\sum_{v\in P_{s,t}}\sum_{e\in A(v)\cap P_{s,t}}\Big(\beta_0(e) - z(v)\Big(\beta_0(e)-\beta_1(e)\Big)\Big)\\
&= \sum_{t\in Y}\sum_{v\in P_{s,t}}\sum_{e\in A(v)\cap P_{s,t}}\beta_0(e) - \max_z \sum_{t\in Y}\sum_{v\in P_{s,t}}\sum_{e\in A(v)\cap P_{s,t}} z(v)\Delta w(e)\\
&= \sum_{t\in Y} d_{\beta_0}(s,t) - \max_z \sum_{v\in V}\sum_{e\in A(v)}\sum_{\substack{e\in P_{s,t}\\ t\in Y}}\Delta w(e)z(v)\\
&= d_{\beta_0}(T) - \max_z \sum_{v\in V}\sum_{e\in A(v)}|L(e)|\Delta w(e)\cdot z(v)\\
&= d_{\beta_0}(T) - \max_z \sum_{v\in V}\sum_{e\in A(v)} Q(e)z(v)\\
&= d_{\beta_0}(T) - \max_z \sum_{v\in V} B(v)z(v) \qquad (6.1)
\end{aligned}
$$

Hence, the problem (1.1) is equivalent to the following problem.

$$
\begin{aligned}
&\max_z \sum_{v\in V} B(v)z(v)\\
&s.t.\ \sum_{v\in V} c(v)z(v) \le K,\\
&z(v) = \begin{cases} 1, & \text{if } v \text{ is upgraded},\\ 0, & \text{if } v \text{ is not upgraded}.\end{cases}
\end{aligned} \qquad (6.2)
$$

The problem (6.2) is just a 0–1 knapsack problem which is NP-hard (Martello and Toth 1990). □

In a similar way, the problem **(MCSDIPT-UN)** can also be proved to be equivalent to a 0–1 minimization knapsack problem.

Theorem 15 *The problem* **(MCSDIPT-UN)** *is NP-hard.*

Proof Similar to the derivation process in the formula (6.1), the constraint in model (1.2) can be similarly transformed into $d_{\beta_0}(T) - \sum_{v\in V} B(v)z(v) \le D$ and then we have $\sum_{v\in V} B(v)z(v) \ge D^0$, where $D^0 = d_{\beta_0}(T) - D$.

Hence, the problem (1.2) is equivalent to the following problem.

$$
\begin{aligned}
&\min_z \sum_{v\in V} c(v)z(v)\\
&s.t.\ \sum_{v\in V} B(v)z(v) \ge D^0,\\
&z(v) = \begin{cases} 1, & \text{if } v \text{ is upgraded},\\ 0, & \text{if } v \text{ is not upgraded}.\end{cases}
\end{aligned} \qquad (6.3)
$$

The problem (6.3) is a 0–1 minimization knapsack problem which is NP-hard (Martello and Toth 1990). □

7 An $O(n)$ time algorithm to solve problem (SDIPT-UN) with unit cost

The problem **(SDIPT-UN)** with unit cost, denoted by **(SDIPT-UN$_u$)**, can be formulated from (6.2) as the following form.

$$\begin{aligned} &\max_z \sum_{v\in V} B(v)z(v) \\ \textbf{(SDIPT-UN}_u\textbf{)}\quad &s.t.\ \sum_{v\in V} z(v) \le K, \\ &z(v) = \begin{cases} 1, & \text{if } v \text{ is upgraded}, \\ 0, & \text{if } v \text{ is not upgraded}. \end{cases} \end{aligned} \tag{7.1}$$

It is shown in model (7.1) that the problem **(SDIPT-UN$_u$)** aims to upgrade K nodes to maximize their relevant sum of $B(v)$-value. Thus we can sort the nodes by the values of $B(v)$ in non-increasing order and upgrade the first K largest $B(v)$-value nodes.

Sort the nodes by the values of $B(v)$ in non-increasing order as follows.

$$B(v_{i_1}) \ge B(v_{i_2}) \ge \cdots \ge B(v_{i_K}) \ge \cdots \ge B(v_{i_{n+1}}).$$

Theorem 16 *Let $\bar{V}_K = \{v_{i_\tau} | \tau = 1, 2, \ldots, K\}$ be the set of the first K largest $B(v)$-value nodes in V. Then $\bar{w}(e) = \begin{cases} \beta_1(e), & e \in A(v), v \in \bar{V}_K, \\ \beta_0(e), & \text{otherwise}. \end{cases}$ is an optimal solution of the problem* **(SDIPT-UN$_u$)**.

Proof Suppose $\bar{w}$ is not an optimal solution of the problem **(SDIPT-UN$_u$)**, but $\hat{w}(e) = \begin{cases} \beta_1(e), & e \in A(v), v \in \hat{V}_K, \\ \beta_0(e), & \text{otherwise}. \end{cases}$ is, where $\hat{V}_K = \{v_{j_\tau} | \tau = 1, 2, \ldots, K\}$ is the set of K nodes different from $\bar{V}_K$. Then we have $\sum_{v\in\hat{V}_K} B(v) \ge \sum_{v\in\bar{V}_K} B(v)$ and $|\hat{V}_K| = |\bar{V}_K| = K$. If $\sum_{v\in\hat{V}_K} B(v) = \sum_{v\in\bar{V}_K} B(v)$, then $\hat{V}$ is also the set of the first K largest $B(v)$-value nodes and the theorem holds. If $\sum_{v\in\hat{V}_K} B(v) > \sum_{v\in\bar{V}_K} B(v)$, then it contradicts that $\bar{V}_K$ is the first K largest $B(v)$-value nodes. Hence, $\bar{w}$ is an optimal solution of the problem **(SDIPT-UN$_u$)**. □

We can solve the problem **(SDIPT-UN$_u$)** similar to the problem **(SDIPT-UE$_{uH}$)**. We first find the K-th largest element q of an array B by the selection algorithm (Thoms et al. 2009, pp 220–222) and then determine the sets V_1 and V_2 of nodes whose value is larger than and equal to q, respectively. Finally, we can obtain the set $S_K := V_1 \cup V_2(1 : K - |V_1|)$ of the first K largest $B(v)$-value edges in the array B. The above steps can both be completed in $O(n)$ time.

Theorem 17 *Algorithm 3 can solve the problem* **(SDIPT-UN$_u$)** *in $O(n)$ time.*

Algorithm 3 A greedy algorithm to solve the problem (**SDIPT-UN**$_u$).

Require: A tree $T(V, E)$ rooted at s, the number n of edges, the set Y of leaf nodes, two edge weight vectors β_0, β_1 and the number K of upgrade nodes.
Ensure: The set S_K of upgrade nodes, an optimal solution $\bar{w}$ and the relevant value $d_{\bar{w}}(T)$.
1: **if** $K > n + 1$ **then**
2: **return** "The problem has no feasible solution!"
3: **end if**
4: Calculate the set $L(e)$ of leaf nodes controlled by the edge $e \in E$ and the set $A(v)$ of adjacent edges of the node $v \in V$. Calculate $\Delta w(e) := \beta_0(e) - \beta_1(e)$ and $Q(e) := |L(e)| \cdot \Delta w(e)$ for any $e \in E$. Calculate $B(v) := \sum_{e \in A(v)} Q(e)$ for each node $v \in V$.
5: Call $q := Selection(B, K)$.
6: Let $V_1 := \{v_i | B(v_i) > q\}$, and $V_2 := \{v_i | B(v_i) = q\}$. Then $S_K := V_1 \cup V_2(1 : K - |V_1|)$.
7: The optimal solution is $\bar{w}(e) := \begin{cases} \beta_1(e), & e \in A(v), v \in S_K, \\ \beta_0(e), & otherwise. \end{cases}$ and the relevant value is $d_{\bar{w}}(T) := d_{\beta_0}(T) - \sum_{v \in S} B(v)$.

8 An $O(n \log n)$ time algorithm to solve the problem (MCSDIPT-UN) with unit cost

The problem (**MCSDIPT-UN**) with unit cost, denoted by (**MCSDIPT-UN**$_u$), can be formulated from (6.3) as follows.

$$\text{(MCSDIPT-UN}_u\text{)} \quad \begin{array}{l} \min_z \sum\limits_{v \in V} z(v) \\ s.t. \sum\limits_{v \in V} B(v) z(v) \geq D^0, \\ z(v) = \begin{cases} 1, & \text{if } v \text{ is upgraded}, \\ 0, & \text{if } v \text{ is not upgraded}. \end{cases} \end{array} \tag{8.1}$$

where $D^0 = d_{\beta_0}(T) - D$.

As shown in model (8.1), the problem (**MCSDIPT-UN**$_u$) aims to upgrade the least number of nodes such that the sum of the total reduction amount is no less than D^0. Obviously, we can first sort the nodes by the values of $B(v)$ non-increasingly and find the minimum number of nodes to be upgraded by a binary search such that the sum of the total reduction amount satisfies the constraint.

Theorem 18 *Let* $\bar{V}_k = \{v_{i_\tau} | \tau = 1, \ldots, k\}$ *be the set with minimum number of nodes satisfying* $\sum_{\tau=1}^{k} B(v_{i_\tau}) \geq D^0$. *Then for any* $\tilde{V}_k = \{v_{\alpha_\tau} | \tau = 1, \ldots, k\}$ *with* $\sum_{\tau=1}^{k} B(v_{\alpha_\tau}) \geq D^0$, $\tilde{w}(e) = \begin{cases} \beta_1(e), & e \in A(v), v \in \tilde{V}_k, \\ \beta_0(e), & otherwise. \end{cases}$ *is an optimal solution of* (**MCSDIPT-UN**$_u$).

Proof Suppose $\tilde{w}$ is not an optimal solution of the problem (**MCSDIPT-UN**$_u$), but $\hat{w}(e) = \begin{cases} \beta_1(e), & e \in A(v), v \in \hat{V}_{k'}, \\ \beta_0(e), & otherwise. \end{cases}$ is, where $\hat{V}_{k'} = \{v_{j_\tau} | \tau = 1, 2, \ldots, k'\}$ with $\sum_{\tau=1}^{k'} B(v_{j_\tau}) \geq D^0$. Then we have $k' < k$, it contracts that $\bar{V}_k$ is the set with minimum number of nodes satisfying $\sum_{\tau=1}^{k} B(v_{i_\tau}) \geq D^0$. Hence, $\tilde{w}$ is an optimal solution of the problem (**MCSDIPT-UN**$_u$). □

We can perform a greedy algorithm similar to Algorithm 2 to solve the problem (**MCSDIPT-UN**$_u$). Similar to Theorem 9, we can conclude that

Algorithm 4 A greedy algorithm to solve the problem (**MCSDIPT-UN**$_u$).

Require: A tree $T(V, E)$ rooted at s, the set Y of leaf nodes, two edge weight vectors β_0 and β_1 and the number K of upgrade nodes.

Ensure: The set S_{k^*} of upgrade nodes, the optimal solution $\bar{w}$ and the relevant value k^*.

1: Calculate the set $L(e)$ of leaf nodes controlled by the edge $e \in E$ and the set $A(v)$ of adjacent edges of the node $v \in V$. Calculate $\Delta w(e) := \beta_0(e) - \beta_1(e)$ and $Q(e) := |L(e)| \cdot \Delta w(e)$ for any $e \in E$. Calculate $B(v) := \sum_{e \in A(v)} Q(e)$, $v \in V$. Calculate $D^0 = d_{\beta_0}(T) - D$.

2: **if** $D^0 > \sum_{v \in V} B(v)$ **then**

3: **return** "The problem has no feasible solution!"

4: **end if**

5: Sort and relabel nodes in non-increasing order by their values of $B(v)$ as follows.

$$B(v_{i_1}) \geq B(v_{i_2}) \geq \cdots \geq B(v_{i_{n+1}}).$$

6: Initialization: $a := 1$, $b := m$, $k := \lfloor \frac{a+b}{2} \rfloor$, OPT='NO'.

7: **while** OPT='NO' **do**

8: **if** $\sum_{j=1}^{k} B(v_{i_j}) < D^0$ **then**

9: update $a := k$.

10: **else if** $\sum_{j=1}^{k-1} B(v_{i_j}) < D^0$ **then**

11: OPT='YES'.

12: **return** $k^* := k$.

13: **else**

14: update $b := k$.

15: **end if**

16: update $k := \lfloor \frac{a+b}{2} \rfloor$.

17: **end while**

18: The optimal solution is $\bar{w}(e) := \begin{cases} \beta_1(e), & e \in A(v), v \in S_{k^*}, \\ \beta_0(e), & otherwise. \end{cases}$ where $S_{k^*} := \{v_{i_\tau} | \tau = 1, \ldots, k^*\}$.

Corollary 19 *The problem* (**MCSDIPT-UN**$_u$) *can be solved in* $O(n \log n)$ *time by Algorithm 4.*

9 Computational experiments

Now we present computational experiments of Algorithms 1, 2, 3 and 4 in Table 2. The programs were coded in Matlab 7.0 and run on a PC Intel(R), Core(TM)i7-10750H CPU @ 2.60 GHz 2.59 GHz under Windows 10. We have tested the algorithms on 6 classes of random trees with the number n of vertices varying from 1000 to 100,000. For each class, we randomly generated 500 instances on randomly generated trees. We randomly generated two vectors w, l satisfying $l < w$ in Algorithms 1, 2 and two vectors β_0, β_1 satisfying $\beta_1 < \beta_0$ in Algorithms 3, 4, respectively. For each randomly generated tree, we solved the four problems (**SPIT-UE**$_{uH}$),(**MCSPIT-UE**$_{uH}$),(**SPIT-UN**$_u$) and (**MCSPIT-UN**$_u$) for comparison, respectively. Let T_1, T_2, T_3, T_4 be the

Table 2 Performances of Algorithms 1, 2, 3 and 4

Complexity	n	1000	5000	10,000	30,000	50,000	100,000
$O(n)$	T_1	**0.0017**	**0.0088**	**0.0173**	**0.0508**	**0.0925**	**0.1771**
	T_1^{max}	0.0470	0.0240	0.0320	0.0700	0.1160	0.2360
	T_1^{min}	0	0	0.0020	0.0360	0.0780	0.1540
$O(n \log n)$	T_2	**0.0002**	**0.0012**	**0.0018**	**0.0067**	**0.0116**	**0.0234**
	T_2^{max}	0.0150	0.0160	0.0160	0.0230	0.0270	0.0380
	T_2^{min}	0	0	0	0	0	0.0100
$O(n)$	T_3	**0.0011**	**0.0079**	**0.0165**	**0.0523**	**0.0915**	**0.1845**
	T_3^{max}	0.0170	0.0230	0.0320	0.0690	0.1100	0.2240
	T_3^{min}	0	0	0	0.0380	0.0620	0.1250
$O(n \log n)$	T_4	**0.0002**	**0.0013**	**0.0028**	**0.0115**	**0.0205**	**0.0435**
	T_4^{max}	0.0160	0.0160	0.0160	0.0250	0.0350	0.0560
	T_4^{min}	0	0	0	0	0.0100	0.0280

average CPU time of Algorithms 1, 2, 3 and 4, respectively. The relevant maximum and minimum running time, denoted by T_i^{max}, $T_i^{min}(i = 1, 2, 3, 4)$, respectively, are recorded as well.

As shown in Table 2, the four algorithms are all very efficient and they follow their own time complexities very well. Notice that T_3, T_4 are relatively slower than T_1, T_2 as the calculation of $B(v)$ in Algorithm 3, 4 is a bit more complicated than that of $Q(e)$ in Algorithm 1, 2.

Furthermore, we can take the ratios between T_3 and T_4 into consideration. In Table 2, we can calculate the ratios $\frac{T_3}{T_4} = \{5.5, 6.0, 5.8, 4.5, 4.4, 4.2\}$ for the 6 classes of random trees. Notice the ratios are in a decreasing trend, which is consistent with their time complexities $O(n)$ and $O(n \log n)$. It can be predicted that the ratio between T_3 and T_4 may be smaller as n becomes larger, and finally, T_3 will be faster than T_4. Similarly, T_1 will also be faster than T_2 when n is large enough.

10 Conclusion and further research

We considered a class of the sum of root-leaf distance interdiction problems by *upgrading edges/nodes* on trees including **(SDIPT-UE/N)** and their minimum cost problem **(MCSDIPT-UE/N)**. We considered total 10 problems by using different norms to measure the upgrade cost and listed the research results in Table 1 for the sake of convenience in comparing the results. We proved the problems **(SDIPT-UE$_H$)** and **(MCSDIPT-UE$_H$)** under weighted Hamming distance, and the problems **(SDIPT-UN)**, **(MCSDIPT-UN)** with general node cost are NP-hard by showing their equivalence to 0–1 knapsack problems. However, under unit Hamming distance or with unit node cost, the problems **(SDIPT-UE$_{uH}$)** and **(SDIPT-UN$_u$)** can be solved in linear time based on the selection algorithm, while the minimum cost problems **(MCSDIPT-UE$_{uH}$)** and **(MCSDIPT-UN$_u$)** can be solved in $O(n \log n)$ time by a

binary search method. Additionally, the problems (**SDIPT-UE$_1$**) and (**MCSDIPT-UE$_1$**) under weighted l_1 norm were transformed into continuous knapsack problems which render two $O(n)$ time algorithms. The efficiency of the four polynomial time algorithms were tested by some numerical experiments.

For further research, we can consider the sum of root-leaf distance interdiction problem on a series-parallel graph or even on a general graph. Moreover,the interdiction problems under other network performance can be studied, such as the shortest path interdiction problem on a general graph or minimum spanning tree interdiction problems by upgrading edges/nodes.

References

Albert R, Jeong H, Barabasi A (2000) Error and attack tolerance of complex networks. Nature 406(6794):378–382

Ayyldz E, Zelik G, Gencer CT (2019) Determining the most vital arcs on the shortest path for fire trucks in terrorist actions that will cause fire. Commun Fac Sci Univ Ankara Ser A1 Math Stat 68(1):441–450

Ball MO, Golden BL, Vohra RV (1989) Finding the most vital arcs in a network. Oper Res Lett 8(2):73–76

Bazgan C, Nichterlein A et al (2015) A refined complexity analysis of finding the most vital edges for undirected shortest paths. In: Algorithms and complexity: lecture notes in computer science, vol 9079, pp 47–60

Bazgan C, Fluschnik T, Nichterlein A, Niedermeier R, Stahlberg M (2019) A more fine-grained complexity analysis of finding the most vital edges for undirected shortest paths. Networks 73(1):23–37

Corley HW, Sha DY (1982) Most vital links and nodes in weighted networks. Oper Res Lett 1:157–161

COVID-19 Global Outbreak Live (2021) Phoenix News in China. https://news.ifeng.com/c/special/7uLj4F83Cqm. Accessed 16 Sept 2021

Ibarra OH, Kim CE (1975) Fast approximation algorithms for the knapsack and sum of subset problems. J Assoc Comput Mach 22(4):463–468

Infectious Diseases, Characteristics of Infectious Diseases: Infectivity. Encyclopedia 360 in China. https://baike.so.com/doc/5378322-5614504.html. Accessed 3 Apr 2021

Italy, Iran in the Middle East, Korea spiralling out of control. Asymptomless people with poison are simply impossible to guard against! WeChat in China: things like the UK. https://mp.weixin.qq.com/s/ehvC-uQw3J71jH-h6lMjSQ. Accessed 23 Feb 2020

Jin C (2019) An improved FPTAS for 0-1 knapsack. In: Leibniz international proceedings in informatics, LIPIcs vol 132, no 76. https://doi.org/10.4230/LIPIcs.ICALP.2019.76

Khachiyan L, Boros E, Borys K, Elbassioni K, Gurvich V, Rudolf G, Zhao J (2008) On short paths interdiction problems: total and node-wise limited interdiction. Theory Comput Syst 43(2):204–233

Magnouche Y, Martin S (2020) Most vital vertices for the shortest s-t path problem: complexity and Branch-and-Cut algorithm. Optim Lett 14(2):2039–2053

Martello S, Toth P (1990) Knapsack Problem: algorithms and computer implementations. John Wiley & Sons, Chichester

Thoms HC, Charles EL, Ronald LR, Clifford S (2009) Introduction to algorithms, 3rd edn. The MIT Press, Cambridge

Will COVID-19 patients be re-infected after they are cured and discharged from hospital? Xinhuanet in China. http://www.xinhuanet.com/politics/2020-02/26/c_1210491141.htm. Accessed 26 Feb 2020

Zhang Q, Guan XC, Pardalos PM (2021a) Maximum shortest path interdiction problem by upgrading edges on trees under weighted l_1 norm. J Global Optim 79(4):959–987

Zhang Q, Guan XC, Wang H, Pardalos PM (2021b) Maximum shortest path interdiction problem by upgrading edges on trees under Hamming distance. Optim Lett 15(8): 2661–2680