Maximum shortest path interdiction problem by upgrading edges on trees under weighted l1 norm

Article *in* Journal of Global Optimization · April 2021

DOI: 10.1007/s10898-020-00958-0

CITATIONS
15

READS
116

3 authors:

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
1,728 PUBLICATIONS **48,337** CITATIONS
SEE PROFILE

Journal of Global Optimization (2021) 79:959–987
https://doi.org/10.1007/s10898-020-00958-0

Maximum shortest path interdiction problem by upgrading edges on trees under weighted l_1 norm

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

Received: 18 January 2020 / Accepted: 28 September 2020 / Published online: 7 October 2020

Abstract
Network interdiction problems by deleting critical edges have wide applicatio ns. However, in some practical applications, the goal of deleting edges is difficult to achieve. We consider the maximum shortest path interdiction problem by upgrading edges on trees (MSPIT) under unit/weighted l_1 norm. We aim to maximize the the length of the shortest path from the root to all the leaves by increasing the weights of some edges such that the upgrade cost under unit/weighted l_1 norm is upper-bounded by a given value. We construct their mathematical models and prove some properties. We propose a revised algorithm for the problem (MSPIT) under unit l_1 norm with time complexity $O(n)$, where n is the number of vertices in the tree. We put forward a primal dual algorithm in $O(n^2)$ time to solve the problem (MSPIT) under weighted l_1 norm, in which a minimum cost cut is found in each iteration. We also solve the problem to minimize the cost to upgrade edges such that the length of the shortest path is lower bounded by a value and present an $O(n^2)$ algorithm. Finally, we perform some numerical experiments to compare the results obtained by these algorithms.

Keywords Network interdiction problem · Upgrading critical edges · Shortest path · Weighted l_1 norm · Primal dual algorithm · Minimum cost cut

1 Introduction

Network interdiction problems by deleting critical edges (denoted by **(NIP-DE)**) have been studied in recent twenty years. The classical problem **(NIP-DE)** mainly has two types. One is the K-most-critical-edge problem [1–3,5–7,10,12,13], which aims at making some network performance as poor as possible by deleting at most K edges, and the other one is the critical edge interdiction problem [4,17], which aims to delete as fewer edges as possible to assure some network performance bounded by a constant. The problem **(NIP-DE)** has been

✉ 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

widely studied in some main network performances including a shortest path [1,2,6,11,13], a minimum spanning tree [5,8,10,12,15], a maximum matching [4,16,17,19], a maximum flow [18,20] and a center or median location [3] etc. They have wide applications in communication networks, transportation networks, network war and terrorist networks [7,11].

The problem **(NIP-DE)** was first applied to the shortest path problem by Corley and Sha in [6]. For any K, Bar-Noy et al. [1] showed that it is NP-hard. Khachiyan et al. [11] showed that there is no approximation algorithm with approximation ratio 2, which is the current best result. When $K = 1$ Nardelli et al. [13] proposed an $O(m\alpha(m, n))$ algorithm to solve the problem on undirected networks, where α is the inverse Ackermann function, m, n are the number of edges and vertices in the network.

From the perspective of parameterization, Bazgan et al. [2] gave the complexity and approximation analysis of the shortest path interdiction problem based on the relationship among the three parameters: shortest path whose length is at least l, the increment b of the length of the path and the diameter or a graph. An algorithm with time complexity $O(mn)$ was proposed when $b = 1$, 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, in which some vehicles go from a source vertex s to a destination vertex t. 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 block at least K edges. 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. What we can do is only to lengthen or shorten the weights of some edges to prolong service due to some emergence schemes or alternative schemes which are always available. For example, in network war, our aim is to block support of enemies, but it is hard to achieve and we can only prolong their support time. In terrorist networks given in Fig. 1 [7], once a terrorist attacks node k_{10} which will cause fire, terrorists want to prevent the fire trucks' transportation from node k_1 to k_{10} via the shortest path, that is, to maximize the shortest path of fire trucks depending on limited interdiction budget by interdicting some arcs. Correspondingly, defenders should determine in advance the risky arc(s) that will interdict 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, we put forward a concept of upgrading critical edges, based on which we consider the shortest path interdiction problems on trees.

The maximum shortest path interdiction problem by upgrading edges on trees (denoted by **(MSPIT-UE)** and **(MSPIT)** in brief) can be defined as follows. Let $T = (V, E, w)$ be an edge-weighed tree rooted at v_1, where $V = \{v_1, v_2, \ldots, v_n\}$ and $E = \{e_2, e_3, \ldots, e_n\}$ are the sets of vertices and edges, respectively. Let $Y = \{t_1, t_2, \ldots, t_l\}$ be the set of leaves. Let $w(e)$ be an original length and $u(e)$ be an upper-bound of $w(e)$ after upgrade for each edge $e \in E$, where $w(e) \leq u(e)$. Denote by $d(e) = u(e) - w(e)$ the deviation between $u(e)$ and $w(e)$. Let $c(e)$ be a cost to upgrade the edge $e \in E$. Denote by P_{v_i,v_j} the path from v_i to v_j in T. Define $f(v_i, v_j) = \sum_{e \in P_{v_i,v_j}} f(e)$ for a vector f and a path P_{v_i,v_j}. The problem **(MSPIT)** aims to find an upgrade scheme r to maximize the shortest path of the tree from the root v_1 to all the leaves on the premise that the total upgrade cost under some norm is upper-bounded by a given value M. The mathematical model can be stated as follows.

$$\textbf{(MSPIT)} \quad \begin{aligned} &\max \ \min_{t \in Y}(w(v_1, t) + r(v_1, t)) \\ &s.t. \ \ \|r\| \leq M, \\ &\quad 0 \leq r \leq d. \end{aligned}$$

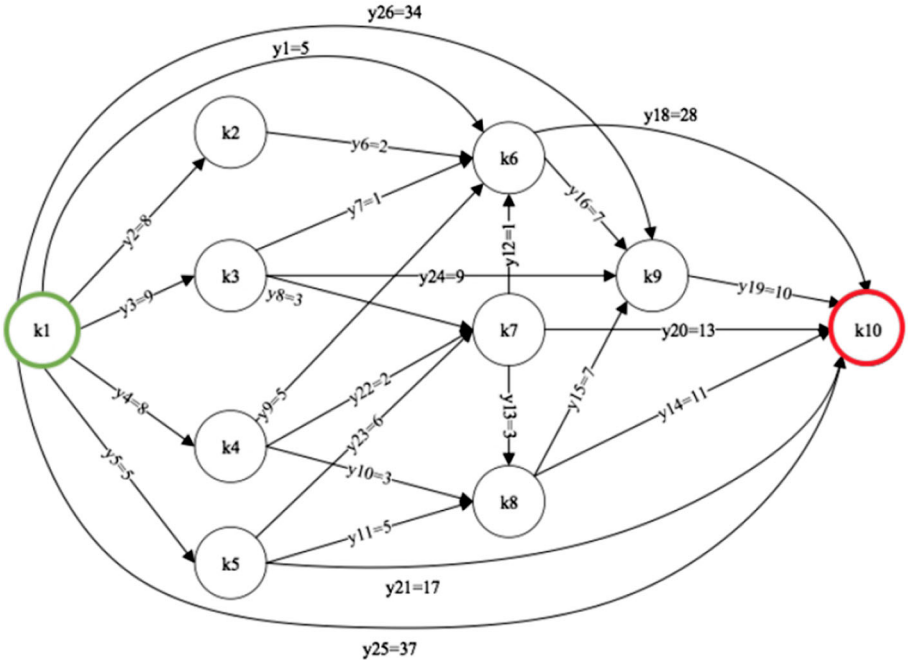

Fig. 1 A terrorist network consisting of 10 nodes and 26 arcs [7]. The interdiction costs are shown on the relevant arcs

When the weighted l_1 norm is applied to the cost $\|r\|$, the problem (**MSPIT**$_1$) under weighted l_1 norm can be formulated in the following form.

$$(\mathbf{MSPIT}_1)\quad \begin{aligned} &\max \ \min_{t\in Y}(w(v_1,t)+r(v_1,t))\\ &s.t.\ \ \sum_{e\in E} c(e)r(e)\le M,\\ &\quad 0\le r(e)\le d(e), e\in E. \end{aligned} \tag{1}$$

The problem (**MSPIT**) under unit l_1 norm ($c(e) = 1$ for each $e \in E$) is denoted by (**MSPIT**$_{u1}$). Hambrusch and Tu [9] considered a similar problem, the edge weight reduction problem in directed trees (denoted by (**EWRT**)), in which some edge weights are reduced to minimize the length of the longest paths from the root to the leaves and the total cost under unit l_1 norm does not exceed a given value. The relative mathematical model is as follows.

$$(\mathbf{EWRT})\quad \begin{aligned} &\min \ \max_{t\in Y}(w(v_1,t)-r(v_1,t))\\ &s.t.\ \ \sum_{e\in E} r(e)\le M,\\ &\quad 0\le r(e)\le d(e), e\in E. \end{aligned}$$

They proposed an $O(n)$ algorithm in two steps. The first step is to determine a range $[L_{k-1}, L_k)$ in which the optimal objective value L^* lies by a binary search algorithm in $O(n)$ time. The second step is to obtain the exact value L^* and an optimal solution r^* according to the two reduction schemes obtained by L_{k-1} and L_k. However, we found a mistake in the formula to calculate r^* in the second step. Thus, we revised the algorithm, whose time complexity is still $O(n)$ for our problem (**MSPIT**$_{u1}$) as well as their problem (**EWRT**). In this paper, we mainly consider the problem (**MSPIT**$_1$) under weighted l_1 norm

Table 1 The relationship between the previous research and our research

Problem	Graph	$K/b/c$		Complexity	Reference
NIP-DE on shortest path	General graph	any K		NP-hard	[1]
				Not approximable within ratio 2	[11]
	Undirected networks	$K=1$		$O(m\alpha(m,n))$	[13]
		$b=1$		$O(mn)$	[2]
		$K=1$		$O(n^2)$	[20]
		$K>1$		Strongly polynomial time	
EWRT	Tree	$K>1, c=1$		$O(n)$	[9] & Alg. 3
MSPIT$_1$-UE	Tree	$K>1$	$c=1$	$O(n)$	Alg. 3
			$c>1$	$O(n^2)$	Alg. 5
MCSPIT$_1$-UE			$c=1$	$O(n)$	Alg. 2
			$c>1$	$O(n^2)$	Alg. 6

and proposed an $O(n^2)$ algorithm, which was not studied in [9] and other references as far as we know.

Furthermore, we consider a minimum cost shortest path interdiction problem by upgrading edges on trees (denoted by (**MCSPIT**$_1$)) under weighted l_1 norm, which is similar to the problem (**MSPIT**$_1$). We aim to upgrade some edges to minimize the total cost under weighted l_1 norm on the premise that the length of the shortest path of the tree is lower-bounded by a given value L.

$$(\textbf{MCSPIT}_1)\quad \begin{array}{l} \min \sum_{e\in E} c(e)r(e), \\ s.t.\ \min_{t\in Y}(w(v_1,t)+r(v_1,t)) \geq L \\ 0 \leq r(e) \leq d(e), e \in E. \end{array} \tag{2}$$

The relationship between the previous research and our research can be shown in Table 1. Upgrading edges rather than deleting edges is the biggest difference.

The paper is organized as follows. In Sect. 2, we present a preprocessing algorithm to delete some edges which are not needed to be upgraded for the current problem. In Sect. 3, we prove some properties of the problem (**MSPIT**$_{u1}$) and propose a revised algorithm with time complexity $O(n)$. In Sect. 4, we study the problem (**MSPIT**$_1$) under weighted l_1 norm and propose a primal-dual algorithm in $O(n^2)$ time. In Sect. 5, we solve the problem (**MCSPIT**$_1$) in $O(n^2)$ time. In Sect. 6, we present some computational experiments to show the effectiveness of the algorithms. In Sect. 7, we draw a conclusion and present future research.

2 A preprocessing algorithm

In this section, we first present some important definitions to identify a storage structure of the rooted tree. Then we propose a preprocessing algorithm to delete the edges which are definitely unnecessary to be upgraded for the current problem.

Given an upgrade scheme r for tree T, the upgraded tree T_r is obtained from T by replacing the edge weight vector w by $w+r$. Let

$$R(T_r) = \min_{t\in Y}(w(v_1,t)+r(v_1,t)) \tag{3}$$

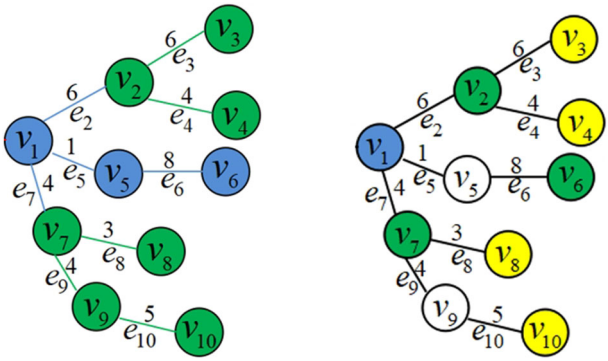

Fig. 2 A tree with cost $c(e)$ on edge e. In the left tree, the Tab of the blue vertices is 1 and of the green vertices is 2. In the right tree, the green vertices are the critical children of v_1, and the chains are stored in the vertices in green and yellow

be the length of the shortest path in T_r. Obviously, $R(T_d)$ is the length of the shortest path in T_d when all the modifications of edges are upgraded to the upper-bound vector d. Let

$$L_{max} = R(T_d). \tag{4}$$

Obviously, we can conclude that

Lemma 1 *The maximum length $R(T_{r^*})$ of the shortest path in T_{r^*} with respect to an optimal upgrade scheme r^* of the problem (**MSPIT**$_{u1}$) or (**MSPIT**$_1$) is not greater than L_{max}.*

2.1 Some important definitions

In this subsection, we present some important definitions to identify a storage structure of the rooted tree. We define a set of critical children for a vertex v whose degree $deg(v)$ is larger than 2 and its corresponding critical father, then store a chain from a critical father to one of its critical child in the child vertex.

For convenience, the label of an edge $e_j = (v_i, v_j)$ is defined as the label of the endpoint v_j farther to the root v_1 in T. For example, $e_8 = (v_7, v_8)$ in Fig. 2. Let $P_j = P_{v_1,t_j}$ be the path from v_1 to a leaf $t_j \in Y$. Let $V^* = \{v \in V | deg(v) > 2\}$ be the set of vertices whose degree is more than 2.

Definition 2 For $e_j = (v_i, v_j)$, where v_i is closer to the root v_1, we call v_i is the **father** of v_j. Define $Tab(v_1) = 1$ and the Tab of any other vertex $v \in V \setminus \{v_1\}$ by

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

Definition 2 gives a Tab for each vertex. As shown in Fig. 2, $deg(v_5) \le 2$, $father(v_5) = v_1$, so $Tab(v_5) = Tab(v_1) = 1$. For v_2, $deg(v_2) > 2$, $father(v_2) = v_1$, thus, $Tab(v_2) = 1 + 1 = 2$.

Definition 3 For a vertex $u \in \{v_1\} \cup V^*$, we define a set $CC(u)$ of **Critical Children**. Let u be in the path from v_1 to $v \in Y \cup V^*$. If $v \in V^*$ and $Tab(v) = Tab(u) + 1$, then $v \in CC(u)$; if $v \in Y$ and $Tab(v) = Tab(u)$, then $v \in CC(u)$. Correspondingly, we call u is the **Critical Father** of v, denoted by $CF(v) = u$.

For a vertex $u \in \{v_1\} \cup V^*$, the set $CC(u)$ of critical children is composed of the leaves with the same Tab as u's and the vertices in V^* whose Tab is one more than u's. In Fig. 2, $CC(v_1) = \{v_2, v_6, v_7\}$, $CF(v_2) = v_1$ and $CF(v_6) = v_1$.

Definition 4 For any vertex $v \in Y \cup V^* \backslash \{v_1\}$, define $chain(v) = P_{CF(v),v}$ as the chain from $CF(v)$ to v and the minimum cost of the chain is defined as $C(chain(v)) = \min\limits_{e \in chain(v), d(e)>0} c(e)$.

Definition 4 shows the storage of chains. For any $v \in Y \cup V^* \backslash \{v_1\}$, $chain(v)$ contains only one path, on which all the vertices exclude $CF(v)$ and v have degrees 2. In Fig. 2, for v_2, $CF(v_2) = v_1$, thus $chain(v_2) = P_{v_1,v_2}$; for v_6, $chain(v_6) = P_{v_1,v_6} = \{v_1, v_5, v_6\} = \{e_5, e_6\}$ and $C(chain(v_6)) = \min\{1, 8\} = 1$.

Definition 5 Define $Layer(v_1) = 0$ and $Layer(v_j) = Tab(father(v_j))$ as the layer number of vertex v_j. Let $\beta = \max_{v_j \in V} Layer(v_j)$.

2.2 A preprocessing algorithm

The preprocessing algorithm aims to reduce the unnecessary steps by deleting the leaves, to which the length from the root is no less than a given length L, and the relevant stored chains excluding the critical fathers.

Let $b(t_i) = CC(CF(t_i))$ be the set of leaf brothers for a leaf $t_i \in Y$. Let

$$b^*(t_i) = \{t_j \in b(t_i) \cap Y | w(P_j) \geq L\} \tag{5}$$

be the set of useless leaf brothers in $b(t_i)$ to which the length from the root is no less than a given value L. Then we should delete the chains stored in the leaves in $b^*(t_i)$ excluding $CF(t_i)$. Then we update $b'(t_i) = b(t_i) \backslash b^*(t_i)$ and $Y' = Y \backslash b^*(t_i)$. Moreover, if $b'(t_i) = \emptyset$, then $CF(t_i)$ becomes a vertex with degree 1, which is not a leaf in the original tree, and hence we need to delete the chain stored in the leaves' critical father $CF(t_i)$. We also delete $CF(t_i)$ from the set of critical children of $CF(CF(t_i))$, that is, $CC(CF(CF(t_i))) := CC(CF(CF(t_i))) \backslash CF(t_i)$. To assure the work of deleting chains run correctly, we perform the above process in a bottom-top fashion and choose the leaves in the non-increasing order of layers.

Note that for a given tree T, we first run the preprocessing algorithm for $L = L_{max}$, then we need to delete all the paths P whose length $w(P) \geq L_{max}$. For example, for the left tree in Fig. 3, $L_{max} := 38$, $w(P_6) = w(v_1, v_6) := 43$, $w(P_7) := 52$, $w(P_8) := 43$, $w(P_{14}) := 43$, $Layer(v_6) := 3$, $Layer(v_7) := 3$, $Layer(v_8) := 2$, $Layer(v_{14}) = 2$, and $\beta = \max_{v \in V} Layer(v) := 3$. For $j := 3$, $Y^3 := \{v_6, v_7\}$, choose v_6, then $b(v_6) := \{v_6, v_7\}$ and $b^*(v_6) := \{v_6, v_7\}$. We delete $chain(v_6) \backslash \{v_5\}$, $chain(v_7) \backslash \{v_5\}$, and then $b(v_6) = \emptyset$, delete $chain(v_5) \backslash \{v_4\}$ and update $CC(v_4) = \{v_5, v_8\} \backslash \{v_5\} := \{v_8\}$, as shown in Fig. 3. For $j := 2$, $Y^2 := \{v_8, v_{11}, v_{13}, v_{14}, \}$, choose v_8, then $b(v_8) := b^*(v_8) := \{v_8\}$. We delete $chain(v_8) \backslash \{v_4\}$, then $b(v_8) := \emptyset$ and delete $chain(v_4) \backslash \{v_1\}$. Choose v_{14} and delete $chain(v_{14}) \backslash \{v_9\}$, as shown in Fig. 4. Finally, after the preprocessing, we obtain the right tree in Fig. 4.

We can easily conclude that

Algorithm 1 $(T', Y', E_{del}) = Preprocess(T, Y, CC, CF, chain, Layer, w, L)$

Require: A tree $T := (V, E)$, the set Y of leaves; the set CF of critical fathers and the chains; the Layer of vertices, an edge weight vector w, and a given value L.

Ensure: A tree T', the set Y' of leaves and the set E_{del} of deleted edges.

For tree T rooted at v_1, let $Y := \{t_1, t_2, \ldots, t_l\}$, $Y' := Y$ and $T' := T$. Initialize $E_{del} := \emptyset$.

Let $P_i := P_{v_1,t_i}$ and $w(P_i) := \sum_{e\in P_i} w(e)$ for each $t_i \in Y'$. Let $\beta := \max_{v\in V} Layer(v)$.

for $j = \beta : -1 : 1$ **do**

  let $Y^j := \{t_i \in Y | Layer(t_i) = j\}$.

  while $Y^j \neq \emptyset$ **do**

    choose $t_i^j \in Y^j$. Let $b(t_i^j) := CC(CF(t_i^j))$ and $b^*(t_i^j) := \{t_k \in b(t_i^j) \cap Y' | w(v_1, t_k) \geq L\}$.

    if $b^*(t_i^j) \neq \emptyset$ **then**

      delete $V_{del} := \bigcup_{t_k\in b^*(t_i^j)} (chain(t_k)\backslash CF(t_i^j))$ and $E_{del}^j := \{e_k | v_k \in V_{del}\}$ from T', $E_{del} := E_{del} \cup E_{del}^j$, $b(t_i^j) := b(t_i^j)\backslash b^*(t_i^j)$, $Y' := Y'\backslash b^*(t_i^j)$.

      if $b(t_i^j) = \emptyset$ **then**

        delete $V_{del} := (chain(CF(t_i^j))\backslash CF(CF(t_i^j)))$ and $E_{del}^j := \{e_k | v_k \in V_{del}\}$ from T', $E_{del} := E_{del} \cup E_{del}^j$, $CC(CF(CF(t_i^j))) := CC(CF(CF(t_i^j)))\backslash CF(t_i^j)$.

      end if

    end if

  end while

end for

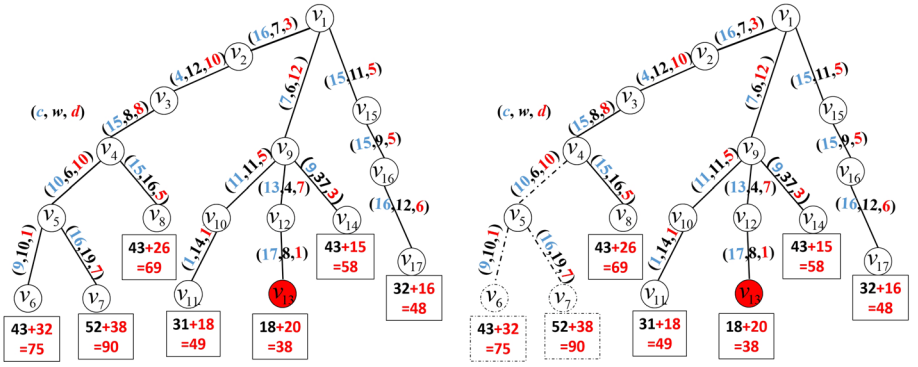

Fig. 3 For the left tree $L_{max} = 38$, delete the $chain(v_6)\backslash\{v_5\}$, $chain(v_7)\backslash\{v_5\}$ and $chain(v_5)\backslash\{v_4\}$

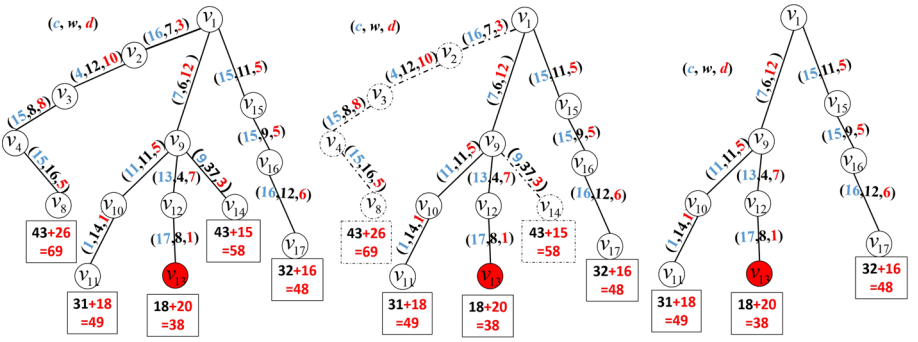

Fig. 4 For the left tree $L_{max} = 38$, delete the $chain(v_8)\backslash\{v_4\}$, $chain(v_4)\backslash\{v_1\}$, $chain(v_{14})\backslash\{v_9\}$

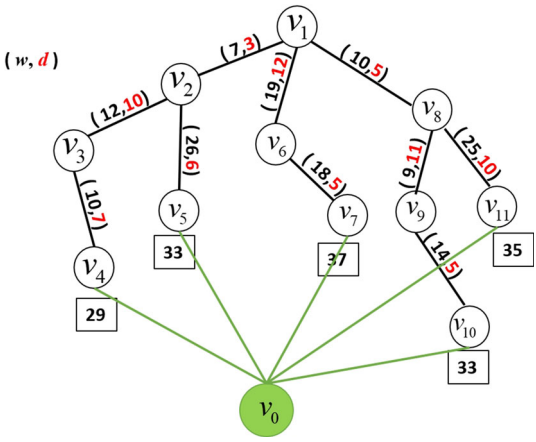

Fig. 5 An example of T_{v_0} and $T = T_{v_1}$ obtained by deleting v_0 and its adjacent edges

Theorem 6 *The preprocessing Algorithm 1 can be done in $O(n)$ time.*

3 Solve the problem (MSPIT$_{u1}$)

Let $M(T_r) = \sum_{e\in E} r(e)$ be the total upgrade cost. Then the problem (**MSPIT**$_{u1}$) can be formulated as (6).

$$\begin{aligned} &\max\ R(T_r) = \min_{t\in Y}(w(v_1,t) + r(v_1,t)) \\ (\mathbf{MSPIT}_{u1})\ &s.t.\ \ M(T_r) \le M, \\ &\quad 0 \le r(e) \le d(e), e \in E. \end{aligned} \tag{6}$$

When solving the problem (**MSPIT**$_{u1}$), we need to solve a sub-problem, minimum cost shortest path interdiction problem on trees by upgrading critical edges under unit l_1 norm, which is denoted by (**MCSPIT**$_{u1}$). We aim to minimize the cost to upgrade some edges on the premise that the shortest path of the tree is lower-bounded by a given value L.

$$\begin{aligned} &\min\ M(T_r) \\ (\mathbf{MCSPIT}_{u1})\ &s.t.\ \ R(T_r) \ge L, \\ &\quad 0 \le r(e) \le d(e), e \in E. \end{aligned} \tag{7}$$

To solve the problem (**MCSPIT**$_{u1}$), we construct an auxiliary network T_{v_0} by adding an artificial terminal v_0 and some edges (t, v_0) with $w(t, v_0) = 0$ and $d(t, v_0) = 0$ for every leaf $t \in Y$, just as shown in Fig. 5.

In this section, we first analyze some properties of the problem (**MCSPIT**$_{u1}$), then propose a linear time algorithm to solve it. Finally, we present a linear time algorithm to solve the problem (**MSPIT**$_{u1}$) followed by complexity analysis and an example.

3.1 Solve the problem (MCSPIT$_{u1}$)

To solve the problem (**MCSPIT**$_{u1}$), we aim to generate an upgrade scheme r^* satisfying $R(T_{r^*}) \ge L$ and minimizing $M(T_{r^*})$. We first analyze some properties of an optimal scheme r^*, then propose a linear time algorithm.

Let r be an upgrade scheme. Define

$$R_r(v_i, v_j) = \begin{cases} w(v_i, v_j) + r(v_i, v_j), & if\ v_j \neq v_0, \\ \min_{P \in P_{v_i, v_j}} (w(P) + r(P)), & if\ v_j = v_0. \end{cases} \tag{8}$$

Hence, $R_r(v_1, v_0) = R(T_r)$. Specially, when $r(v_i, v_j) = 0$ for all $(v_i, v_j) \in E$, $R_0(v_i, v_j) = w(v_i, v_j)$ if $v_j \neq v_0$ and $R_0(v_i, v_0) = \min_{P \in P_{v_i, v_0}} w(P)$.

Definition 7 [9] An upgrade scheme r is **canonical** if $R_r(v_1, v_i) \geq R_{r'}(v_1, v_i)$ for all $v_i \in V \cup \{v_0\}$ and for any other scheme r' with $M(T_r) = M(T_{r'})$.

Notice that the upgrades occur as close to the root as possible in a canonical upgrade scheme. Furthermore, we can divide the edges into three classes due to their upgrade weight.

Definition 8 [9] We refer to an edge e with $r(e) = d(e)$ (resp. $r(e) = 0$) as an edge with full (resp. zero) upgrade. An edge e with $0 < r(e) < d(e)$ is called an edge with partial upgrade.

The following lemma gives a characterization of edge upgrade in an optimal canonical upgrade scheme.

Lemma 9 [9] *Assume that r is an optimal upgrade scheme. Then r is the optimal canonical upgrade scheme if and only if for every path P from v_1 to v_0, if P contains upgraded edges, then there exists one edge (v_i, v_j) on P (v_i is closer to the root v_1) such that each edge on P from v_1 to v_i has full upgrade and each edge on P from v_j to v_0 has zero upgrade.*

According to Lemma 9, we can obtain Algorithm 2 to solve the problem (**MCSPIT**$_{u1}$), which is similar to the method given in Page 71 of [9]. For each edge $(v_i, v_j) \in E$, let $\Delta R_{ij} = L - w(v_1, v_i) - R_0(v_i, v_0)$, then we can determine an optimal upgrade scheme $r^*(v_i, v_j)$ based on the relationships among $d(v_1, v_i)$, ΔR_{ij} and $d(v_1, v_i) + d(v_i, v_j)$.

Algorithm 2 Solve the problem (**MCSPIT**$_{u1}$): $r^* = OPT(L)$

Require: An auxiliary tree T_{v_0}, the set E_{del} of deleted edges in the preprocessing algorithm; two edge weight vectors w, d, and two values L and L_{max}.
Ensure: An optimal canonical upgrade scheme r^*.
1: **if** $L > L_{max}$ **then**
2: Output "The upgraded length of the shortest path is impossible to be L, and its maximum length is L_{max}". **Return.**
3: **end if**
4: **if** $R_0(v_1, v_0) \geq L$ **then**
5: output $r^*(v_i, v_j) := 0$ for all $(v_i, v_j) \in E$ and **Return.**
6: **else**
7: Let $r^*(v_i, v_j) := 0$ for $(v_i, v_j) \in E_{del}$.
8: **for** $(v_i, v_j) \in E \setminus E_{del}$ **do**
9: Let $\Delta R_{ij} := L - w(v_1, v_i) - R_0(v_i, v_0)$ and $d(v_1, v_i) := \sum_{e \in P_{v_1, v_i}} d(e)$.

$$r^*(v_i, v_j) := \begin{cases} 0, & if\ d(v_1, v_i) \geq \Delta R_{ij}; \\ \Delta R_{ij} - d(v_1, v_i), & if\ d(v_1, v_i) < \Delta R_{ij} < d(v_1, v_i) + d(v_i, v_j); \\ d(v_i, v_j), & if\ \Delta R_{ij} \geq d(v_1, v_i) + d(v_i, v_j). \end{cases} \tag{9}$$

10: **end for**
11: **end if**

Theorem 10 *Algorithm 2 can solve the problem (**MCSPIT**$_{u1}$) in $O(n)$ time.*

3.2 Solve the problem (MSPIT$_{u1}$)

In this subsection, we first analyze some properties of the problem (**MSPIT**$_{u1}$), then propose an algorithm to solve it, finally we show its linear time complexity.

Every vertex $v_j \in V$ induces the entry

$$L_j = w(v_1, v_j) + R_0(v_j, v_0) + d(v_1, v_j). \tag{10}$$

Let $\angle = \{L_j | v_j \in V\}$ be the list containing all the entries.

Lemma 11 *For each edge $e_j = (v_i, v_j) \in E$, the lengths of entries are nondecreasing, that is, $L_i \leq L_j$.*

Proof For any $e_j = (v_i, v_j) \in E$, we have

$$\begin{aligned} & L_j - L_i \\ &= (w(v_1, v_j) + R_0(v_j, v_0) + d(v_1, v_j)) - (w(v_1, v_i) + R_0(v_i, v_0) + d(v_1, v_i)) \\ &= (w(v_1, v_i) + w(v_i, v_j) + R_0(v_j, v_0) + d(v_1, v_i) + d(v_i, v_j)) \\ &\quad -(w(v_1, v_i) + R_0(v_i, v_0) + d(v_1, v_i)) \\ &= w(v_i, v_j) + R_0(v_j, v_0) + d(v_i, v_j) - R_0(v_i, v_0) \end{aligned}$$

If $w(v_i, v_j) + R_0(v_j, v_0) > R_0(v_i, v_0)$, then $L_j - L_i > d(v_i, v_j) \geq 0$. Otherwise, $w(v_i, v_j) + R_0(v_j, v_0) = R_0(v_i, v_0)$, then $L_j - L_i = d(v_i, v_j) \geq 0$. □ □

From Lemma 11, we can see that the smallest entry is $L_1 = R_0(v_1, v_0)$ induced by the root v_1, and the largest efficient entry is L_{max}. Therefore, we update $\angle = \{L_i \in \angle | L_i \leq L_{max}\}$ as the list of **efficient entries** in the remainder of the paper. Furthermore, based on Lemmas 11 and 1, we can conclude that

Corollary 12 *If $L_j > L_{max}$ for vertex v_j, then $r^*(e) = 0$ for any edge e on the path from v_j to a leaf in any optimal upgrade scheme r^*.*

For two given entries L_a and L_b, we call Algorithm 2 and obtain $r_a = OPT(L_a)$ and $r_b = OPT(L_b)$. Determine the sets E_z, E_f, E_p, E_s and E_u on tree T as follows.

- Set $E_z = \{e_i \in E | r_a(e_i) = r_b(e_i) = 0\}$ contains the edges which have zero upgrade in both r_a and r_b.
- Set $E_f = \{e_i \in E | r_a(e_i) = r_b(e_i) = d(e_i)\}$ contains the edges which have full upgrade in both r_a and r_b.
- Set $E_p = \{e_i \in E | 0 < r_a(e_i) < d(e_i), 0 < r_b(e_i) \leq d(e_i), r_b(e_j) = 0, \forall e_j \in P_{v_i,t}\}$ includes the edges having partial/full upgrade in both r_a and r_b.
- Set $E_s = E \setminus (E_z \cup E_f)$ is the set of edges not included in sets E_z and E_f.
- Set $E_u = E_s \setminus E_p$ contains all edges not included in sets E_z, E_p and E_f.

If the entries in $\angle =< L_1, L_2, \ldots, L_n >$ are in nondecreasing order and $r_i = OPT(L_i)$, we have $M(T_{r_{i-1}}) \leq M(T_{r_i})$ and $r_{i-1} \leq r_i$ as $L_{i-1} \leq L_i$.

Let k be the index such that $M(T_{r_{k-1}}) \leq M < M(T_{r_k})$. Next we suppose that E_z, E_f and E_s are defined for the indexes $a = k - 1$ and $b = k$.

Lemma 13 *Let k be the index such that $M(T_{r_{k-1}}) \leq M < M(T_{r_k})$. Then $E_z = \{e \in E | r_k(e) = 0\}$, $E_f = \{e \in E | r_{k-1}(e) = d(e)\}$ and $E_s = \{e | r_{k-1}(e) < r_k(e)\}$.*

Proof On one hand, an edge e having zero upgrade in r_k also has zero upgrade in r_{k-1} and thus $E_z = \{e \in E | r_k(e) = 0\}$. On the other hand, an edge e having full upgrade in r_{k-1} also has full upgrade in r_k and thus $E_f = \{e \in E | r_{k-1}(e) = d(e)\}$. Furthermore, we show that $r_{k-1}(e) < r_k(e)$ for $e \in E_s = E \setminus (E_z \cup E_f)$. It holds obviously for $e \notin E_z$ when $r_{k-1}(e) = 0 < r_k(e) \leq d(e)$, and for $e \notin E_f$ when $0 \leq r_{k-1}(e) < d(e) = r_k(e)$. If $e \in E \setminus (E_z \cup E_f)$ with $0 < r_k(e) < d(e)$ and $0 < r_{k-1}(e) < d(e)$, suppose $r_{k-1}(e) = r_k(e)$. Then by (9) in Algorithm 2, we have $r_{k-1}(e) = r_{k-1}(v_i, v_j) = \Delta R_{ij}^{k-1} - d(v_1, v_i) = \Delta R_{ij}^{k} - d(v_1, v_i) = r_k(e)$ and $L_{k-1} = L_k$, which follows a contradiction $M(T_{r_{k-1}}) = M(T_{r_k})$. Hence, we have $r_{k-1}(e) < r_k(e)$, $e \in E \setminus (E_z \cup E_f)$ and $E_s = \{e | r_{k-1}(e) < r_k(e)\}$. □ □

We will determine an optimal upgrade scheme r^* satisfying $r_{k-1}(e) \leq r^*(e) \leq r_k(e)$ for all $e \in E$ and the length $R(T_{r^*})$ of the shortest path. For an edge $(v_i, v_j) \in E_s$, let

$$\hat{R}_{r_{k-1}}(P_{ij}^*) = R_{r_{k-1}}(v_1, v_i) + w(v_i, v_j) + r_{k-1}(v_i, v_j) + R_0(v_j, v_0) \tag{11}$$

be the length of the shortest path from v_1 to v_0 that goes through edge (v_i, v_j) with respect to r_{k-1}. Let $E_s^* = \{(v_i, v_j) \in E_s | \hat{R}_{r_{k-1}}(P_{ij}^*) < R(T_{r^*})\}$. We have the following property.

Lemma 14 *Let k be the index such that $M(T_{r_{k-1}}) \leq M < M(T_{r_k})$. Then the following statements hold. (1) Every path P from v_1 to v_0 contains at most one edge belonging to set $E_s = \{e | r_{k-1}(e) < r_k(e)\}$. (2) If*

$$R(T_{r^*}) = \frac{M - M(T_{r_{k-1}}) + \sum_{(v_i, v_j) \in E_s^*} \hat{R}_{r_{k-1}}(P_{ij}^*)}{|E_s^*|}, \tag{12}$$

then an optimal upgrade scheme r^* can be calculated by

$$r^*(v_i, v_j) = \begin{cases} d(v_i, v_j), & if\ (v_i, v_j) \in E_f, \\ 0, & if\ (v_i, v_j) \in E_z, \\ r_{k-1}(v_i, v_j), & if\ (v_i, v_j) \in E_s \setminus E_s^*, \\ r_{k-1}(v_i, v_j) + R(T_{r^*}) - \hat{R}_{r_{k-1}}(P_{ij}^*), & if\ (v_i, v_j) \in E_s^*. \end{cases} \tag{13}$$

Proof (1) Suppose there are two edges $(v_{\bar{b}}, v_{\bar{a}})$ and $(v_{\bar{a}}, v_j)$ in one path P from v_1 to v_0, in which $v_{\bar{b}}$ is closer to v_1 and v_j is closer to v_0. Then we have $0 \leq r_{k-1}(v_{\bar{b}}, v_{\bar{a}}) < d(v_{\bar{b}}, v_{\bar{a}})$. **Case 1.** $0 < r_{k-1}(v_{\bar{b}}, v_{\bar{a}}) < d(v_{\bar{b}}, v_{\bar{a}}) = r_k(v_{\bar{b}}, v_{\bar{a}})$, then $r_{k-1}(v_{\bar{a}}, v_j) = 0 < r_k(v_{\bar{a}}, v_j) \leq d(v_{\bar{a}}, v_j)$. By (9) in Algorithm 2, we have $L_k - w(v_1, v_{\bar{a}}) - R_0(v_{\bar{a}}, v_0) > d(v_1, v_{\bar{a}})$, then $L_k > w(v_1, v_{\bar{a}}) + R_0(v_{\bar{a}}, v_0) + d(v_1, v_{\bar{a}}) = L_{\bar{a}}$. Similarly, we have $L_{k-1} \leq L_{\bar{a}}$. Additionally, if $L_{k-1} = L_{\bar{a}}$, we have

$$\begin{aligned} L_{k-1} = L_{\bar{a}} &= w(v_1, v_{\bar{a}}) + R_0(v_{\bar{a}}, v_0) + d(v_1, v_{\bar{a}}) \\ &= w(v_1, v_{\bar{b}}) + w(v_{\bar{b}}, v_{\bar{a}}) + R_0(v_{\bar{a}}, v_0) + d(v_1, v_{\bar{b}}) + d(v_{\bar{b}}, v_{\bar{a}}) \\ &\geq w(v_1, v_{\bar{b}}) + R_0(v_{\bar{b}}, v_0) + d(v_1, v_{\bar{b}}) + d(v_{\bar{b}}, v_{\bar{a}}), \end{aligned}$$

as $w(v_{\bar{b}}, v_{\bar{a}}) + R_0(v_{\bar{a}}, v_0) \geq R_0(v_{\bar{b}}, v_0)$. Then $L_{k-1} - w(v_1, v_{\bar{b}}) - R_0(v_{\bar{b}}, v_0) \geq d(v_1, v_{\bar{b}}) + d(v_{\bar{b}}, v_{\bar{a}})$ holds in (9), thus $r_{k-1}(v_{\bar{b}}, v_{\bar{a}}) = d(v_{\bar{b}}, v_{\bar{a}})$ which contradicts to $r_{k-1}(v_{\bar{b}}, v_{\bar{a}}) < d(v_{\bar{b}}, v_{\bar{a}})$. Hence we have $L_{k-1} < L_{\bar{a}} < L_k$ which contradicts L_{k-1} and L_k are sequential in $\angle$. **Case 2.** $0 = r_{k-1}(v_{\bar{b}}, v_{\bar{a}}) < d(v_{\bar{b}}, v_{\bar{a}}) = r_k(v_{\bar{b}}, v_{\bar{a}})$, then $r_{k-1}(v_{\bar{a}}, v_j) = 0 < r_k(v_{\bar{a}}, v_j) \leq d(v_{\bar{a}}, v_j)$. Similarly we have $L_{k-1} \leq L_{\bar{b}} < L_{\bar{a}}$ and $L_k > L_{\bar{a}}$, thus $L_{k-1} < L_{\bar{a}} < L_k$ results in a contradiction.

(2) Obviously, an edge $e \in E_f$ has full upgrade in r^* and an edge $e \in E_z$ has zero upgrade in r^*. Additionally, for $(v_i, v_j) \in E_s \setminus E_s^*$, we have $\hat{R}_{r_{k-1}}(P_{ij}^*) \geq R(T_{r^*})$ and $r^*(v_i, v_j) = r_{k-1}(v_i, v_j)$.

For an edge (v_i, v_j) in E_s^*, if there is only one path $P = P_{ij}^*$ going through (v_i, v_j), then $r^*(v_i, v_j) = r_{k-1}(v_i, v_j) + R(T_{r^*}) - (w(P) + r_{k-1}(P))$; if there are more than one path going through (v_i, v_j), we find the maximum upgrade amount from the shortest path P_{ij}^*, which results in $r^*(v_i, v_j) = r_{k-1}(v_i, v_j) + R(T_{r^*}) - \hat{R}_{r_{k-1}}(P_{ij}^*)$. Furthermore,

$$\begin{aligned} r^*(v_i, v_j) &= r_{k-1}(v_i, v_j) + R(T_{r^*}) - \hat{R}_{r_{k-1}}(P_{ij}^*) \\ &< r_{k-1}(v_i, v_j) + R(T_{r_k}) - \hat{R}_{r_{k-1}}(P_{ij}^*) \\ &= r_k(v_i, v_j) \le d(v_i, v_j). \end{aligned}$$

The total upgrade cost is

$$\begin{aligned} & M(T_{r^*}) \\ &= \sum_{e \in E_f} d(e) + \sum_{e \in E_s \setminus E_s^*} r_{k-1}(e) + \sum_{(v_i, v_j) \in E_s^*} (r_{k-1}(e) + R(T_{r^*}) - \hat{R}_{r_{k-1}}(P_{ij}^*)) \\ &= M(T_{r_{k-1}}) + \sum_{(v_i, v_j) \in E_s^*} R(T_{r^*}) - \sum_{(v_i, v_j) \in E_s^*} \hat{R}_{r_{k-1}}(P_{ij}^*) \\ &= M(T_{r_{k-1}}) + |E_s^*| R(T_{r^*}) - \sum_{(v_i, v_j) \in E_s^*} \hat{R}_{r_{k-1}}(P_{ij}^*) \\ &= M(T_{r_{k-1}}) + M - M(T_{r_{k-1}}) + \sum_{(v_i, v_j) \in E_s^*} \hat{R}_{r_{k-1}}(P_{ij}^*) - \sum_{(v_i, v_j) \in E_s^*} \hat{R}_{r_{k-1}}(P_{ij}^*) \\ &= M. \end{aligned}$$

Next we will show that $R(T_{r^*})$ is the length of the shortest path under r^*. For $(v_i, v_j) \in E_s \setminus E_s^*$, we have $\hat{R}_{r_{k-1}}(P_{ij}^*) \ge R(T_{r^*})$. For $(v_i, v_j) \in E_s^*$, we have

$$\begin{aligned} & \hat{R}_{r^*}(P_{ij}^*) \\ &= R_{r^*}(v_1, v_i) + w(v_i, v_j) + r^*(v_i, v_j) + R_0(v_j, v_0) \\ &= R_{r_{k-1}}(v_1, v_i) + w(v_i, v_j) + (r_{k-1}(v_i, v_j) + R(T_{r^*}) - \hat{R}_{r_{k-1}}(P_{ij}^*)) + R_0(v_j, v_0) \\ &= R(T_{r^*}). \end{aligned}$$

Combining with (1), we can conclude that the $R(T_{r^*})$ is the length of the shortest path in T_{r^*}. □ □

To determine the set E_s^* and the length $R(T_{r^*})$, we consider a series of $r^\kappa (\kappa = 0, 1, \ldots)$ such that

$$R(T_{r^\kappa}) = \frac{M - M(T_{r_{k-1}}) + \sum_{(v_i, v_j) \in E_s^\kappa} \hat{R}_{r_{k-1}}(P_{ij}^*)}{|E_s^\kappa|}, \tag{14}$$

where $E_s^0 = E_s$, $E_s^{\kappa+1} = \{(v_i, v_j) \in E_s^\kappa | \hat{R}_{r_{k-1}}(P_{ij}^*) < R(T_{r^\kappa})\}$, $\kappa \ge 0$. Such a circulation terminates until $E_s^{\kappa+1} = E_s^\kappa$. In this case, $R(T_{r^*}) = R(T_{r^\kappa})$.

Lemma 15 *If $M(T_{r_{k-1}}) < M$, then $E_s^1 \ne \emptyset$ and $R(T_{r^1}) \le R(T_{r^0})$. Furthermore, $R(T_{r^\kappa}) \le R(T_{r^{\kappa-1}}) \le \cdots \le R(T_{r^1}) \le R(T_{r^0})$. When $E_s^\kappa \setminus E_s^{\kappa+1} = \emptyset$, $R(T_{r^*}) = R(T_{r^\kappa})$.*

Proof If $E_s^1 = \emptyset$, then $\hat{R}_{r_{k-1}}(P_{ij}^*) \ge R(T_{r^0})$ for any edge in E_s. Hence, it follows from (14) that

$$R(T_{r^0}) \ge \frac{M - M(T_{r_{k-1}}) + |E_s| R(T_{r^0})}{|E_s|} = R(T_{r^0}) + \frac{M - M(T_{r_{k-1}})}{|E_s|} > R(T_{r^0}),$$

which is a contradiction.

It follows from (14) that

$$M - M(T_{r_{k-1}}) + \sum_{(v_i,v_j)\in E_s} \hat{R}_{r_{k-1}}(P^*_{ij}) = |E_s|R(T_{r^0}) \tag{15}$$

$$M - M(T_{r_{k-1}}) + \sum_{(v_i,v_j)\in E^1_s} \hat{R}_{r_{k-1}}(P^*_{ij}) = |E^1_s|R(T_{r^1}) \tag{16}$$

By (15) minus (16), we can obtain

$$(|E_s| - |E^1_s|)R(T_{r^0}) \le \sum_{(v_i,v_j)\in E_s\setminus E^1_s} \hat{R}_{r_{k-1}}(P^*_{ij}) = |E_s|R(T_{r^0}) - |E^1_s|R(T_{r^1}). \tag{17}$$

Then $|E^1_s|R(T_{r^1}) \le |E^1_s|R(T_{r^0})$. It follows from $E^1_s \neq \emptyset$ that $|E^1_s| > 0$ and $R(T_{r^1}) \le R(T_{r^0})$. Similarly, we can show that $R(T_{r^\kappa}) \le R(T_{r^{\kappa-1}}) \le \cdots \le R(T_{r^1}) \le R(T_{r^0})$.

When $E^\kappa_s \setminus E^{\kappa+1}_s = \emptyset$, then $E^{\kappa+1}_s = E^\kappa_s$ and $\hat{R}_{r_{k-1}}(P^*_{ij}) < R(T_{r^\kappa})$ holds for each $(v_i, v_j) \in E^\kappa_s$. Hence, $E^\kappa_s = E^*_s = \{(v_i, v_j) \in E_s | \hat{R}_{r_{k-1}}(P^*_{ij}) < R(T_{r^*})\}$ and $R(T_{r^*}) = R(T_{r^\kappa})$. □ □

Based on the above properties, we can state the Algorithm 3 to solve the problem (**MSPIT**$_{u1}$). The main idea is in two steps. The first step is to determine a range $[L_{k-1}, L_k)$ of the optimal value $R(T^*_r)$ by a binary search method in $O(n)$ time without sorting the entries in the list $\angle$ [9]. The second step is to obtain the exact value $R(T^*_r)$ and an optimal solution r^* according to the two upgrade schemes $r_{k-1} = OPT(L_{k-1})$ and $r_k = OPT(L_k)$.

Here are some notations. Let $\angle_{ab}$ be the sublist of $\angle$ containing the entries L_j with $L_a < L_j < L_b$ and $n_{ab} = |\angle_{ab}|$. Let $M_f = \sum_{(v_i,v_j)\in E_f} d(v_i, v_j)$ be the total upgrade spent on the edges in set E_f. Let $M_{p,b} = \sum_{e\in E_p} r_b(e)$ be the total upgrade made on the set E_p in upgrade scheme r_b. Let L_q be the $(\lfloor \frac{n_{ab}}{2} \rfloor)$th smallest element in list $\angle_{ab}$. Let $\delta = L_b - L_q$. Let $E_{u,z}$, $E_{u,p}$ and $E_{u,f}$ be the sets of edges in E_u have zero, partial and full upgrade in r_q, respectively. Then $E_u = E_{u,z} \cup E_{u,p} \cup E_{u,f}$. The total upgrade of scheme r_q is determined below.

$$M(T_{r_q}) = M_f + (M_{p,b} - \delta \times |E_p|) + M_{u,f} + M_{u,p}, \tag{18}$$

with $M_{u,f} = \sum_{(v_i,v_j)\in E_{u,f}} d(v_i, v_j)$, $M_{u,p} = \sum_{(v_i,v_j)\in E_{u,p}} r_q(v_i, v_j)$.

Now we analyze the time complexity of Algorithm 3. It is easy to see that work done in Lines 11–22, an iteration in while circulation, is bounded by $O(|E_u|)$. The authors in [9] showed that $|E_u| \le 2n_{ab}$ and n_{ab} reduces by half from one iteration to the next. The edge $(v_i, v_q) \in E_u$ inducing the entry L_q is no longer in E_u by the end of the current iteration. When $L_q < L_k < L_b$, we have $n_{qb} \le \frac{n_{ab}}{2}$ and when $L_a < L_k < L_q$ we have $n_{aq} \le \frac{n_{ab}}{2}$. Hence, searching for index k in the whole while circulation takes $O(n)$ time. Obtaining the exact value $R(T^*_r)$ in lines 25–32 can be completed in $O(n)$ time and the optimal upgrade scheme r^* is generated from r_k and r_{k-1} in Line 33 by (13) in $O(n)$ time. Thus, the $O(n)$ time bound for the problem (**MSPIT**$_{u1}$) follows.

Theorem 16 *Algorithm 3 can solve the problem (**MSPIT**$_{u1}$) in $O(n)$ time.*

Remark 17 The optimal upgrade scheme r^* generated from r_k and r_{k-1} in (1)-(3) in page 75 of [9] is given below.

$$r^*(v_i, v_j) = \begin{cases} d(v_i, v_j), & if\ r_{k-1}(v_i, v_j) = d(v_i, v_j), \\ 0, & if\ r_k(v_i, v_j) = 0, \\ r_{k-1}(v_i, v_j) + \frac{M - M(T_{r_{k-1}})}{|E_s|}, & if\ (v_i, v_j) \in E_s. \end{cases} \tag{19}$$

Algorithm 3 An algorithm to solve the problem (**MSPIT**$_{u1}$).

Require: A tree $T = (V, E)$, the set Y of leaves; the set CC of critical children, the set CF of critical fathers and the chains; the Layer of vertices, two edge weight vectors w, u and the maximum cost M.
Ensure: The optimal canonical upgrade scheme r^*.
1: Calculate $d := u - w$ and $L_{max} := R(T_d)$.
2: Call $(T, Y, E_{del}) := Preprocess(T, Y, CC, CF, chain, Layer, w, L_{max})$.
3: Determine $r_{max} := OPT(L_{max})$, get the minimum cost $M_{max} := \sum_{e \in E \setminus E_{del}} r_{max}(e)$.
4: **if** $M \geq M_{max}$ **then**
5: output an optimal solution $r^* := r_{max}$ and the optimal value L_{max}, **return.**
6: **else**
7: Construct an auxiliary network T_{v_0} by adding an artificial terminal v_0 and some edges (t, v_0) with $w(t, v_0) := 0$ and $d(t, v_0) := 0$ for every leaf $t \in Y$.
8: Calculate L_j by (10) for each vertex $v_j \in V$. Determine the list $\angle$ of the efficient entries induced by L_j which is no more than L_{max}.
9: Set $L_a := L_1$, $L_b := L_{max}$, $\angle_{ab} := \angle$, $n_{ab} := |\angle_{ab}|$, $r_a := OPT(L_a)$ and $r_b := OPT(L_b)$. For tree T, determine the sets E_z, E_u, E_p and E_f.
10: **while** $n_{ab} \neq 0$ **do**
11: Let L_q be the $(\lfloor \frac{n_{ab}}{2} \rfloor)$th smallest element in list $\angle_{ab}$. Obtain $r_q(v_i, v_j)$ by (9) and determine the sets $E_{u,z}$, $E_{u,p}$ and $E_{u,f}$. Calculate $M(T_{r_q})$ according to (18).
12: **if** $M(T_{r_q}) < M$ **then**
13: let $L_a := L_q$, $E_f := E_f \cup E_{u,f}$, $E_u := E_u \setminus E_{u,f}$.
14: Edges from $E_{u,p}$ that qualify for E_p are moved from set E_u to E_p.
15: Update $\angle_{ab}$ by deleting the entries smaller than L_q. Let $n_{ab} := |\angle_{ab}|$.
16: **else if** $M(T_{r_q}) > M$ **then**
17: let $L_b := L_q$, $E_z := E_z \cup E_{u,z}$, $E_u := E_u \setminus E_{u,z}$.
18: Edges from $E_{u,p}$ and $E_{u,f}$ that qualify for E_p are moved from set E_u to E_p.
19: Update $\angle_{ab}$ by deleting the entries larger than L_q. Let $n_{ab} := |\angle_{ab}|$.
20: **else**
21: let $L_k := L_q$, **Return** $r^* := OPT(L_q)$.
22: **end if**
23: **end while**
24: Let $L_{k-1} := L_a$, $L_k := L_b$ and $E_s := E \setminus (E_z \cup E_f)$.
25: Determine $r_k := OPT(L_k)$ and $r_{k-1} := OPT(L_{k-1})$.
26: Let $R(T_{r^0}) := \frac{M - M(T_{r_{k-1}}) + \sum_{(v_i, v_j) \in E_s} \hat{R}_{r_{k-1}}(P^*_{ij})}{|E_s|}$.
27: Let $E_s^0 := E_s$, $E_s^1 := \{(v_i, v_j) \in E_s^0 | \hat{R}_{r_{k-1}}(P^*_{ij}) < R(T_{r^0})\}$, $R(T_{r^1}) := R(T_{r^0})$, $\kappa := 1$.
28: **while** $E_s^{\kappa-1} \setminus E_s^{\kappa} \neq \emptyset$ **do**
29: $R(T_{r^\kappa}) := \frac{M - M(T_{r_{k-1}}) + \sum_{(v_i, v_j) \in E_s^{\kappa}} \hat{R}_{r_{k-1}}(P^*_{ij})}{|E_s^{\kappa}|}$,
30: Update $E_s^{\kappa+1} = \{(v_i, v_j) \in E_s^{\kappa} | \hat{R}_{r_{k-1}}(P^*_{ij}) < R(T_{r^\kappa})\}$ and $\kappa := \kappa + 1$.
31: **end while**
32: Let $E_s^* := E_s^{\kappa}$ and $R(T_{r^*}) := R(T_{r^\kappa})$.
33: Calculate $r^*(v_i, v_j)$ by (13) for $(v_i, v_j) \in E \setminus E_{del}$ and $r^*(v_i, v_j) := 0$ for $(v_i, v_j) \in E_{del}$.
34: **end if**

This implies that the rest cost $(M - M(T_{r_{k-1}}))$ is distributed to $r^*(v_i, v_j)$ for $(v_i, v_j) \in E_s$ in average, which is not actually true. The distribution must be made according to the length of every path, which may not be equal, just as (13) shows.

For example, in Fig. 5, for a given $M := 30$, according to lines 7–24 in Algorithm 3, we can determine two consecutive entries $L_8 := 38$ and $L_5 := 42$ with $M_5 := 35 > M > M_8 := 17$. We have

$$r_8 = OPT(L_8) := (3, 6, 0, 2, 1, 0, 5, 0, 0, 0),$$
$$r_5 = OPT(L_5) := (3, 10, 0, 6, 5, 0, 5, 4, 0, 2),$$

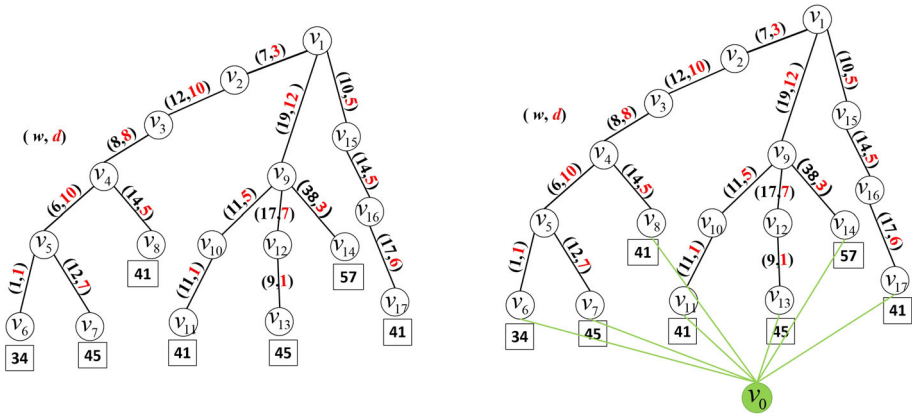

Fig. 6 A tree and its auxiliary network T_{v_0} in Example 1

and $E_s := \{e_3, e_5, e_6, e_9, e_{11}\}$. The solution calculated by (19) is

$$r^* := (3, 8.6, 0, 4.6, 3.6, 0, 5, 2.6, 0, 2.6),$$

and the length of the shortest path is 40.6. But the solution obtained by Lines 25–33 in Algorithm 3 is

$$r^* := (3, 9, 0, 5, 4, 0, 5, 3, 0, 1),$$

and the length of the shortest path is 41 which is larger than the former one.

3.3 Two examples of the problem

For the better understanding of Algorithm 3, Example 1 and Example 2 are given to show the detailed computing process.

Example 1 As is shown in Fig. 6, $V := \{v_1, \ldots, v_{17}\}$, $E := \{e_2, \ldots, e_{17}\}$, $M := 40$, $t_1 := v_6$, $t_2 := v_7$, $t_3 := v_8$, $t_4 := v_{11}$, $t_5 := v_{13}$, $t_6 := v_{14}$, $t_7 := v_{17}$,

$$w := (7,\ 12,\ 8,\ 6,\ 1,\ 12,\ 14,\ 19,\ 11,\ 11,\ 17,\ 9,\ 38,\ 10,\ 14,\ 17),$$
$$u := (10,\ 22,\ 16,\ 16,\ 2,\ 19,\ 19,\ 31,\ 16,\ 12,\ 24,\ 10,\ 41,\ 15,\ 19,\ 23).$$

It is easy to have $d = u - w := \{3, 10, 8, 10, 1, 7, 5, 12, 5, 1, 7, 1, 3, 5, 5, 6\}$, $L_{max} := 57$. By calling the Preprocess Algorithm, we can delete the leaf v_{14} from T as $w(P_{v_{14}}) := 57$ and obtain $E_{del} := \{e_{14}\}$. Then determine

$$r_{max} := OPT(L_{max}) = (3, 10, 8, 2, 0, 0, 0, 12, 4, 0, 0, 0, 5, 5, 6), \quad M_{max} := 55.$$

By Line 8, for every vertex $v_i \in V$, determine the entry list

$$\angle := \{34, 37, 47, 55, 65, 66, 83, 67, 53, 68, 59, 64, 65, 46, 51, 57\}$$

and the efficient list obtained by deleting the entries larger than $L_{max} = 57$ is $\angle := \{L_1, L_2, L_3, L_4, L_9, L_{15}, L_{16}, L_{17}\} = \{34, 37, 47, 55, 53, 46, 51, 57\}$. $L_a := 34$, $L_b := 57$. Call $r_a := OPT(L_a) = (0, \ldots, 0)$ and $r_b := OPT(L_b) = (3, 10, 8, 2, 0, 0, 0, 12, 4, 0, 0, 0, 5, 5, 6)$. Determine $E_z := \{e_6, e_7, e_8, e_{11}, e_{12}, e_{13}, e_{14}\}$, $E_f := \emptyset$, $E_p :=$

Table 2 The results for the detailed 3 iterations in Example 1 by Algorithm 3

Iteration	1	2	3
L_q	47	51	53
r_q	(3, 10, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 5, 1, 0)	(3, 10, 4, 0, 0, 0, 0, 10, 0, 0, 0, 0, 0, 5, 5, 0)	(3, 10, 6, 0, 0, 0, 0, 12, 0, 0, 0, 0, 0, 5, 5, 2)
$E_{u,z}$	$\{e_4, e_5, e_{10}, e_{17}\}$	$\{e_5, e_{10}, e_{17}\}$	$\{e_5, e_{10}\}$
$E_{u,p}$	$\{e_9, e_{16}\}$	$\{e_4, e_9\}$	$\{e_4, e_{17}\}$
$E_{u,f}$	$\{e_2, e_3, e_{15}\}$	$\{e_{16}\}$	$\{e_9\}$
$M(T_{r_q})$	$25 < M$	$37 < M$	$43 > M$
L_a	47	51	51
L_b	57	57	53
E_z	$\{e_6, e_7, e_8, e_{11}, e_{12}, e_{13}, e_{14}\}$	$\{e_6, e_7, e_8, e_{11}, e_{12}, e_{13}, e_{14}\}$	$\{e_5, e_6, e_7, e_8, e_{10}, e_{11}, e_{12}, e_{13}, e_{14}\}$
E_p	$\emptyset$	$\emptyset$	$\{e_4\}$
E_f	$\{e_2, e_3, e_{15}\}$	$\{e_2, e_3, e_{15}, e_{16}\}$	$\{e_2, e_3, e_{15}, e_{16}\}$
E_u	$\{e_4, e_5, e_9, e_{10}, e_{16}, e_{17}\}$	$\{e_4, e_5, e_9, e_{10}, e_{17}\}$	$\{e_9, e_{17}\}$
$\angle_{ab}$	$\{L_4, L_9, L_{16}\} = \{55, 53, 51\}$	$\{L_4, L_9\} = \{55, 53\}$	$\emptyset$
n_{ab}	3	2	0

$\emptyset$, $E_u := \{e_2, e_3, e_4, e_5, e_9, e_{10}, e_{15}, e_{16}, e_{17}\}$, $\angle_{ab} := \{L_2, L_3, L_4, L_9, L_{15}, L_{16}\} = \{37, 47, 55, 53, 46, 51\}$, $n_{ab} := 6$.

In the first iteration, $L_q := L_3 = 47$ is the third smallest value in $\angle_{ab}$. Obtain $r_q := (3, 10, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 5, 1, 0)$ for edges in E_u by (9) when $L := L_q = 47$. Then we have $E_{u,z} := \{e_4, e_5, e_{10}, e_{17}\}$, $E_{u,p} := \{e_9, e_{16}\}$ and $E_{u,f} := \{e_2, e_3, e_{15}\}$. We get $M(T_{r_q}) := 25 < M := 40$ by (18). Thus $L_a := 47$, $E_f := E_f \cup E_{u,f} = \{e_2, e_3, e_{15}\}$, $E_u := E_u \backslash E_{u,f} = \{e_4, e_5, e_9, e_{10}, e_{16}, e_{17}\}$, $E_p := \emptyset$. The results for the first three iterations can be shown in Table 2.

After the third iteration, $n_{ab} := 0$, so the iteration terminates and we have $L_{k-1} := L_a = L_{16} = 51$, $L_k := L_b = L_9 = 53$.

Determine $R(T_{r^*})$: $R(T_{r^0}) := 52$, $E_s^0 := E_s = E \setminus (E_z \cup E_f) = \{e_4, e_9, e_{17}\}$, $E_s^1 := \{e_4, e_9, e_{17}\}$, $R(T_{r^1}) = R(T_{r^0})$, $E_s^1 \setminus E_s^0 = \emptyset$, then $R(T_{r^*}) := R(T_{r^1}) = 52$ with $E_s^* := \{e_4, e_9, e_{17}\}$.

Generate $r^* := (3, 10, 5, 0, 0, 0, 0, 11, 0, 0, 0, 0, 0, 5, 5, 1)$ by (13).

Example 2 In Fig. 5, $V := \{v_1, \ldots, v_{11}\}$, $E := \{e_2, \ldots, e_{11}\}$, $t_1 := v_4$, $t_2 := v_5$, $t_3 := v_7$, $t_4 := v_{10}$, $t_5 := v_{11}$, $w := (7, 12, 10, 26, 19, 18, 10, 9, 14, 25)$, $u := (10, 22, 17, 32, 31, 23, 15, 20, 19, 35)$, $M := 10$. Then $d := (3, 10, 7, 6, 12, 5, 5, 11, 5, 10)$, $L_{max} := 42$, the efficient list $\angle := \{L_1, L_2, L_3, L_5, L_8\} = \{29, 32, 42, 42, 38\}$. $L_a := 29$, $L_b := 42$. Then $r_{max} := (3, 10, 0, 6, 5, 0, 5, 4, 0, 2)$ and $M_{max} := 35$.

For the given $M := 10 < M_{max}$, according to lines 7–24 in Algorithm 3, we can determine two consecutive entries $L_2 := 32$ and $L_8 := 38$ with $M_8 := 17 > M > M_2 := 3$ in the list $\angle$. We have

$$r_2 := OPT(L_2) = (3, 0, 0, 0, 0, 0, 0, 0, 0, 0), r_8 := (3, 6, 0, 2, 1, 0, 5, 0, 0, 0).$$

Through lines 26–33, determine $R(T_{r^*})$: $E_s := \{e_3, e_5, e_6, e_8\}$,

$$R(T_{r^0}) := \frac{10-3+(32+36+37+33)}{4} = 36\frac{1}{4},$$

$E_s^0 := E_s = \{e_3, e_5, e_6, e_8\}$, $E_s^1 = \{e_3, e_8\}$, $R(T_{r^1}) := R(T_{r^0}) = 36\frac{1}{4}$, $\kappa := 1$. $E_s^0 \setminus E_s^1 := \{e_5, e_6\} \neq \emptyset$, then do the first while-iteration. We have $R(T_{r^1}) := \frac{10-3+(32+33)}{2} = 36$, $E_s^2 := \{e_3, e_8\}$, $\kappa := 2$, $E_s^1 \setminus E_s^2 := \emptyset$, thus the iteration terminates. Hence, $E_s^* := \{e_3, e_8\}$, $R(T_{r^*}) := R(T_{r^1}) = 36$. Finally, generate $r^* := (3, 4, 0, 0, 0, 0, 3, 0, 0, 0)$ by (13).

4 Solve the problem ($\mathbf{MSPIT_1}$)

In this section, we solve the problem ($\mathbf{MSPIT_1}$) under weighted l_1 norm through a primal dual algorithm. Based on two sub-algorithms, the preprocessing algorithm (Algorithm 1) and the minimum cost cut algorithm (Algorithm 4), we give the primal dual algorithm followed by time complexity analysis and a computational example.

4.1 An algorithm to find a minimum cost cut

In this subsection, we firstly define a minimum cost cut, then give an algorithm to find such a cut.

Definition 18 A set $E_d(T_v)$ of edges is called a feasible cut of a tree T_v rooted at v with respect to d if there is one and only one edge $e \in E(T_v)$ on every path from v to every leaf $t_j \in Y \cap T_v$, where $d(e) > 0$. A minimum cost cut $E_d^*(T_v)$ is a feasible cut whose cost $C(T_v) = C(E_d^*(T_v)) = \sum_{e \in E_d^*(T_v)} c(e)$ is minimum.

The main idea of the algorithm to find a minimum cost cut is as follows. For a given length z, we firstly call the preprocessing algorithm $(T', Y', E_{del}) = Preprocess(T, Y, CC, CF, chain, Layer, w, z)$ to delete the paths whose length are no less than z. Then for the tree T', we determine a minimum cost cut for each subtree T_v when $v \in V^* \cup \{v_1\}$ in a decreasing order of Layers. For $v \in V^* \cup \{v_1\}$, let $CC(v) = \{v_{h_1}, v_{h_2}, \ldots, v_{h_p}\}$ be the set of critical children of v, and let $\bar{T}_{v_h} = chain(v_h) \cup T_{v_h}$ be a branch of T_v rooted at v_h. Then we can divided T_v into the union of all the branches $\bar{T}_{v_h}$, that is, $T_v = \bigcup_{v_h \in CC(v)} \bar{T}_{v_h}$. Based on this structure and the definition of a minimum cost cut, we can calculate $C(\bar{T}_{v_h}) = \min\{C(chain(v_h)), C(T_{v_h})\}$. Hence, $C(T_v) = \sum_{v_h \in CC(v)} C(\bar{T}_{v_h})$ is the sum of the minimum costs $C(\bar{T}_{v_h})$ for all the branches $\bar{T}_{v_h}$, $v_h \in CC(v)$.

Obviously, we need to transverse the tree from bottom to up to find a minimum cost cut. Then we can conclude that

Lemma 19 *Algorithm 4 can find a minimum cost cut in $O(n)$ time.*

Given an example in Fig. 7, we call Algorithm 4 in details as follows.

$$\begin{aligned}
V^2 &:= \{v_5\}, C(T_{v_5}) := C(\bar{T}_{v_6}) + C(\bar{T}_{v_7}) = \min\{9, +\infty\} + \min\{16, +\infty\} = 25;\\
V^1 &:= \{v_4, v_9\}, C(T_{v_4}) := C(\bar{T}_{v_5}) + C(\bar{T}_{v_8}) = 10 + 15 = 25;\\
C(T_{v_9}) &:= C(\bar{T}_{v_{11}}) + C(\bar{T}_{v_{13}}) = \min\{1, +\infty\} + \min\{13, +\infty\} = 1 + 13 = 14;\\
V^0 &:= \{v_1\}, C(T_{v_1}) := C(\bar{T}_{v_4}) + C(\bar{T}_{v_9}) + C(\bar{T}_{v_{17}}) = 4 + 7 + 15 = 26.
\end{aligned}$$

Therefore, $C(T) := C(T_{v_1}) = 26$ and a minimum cost cut of T given in Fig. 7 is $E_d^*(T) := \{(v_2, v_3), (v_1, v_9), (v_1, v_{15})\}$.

Algorithm 4 $(E_d^*(T), C(T)) = MinCut(T, Y, CC, CF, chain, Layer, c, w, d, z)$

Require: A tree $T = (V, E)$, the set Y of leaves, the set CC of critical children, the set CF of critical fathers and the chains, the Layer of vertices; a cost vector c, two edge weight vectors w, d and the length z of shortest path;

Ensure: The minimum cost cut $E_d^*(T)$ of the tree and the relative cost $C(T)$.

Call $(T, Y, E_{del}) = Preprocess(T, Y, CC, CF, chain, Layer, w, z)$.
for $t \in Y$ **do**
  $T_t := \emptyset, C(T_t) := +\infty, E_d^*(T_t) := \emptyset$.
end for
Let $\beta := \max_{v \in V} Layer(v)$.
for $j = \beta - 1 : -1 : 0$ **do**
  let $V^j := \{v \in V^* \cup \{v_1\} | Layer(v) = j\}$.
  while $V^j \neq \emptyset$ **do**
    choose $v \in V^j$. Determine $CC(v) := \{v_{h_1}, v_{h_2}, \ldots, v_{h_p}\}$, where $p := degree(v) - 1$ if $v \neq v_1$ and $p := degree(v)$ if $v = v_1$.
    for $v_h \in CC(v)$ **do**
      let $\bar{T}_{v_h} := chain(v_h) \cup T_{v_h}, C(chain(v_h)) = \min_{e \in chain(v_h), d(e) > 0} c(e) := c(\hat{e}_h)$.
      if $C(chain(v_h)) \leq C(T_{v_h})$ **then**
        $C(\bar{T}_{v_h}) := C(chain(v_h)), E_d^*(\bar{T}_{v_h}) := \{\hat{e}_h\}$;
      else
        $C(\bar{T}_{v_h}) := C(T_{v_h}), E_d^*(\bar{T}_{v_h}) := E_d^*(T_{v_h})$.
      end if
    end for
    Calculate $C(T_v) := \sum_{v_h \in CC(v)} C(\bar{T}_{v_h}), E_d^*(T_v) := \bigcup_{v_h \in CC(v)} E_d^*(\bar{T}_{v_h})$.
    Update $V^j := V^j \backslash \{v\}$.
  end while
end for
Return $E_d^*(T) := E_d^*(T_{v_1}), C(T) := C(T_{v_1})$.

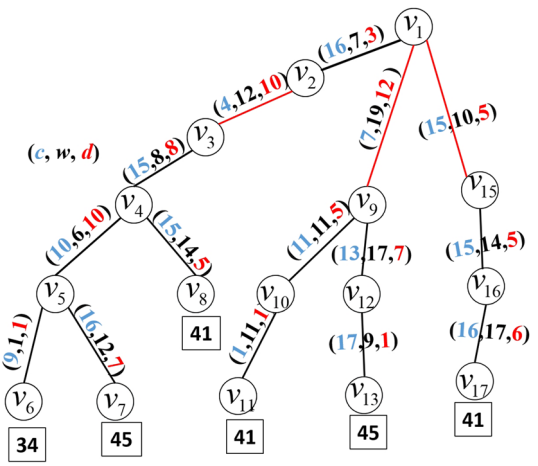

Fig. 7 The minimum cost cut is $\{(v_2, v_3), (v_1, v_9), (v_1, v_{15})\}$

4.2 A primal dual algorithm to solve the problem (MSPIT$_1$)

In this subsection, we first present the main idea of the primal dual algorithm [14, Chapter 5–6] to solve the problem (**MSPIT**$_1$), then describe the algorithm in details and analyze its time complexity.

The model (1) of the problem (**MSPIT**$_1$) is equivalent to the following model.

$$
(D)\quad \begin{array}{l} \max\ z \\ s.t.\ \ w(P_i)+r(P_i)\geq z, t_i\in Y \\ \qquad \sum_{e\in E} c(e)r(e)\leq M, \\ \qquad 0\leq r(e)\leq d(e), e\in E, \\ \qquad z\geq 0. \end{array} \tag{20}
$$

Obviously, the problem (20) is a linear programming problem with $l+n$ constraints and n variables. Now we present the main idea of the primal dual algorithm [14, Chapter 5-6]. Firstly, we consider the problem (20) as a dual problem **(D)**. Given a dual feasible solution $\pi^k=(r^k,z^k)$ of the problem **(D)**, we can determine a set J^k of admissible rows where equality constraints hold in the constraint conditions of **(D)** for π^k. Based on the set J^k, we can get the dual of the restricted primal problem, denoted by (**DRP**^{k}). Secondly, we solve the problem (**DRP**^{k}) by finding a minimum cost cut and obtain its optimal solution $\bar{\pi}^k=(\bar{r}^k,\bar{z}^k)$. If the optimal objective value $\bar{z}^k$ of (**DRP**^{k}) is positive, then we determine the adjustment amount θ^k to obtain a better dual feasible solution $\pi^{k+1}=\pi^k+\theta^k\bar{\pi}^k$ whose objective value $z^{k+1}=z^k+\theta\bar{z}^k$ is larger than z^k and continue to find a set J^{k+1} for π^{k+1}. The above iteration terminates until the optimal objective value $\bar{z}^k$ of **(DRP)** is zero, we obtain an optimal dual solution π^k of **(D)**.

1. **Transform the problem** (20) **as a standard dual problem** (**D**$_s$), where all the constraints are in the form of "less than and equal to" and all the variables are unconstrained.

$$
(D_s)\quad \begin{array}{l} \max\ z \\ s.t.\ \ -r(P_i)+z\leq w(P_i), t_i\in Y \\ \qquad \sum_{e\in E} c(e)r(e)\leq M, \\ \qquad r(e)\leq d(e), e\in E \\ \qquad -r(e)\leq 0, e\in E \\ \qquad -z\leq 0. \end{array}
$$

2. **Determine an initial dual feasible solution** (r^k,z^k) **when** $k=0$.

Let $P_{i^*}=\arg\min_{t_i\in Y}\{w(P_i)\}$ be the shortest path of tree T, $w(P_{i1^*})=\min\{w(P_i)|w(P_i)>w(P_{i^*})\}$ and $L^0=\min\{w(P_{i1^*}),L_{max}\}$, where $L_{max}=R(T_d)$. Initialize $g^0=0, r^0=\mathbf{0}, z^0=w(P_{i^*})$.

(1) Call $(E_d^*(T),C(T))=MinCut(T,Y,CC,CF,chain,Layer,c,w,d,z^0)$ to determine a minimum cost cut $E_d^*(T)$. Let $d(\widetilde{e})=\min_{e\in E_d^*(T)} d(e)$ and $\theta=\min\{\frac{M}{C(T)}, d(\widetilde{e}), L^0-z^0\}$. Let $\bar{r}^0(e)=\theta$ for $e\in E_d^*(T)$ and $\bar{r}^0(e)=0$ for $e\notin E_d^*(T)$. Then there are three possible cases below.

(a) If $\theta=\frac{M}{C(T)}$, then the cost M is used up and $z^0=z^0+\theta$ is the length of the shortest path with respect to an optimal upgrade scheme $r^0=r^0+\bar{r}^0$ and the upgrade cost $g^0:=g^0+\theta C(T)=M$.

(b) If $\theta=L^0-z^0$, update $z^0=z^0+\theta$, then $z^0=L^0$ is the length of the shortest path with a feasible upgrade scheme $r^0=r^0+\bar{r}^0$, and hence (r^0,z^0) is a dual feasible solution.

(c) If $\theta=d(\widetilde{e})$, then update d by $d(e)=d(e)-\theta$ for $e\in E_d^*(T)$. Go back to (1) to find the next minimum cost cut by the new weight d and $z^0=z^0+\theta$. Such a process terminates until case (b) or (c) occurs.

Therefore, we can either find a dual feasible solution (r^0, z^0) or obtain a dual optimal solution (r^0, z^0).

3. **Determine the sets of admissible rows according to the dual feasible solution** (r^k, z^k).

$$J_1^k = \{t_i \in Y | -r^k(P_i) + z^k = w(P_i)\}, \tag{21}$$

$$J_2^k = \begin{cases} \{1\}, & if \ \sum_{e\in E} c(e)r^k(e) = M; \\ \emptyset, & Otherwise. \end{cases} \tag{22}$$

$$J_3^k = \{e_j | r^k(e_j) = d(e_j)\}, \tag{23}$$

$$J_4^k = \{e_j | r^k(e_j) = 0\}, \tag{24}$$

$$J_5^k = \begin{cases} \{1\}, & if \ z^k = 0; \\ \emptyset, & if \ z^k \neq 0. \end{cases} \tag{25}$$

4. **Generate the problem (DRP^k) based on the sets of admissible rows and the relationship between the problems (D) and (DRP^k)**. Then solve the problem (**DRP**^{k}) and obtain its optimal solution $(\bar{r}^k, \bar{z}^k)$.

$$\textbf{(DRP}^k\textbf{)} \quad \begin{array}{lll} \max & \bar{z}^k & \\ s.t. & -\bar{r}^k(P_i) + \bar{z}^k \leq 0, & t_i \in J_1^k, \\ & \bar{r}^k(e_j) \leq 0, & e_j \in J_3^k, \\ & -\bar{r}^k(e_j) \leq 0, & e_j \in J_4^k, \\ & \bar{r}^k(e_j) \leq 1, & j = 2, \ldots, n, \\ & \bar{z}^k \leq 1. & \end{array}$$

An optimal solution to the problem (**DRP**^{k}) is not unique. We try to find an optimal solution $(\bar{r}^k, \bar{z}^k)$ for the set $P_1^k = \{P_i = P_{v_1,t_i} | t_i \in J_1^k\}$ of paths in the problem (**DRP**^{k}) by a minimum cost cut $E^*_{d^k}(P_1^k)$ on the edges with $d^k(e) > 0$. Let $C(P_1^k) = C(E^*_{d^k}(P_1^k))$ for simplicity. Then we can obtain the following lemma.

Lemma 20 *If* $C(P_1^k) > 0$, *then*

$$\bar{z}^k = 1, \ \ \bar{r}^k(e) = \begin{cases} 1, \ if \ e \in E^*_{d^k}(P_1^k) \\ 0, \ \ otherwise \end{cases} \tag{26}$$

*is an optimal solution of the problem (**DRP**^{k}).*

Proof Notice that $J_3^k = \{e_j | r^k(e_j) = d(e_j)\}$, then $d^k(e_j) = d(e_j) - r^k(e_j) = 0$ and $\bar{r}^k(e_j) \leq 0$ for edge $e_j \in J_3^k$. Moreover, the minimum cost cut $E^*_{d^k}(P_1^k)$ contains the edges with $d^k(e) > 0$. Hence, $J_3^k \cap E^*_{d^k}(P_1^k) = \emptyset$ and $\bar{r}^k$ given by (26) satisfies the last four constraints of the problem (**DRP**^{k}). It follows from the definition of the minimum cost cut that there is one and only one edge of $E^*_{d^k}(P_1^k)$ on each path $P_i \in P_1^k$. Then $\bar{r}^k(P_i) = \sum_{e\in P_i} \bar{r}^k(e) = \sum_{e\in P_i \cap E^*_{d^k}(P_1^k)} \bar{r}^k(e) = 1 = \bar{z}^k$. Therefore, $(\bar{r}^k, \bar{z}^k)$ is a feasible solution of the problem (**DRP**^{k}) with the maximum objective value $\bar{z}^k = 1$, and hence it is also an optimal solution. □ □

5. Determine the adjustment amount θ^k **of the cut** $E^*_{d^k}(P_1^k)$.

Let

$$\theta_1^k = \min_{t_i \notin J_1^k, P_i \cap E^*_{d^k}(P_1^k) = \emptyset} \{\theta_1(t_i) := w(P_i) + r^k(P_i) - z^k\}; \tag{27}$$

$$\theta_2^k = \frac{M - g^k}{C(P_1^k)}; \tag{28}$$

$$\theta_3^k = \min_{e_j \notin J_3^k, e_j \in E^*_{d^k}(P_1^k)} \{d(e_j) - r^k(e_j)\}. \tag{29}$$

For $\theta_4^k = \min_{e_j \notin J_4^k, -\bar{r}_j^k(e_j) > 0} \frac{r^k(e_j)}{-\bar{r}^k(e_j)}$, we have $-\bar{r}^k(e_j) \leq 0$ by (26) before reaching the optimality, and thus $\theta_4^k = +\infty$. Similarly, for $\theta_5^k = \frac{z^k}{-\bar{z}^k} > 0$, we have $\bar{z}^k = 1 > 0$ before reaching the optimality. Thus $-\bar{z}^k < 0$, and $\theta_5^k = +\infty$.

Hence $\theta^k = \min\{\theta_1^k, \theta_2^k, \theta_3^k\}$ is the the adjustment amount of the cut $E^*_{d^k}(P_1^k)$. If $\theta^k = \theta_1^k$, then the minimum length of the paths will reach to $z^{k+1} = z^k + \theta_1^k$ after this iteration and at least one more path will be added to the set P_1^{k+1} of paths. Then we need to find a new minimum cost cut on the new set $P_1^{k+1} = \{P_i | w(P_i) + r^{k+1}(P_i) = z^{k+1}\} = P_1^k \cup P(\theta_1^k)$ of paths, where $P(\theta_1^k) = \{P_i | w(P_i) + r^k(P_i) - z^k = \theta_1^k, t_i \notin J_1^k, P_i \cap E^*_{d^k}(T) = \emptyset\}$. If $\theta^k = \theta_2^k$, then the total cost M has been used up and the number θ_2^k of units can be distributed to the current minimum cost cut $E^*_{d^k}(P_1^k)$, which implies the iteration terminates. If $\theta^k = \theta_3^k$, then the upgrade amount of at least one edge e_j in the cut $E^*_{d^k}(P_1^k)$ will achieve its upper-bound $d(e_j)$ and we have $d^{k+1}(e_j) = d^k(e_j) - \theta_3^k = d^k(e_j) - d(e_j) + r^k(e_j) = 0$. In this case, we can update $J_3^{k+1} = J_3^k \cup \{e_j \in E^*_{d^k}(T) | d^{k+1}(e_j) = 0\}$.

6. Update a better dual feasible solution.

Lemma 21 *Suppose (r^k, z^k) is a dual feasible solution of **(D)** and $(\bar{r}^k, \bar{z}^k)$ is a dual optimal solution of **(DRP**^{k}**)** given by (26). Let $r^{k+1} = r^k + \theta^k \bar{r}^k$ and $z^{k+1} = z^k + \theta^k$. Then (r^{k+1}, z^{k+1}) is a better dual feasible solution of **(D)** .*

Proof For the edges $e_j \in E^*_{d^k}(P_1^k)$, we have $0 \leq r^k(e_j) \leq r^{k+1}(e_j) = r^k(e_j) + \theta^k \leq r^k(e_j) + \theta_3^k \leq r^k(e_j) + d(e_j) - r^k(e_j) = d(e_j)$. For the edges $e_j \notin E^*_{d^k}(P_1^k)$, we have $0 \leq r^{k+1}(e_j) = r^k(e_j) \leq d(e_j)$.

$$\begin{aligned}\sum_{e \in E} c(e) r^{k+1}(e) &= \sum_{e \in E^*_{d^k}(P_1^k)} c(e)(r^k(e) + \theta^k) + \sum_{e \notin E^*_{d^k}(P_1^k)} c(e) r^k(e) \\ &= \sum_{e \in E} c(e) r^k(e) + \theta^k C(P_1^k) \leq g^k + (M - g^k) = M.\end{aligned}$$

$$-r^{k+1}(P_i) + z^{k+1} = -r^k(P_i) - \theta^k + z^k + \theta^k = -r^k(P_i) + z^k \leq w(P_i).$$

Furthermore, $z^{k+1} = z^k + \theta^k > z^k$ as $\theta^k > 0$. As a conclusion, (r^{k+1}, z^{k+1}) is a better dual feasible solution of **(D)**. □ □

7. Satisfy the termination condition.

The above iteration terminates until $\theta^k = \theta_2^k$ in some iteration. In this case, the total cost M is used up, and we can find an optimal solution of the dual problem.

We summarize the steps above in Algorithm 5, and analyze its time complexity.

Theorem 22 *Algorithm 5 can solve the problem (**MSPIT**$_1$) in $O(n^2)$ time.*

Proof In Lines 1–4, the algorithms of Preprocess and MinCut can all be completed in $O(n)$ time. The number of calling the MinCut Algorithm 4 is upper-bounded by $n - 1$ because

there is at least one upgrade $r^{k+1}(e)$ achieving its upper-bound value $d(e)$ when finding a dual feasible solution in Lines 5–16 and in Lines 17–35 in the worst case. Hence, Algorithm 5 can be completed in $O(n^2)$ time. □ □

Algorithm 5 The primal dual algorithm to the problem (**MSPIT**$_1$)

Require: A tree $T = (V, E)$, the set Y of leaves; the set CC of critical children, the set CF of critical fathers and the chains; the Layer of vertices, three edge vectors w, u, c and the maximum cost M.
Ensure: An optimal upgrade scheme r^k and the length z^k of the shortest path.
1: Calculate $d := u - w$ and $L_{max} := R(T_d)$.
2: Call $(T, Y, E_{del}) := Preprocess(T, Y, CC, CF, chain, Layer, w, L_{max})$.
3: Initialization: $k := 0$, $w(P_{i^*}) := \min_{t_i \in Y}\{w(P_i)\}$, $w(P_{i1^*}) := \min\{w(P_i)|w(P_i) > w(P_{i^*}), t_i \in Y\}$, $L^0 := \min\{w(P_{i1^*}), L_{max}\}$, $d^k := d$, $r^k := 0$, $g^k := 0$, $z^k := w(P_{i^*})$.
4: Call $(E^*_{d^k}(T), C(T)) := MinCut(T, Y, CC, CF, chain, Layer, c, w, d^k, z^k)$.
5: **while** $g^k < M$ **and** $z^k < L^0$ **do**
6: Let $d^k(\widetilde{e}) := \min_{e \in E^*_{d^k}(T)} d^k(e)$, $\theta^k := \min\{\frac{M-g^k}{C(T)}, d^k(\widetilde{e}), L^0 - z^k\}$.
7: Update $g^{k+1} := g^k + \theta^k C(T)$, $z^{k+1} := z^k + \theta^k$, $r^{k+1}(e) := r^k(e) + \theta^k$ and $d^{k+1}(e) := d^k(e) - \theta^k$ for $e \in E^*_{d^k}(T)$, $r^{k+1}(e) := r^k(e)$ and $d^{k+1}(e) := d^k(e)$ for $e \notin E^*_{d^k}(T)$; $w^{k+1} := w + r^{k+1}$ and $k := k + 1$,
8: **if** $z^k < L^0$ **then**
9: call $(E^*_{d^k}(T), C(T)) := MinCut(T, Y, CC, CF, chain, Layer, c, w^k, d^k, z^k)$.
10: **end if**
11: **end while**
12: **if** $g^k = M$ **then**
13: output an optimal solution r^k and the length z^k of the shortest path, **Return**.
14: **else if** $z^k = L^0$ **then**
15: let $J_1^k := \{t_i \in Y| - r^k(P_i) + z^k = w(P_i)\}$, $J_3^k := \{e_i \in E|r^k(e_i) = d(e_i)\}$.
16: **end if**
17: **while** $g^k < M$ **do**
18: **if** $z^k = L_{max}$ **then**
19: z^k reaches the maximum length of the shortest path, and output an optimal solution r^k and z^k, **Return**.
20: **else**
21: let $P_1^k := \{P_i|t_i \in J_1^k\}$ be a subset of the tree T including all the paths for $t_i \in J_1^k$.
22: Call $(E^*_{d^k}(P_1^k), C(P_1^k)) := MinCut(P_1^k, J_1^k, CC, CF, chain, Layer, c, w^k, d^k, z^k)$.
23: **if** $C(P_1^k) > 0$ **then**
24: The optimal solution of the problem (**DRP**) is $\bar{z}^k := 1, \bar{r}^k(e) := 1, e \in E^*_{d^k}(P_1^k), \bar{r}^k(e) := 0, otherwise$.
25: Calculate $\theta^k := \min\{\theta_1^k, \theta_2^k, \theta_3^k\}$, where $\theta_1^k, \theta_2^k, \theta_3^k$ are defined as in (27)-(29).
26: Update $g^{k+1} := g^k + \theta^k C(P_1^k)$, $z^{k+1} := z^k + \theta^k$, $r^{k+1} := r^k + \theta^k \bar{r}^k$, $d^{k+1} := d^k - \theta^k \bar{r}^k$, $w^{k+1} := w + r^{k+1}$, $J_3^{k+1} := J_3^k \cup \{e_j \in E^*_{d^k}(P_1^k)|d^{k+1}(e_j) = 0\}$.
27: **if** $\theta^k = \theta_2^k$ **then**
28: $g^{k+1} := M$, the cost M is used up and output an optimal solution r^{k+1} and z^{k+1}.
29: **else if** $\theta^k = \theta_1^k$ **then**
30: let $J_1^{k+1} := J_1^k \cup \{t_j|\theta_1(t_j) = \theta_1\}$.
31: **end if**
32: Update $k := k + 1$.
33: **end if**
34: **end if**
35: **end while**

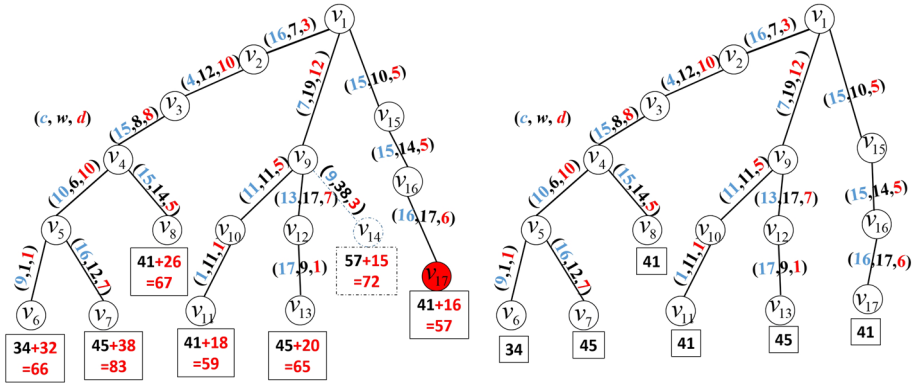

Fig. 8 For the left tree $L_{max} = 57$, delete the $chain(v_{14})\backslash v_9 = \{v_{14}\}$

4.3 An example to show the computing process of Algorithm 5

For the better understanding of Algorithm 5, Example 3 is given to show the detailed computing process.

Example 3 For the left tree in Fig. 8, $V := \{v_1, \ldots, v_{17}\}$, $E := \{e_2, \ldots, e_{17}\}$, $t_1 := v_6$, $t_2 := v_7$, $t_3 := v_8$, $t_4 := v_{11}$, $t_5 := v_{13}$, $t_6 := v_{14}$, $t_7 := v_{17}$, $M := 150$.

$$
\begin{aligned}
c &:= (16, 4, 15, 10, 9, 16, 15, 7, 11, 1, 13, 17, 9, 15, 15, 16),\\
w &:= (7, 12, 8, 6, 1, 12, 14, 19, 11, 11, 17, 9, 38, 10, 14, 17),\\
u &:= (10, 22, 16, 16, 2, 19, 19, 31, 16, 12, 24, 10, 41, 15, 19, 23),\\
d &:= u - w = (3, 10, 8, 10, 1, 7, 5, 12, 5, 1, 7, 1, 3, 5, 5, 6).
\end{aligned}
$$

1. Call the preprocess algorithm. (Lines 1–2)

We have $L_{max} := 57$. Call the preprocess algorithm, we have $w(v_1, v_{14}) := 57$, then $b(v_{14}) := \{v_{11}, v_{13}, v_{14}\}$ and $b^*(v_{14}) := \{v_{14}\}$. So we delete $chain(v_{14})\backslash v_9 = \{v_{14}\}$ and obtain the right tree in Fig. 8.

2. Find a dual feasible solution of the problem (D). (Lines 3–16)

Initialization: $k := 0$, $w(P_{i^*}) := 34$, $w(P_{i1^*}) := 41$, $L^0 := 41$, $d^0 := d$, $r^0 := (0, \ldots, 0)$, $g^0 := 0$, $z^0 := 34$.

Call $(E^*_{d^k}(T), C(T)) := MinCut(T, Y, CC, CF, chain, Layer, c, w, d^0, z^0)$, and obtain $E^*_{d^0}(T) = \{e_3\}$, $C(T) = 4$. In Line 6, $d^0(\widetilde{e}) := 10$, $\theta^0 := \min\{\frac{M-g^0}{C(T)}, d^0(\widetilde{e}), L^0 - z^0\} = \min\{\frac{400}{4}, 10, 7\} = 7$, shown in the left tree in Fig. 9.

Update $g^1 := 28$, $z^1 := 34 + 7 = 41$, $r^1(e_3) := 7$, $d^1(e_3) := 10 - 7 = 3$ and $k := 1$. Please see the right tree in Fig. 9. In this case, $z^1 = L^0$, let $J^1_1 = \{t_i \in Y | - r^1(P_i) + z^1 = w(P_i)\} := \{t_1, t_4, t_6\}$, $J^1_3 = \{e_i \in E | r^k(e_i) = d(e_i)\} := \emptyset$.

3. Find a dual optimal solution of the problem (D). (Lines 17–35)

The first iteration, $k := 1$. Let $P^1_1 = \{P_i | t_i \in J^1_1\} := \{P_1, P_4, P_6\}$. Call the Algorithm MinCut to obtain $E^*_{d^1}(P^1_1) := \{e_3, e_{11}, e_{15}\}$, $C(P^1_1) := 20$. Then $\bar{z}^1 := 1$, $\bar{r}^1(e_3) := 1$, $\bar{r}^1(e_{11}) := 1$, $\bar{r}^1(e_{15}) := 1$, $\bar{r}^1(e) := 0$, $e \notin E^*_{d^1}(P^1_1)$. Calculate $\theta^1_1 := 4$, $\theta^1_2 := 18.6$, $\theta^1_3 := 1$, $\theta^1 := \min\{\theta^1_1, \theta^1_2, \theta^1_3\} := 1$. Update $g^2 = g^1 + \theta^1 C(P^1_1) := 48$, $z^2 = z^1 + \theta^1 := 42$, $r^2(e_3) := 8$, $r^2(e_{11}) := 1$, $r^2(e_{15}) := 1$, $r^2(e) := 0$ for $e \notin E^*_{d^1}(P^1_1)$;

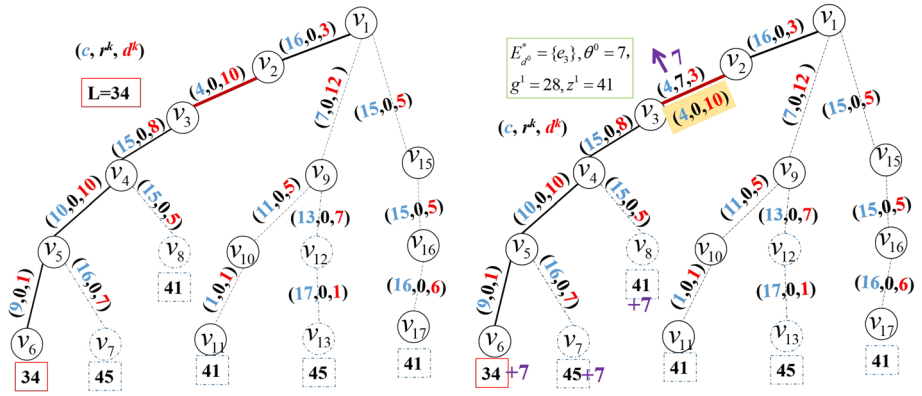

Fig. 9 A dual feasible solution

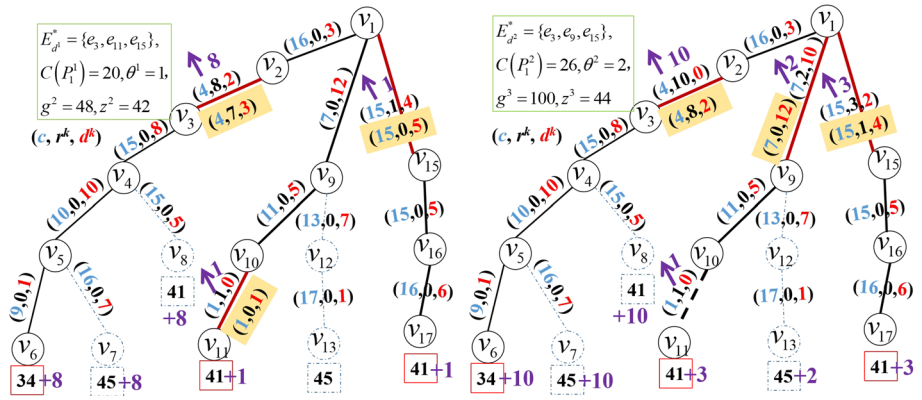

Fig. 10 The first and second iterations

$d^2(e_3) := 2$, $d^2(e_{11}) := 0$, $d^2(e_{15}) := 4$, $d^2(e) := d^1(e) = d(e)$ for $e \notin E^*_{d^1}(P_1^1)$; $J_3^2 = J_3^1 \cup \{e_j \in E^*_{d^1}(P_1^1) | d^2(e_j) = 0\} := \{e_{11}\}$; $k := 2$. Please see the left tree in Fig. 10.

The second iteration, $k := 2$. Let $P_1^2 := \{P_1, P_4, P_6\}$. Call the Algorithm MinCut to obtain $E^*_{d^2}(P_1^2) := \{e_3, e_9, e_{15}\}$, $C(P_1^2) := 26$. Then $\bar{z}^2 := 1, \bar{r}^2(e_3) := 1, \bar{r}^2(e_9) := 1, \bar{r}^2(e_{15}) := 1, \bar{r}^2(e) := 0, e \notin E^*_{d^2}(P_1^2)$. Calculate $\theta_1^2 := +\infty, \theta_2^2 := 13\frac{8}{13}, \theta_3^2 := 2$, $\theta^2 := 2$. Update $g^3 := 100, z^3 := 44, r^3(e_3) := 10, r^3(e_9) := 2, r^3(e_{15}) := 3, r^3(e_{11}) := 1$, $r^3(e) := 0$ for other e; $d^3(e_3) := 0, d^3(e_9) := 10, d^3(e_{15}) := 2, d^3(e_{11}) := 0, d^3(e) := d(e)$ for other e; $J_3^3 := \{e_3, e_{11}\}$; $k := 3$. Please see the right tree in Fig. 10.

The results of the following third and fourth iterations are shown in the left and right tree in Fig. 11, respectively. Table 3 shows the results of the four iterations in details.

The value θ^4 is obtained at θ_2^4, which implies that the cost $M = 150$ is used up and the iteration terminates. Then we output the optimal length $z^5 := 45\frac{19}{32}$ of the shortest path and an optimal upgrade scheme

$$r^5 := (0, 10, 0, \frac{19}{32}, 1, 0, 0, 3\frac{19}{32}, 0, 1, 0, 0, 0, 4\frac{19}{32}, 0, 0).$$

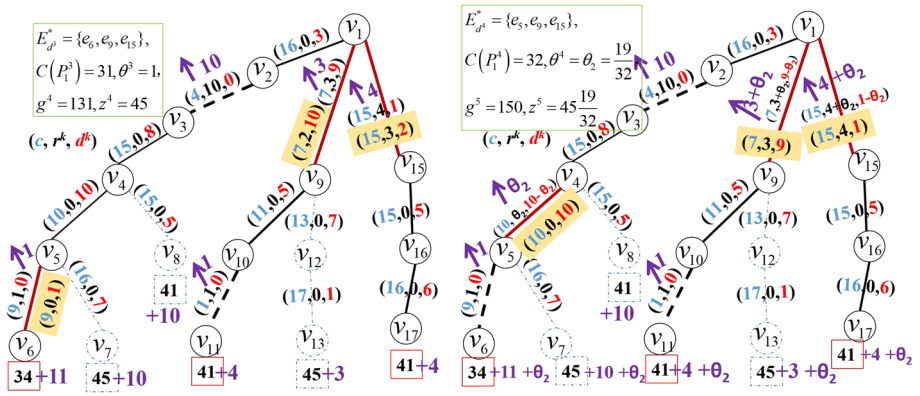

Fig. 11 The third and fourth iterations

Table 3 The results for the detailed 4 iterations in Example 3 by Algorithm 5

k	1	2	3	4
P_1^k	$\{P_1, P_4, P_6\}$	$\{P_1, P_4, P_6\}$	$\{P_1, P_4, P_6\}$	$\{P_1, P_4, P_6\}$
$E^*_{d^k}(P_1^k)$	$\{e_3, e_{11}, e_{15}\}$	$\{e_3, e_9, e_{15}\}$	$\{e_6, e_9, e_{15}\}$	$\{e_5, e_9, e_{15}\}$
$C(P_1^k)$	20	26	31	32
θ_1^k	4	$+\infty$	7	6
θ_2^k	18.6	$13\frac{8}{13}$	$9\frac{21}{31}$	$\frac{19}{32}$
θ_3^k	1	2	1	1
θ^k	1	2	1	$\frac{19}{32}$
g^{k+1}	48	100	131	150
z^{k+1}	42	44	45	$45\frac{19}{32}$
r^{k+1}	(0, **8**, 0, 0, 0, 0, 0, 0, 0, **1**, 0, 0, 0, **1**, 0, 0)	(0, **10**, 0, 0, 0, 0, 0, 0, **2**, 0, 1, 0, 0, 0, **3**, 0, 0)	(0, 10, 0, 0, **1**, 0, 0, **3**, 0, 1, 0, 0, 0, **4**, 0, 0)	(0, 10, 0, $\frac{19}{32}$, 1, 0, 0, $\mathbf{3}\frac{19}{32}$, 0, 1, 0, 0, 0, $\mathbf{4}\frac{19}{32}$, 0, 0).
d^{k+1}	(3, **2**, 8, 10, 1, 7, 5, 12, 5, **0**, 7, 1, 3, **4**, 5, 6)	(3, **0**, 8, 10, 1, 7, 5, **10**, 5, 0, 7, 1, 3, **2**, 5, 6)	(3, 0, 8, 10, **0**, 7, 5, **9**, 5, 0, 7, 1, 3, **1**, 5, 6)	(3, 0, 8, $\mathbf{9}\frac{\mathbf{13}}{\mathbf{32}}$, 0, 7, 5, $\mathbf{8}\frac{\mathbf{13}}{\mathbf{32}}$, 5, 0, 7, 1, 3, $\frac{\mathbf{13}}{\mathbf{32}}$, 5, 6)
J_3^{k+1}	$\{e_{11}\}$	$\{e_3, e_{11}\}$	$\{e_3, e_6, e_{11}\}$	$\{e_3, e_6, e_{11}\}$

5 Solve the problem (MCSPIT$_1$)

In this section, we solve the problem (**MCSPIT**$_1$) under weighted l_1 norm, which can be formulated as in (2). We aim to upgrade some edges to minimize the total cost under weighted l_1 norm on the premise that the length of the shortest path of the tree is lower-bounded by a given value L.

Similar to Algorithm 5, replace (28) by $\theta_2^k = L - \max_{t_i \in J_1^k} w^k(v_1, t_i)$, which is the minimum length of the paths to be upgraded. Then we can obtain the relevant primal dual algorithm to the problem (**MCSPIT**$_1$) in Algorithm 6.

Algorithm 6 The primal dual algorithm to the problem (**MCSPIT**$_1$)

Require: A tree $T = (V, E)$, the set Y of leaves; the set CC of critical children, the set CF of critical fathers and the chains; the Layer of vertices, three edge vectors w, u, c and the given length L.
Ensure: An optimal upgrade scheme r^k and the relevant upgrade cost g^k.
1: Calculate $d := u - w$ and $L_{max} := R(T_d)$.
2: Call $(T, Y, E_{del}) := Preprocess(T, Y, CC, CF, chain, Layer, w, L_{max})$.
3: Initialization: $k := 0$, $w(P_{i*}) := \min_{t_i \in Y}\{w(P_i)\}$, $w(P_{i1*}) := \min\{w(P_i)|w(P_i) > w(P_{i*}), t_i \in Y\}$, $d^k := d$, $r^k := 0$, $w^k := w$, $g^k := 0$, $z^k := w(P_{i*})$.
4: **if** $L > L_{max}$ **then**
5: Output "The upgraded length of the shortest path is impossible to be L, and its maximum length is L_{max}". **Return.**
6: **end if**
7: Initialize $J_1^k := \{t_i \in Y | w(P_i) = w(P_{i*})\}$, $J_3^k := \emptyset$.
8: **while** $z^k < L$ **do**
9: let $P_1^k := \{P_i | t_i \in J_1^k\}$ be a subset of the tree T including all the paths for $t_i \in J_1^k$.
10: Call $(E^*_{d^k}(P_1^k), C(P_1^k)) := MinCut(P_1^k, J_1^k, CC, CF, chain, Layer, c, w^k, d^k, z^k)$.
11: **if** $C(P_1^k) > 0$ **then**
12: The optimal solution of the problem **(DRP)** is $\bar{z}^k := 1$, $\bar{r}^k(e) := 1$, $e \in E^*_{d^k}(P_1^k)$, $\bar{r}^k(e) := 0, otherwise.$
13: Calculate $\theta^k := \min\{\theta_1^k, \theta_2^k, \theta_3^k\}$, where θ_1^k, θ_3^k are defined as in (27), (29), $\theta_2^k := L - \max_{t_i \in J_1^k} w^k(v_1, t_i)$.
14: Update $g^{k+1} := g^k + \theta^k C(P_1^k)$, $z^{k+1} := z^k + \theta^k$, $r^{k+1} := r^k + \theta^k \bar{r}^k$, $d^{k+1} := d^k - \theta^k \bar{r}^k$, $w^{k+1} := w + r^{k+1}$, $J_3^{k+1} := J_3^k \cup \{e_j \in E^*_{d^k}(P_1^k) | d^{k+1}(e_j) = 0\}$.
15: **if** $\theta^k = \theta_1^k$ **then**
16: let $J_1^{k+1} := J_1^k \cup \{t_j | \theta_1^k(t_j) = \theta_1^k\}$.
17: **end if**
18: Update $k := k + 1$.
19: **end if**
20: **end while**
21: Output an optimal solution r^k and its cost g^k.

Corollary 23 *Algorithm 6 can solve the problem (**MCSPIT**$_1$) in $O(n^2)$ time.*

6 Computational experiments

Now we present computational experiments of Algorithms 2, 3, 5 and 6. 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 10 classes of random trees, with the number n of vertices varying from 100 to 20000. For each class, we randomly generate 30 instances. For each instance, we use two numbers to show the structure property of a randomly generated tree. One is the max number λ of edges in each path from the root v_1 to a leaf and the other is the number l of leaves. Let λ_{ave} and l_{ave} be the average values of λ and l, respectively. We randomly generated three vectors w, u, c satisfying $w < u$ which means that $d = u - w > 0$. For each randomly generated tree, under weighted l_1 norm, we first solve the problem (**MCSPIT**$_1$) with a randomly generated integer L in the range $[L_{min}, L_{max}]$ by Algorithm 6, where $L_{min} = R(T_0)$ and $L_{max} = R(T_d)$, respectively. Next, solve the problem (**MCSPIT**$_1$) with $L = L_{max}$ by Algorithm 6 to obtain the cost M_{max}. Then solve the problem (**MSPIT**$_1$) by Algorithm 5 with a randomly generated integer M in the range $[1, M_{max}]$. Similarly, we solve the problems (**MCSPIT**$_{u1}$) and (**MSPIT**$_{u1}$) under unit l_1 norm

Table 4 Performances of Algorithms 2, 3, 5 and 6

n	100	200	500	1000	3000	5000	7000	10,000	15,000	20,000
λ_{ave}	7.03	16.47	43.27	78.50	245.47	455.27	613.43	792.73	1277.50	1619.10
l_{ave}	50.80	60.77	75.17	85.07	93.63	70.63	63.67	91.70	99.97	95.83
M_{1ave} $(\times 10^3)$	0.06	0.12	0.89	3.04	27.77	78.74	232.93	288.42	662.29	997.48
T_3	0.02	0.03	0.07	0.17	1.52	8.72	24.12	42.53	142.63	343.75
T_{3min}	0.01	0.02	0.06	0.14	0.85	3.64	11.16	23.84	65.63	133.36
T_{3max}	0.06	0.03	0.11	0.23	2.31	12.05	34.82	65.81	270.44	633.48
T_5^{u1}	0.12	0.12	0.44	0.96	7.78	26.71	117.36	115.08	408.74	831.80
T_{5min}^{u1}	0.04	0.05	0.09	0.17	0.48	1.81	1.98	3.94	8.87	10.47
T_{5max}^{u1}	0.40	0.31	2.34	3.38	33.70	273.98	1229.20	672.13	2023.10	6764
M_{2ave} $(\times 10^5)$	0.008	0.03	0.64	3.76	101.55	475	1918.70	3294.10	12781	32169
T_5	0.16	0.13	0.57	1.22	8.85	28.40	120.95	123.98	1312.40	1198.30
T_{5min}	0.04	0.05	0.10	0.20	0.55	2.69	5.60	5.32	6.24	27.18
T_{5max}	0.72	0.41	3.65	6.12	48.92	194.85	747.96	473.15	23446	10074
L_{ave} $(\times 10^3)$	0.05	0.18	1.22	4.44	41.39	144	281.73	458.43	1086.2	1695.2
T_2	0.01	0.02	0.05	0.10	0.61	2.57	6.68	11.70	38.79	90.37
T_{2min}	0.01	0.02	0.04	0.10	0.43	1.33	3.56	7.11	17.66	41.37
T_{2max}	0.04	0.04	0.08	0.15	0.88	3.39	9.30	19.16	70.63	162.82
T_6^{u1}	0.10	0.11	0.42	0.78	7.24	22.34	94.38	90.31	299.07	540.56
T_{6min}^{u1}	0.04	0.06	0.12	0.23	0.96	2.21	2.93	5.32	7.95	14.37
T_{6max}^{u1}	0.45	0.34	1.75	2.89	44.15	287.95	1186.20	670.77	2780.60	2408.60
T_6	0.16	0.13	0.46	0.86	7.78	24.15	106.08	98.63	327.67	592.60
T_{6min}	0.05	0.07	0.11	0.25	0.91	1.84	2.78	5.40	8.05	13.59
T_{6max}	1.01	0.37	1.90	2.79	47.36	308.36	1309.80	731.62	3151.70	2256.10

by Algorithms 2 and 3, respectively. For comparison, we also solve the weighted problems (**MCSPIT**$_1$) and (**MSPIT**$_1$) by Algorithms 6 and 5 when the cost vector $c = (1, \ldots, 1)$, respectively. That is, solve the problems under unit l_1 norm by the primal dual algorithms for the weighted l_1 norm, whose time complexity is $O(n^2)$ and much worse than $O(n)$ for the the problems under unit l_1 norm. We recorded the average CPU time of the six algorithms. Let T_2, T_3, T_5, T_6 be the average CPU time of Algorithms 2, 3, 5 and 6, respectively. Let T_5^{u1}, T_6^{u1} be the average CPU time of Algorithms 5 and 6 when $c = (1, \ldots, 1)$, respectively. For each algorithm, the relevant minimum and maximum running time are also recorded, denoted by T_{min} and T_{max}, respectively.

It can be seen from Table 4 that Algorithm 3 for the problem (**MSPIT**$_{u1}$), which has time complexity $O(n)$, is really more efficient than the primal dual algorithm 5 that runs in $O(n^2)$ time. Correspondingly, Algorithm 2 for the problem (**MCSPIT**$_{u1}$) with time complexity $O(n)$ also operates better than Algorithm 6 running in $O(n^2)$ time.

7 Conclusion and further research

In this paper, we consider the maximum shortest Path interdiction problem by upgrading edges on tree network under unit/weighted l_1 norm. Under the unit l_1 norm, we present two linear time algorithms for the problems (**MCSPIT**$_{u1}$) and (**MSPIT**$_{u1}$), respectively. Furthermore, we revised a mistake in the method given in [9] and improved the algorithm to solve the problem (**MSPIT**$_{u1}$). Under weighted l_1 norm, we propose two primal dual algorithms for the problems (**MSPIT**$_1$) and (**MCSPIT**$_1$) in $O(n^2)$ time, respectively, in which a minimum cost cut is solved in each iteration.

For further research, we can focus on the maximum shortest Path interdiction problem on some networks including series parallel graphs and general graphs. On the other hand, we can consider other network interdiction problem based on network performances such as the minimum spanning tree and the maximum matching.

Acknowledgements Research is supported by National Natural Science Foundation of China (11471073). The work of P.M. Pardalos was conducted within the framework of the Basic Research Program at the National Research University Higher School of Economics (HSE).

References

1. 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)
2. Bazgan, C., Nichterlein, A., et al.: A refined complexity analysis of finding the most vital edges for undirected shortest paths: algorithms and complexity. Lect. Notes Comput. Sci. **9079**, 47–60 (2015)
3. Bazgan, C., Toubaline, S., Vanderpooten, D.: Complexity of determining the most vital elements for the p-median and p-center location problems. J. Comb. Optim. **25**(2), 191–207 (2013)
4. Bazgan, C., Toubaline, S., Vanderpooten, D.: Critical edges for the assignment problem: complexity and exact resolution. Oper. Res. Lett. **41**, 685–689 (2013)
5. 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)
6. Corley, H.W., Sha, D.Y.: Most vital links and nodes in weighted networks. Oper. Res. Lett. **1**, 157–161 (1982)
7. 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)
8. 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), pp. 539–546 (1996)
9. Hambrusch, S.E., Tu, H.Y.: Edge weight reduction problems in directed acyclic graphs. J. Algorithms **24**(1), 66–93 (1997)
10. Iwano, K., Katoh, N.: Efficient algorithms for finding the most vital edge of a minimum spanning tree. Inf. Process. Lett. **48**(5), 211–211-213 (1993)
11. 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)
12. Liang, W.: Finding the k most vital edges with respect to minimum spanning trees for fixed k. Discrete Appl. Math. **113**(2–3), 319–327 (2001)
13. 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)
14. Papadimitriou, C.H., Steiglitz, K.: Combinatorial Optimization: Algorithms and Complexity, 2nd edn. Dover Publications, New York (1988)
15. 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, pp. 964–73 (2005)

16. Ries, B., Bentz, C., Picouleau, C., Werra, D., de Costa, M., Zenklusen, R.: Blockers and transversals in some subclasses of bipartite graphs: when caterpillars are dancing on a grid. Discrete Math. **310**(1), 132–146 (2010)
17. Zenklusen, R.: Matching interdiction. Discrete Appl. Math. **158**(15), 1676–1690 (2010)
18. Zenklusen, R.: Network flow interdiction on planar graphs. Discrete Appl. Math. **158**(13), 1441–1455 (2010)
19. Zenklusen, R., Ries, B., Picouleau, C., de Werra, D., Costa, M., Bentz, C.: Blockers and transversals. Discrete Math. **309**(13), 4306–4314 (2009)
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)