The sum of root-leaf distance interdiction problem with cardinality constraint by upgrading edges on trees

Xiao Li · Xiucui Guan · Qiao Zhang · Xinyi Yin · Panos M. Pardalos

Abstract A network for the transportation of supplies can be described as a rooted tree with a weight of a degree of congestion for each edge. We take the sum of root-leaf distance (SRD) on a rooted tree as the whole degree of congestion of the tree. Hence, we consider the SRD interdiction problem on trees with cardinality constraint by upgrading edges (denoted by (**SDIPTC**) in brief). It aims to maximize the SRD by upgrading the weights of N critical edges such that the total upgrade cost under some measurement is upper-bounded by a given value. The relevant minimum cost problem(**MCSDIPTC**) aims to minimize the total upgrade cost on the premise that the SRD is lower-bounded by a given value. We develop two different norms including weighted l_∞ norm and weighted bottleneck Hamming distance to measure the upgrade cost. We propose two binary search algorithms within $O(n \log n)$ time for the problems (**SDIPTC**) under the two norms, respectively. For problems (**MCSDIPTC**),we propose two binary search algorithms within $O(Nn^2)$ and $O(n \log n)$ under weighted l_∞ norm and weighted bottleneck Hamming distance, respectively. These problems are solved through their subproblems (**SDIPT**) and (**MCSDIPT**), in which we ignore the cardinality constraint on the number of upgraded edges. Finally, we design numerical experiments to show the effectiveness of these algorithms.

Keywords Network interdiction problems · Tree · Greedy algorithm · Binary search method · l_∞ norm · Bottleneck Hamming distance

X. Li · X.C. Guan · X.Y. Yin
School of Mathematics, Southeast University, Nanjing 210096, Jiangsu, China
E-mail: xcguan@163.com (Corresponding author: Xiucui Guan)

Q. Zhang
Aliyun School of Big Data, Changzhou University, Changzhou 213164, Jiangsu, China

P.M. Pardalos
Center for Applied Optimization, Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL, USA

Mathematics Subject Classification (2020) 90C27 · 90C35

1 Introduction

Terrorism is a persistent challenge for many nations worldwide, and it is imperative that governments take appropriate measures to prevent and mitigate the impact of terrorist attacks. Among the diverse tactics used by terrorists, actions that trigger fires and hinder the aid process are recognized as Network Interdiction Problems (NIPs), which are based on game theory.

An NIP involves two main actors, a leader and a follower, who have competing objectives. The follower aims to optimize their objectives by using a network to facilitate the movement of resources such as supply convoys or maximizing the amount of material transported through the network. Conversely, the leader seeks to impede the follower's objectives by disrupting the network's structure through interdicting arcs or nodes. The leader can do so by attacking nodes to reduce their capacity or destroy them, ultimately hindering the follower's ability to move through the network [8].

Some transportation networks can be represented as rooted trees with weighted edges, where the root node is the critical infrastructure or facility that the leader intends to attack, the internal nodes are served as transportation hubs, and leaf nodes are emergency agencies including fire stations, police stations, hospitals etc. The edge weight indicates the degree of congestion of the roads, and targeting these edges can disrupt the network by impeding rescue. However, decision-makers with limited resources must identify the most strategic edges to attack in order to maximize the network's overall performance degradation.

In a broader sense, tree networks can be used to describe networks with one-directional links. For instance, during wartime, ammunition supplies may only flow from logistics centers to the front lines. The related issues can be formulated as follows.

Let $T = (V, E, w, u, c)$ be an edge-weighted 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 leaf nodes. Let $w(e)$ and $u(e)$ be the original weight and upper bound of edge $e \in E$, respectively, where $u(e) \geq w(e) \geq 0$. Let $c(e) > 0$ be the unit modification cost of edge $e \in E$. Let $P_k := P_{t_k} := P_{s,t_k}$ be the root-leaf path from the root node s to the leaf node t_k. Let $w(P_k) := \sum_{e \in P_{s,t_k}} w(e)$ and $w(T) := \sum_{t_k \in Y} w(P_k)$ be the weight of the path P_k and the sum of root-leaf distance (SRD) under the edge weight w, respectively. The sum of root-leaf distance interdiction problem on trees with cardinality constraint (**SDIPTC**) aims to maximize the SRD by determining an edge weight vector $\tilde{w}$ such that the modification cost $\|\tilde{w} - w\|$ in a certain norm and the number of updated edges do not exceed two given values K and

N, respectively. Its mathematical model can be stated as follows.

$$\max_{\tilde{w}} \tilde{w}(T) := \sum_{t_k \in Y} \tilde{w}(P_k)$$

$$\textbf{(SDIPTC)} \quad s.t. \quad \|\tilde{w} - w\| \leq K,$$

$$\sum_{e \in E} H(\tilde{w}(e), w(e)) \leq N, \tag{1}$$

$$w(e) \leq \tilde{w}(e) \leq u(e), \quad e \in E,$$

where $H(\tilde{w}(e), w(e)) = \begin{cases} 0, & \tilde{w}(e) = w(e) \\ 1, & \tilde{w}(e) \neq w(e) \end{cases}$ is the Hamming distance between $\tilde{w}(e)$ and $w(e)$.

We also consider its relevant minimum cost problem with cardinality constraint (**MCSDIPTC**) by exchanging its objective function and the norm constraint. The mathematical model of the problem is as follows.

$$\min_{\tilde{w}} C(\tilde{w}) := \|\tilde{w} - w\|$$

$$\textbf{(MCSDIPTC)} \quad s.t. \quad \sum_{t_k \in Y} \tilde{w}(P_k) \geq D,$$

$$\sum_{e \in E} H(\tilde{w}(e), w(e)) \leq N, \tag{2}$$

$$w(e) \leq \tilde{w}(e) \leq u(e), \quad e \in E.$$

Most network interdiction problems aim to make the network's performance as poor as possible by deleting critical edges or nodes. Albert (2000) [1] found that in a scale-free network when the most connected nodes are removed, the diameter of the network rapidly increases. They pointed out that this vulnerability that is susceptible to attack is due to the uneven distribution of connections and the removal of a limited number of nodes in the network can have a significant impact, giving rise to the concept of critical nodes and critical edges. Subsequently, Ayyildiz et al. (2019) [2] analyzed how to maximize the length of the shortest path between two nodes based on a limited interdicting budget from two opposing aspects (breaking and maintaining the network). On the other hand, Magnouche and Martin (2020) [11] studied how to delete the minimum number of critical nodes to make the root-leaf path length in the remaining graph at least d. They proved the $\mathcal{NP}$-hardness of the problem, proposed an integer linear programming model with an exponent p constraints and designed a branch and bound algorithm.

Corley and Sha (1982) [6] first applied the NIP by deleting K critical edges to maximize the length of the shortest path. Ball et al. (1989) [3] proved that the problem is $\mathcal{NP}$-hard. Khachiyan et al. (2008) [9] proved that there is no approximation algorithm with a performance ratio of 2. Bazgan et al. (2015) [5] provided an $O(mn)$ time algorithm when the increment $b = 1$ of the path length, and further (2019) [4] proved that the problem is $\mathcal{NP}$-hard when $b \geq 2$.

However, it is often difficult to implement the deletion of critical edges/nodes in practical applications. Zhang et al. (2021) [13] proposed a concept of updating critical edges to replace the limitations of the traditional NIP caused by deleting critical edges, and studied the shortest path interdiction problem (SPIT) and its minimum cost problem (MCSPIT) on trees. For the problem (SPIT), they provided an $O(n^2)$ primal-dual algorithm under the weighted l_1 norm and improved the time complexity to $O(n)$ under the unit l_1 norm. For the problem (SPIT$_{uH}$) under unit Hamming distance, they [14] devised algorithms with time complexities $O(N+l\log l)$ and $O(n(\log n + K^3))$ when $K=1$ and $K>1$, respectively, where n, K and l are the numbers of tree nodes, upgraded edges and leaves respectively. Afterward, Yi et al. (2023) [10] improved the time complexities to $O(n)$ and $O(nK^2)$, respectively. For the relevant minimum cost problem (MCSPIT$_{uH}$), based on dynamic programming ideas and binary search methods, Zhang et al. (2020) [14] provided an $O(n^4\log n)$ algorithm. Yi et al. (2023) [10] improved the complexity to $O(n^3\log n)$ through a binary search. Zhang et al. (2020) [14] subsequently demonstrated that the problems (SPIT$_H$) and (MCSPIT$_H$) under the weighted Hamming distance are equivalent to the 0-1 knapsack problem and therefore $\mathcal{NP}$-hard. Furthermore, for the SRD interdiction problems (SDIPT-UE/N) and (MCSDIPT-UE/N) by upgrading edges/nodes under weighted Hamming distance, Zhang et al. [12] proved its $\mathcal{NP}$-hardness by showing the equivalence between the two problems and the 0–1 knapsack problem. Under weighted l_1 norm, the problems (SDIPT$_1$-UE) and (MCSDIPT$_1$-UE) can be solved in $O(n)$ time by continuous knapsack problems. For the problem (SDIPT$_{uH}$-UE) under unit Hamming distance and unit node cost problem (SDIPT-UN), they designed two linear time greedy algorithms. Moreover, for the minimum cost problems (MCSDIPT$_{uH}$-UE)under unit Hamming distance and unit node cost problem (MCSDIPT-UN), they [12] used a binary search method to provide algorithms with a time complexity of $O(n\log n)$.

Based on the above analysis, we can know that there is no reseach results for the SRD interdiction problems under l_∞ norm and bottleneck Hamming distance, which are sub-problems of the problems (**SDIPTC**) and (**MCSDIPTC**) we mainly studied in this paper. We impose a cardinality constrain on the number of modified edges since there are always many optimal solutions, which include some unnecessary modifications of weights, for the problem (**SDIPT**) under bottleneck norms. We compare our results with existing studies in Table 1, where the subscripts $1, u1, H, uH, BH$ and $_\infty$ denote weighted l_1-norm, unit l_1-norm, weighted Hamming distance, unit Hamming distance, bottleneck Hamming distance, and weighted l_∞ norm, respectively.

The paper is structured as follows. In Section 2, we propose two algorithms for solving problems (**SDIPTC$_\infty$**) and (**MCSDIPTC$_\infty$**) in $O(n)$ and $\mathrm{O}(Nn^2)$ time, respectively, which mainly call the subroutines for solving the two sub-problems (**SDIPT$_\infty$**) and (**MCSDIPT$_\infty$**) in $O(n)$ and $\mathrm{O}(n\log n)$ time, respectively. In Section 3, we solve the problems (**MCSDIPTC$_{BH}$**) and (**SDIPTC$_{BH}$**) under bottleneck Hamming distance in $O(n)$ and $\mathrm{O}(n\log n)$ time, respectively, which have the same time complexities for the sub-problems

Table 1 The relationship between the previous research and our research

Type	Graph	Problem	K/b/c		Complexity	Ref.
Del. nodes	General graph	NIP on shortest path	any K		$\mathcal{NP}$-hard	[11]
					$\mathcal{NP}$-hard	[3]
Del. edges					Not approx. within 2	[9]
	Undirected network		$b=1$		$O(mn)$	[5]
			$b \geq 2$		$\mathcal{NP}$-hard	[4]
Upgrad nodes	Tree	SDIPT-UN	any K	any c	$\mathcal{NP}$-hard	[12]
		MCSDIPT-UN			$\mathcal{NP}$-hard	
		SDIPT-UN$_u$		$c=1$	$O(n)$	
		MCSDIPT-UN$_u$			$O(n \log n)$	
Upgrad edges		MCSPIT$_1$	any K	c=1	$O(n)$	[13]
				any c	$O(n^2)$	
		MSPIT$_1$		$c=1$	$O(n)$	
				any c	$O(n^2)$	
		MSPIT$_H$		any c	$\mathcal{NP}$-hard	[10] [14]
			$K=1$	$c=1$	$O(n)$	
			any K	$c=1$	$O(nK^2)$	
		MCSPIT$_{uH}$			$O(n^3 \log n)$	
		SDIPT-UE$_H$	any K	any c	$\mathcal{NP}$-hard	[13]
		MCSDIPT-UE$_H$			$\mathcal{NP}$-hard	
		SDIPT-UE$_{uH}$	any K	$c=1$	$O(n)$	
		MCSDIPT-UE$_{uH}$			$O(n \log n)$	
		SDIPT-UE$_1$	any K	any c	$O(n)$	
		MCSDIPT-UE$_1$			$O(n)$	
		SDIPT$_\infty$	any K	any c	$O(n)$	(7)
		SDIPTC$_\infty$			$O(n)$	Alg1
		MCSDIPT$_\infty$			$O(n \log n)$	Alg2
		MCSDIPTC$_\infty$			$O(Nn^2)$	Alg3
		SDIPT$_{BH}$			$O(n)$	(26)
		SDIPTC$_{BH}$			$O(n)$	Alg4
		MCSDIPT$_{BH}$			$O(n \log n)$	Alg5
		MCSDIPTC$_{BH}$			$O(n \log n)$	Alg6

without the cardinality constraints. In Section 4, we report on computational experiments that demonstrate the effectiveness of all proposed polynomial time algorithms. Finally, in Section 5, we summarize our findings and suggest some areas for future research.

2 Solve the problems (SDIPTC$_\infty$) and (MCSDIPTC$_\infty$) under weighted l_∞ norm

When the weighted l_∞ norm is applied to the upgrade cost, the problems (SDIPTC$_\infty$) and (MCSDIPTC$_\infty$) under weighted l_∞ norm, can be formulated as the following models (3) and (4), respectively.

$$
\max_{\tilde{w}} \tilde{w}(T) := \sum_{t_k \in Y} \tilde{w}(P_k)
$$

$$
\textbf{(SDIPTC}_\infty) \quad s.t. \quad \max_{e \in E} c(e)(\tilde{w}(e) - w(e)) \leq K, \tag{3}
$$

$$
\sum_{e \in E} H(\tilde{w}(e), w(e)) \leq N,
$$

$$
w(e) \leq \tilde{w}(e) \leq u(e), \quad e \in E.
$$

$$\min_{\tilde{w}} C(\tilde{w}) := \max_{e \in E} c(e)(\tilde{w}(e) - w(e))$$

$$(\textbf{MCSDIPTC}_\infty) \quad s.t. \sum_{t_k \in Y} \tilde{w}(P_k) \geq D, \tag{4}$$

$$\sum_{e \in E} H(\tilde{w}(e), w(e)) \leq N,$$

$$w(e) \leq \tilde{w}(e) \leq u(e), \quad e \in E.$$

This section proves the properties of the problem ($\textbf{SDIPTC}_\infty$) and presents a divide-and-conquer algorithm with time complexity of $O(n)$ for it. For the relevant minimum cost problem ($\textbf{MCSDIPTC}_\infty$), this section provides an algorithm with a time complexity of $O(Nn^2)$.

Definition 1 [12] For any $e \in E$, define $L(e) := \{t_k | e \in P_{s,t_k}, k = 1, 2, \ldots, r\}$ 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.

2.1 An $O(n)$ time algorithm to solve the problem ($\textbf{SDIPTC}_\infty$)

In order to solve problem ($\textbf{SDIPTC}_\infty$), we first consider its sub-problem (SDIPT$_\infty$) by deleting the constraint of cardinality. Its mathematical model can be described as follows.

$$\max_{\bar{w}} \bar{w}(T) := \sum_{t_k \in Y} \bar{w}(P_k)$$

$$(\textbf{SDIPT}_\infty) \quad s.t. \max_{e \in E} c(e)(\bar{w}(e) - w(e)) \leq K, \tag{5}$$

$$w(e) \leq \bar{w}(e) \leq u(e), \quad e \in E.$$

In the problem ($\textbf{SDIPT}_\infty$), we can maximize the weight of each edge as much as possible under the constraint of cost K and the upper bound $u(e)$ of the edge weight $u(e)$. Therefore, we have the following theorem.

Theorem 1 *The weight vector*

$$\bar{w}(e) = w(e) + \min \left\{ \frac{K}{c(e)}, u(e) - w(e) \right\} (e \in E) \tag{6}$$

is an optimal solution to the problem ($\textbf{SDIPT}_\infty$).

Proof **(1)** Apparently, the solution $\bar{w}$ defined by (6) satisfies feasibility.

(2) Suppose that $\bar{w}$ is not the optimal solution, and there exists another edge weight vector w' such that $w'(T) > \bar{w}(T)$. Then at least one edge $e_i \in E$ satisfies $w'(e_i) > \bar{w}(e_i)$. We have $w'(e_i) - w(e_i) > \bar{w}(e_i) - w(e_i) = \min \left\{ \frac{K}{c(e)}, u(e) - w(e) \right\}$, which implies $c(e_i)(w'(e_i) - w(e_i)) > K$ or $w'(e_i) >$

$u(e_i)$, either of which contradicts the cost constraint condition or the upper bound constraint condition, respectively. Therefore, $\bar{w}$ is an optimal solution to problem (**SDIPT**$_\infty$). $\square$

Based on Theorem 1, for any given T, n, w, u, c, K, we can obtain an optimal solution $\bar{w}$ and its corresponding optimal value $\bar{w}(T)$ for problem (**SDIPT**$_\infty$). For convenience, we denote this subroutine as:

$$[\bar{w}, \bar{w}(T)] = SDIPT_\infty(T, n, w, u, c, K) \tag{7}$$

We just update every edges in the subroutine, therefore, we have the following conclusion.

Theorem 2 *Problem (**SDIPT**$_\infty$) can be solved by subroutine (7) in $O(n)$ time.*

Now we can come to the problem (**SDIPT**$_\infty$) with cardinality constraint. First, call $[\bar{w}, \bar{w}(T)] := SDIPT_\infty(T, n, w, u, c, K)$. Then, we define $\bar{S}(e) = |L(e)|(\bar{w}(e) - w(e))(e \in E)$, which means the increment of SRD for e under $\bar{w}$. Based on that, we can select the N edges with the largest $\bar{S}(e)$, and update them to obtain an optimal solution.

Therefore, in the first step, we use the algorithm proposed by Thoms (2009, pp.220-222) [7] called $\bar{S}^N := Selection(\bar{S}, N)$ to find the N-th largest element $\bar{S}^N$ in the set $\bar{S} := \{\bar{S}(e)|e \in E\}$, and thus divide the edge set into two parts, $\bar{E}_N := \{e \in E|\bar{S}(e) \geq \bar{S}^N\}$ and $E \backslash \bar{E}_N$.

Theorem 3 *The weight vector*

$$\tilde{w}(e) = \begin{cases} \bar{w}(e), & e \in \bar{E}_N \\ w(e), & otherwise \end{cases} \tag{8}$$

*is an optimal solution of the problem (**SDIPTC**$_\infty$).*

Proof Feasibility is obviously satisfied. Suppose $\tilde{w}$ is not optimal, but $\hat{w}$ is. Then $\hat{w}(T) > \tilde{w}(T)$, and there exists at least one edge e_k, such that $\hat{w}(e_k) > \tilde{w}(e_k)$.

(1) If $e_k \in \bar{E}_N$, then $c(e_k)(\hat{w}(e_k) - w(e_k)) > c(e_k)(\tilde{w}(e_k) - w(e_k)) = \min\{u(e_k), K\}$, which contradicts the upper bound or update cost constraints.

(2) If $e_k \in E \backslash \bar{E}_N$, then $\bar{w}(e_k) \geq \hat{w}(e_k) > \tilde{w}(e_k) = w(e_k)$. Moreover, there must exist an edge $\hat{e}$ in $\bar{E}_N$ such that $\hat{w}(\hat{e}) = w(\hat{e}) < \tilde{w}(\hat{e}) = \bar{w}(e_k)$. Since $\bar{S}(\hat{e}) > \bar{S}(e_k)$, there exists a feasible solution $\breve{w}(e)$ given by

$$\breve{w}(e) = \begin{cases} w(e), & e = e_k \\ \bar{w}(e), & e = \hat{e} \\ \hat{w}(e), & otherwise \end{cases}$$

thus we have

$$
\begin{aligned}
\breve{w}(T) - w(T) &= \sum_{e \in E \setminus \{e_k, \hat{e}\}} |L(e)|(\hat{w}(e) - w(e)) + \bar{S}(\hat{e}) \\
&> \sum_{e \in E \setminus \{e_k\}} |L(e)|(\hat{w}(e) - w(e)) + \bar{S}(e_k) \\
&\geq \sum_{e \in E \setminus \{e_k\}} |L(e)|(\hat{w}(e) - w(e)) + |L(e_k)|(\hat{w}(e_k) - w(e_k)) \\
&= \hat{w}(T) - w(T),
\end{aligned}
$$

this indicates that $\breve{w}(T) > \bar{w}(T)$, which contradicts the optimality of $\hat{w}$. $\qquad\square$

Based on theorem 3, we present Algorithm 1 below to solve the problem (**SDIPTC$_\infty$**).

Algorithm 1 $[\tilde{w}, \tilde{w}(T)] :=$ SDIPTC$_\infty(T, n, w, u, c, K, N)$

Require: A rooted tree $T(V, E)$, the number n of edges, the weight vector w, an upper bound vector u, a cost vector c, and given K and N.
Ensure: The optimal vector $\tilde{w}$ and the corresponding SRD $\tilde{w}(T)$.
1: Call $[\bar{w}, \bar{w}(T)] := SDIPT_\infty(T, n, w, u, c, K)$.
2: For each e, determine the set $L(e)$, and calculate $\bar{S}(e) := |L(e)|(\bar{w}(e) - w(e))$.
3: $\bar{S}^N := Selection(\bar{S}, N), \bar{E}_N := \{e \in E | \bar{S}(e) \geq \bar{S}^N\}$.
4: Compute $\tilde{w}$ by (8) and $\tilde{w}(T) := \sum_{t_k \in Y} \tilde{w}(P_k)$.
5: **return** $[\tilde{w}, \tilde{w}(T)]$.

Theorem 4 *Problem (**SDIPTC$_\infty$**) can be solved by Algorithm 1 in $O(n)$ time.*

Proof In Algorithm 1, line 1 solves problem (**SDIPT$_\infty$**) in linear time, the Selection algorithm in Line 3 also takes linear time [7], and Line 4 can determine the optimal solution $\tilde{w}$ in $O(n)$ time. Therefore, problem (**SDIPTC$_\infty$**) can be solved by Algorithm 1 in $O(n)$ time. $\qquad\square$

2.2 An $O(Nn^2)$ time algorithm to solve the problem (**MCSDIPTC$_\infty$**)

Solving problem (**MCSDIPTC$_\infty$**) directly is challenging, and therefore, we first focus on solving its sub-problem (**MCSDIPT$_\infty$**) without the cardinality constraint. This sub-problem can be described as follows:

$$
\begin{aligned}
&\min_{\bar{w}} C(\bar{w}) := \max_{e \in E} c(e)(\bar{w}(e) - w(e)) \\
(\textbf{MCSDIPT}_\infty) \quad &s.t. \sum_{t_k \in Y} \bar{w}(P_k) \geq D, \\
&\quad\quad w(e) \leq \bar{w}(e) \leq u(e), \quad e \in E.
\end{aligned}
\tag{9}
$$

For convenience, let $\Delta D := D - w(T)$.

*2.2.1 An $O(n \log n)$ algorithm to solve (**MCSDIPT$_\infty$**)*

For the problem (**MCSDIPT$_\infty$**), we can solve two special cases based on the relationship between $u(T)$ and D, as well as $w(T)$ and D.

Lemma 1 *If $D \leq w(T)$, then w is an optimal solution to (**MCSDIPT$_\infty$**).*

Lemma 2 *If $u(T) < D$, then the problem (**MCSDIPT$_\infty$**) is infeasible.*

The proofs of the above two lemmas are obvious. Then we consider the situation that $w(T) < D \leq u(T)$. First, we will prove an interesting property of the optimal solution.

Lemma 3 *Let $\bar{w}$ be an optimal solution of problem (**MCSDIPT$_\infty$**), then $\bar{w}(T) = \sum_{t_k \in Y} \bar{w}(P_k) = D$.*

Proof Suppose SRD under optimal solution $\bar{w}$ is $\bar{w}(T) = D + d(d > 0)$. Let $\bar{w}'(e) = \bar{w}(e) - \frac{d}{c(e) \sum_{e_i \in E} \frac{|L(e_i)|}{c(e_i)}} (e \in E)$, then

$$
\begin{aligned}
\bar{w}'(T) &= \sum_{t_k \in Y} \bar{w}'(P_k) \\
&= \sum_{e \in E} |L(e)|(\bar{w}'(e) - \bar{w}(e)) + \sum_{e \in E} |L(e)|\bar{w}(e) \\
&= -\sum_{e \in E} \frac{d}{c(e) \sum_{e_i \in E} \frac{|L(e_i)|}{c(e_i)}} |L(e)| + \bar{w}(T) \\
&= -\frac{d}{\sum_{e_i \in E} \frac{|L(e_i)|}{c(e_i)}} \sum_{e \in E} \frac{|L(e)|}{c(e)} + D + d \\
&= D
\end{aligned}
$$

while the maximum cost

$$
C(\bar{w}') = \max_{e \in E} c(e)(\bar{w}'(e) - w(e)) < \max_{e \in E} c(e)(\bar{w}(e) - w(e)) = C(\bar{w})
$$

which contradicts the optimality of $\bar{w}$. $\qquad\square$

Here we construct an optimal solution to the problem (**MCSDIPT$_\infty$**) where $w(T) < D \leq u(T)$. Using the property of the l_∞ norm, we know that when the update costs are fixed, updating every edge can increase SRD by the maximum amount. For each edge $e \in E$, let $F(e) = c(e)(u(e) - w(e))$ represent the total cost required to achieve the maximum possible update. Then we sort the $F(e)$ values of all edges in a non-decreasing order and renumber them to obtain the sequence $\{F(e_{m_k})\}$ as follows:

$$
F(e_{m_1}) \leq F(e_{m_2}) \leq \cdots \leq F(e_{m_n}). \tag{10}
$$

For an index k, define a function

$$f(k) = \sum_{e \in E} |L(e)| \cdot \min\left\{u(e) - w(e), \frac{F(e_{m_k})}{c(e)}\right\} \tag{11}$$

to represent the maximum increment of SRD when the update cost is $F(e_{m_k})$

Next, using the binary search method on the sequence $\{F(e_{m_k})\}$, we can determine a critical index k^* such that $f(k^*) \leq D - w(T) \leq f(k^* + 1)$. This allows us to obtain a unique optimal solution to the problem ($\mathbf{MCSDIPT}_\infty$).

Theorem 5 *When $w(T) < D \leq u(T)$, let $E_{\leq} := \{e \in E | F(e) \leq F(e_{m_{k^*}})\}$, the weight vector*

$$\bar{w}(e) = \begin{cases} u(e), & e \in E_{\leq} \\ w(e) + \frac{F(e_{m_{k^*}})}{c(e)} + \frac{\Delta D - f(k^*)}{c(e) \sum_{j=k^*+1}^{n} \frac{|L(e_{m_j})|}{c(e_{m_j})}}, & e \in E \backslash E_{\leq} \end{cases} \tag{12}$$

is the unique optimal solution to the problem ($\mathbf{MCSDIPT}_\infty$),

Proof **(1) First, we prove that $\bar{w}$ is a feasible solution to problem MCSDIPT$_\infty$.**

(1.1) We prove that $\bar{w}(e)$ for $e \in E$ satisfies the bound constraint. When $e \in E_{\leq}$, we have $\bar{w}(e) = u(e)$; and when $e \in E \backslash E_{\leq}$, we have $\frac{F(e_{m_{k^*}})}{c(e)} < u(e) - w(e)$. Then

$$f(k^* + 1) - f(k^*)$$
$$= \sum_{i=1}^{n} \min\{F(e_{m_i}), F(e_{m_{k^*+1}})\} \frac{|L(e_{m_i})|}{c(e_{m_i})}$$
$$- \sum_{i=1}^{n} \min\{F(e_{m_i}), F(e_{m_{k^*}})\} \frac{|L(e_{m_i})|}{c(e_{m_i})}$$
$$= \left\{\sum_{i=1}^{k^*} F(e_{m_i}) \frac{|L(e_{m_i})|}{c(e_{m_i})} + \sum_{i=k^*+1}^{n} F(e_{m_{k^*+1}}) \frac{|L(e_{m_i})|}{c(e_{m_i})}\right\}$$
$$- \left\{\sum_{i=1}^{k^*} F(e_{m_i}) \frac{|L(e_{m_i})|}{c(e_{m_i})} + \sum_{i=k^*+1}^{n} F(e_{m_{k^*}}) \frac{|L(e_{m_i})|}{c(e_{m_i})}\right\}$$
$$= \left(F(e_{m_{k^*+1}}) - F(e_{m_{k^*}})\right) \sum_{i=k^*+1}^{n} \frac{|L(e_{m_i})|}{c(e_{m_i})}.$$

Therefore

$$c(e)(\bar{w}(e) - w(e)) = \frac{\Delta D - f(k^*)}{\sum_{i=k^*+1}^{n} \frac{|L(e_{m_i})|}{c(e_{m_i})}} + F(e_{m_k^*})$$

$$< \frac{f(k^*+1) - f(k^*)}{\sum_{i=k^*+1}^{n} \frac{|L(e_{m_i})|}{c(e_{m_i})}} + F(e_{m_{k^*}})$$

$$= \frac{\left(F(e_{m_{k^*+1}}) - F(e_{m_{k^*}})\right) \sum_{i=k^*+1}^{n} \frac{|L(e_{m_i})|}{c(e_{m_i})}}{\sum_{i=k^*+1}^{n} \frac{|L(e_{m_i})|}{c(e_{m_i})}} + F(e_{m_{k^*}})$$

$$= F(e_{m_{k^*+1}}) \le F(e).$$

Hence, $w(e) \le \bar{w}(e) < u(e)$, for any $e \in E \backslash E_{\le}$.

(1.2) Prove that under $\bar{w}$, $\bar{w}(T) \ge D$.

$$\bar{w}(T) - w(T) = \sum_{t_k \in Y} \bar{w}(P_k) - \sum_{t_k \in Y} w(P_k) = \sum_{e \in E} (\bar{w}(e) - w(e))|L(e)|$$

$$= \sum_{e \in E_{\le}} (u(e) - w(e))|L(e)| + \sum_{e \in E \backslash E_{\le}} \left(\frac{F(e_{m_{k^*}})}{c(e)} + \frac{\Delta D - f(k^*)}{\sum_{j=k^*+1}^{n} \frac{|L(e_{m_j})|}{c(e_{m_j})} c(e)} \right)|L(e)|$$

$$= f(k^*) + \frac{\left(\Delta D - f(k^*)\right) \cdot \sum_{j=k^*+1}^{n} \frac{|L(e_{m_j})|}{c(e_{m_j})}}{\sum_{j=k^*+1}^{n} \frac{|L(e_{m_j})|}{c(e_{m_j})}}$$

$$= \Delta D.$$

Therefore, $\bar{w}(T) = D$, and $\bar{w}$ is a feasible solution to problem **MCSDIPT$_\infty$**.

(2) Next, we prove the optimality of $\bar{w}$.

Assume that $\bar{w}$ is not the optimal solution of problem (**MCSDIPT$_\infty$**), but w' is. Then there exists $\bar{e} \in E$ such that $C(\bar{w}) = c(\bar{e})(\bar{w}(\bar{e}) - w(\bar{e})) > C(w') \ge c(\bar{e})(w'(\bar{e}) - w(\bar{e}))$. Therefore, the total increment in SRD due to edge $\bar{e}$ is $|L(\bar{e})|(w'(\bar{e}) - w(\bar{e})) < |L(\bar{e})|(\bar{w}(\bar{e}) - w(\bar{e}))$. According to Lemma 3, and $\bar{w}(T) = w'(T) = D$, there exists an edge $\hat{e} \in E$ such that $|L(\hat{e})|(w'(\hat{e}) - w(\hat{e})) > |L(\hat{e})|(\bar{w}(\hat{e}) - w(\hat{e}))$, which implies that $w'(\hat{e}) > \bar{w}(\hat{e})$.

(2.1) If $\hat{e} \in E_{\le}$, then $w'(\hat{e}) > \bar{w}(\hat{e}) = u(\hat{e})$, which contradicts the feasibility of w'.

(2.2) If $\hat{e} \in E \backslash E_{\le}$, then $C(w') \ge c(\hat{e})(w'(\hat{e}) - w(\hat{e})) > c(\hat{e})(\bar{w}(\hat{e}) - w(\hat{e})) = C(\bar{w})$, which contradicts $C(w') < C(\bar{w})$.

Therefore, $\bar{w}$ is the optimal solution to the problem (**MCSDIPT$_\infty$**).

(3) We prove the uniqueness of $\bar{w}$ by contradiction

Suppose $\hat{w}$ is another optimal solution of the problem (**MCSDIPT$_\infty$**), such that $C(\hat{w}) = C(\bar{w})$ and there exists $e_a \in E$ such that $\hat{w}(e_a) \ne \bar{w}(e_a)$.

(3.1) If $e_a \in E_{\le}$, then we have $\hat{w}(e_a) < \bar{w}(e_a) = u(e_a)$, and we can construct a feasible solution $\hat{w}'(e) = \begin{cases} u(e), & e = e_a \\ \hat{w}(e), & \text{otherwise} \end{cases}$, where $\sum_{t_k \in Y} \hat{w}'(P_k) =$

$D + |L(e_a)|(u(e_a) - \hat{w}(e_a)) > D$. Thus we construct another feasible solution $\hat{w}''(e) = \hat{w}'(e) - \frac{(u(e_a) - \hat{w}(e_a))|L(e_a)|}{c(e) \sum_{e_i \in E} \frac{|L(e_i)|}{c(e_i)}}, e \in E$, we get $C(\hat{w}'') < C(\hat{w}') = \max\{C(\hat{w}), F(e_a)\} = \max\{C(\bar{w}), F(e_a)\} = C(\bar{w})$, which contradicts the optimality of $\bar{w}$.

(3.2) If $e_a \in E \backslash E_{\leq}$ and $\hat{w}(e_a) > \bar{w}(e_a)$, then $C(\hat{w}) \geq c(e_a)(\hat{w}(e_a) - w(e_a)) > c(e_a)(\bar{w}(e_a) - w(e_a)) = C(\bar{w})$, which contradicts to the optimality of $\hat{w}$. Suppose $\hat{w}(e_a) < \bar{w}(e_a)$, similarly we can construct a feasible solution $\hat{w}'(e) = \begin{cases} \bar{w}(e_a), & e = e_a \\ \hat{w}(e), & \text{otherwise} \end{cases}$, where $\sum_{t_k \in Y} \hat{w}'(P_k) = D + |L(e_a)|(\bar{w}(e_a) - \hat{w}(e_a)) > \sum_{t_k \in Y} \bar{w}(P_k) = D$.

Then we construct $\hat{w}''(e) = \hat{w}'(e) - \frac{|L(e_a)|(\bar{w}(e_a) - \hat{w}(e_a))}{c(e) \sum_{e_i \in E} \frac{|L(e_i)|}{c(e_i)}} (e \in E)$, now we have $\hat{w}''(T) = D$ and $C(\hat{w}'') < C(\hat{w}') = C(\bar{w})$, which contradicts to the optimality of $\bar{w}$.

In conclusion, the weight vector defined by (12) is the unique optimal solution to the problem (**MCSDIPT$_\infty$**). $\qquad\square$

Based on Theorem 5, we present Algorithm 2 for problem (**MCSDIPT$_\infty$**).

Theorem 6 *The problem (**MCSDIPT$_\infty$**) can be solved by Algorithm 2 in $O(n \log n)$ time.*

Proof The sorting in Line 8 can be completed in $O(n \log n)$ time. The binary search process in Lines 10-21 takes $O(\log n)$ iterations. So it takes a total of $O(n \log n)$ time. Therefore, the problem (**MCSDIPT$_\infty$**) can be solved in $O(n \log n)$ time. $\qquad\square$

2.2.2 An $O(Nn^2)$ algorithm to solve (**MCSDIPTC$_\infty$**)

For the original problem (**MCSDIPTC$_\infty$**), we first consider its feasibility. We define $S(e) := |L(e)|(u(e) - w(e))(e \in E)$, which represents the maximum increment in SRD that each edge can achieve. We sort the edges in a non-increasing order of $S(e)$ and renumber them to obtain the following sequence:

$$S(e_{\beta_1}) \geq S(e_{\beta_2}) \geq \cdots \geq S(e_{\beta_{n-1}}) \geq S(e_{\beta_n}) \qquad (13)$$

Under the cardinality constraint of N edges, the maximum total increment in SRD is $S_N = \sum_{i=1}^{N} S(e_{\beta_i})$.

If $S_N < \Delta D$, then it is clear that the problem has no solution, as the increment S_N is not sufficient to meet the problem requirements.

Lemma 4 *When $S_N < \Delta D$, problem (**MCSDIPTC$_\infty$**) is infeasible.*

When $S_N \geq \Delta D$, we give the following lemma to determine the number of edges that are updated in any optimal solution.

Algorithm 2 $[\bar{w}, C(\bar{w})] :=$MCSDIPT$_{\infty}(T, n, w, u, c, D)$

Require: A rooted tree $T(V, E)$, the number n of edges, the weight vector w, an upper bound vector u, a cost vector c, and given value D.
Ensure: The optimal vector $\bar{w}$ and the cost $C(\bar{w})$.
1: For each edge $e \in E$, determine the set $L(e)$ and calculate $F(e) := c(e)(u(e) - w(e))$, $u(T) := \sum_{t_k \in Y} u(P_k)$, and let $\Delta D := D - w(T)$
2: **if** $u(T) < D$ **then**
3:    **return** "The problem is infeasible" and $\bar{w} = [\ ], C(\bar{w}) = +\infty$
4: **else**
5:    **if** $w(T) > D$ **then**
6:       **return** $[w, 0]$.
7:    **else**
8:       Sort $F(e)$ in a non-decreasing order by (10).
9:       Initialize $a := 1$, $b := n$, $k^* := 0$.
10:      **while** $b - a > 1$ and $k^* = 0$ **do**
11:         $k := \left\lfloor \frac{a+b}{2} \right\rfloor$.
12:         Calculate the values of $f(k)$ and $f(k+1)$ using (11).
13:         **if** $f(k) \leq \Delta D \leq f(k+1)$ **then**
14:           $k^* := k$.
15:         **else if** $f(k+1) < \Delta D$ **then**
16:           $a := k$.
17:         **else**
18:           $b := k$.
19:         **end if**
20:      **end while**
21:       Calculate $\bar{w}$ using (12), $C(\bar{w}) := F(e_{m_{k^*}}) + \dfrac{\Delta D - f(k^*)}{\sum_{j=k^*+1}^{n} \frac{|L(e_{m_j})|}{c(e_{m_j})}}$.
22:      **return** $[\bar{w}, C(\bar{w})]$.
23:    **end if**
24: **end if**

Lemma 5 *Suppose $S_N \geq \Delta D$, and let $\tilde{w}$ be an optimal solution of problem* **(MCSDIPTC$_{\infty}$)**. *Let $R = \sum_{e \in E} H(w(e), u(e))$. Then, we have*

$$\sum_{e \in E} H(\tilde{w}(e), w(e)) = \min\{R, N\}.$$

Proof (1) When $R \leq N$, it means that the cardinality constraint is always satisfied. So the problem **(MCSDIPTC$_{\infty}$)** can be directly transformed to its sub-problem **(MCSDIPT$_{\infty}$)**. According to Theorem 5, all R edges will be updated, so $\sum_{e \in E} H(\tilde{w}(e), w(e)) = R$ in this case.

(2) When $R > N$, we prove $\sum_{e \in E} H(\tilde{w}(e), w(e)) = N$ by contradiction. Suppose that $\sum_{e \in E} H(\tilde{w}(e), w(e)) < N$. Then, there is an edge e_i such that $\tilde{w}(e_i) = w(e_i) < u(e_i)$. We can construct a solution $\hat{w}(e) = \begin{cases} \tilde{w}(e), & e \neq e_i \\ w(e) + \varepsilon, & e = e_i \end{cases}$,

where $0 < \varepsilon \leq \min \left\{ \frac{C(\tilde{w})}{c(e_i)}, u(e_i) - w(e_i)) \right\}$. Then $C(\hat{w}) \leq C(\tilde{w})$ and thus

$$
\begin{aligned}
\hat{w}(T) &= \sum_{t_k \in Y} \hat{w}(P_k) = \sum_{e \in E} |L(e)| \hat{w}(e) \\
&= \sum_{e \in E \setminus \{e_i\}} |L(e)| \tilde{w}(e) + |L(e_i)| (\tilde{w}(e_i) + \varepsilon) \\
&= \sum_{e \in E} |L(e)| \tilde{w}(e) + |L(e_i)| \cdot \varepsilon \\
&= \tilde{w}(T) + |L(e_i)| \cdot \varepsilon.
\end{aligned} \tag{14}
$$

Let $\tilde{E} = \{e | c(e)(\tilde{w}(e) - w(e)) = C(\tilde{w})\}$, we can construct a solution

$$
\bar{w}(e) = \begin{cases} \hat{w}(e), & e \notin \tilde{E} \\ \hat{w}(e) - \frac{\varepsilon |L(e_i)|}{\sum_{e_j \in \tilde{E}} |L(e_j)|}, & e \in \tilde{E} \end{cases}
$$

then we have

$$
\begin{aligned}
\bar{w}(T) &= \sum_{t_k \in Y} \bar{w}(P_k) = \sum_{t_k \in Y} \hat{w}(P_k) - \frac{\varepsilon |L(e_i)|}{\sum_{e_j \in \tilde{E}} |L(e_j)|} \sum_{e \in \tilde{E}} |L(e)| \\
&= \hat{w}(T) - \varepsilon |L(e_i)| = \tilde{w}(T) \geq D,
\end{aligned}
$$

where the last equality can be obtained from (14). Furthermore, note that

$$
\sum_{e \in E} H(\bar{w}(e), w(e)) = \sum_{e \in E} H(\tilde{w}(e), w(e)) + 1 \leq N,
$$

then $\bar{w}$ is a feasible solution. However, it follows from the definitions of $\hat{w}$ and $\bar{w}$ that $C(\tilde{w}) > C(\bar{w})$, which contradicts the optimality of $\tilde{w}$. □

Based on the analysis above, if we can determine the set $\bar{E}^*$ of N edges that are updated in the optimal solution, then the problem (**MCSDIPTC$_\infty$**) can be transformed into its sub-problem (**MCSDIPT$_\infty$**) which can be solved by Algorithm 2 in $O(n \log n)$ time. Next, we use an iterative method to determine the set $\bar{E}^*$.

Initialization: $\tau := 0$, $\bar{E}^\tau := \{e_{\beta_1}, e_{\beta_2}, \cdots, e_{\beta_N}\}$.

Call Algorithm 2: $[\bar{w}^\tau, C(\bar{w}^\tau)]$=MCSDIPT$_\infty(T, n, w, u^\tau, c, D)$, where $u^\tau(e)$ $= \begin{cases} u(e), & e \in \bar{E}^\tau \\ w(e), \text{ otherwise} \end{cases}$, then the obtained $(\bar{w}^\tau, C(\bar{w}^\tau))$ is the initial feasible solution and the corresponding objective value of the problem.

Next, for each edge in the current edge set $\bar{E}^\tau$, consider whether there exists a better edge in the set $E \setminus \bar{E}^\tau$ that can reduce the upgrade cost by replacing it.

Sort the values $\nu(e) := \frac{|L(e)|}{c(e)}$ in a non-increasing order such that

$$
\nu(e_{\gamma_1}) \geq \nu(e_{\gamma_2}) \geq \cdots \geq \nu(e_{\gamma_n}). \tag{15}
$$

Let $S^\tau(e) := \min\{S(e), \nu(e)C(\bar{w}^\tau)\}$ denote the maximum SRD increment of edge e within the maximum cost $C(\bar{w}^\tau)$.

Let $\bar{E}^\tau_{\underline{=}} := \{e \in E | c(e)(\bar{w}^\tau(e) - w(e)) = C(\bar{w}^\tau)\}$ be the set of edges whose costs reach $C(\bar{w}^\tau)$. For edge $e_i \in \bar{E}^\tau$ and edge $e_j \in E \setminus \bar{E}^\tau$, where j is traversed according to the sequence $\{\gamma_n\}$, there are two replacement scenarios as follows.

(1) If $e_i \in \bar{E}^\tau_{\underline{=}}$, $S^\tau(e_j) = \min\{S(e_j), \nu(e_j)C(\bar{w}^\tau)\} \geq \nu(e_i)C(\bar{w}^\tau) = S^\tau(e_i)$ and $\nu(e_i) < \nu(e_j)$. Then e_i is replaced with e_j and update $\bar{E}^\tau = \bar{E}^\tau \setminus \{e_i\} \cup \{e_j\}$.

(2) If $e_i \in \bar{E}^\tau \setminus \bar{E}^\tau_{\underline{=}}$ and $S^\tau(e_j) > S^\tau(e_i)$, then e_i is replaced with e_j and $\bar{E}^\tau = \bar{E}^\tau \setminus \{e_i\} \cup \{e_j\}$.

After the replacement is done for all $e_i \in \bar{E}^\tau$ and $e_j \in E \setminus \bar{E}^\tau$, set $\bar{E}^{\tau+1} = \bar{E}^\tau$, the first iteration is finished.

In the second iteration, let $u^{\tau+1}(e) = \begin{cases} u(e), e \in \bar{E}^{\tau+1} \\ w(e), \text{otherwise} \end{cases}$, and call Algorithm 2: $[\bar{w}^{\tau+1}, C(\bar{w}^{\tau+1})]=$MCSDIPT$_\infty(T, n, w, u^{\tau+1}, c, D)$. The resulting $\bar{w}^{\tau+1}$ is a better feasible solution to problem (**MCSDIPTC$_\infty$**) than $\bar{w}^\tau$, with a corresponding objective value of $C(\bar{w}^\tau)$.

Theorem 7 *The solution $\bar{w}^{\tau+1}$ is feasible to problem (**MCSDIPTC$_\infty$**), and $C(\bar{w}^\tau) \geq C(\bar{w}^{\tau+1})$.*

Proof Without loss of generality, let $\bar{E}^{\tau+1} = \bar{E}^\tau \setminus \{e_i\} \cup \{e_j\}$, where $e_i \in \bar{E}^\tau$ and $e_j \in E \setminus \bar{E}^\tau$.

(1) **If** $e_i \in \bar{E}^\tau_{\underline{=}}$, then

$$S^\tau(e_j) \geq S^\tau(e_i), \tag{16}$$

$$\nu(e_i) < \nu(e_j). \tag{17}$$

(1.1) **Use induction to prove the feasibility of** $\bar{w}^{\tau+1}$. First, since $\sum_{e \in \bar{E}^0} S(e) = S_N \geq D$, the vector $\bar{w}^0$ is a feasible solution to the original problem.

Assume that the feasibility holds when $\tau = k$ then

$$\sum_{e \in \bar{E}^{\tau+1}} S(e) \geq \sum_{e \in \bar{E}^{\tau+1}} S^\tau(e) \geq \sum_{e \in \bar{E}^\tau} S^\tau(e) = \sum_{e \in \bar{E}^\tau \setminus \{e_i\}} S^\tau(e) + S^\tau(e_i) \geq \Delta D.$$

which means that $u^\tau(T) - w(T) \geq \Delta D$ and $u^\tau(T) \geq D$. Thus we can get a feasible solution after calling Algorithm 2, which satisfies $\bar{w}^{\tau+1}(T) \geq D$. Furthermore, $|\bar{E}^{\tau+1}| = |\bar{E}^\tau| = N$, and for any edge $e \in E$, $w(e) \leq \bar{w}^{\tau+1}(e) \leq u(e)$, so $\bar{w}^{\tau+1}$ is a feasible solution to problem (**MCSDIPTC$_\infty$**).

(1.2) **We show** $C(\bar{w}^\tau) \geq C(\bar{w}^{\tau+1})$ **by contradiction.** Assume $C(\bar{w}^\tau) < C(\bar{w}^{\tau+1})$. Then, $\sum_{e \in \bar{E}^\tau \setminus \{e_i\}} S^\tau(e) \leq \sum_{e \in \bar{E}^{\tau+1} \setminus \{e_j\}} S^{\tau+1}(e)$. Also, by Lemma 3, we have $\sum_{e \in \bar{E}^\tau} S^\tau(e) = \Delta D = \sum_{e \in \bar{E}^{\tau+1}} S^{\tau+1}(e)$ which implies that

$$\sum_{e \in \bar{E}^\tau \setminus \{e_i\}} S^\tau(e) + S^\tau(e_i) = \sum_{e \in \bar{E}^{\tau+1} \setminus \{e_j\}} S^{\tau+1}(e) + S^{\tau+1}(e_j)$$

Therefore,

$$S^\tau(e_i) \geq S^{\tau+1}(e_j) \geq S^\tau(e_j) \tag{18}$$

Combining equations (16) and (18), we have

$$S^\tau(e_i) = S^\tau(e_j) = S^{\tau+1}(e_j), \tag{19}$$

which implies that $\sum_{e \in \bar{E}^\tau \setminus \{e_i\}} S^\tau(e) = \sum_{e \in \bar{E}^{\tau+1} \setminus \{e_j\}} S^{\tau+1}(e)$.

Then, for any edge $e \in \bar{E}^\tau \setminus \{e_i\}$, $\bar{w}^{\tau+1}(e) = \bar{w}^\tau(e)$, combined with the assumption $C(\bar{w}^\tau) < C(\bar{w}^{\tau+1})$, we have

$$C(\bar{w}^\tau) = c(e_i)(\bar{w}^\tau(e_i) - w(e_i)) < c(e_j)(\bar{w}^{\tau+1}(e_j) - w(e_j)) = C(\bar{w}^{\tau+1}).$$

It follows from (19) that

$$\nu(e_i)C(\bar{w}^\tau) = S^\tau(e_i) = S^{\tau+1}(e_j) = \nu(e_j)C(\bar{w}^{\tau+1}),$$

which implies that

$$\nu(e_i) = \nu(e_j)\frac{C(\bar{w}^{\tau+1})}{C(\bar{w}^\tau)}.$$

Therefore, we have $\nu(e_i) > \nu(e_j)$, which contradicts equation (17).

(2) If $e_i \notin \bar{E}^\tau_=$, then $S^\tau(e_j) > S^\tau(e_i)$. Using the same argument as **(1.1)**, we can prove that $\bar{w}^{\tau+1}$ is a feasible solution for problem **(MCSDIPTC$_\infty$)**. Next, we show $C(\bar{w}^\tau) \geq C(\bar{w}^{\tau+1})$ by contradiction. Similar to **(1.2)**, we have equation (18). However, this contradicts the fact that $S^\tau(e_j) > S^\tau(e_i)$. $\square$

Then, update $\tau := \tau + 1$ and continue iterating until there is no edge in $E \setminus \bar{E}^\tau$ better than any edge in the current set $\bar{E}^\tau$. Denote the resulting set of edges by $\bar{E}^*$. Let $u^*(e) = \begin{cases} u(e), e \in \bar{E}^* \\ w(e), \text{otherwise} \end{cases}$. Then, call Algorithm 2:

$$[\tilde{w}, C(\tilde{w})] = MCSDIPT_\infty(T, n, w, u^*, c, D), \tag{20}$$

Theorem 8 *The solution $\tilde{w}$ obtained by (20) is an optimal solution to problem* **(MCSDIPTC$_\infty$)**.

Proof According to Theorem 7, the iteratively obtained solution $\tilde{w}$ is feasible.

Assume that $\tilde{w}$ is not optimal, but $\hat{w}$ is, such that $\max_{e \in E} c(e)(\hat{w}(e) - w(e)) < \max_{e \in E} c(e)(\tilde{w}(e) - w(e))$.
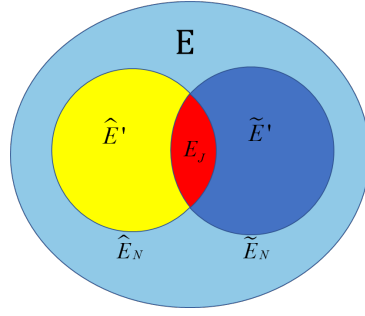
Let $\hat{E}_N = \{e \in E \mid \hat{w}(e) \neq w(e)\}$, $\tilde{E}_N = \{e \in E \mid \tilde{w}(e) \neq w(e)\}$, $\tilde{E}' = \tilde{E}_N \setminus \hat{E}_N$, $\hat{E}' = \hat{E}_N \setminus \tilde{E}_N$, and $E_J = \tilde{E}_N \cap \hat{E}_N$.

(1) If $\hat{E}_N = \tilde{E}_N$, then by the uniqueness of the optimal solution of the problem **(MCSDIPT$_\infty$)**, we have $\hat{w} = \tilde{w}$, which contradicts the assumption.

(2) If $\hat{E}_N \neq \tilde{E}_N$, since $C(\hat{w}) = \max_{e \in E} c(e)(\hat{w}(e) - w(e)) < \max_{e \in E} c(e)(\tilde{w}(e) - w(e)) = C(\tilde{w})$, we have $\max_{e \in E_J} c(e)(\hat{w}(e) - w(e)) \leq \max_{e \in E_J} c(e)(\tilde{w}(e) - w(e))$. Therefore, the total increment in SRD on E_J by $\hat{w}$ should be no more than that by $\tilde{w}$ and $\sum_{e \in E_J} |L(e)|(\hat{w}(e) - w(e)) \leq \sum_{e \in E_J} |L(e)|(\tilde{w}(e) - w(e))$. Moreover, since $\sum_{t_k \in Y} \hat{w}(P_k) \geq D$ and $\sum_{t_k \in Y} \tilde{w}(P_k) = D$, we have

$$\sum_{e \in \hat{E}'} |L(e)|(\hat{w}(e) - w(e)) \geq \sum_{e \in \tilde{E}'} |L(e)|(\tilde{w}(e) - w(e)). \tag{21}$$

Fig. 1 Inter-set relations

(2.1) If ">" holds in equation (21), then there exist $e_a \in \hat{E}'$ and $e_b \in \tilde{E}'$ such that $|L(e_a)|(\hat{w}(e_a) - w(e_a)) > |L(e_b)|(\tilde{w}(e_b) - w(e_b))$, which implies that $\hat{S}(e_a) = |L(e_a)| \min \left\{ u(e_a) - w(e_a), \frac{C(\hat{w})}{c(e_a)} \right\} > |L(e_b)| \min \left\{ u(e_b) - w(e_b), \frac{C(\tilde{w})}{c(e_b)} \right\} = \tilde{S}(e_b)$.

From the replacement case (2) of the iteration $\bar{E}^{\tau}$, it can be concluded that at the end of the iteration, for each edge in $\bar{E}$ under $\tilde{w}$, there is no edge in $E \setminus \bar{E}$ that can replace any edge in $\bar{E}$. However, the contradiction arises from the fact that $\tilde{S}(e_a) \geq \hat{S}(e_a) > \tilde{S}(e_b)$ and edge e_a satisfies the replacement requirement of edge e_b.

(2.2) If the equality holds in (21), it means that the two edge weight vectors achieve the same SRD on the same edge set E_J. Due to the uniqueness of the optimal solution of the problem (MCSDIPT$_\infty$), it follows that SRD increment by each edge in E_J must be equal. Therefore, the maximum cost edges of the two schemes are in $\hat{E}'$ and $\tilde{E}'$, respectively. Moreover, since $C(\hat{w}) < C(\tilde{w})$, it follows that $\hat{E}'$ is better than $\tilde{E}'$, and there must be at least one edge $e_a \in \hat{E}', e_b \in \tilde{E}'$, such that $\tilde{S}(e_a) > \tilde{S}(e_b)$ and edge e_a satisfies the replacement requirement of edge e_b, which leads to a contradiction.

Therefore, $\tilde{w}$ is the optimal solution to the original problem. $\qquad \square$

The following Algorithm 3 can be obtained from the above analysis to solve the problem (MCSDIPT$_\infty$).

Theorem 9 *Problem (**MCSDIPTC$_\infty$**) can be solved by Algorithm 3 in $O(Nn^2)$ time.*

Proof In Algorithm 3, we sort $S(e)$ and $\nu(e)$ in Lines 2 and 7, which takes a total of $O(n \log n)$ time. In the loop from Line 10 to 31, Line 13 calculates the optimal solution $\bar{w}$ and $C(\bar{w})$ for the problem (**MCSDIPT$_\infty$**) obtained by modifying only N edges in $\bar{E}$, which requires $O(n \log n)$ time according to Theorem 6. In the loop from Line 15 to 29, we iterate over the sets $\bar{E}^{\tau}$ and $E \setminus \bar{E}^{\tau}$, which takes Nn iterations. In each iteration we can update the selected edges in $O(n)$ time. Therefore, problem (**MCSDIPT$_\infty$**) can be solved by Algorithm 3 in $O(Nn^2)$ time. $\qquad \square$

Algorithm 3 $[\tilde{w}, C(\tilde{w})]$:=MCSDIPTC$_\infty(T, n, w, u, c, D, N)$

Require: A rooted tree $T(V, E)$, the number n of edges, the weight vector w, an upper bound vector u, a cost vector c, and given value D and N.
Ensure: The optimal vector $\tilde{w}$ and the cost $C(\tilde{w})$.
1: For each $e \in E$, determine the set $L(e)$, calculate $S(e) := |L(e)|(u(e) - w(e))$, and let $\Delta D = D - w(T)$.
2: Sort the $S(e)$ in a non-increasing order by (13).
3: $S_N := \sum_{i=1}^{N} S(e_{\beta_i})$.
4: **if** $S_N < \Delta D$ **then**
5: 　**return** "The problem is infeasible" and $\tilde{w} = [\,], C(\tilde{w}) = +\infty$.
6: **else**
7: 　Sort $\nu(e) := \frac{|L(e)|}{c(e)}, e \in E$ in a non-increasing order by (15).
8: 　$flg := 1, \tau := 0, \bar{E}^\tau := \{e_{\beta_1}, e_{\beta_2}, \ldots, e_{\beta_N}\}$.
9: 　Let $S^\tau(e) := \min\{S(e), \nu(e)C(\bar{w}^\tau)\}, e \in E$.
10: 　**while** $flg = 1$ **do**
11: 　　$flg := 0$.
12: 　　Let $u^\tau(e) := \begin{cases} u(e), e \in \bar{E}^\tau \\ w(e), \text{otherwise} \end{cases}$.
13: 　　$[\bar{w}^\tau, C(\bar{w}^\tau)]$=MCSDIPT$_\infty(T, n, w, u^\tau, c, D)$.
14: 　　$\bar{E}^\tau_= := \{e \in E | c(e)(\bar{w}^\tau(e) - w(e)) = C(\bar{w}^\tau)\}$.
15: 　　**for** $e_i \in \bar{E}^\tau$ **do**
16: 　　　**for** $e_j \in E \setminus \bar{E}^\tau$ in the order by (15) **do**
17: 　　　　**if** $e_i \in \bar{E}^\tau_=$ **then**
18: 　　　　　**if** $S^\tau(e_j) \geqq S^\tau(e_i)$ *and* $\nu(e_i) < \nu(e_j)$ **then**
19: 　　　　　　$\bar{E}^\tau := \bar{E}^\tau \setminus \{e_i\} \cup \{e_j\}, flg := 1$.
20: 　　　　　　**break**
21: 　　　　　**end if**
22: 　　　　**else**
23: 　　　　　**if** $S^\tau(e_j) > S^\tau(e_i)$ **then**
24: 　　　　　　$\bar{E}^\tau := \bar{E}^\tau \setminus \{e_i\} \cup \{e_j\}, flg := 1$.
25: 　　　　　　**break**
26: 　　　　　**end if**
27: 　　　　**end if**
28: 　　　**end for**
29: 　　**end for**
30: 　　Update $\bar{E}^{\tau+1} := \bar{E}^\tau$, $\tau := \tau + 1$.
31: 　**end while**
32: **end if**
33: Update $\tau := \tau - 1$.
34: **return** $[\tilde{w} := \bar{w}^\tau, C(\tilde{w}) := C(\bar{w}^\tau)]$.

---

3 Solve the problems (SDIPTC$_{BH}$) and (MCSDIPTC$_{BH}$) under weighted bottleneck Hamming distance

When the weighted bottleneck Hamming distance is applied to the upgrade cost, the problems (SDIPTC$_{BH}$) and (MCSDIPTC$_{BH}$) can be formulated as the following models (22) and (23), respectively.

$$\max_{\tilde{w}} \tilde{w}(T) := \sum_{t_k \in Y} \tilde{w}(P_k)$$

$(\mathbf{SDIPTC}_{BH})$ $\quad s.t.$ $\max_{e \in E} c(e)H(\tilde{w}(e), w(e)) \leq K,$ $\qquad (22)$

$$\sum_{e \in E} H(\tilde{w}(e), w(e)) \leq N,$$

$$w(e) \leq \tilde{w}(e) \leq u(e), \quad e \in E.$$

$$\min_{\tilde{w}} C(\tilde{w}) := \max_{e \in E} c(e)H(\tilde{w}(e), w(e))$$

$(\mathbf{MCSDIPTC}_{BH})$ $\quad s.t.$ $\sum_{t_k \in Y} \tilde{w}(P_k) \geq D,$ $\qquad (23)$

$$\sum_{e \in E} H(\tilde{w}(e), w(e)) \leq N,$$

$$w(e) \leq \tilde{w}(e) \leq u(e), \quad e \in E.$$

In this section, we first provide an algorithm with time complexity $O(n)$ for problem $(\mathbf{SDIPTC}_{BH})$. For the minimum cost problem $(\mathbf{MCSDIPTC}_{BH})$, we transform it into its sub-problem $(\mathbf{MCSDIPT}_{BH})$ and provide an algorithm with time complexity $O(n \log n)$.

3.1 An $O(n)$ time algorithm to solve the problem $(\mathbf{SDIPTC}_{BH})$

Similarly, we first consider solving the sub-problem $(\mathbf{SDIPT}_{BH})$ without the cardinality constraint, which can be formulated as follows.

$$\max_{\bar{w}} \bar{w}(T) := \sum_{t_k \in Y} \bar{w}(P_k)$$

$(\mathbf{SDIPT}_{BH})$ $\quad s.t.$ $\max_{e \in E} c(e)H(\bar{w}(e), w(e)) \leq K,$ $\qquad (24)$

$$w(e) \leq \bar{w}(e) \leq u(e), \quad e \in E.$$

Based on the characteristic of bottleneck Hamming distance, in order to maximize the objective function, we consider updating the edges $e \in E$ that satisfy $c(e) \leq K$. Thus, we give the following optimality condition.

Theorem 10 *The weight vector*

$$\bar{w}(e) = \begin{cases} u(e), & c(e) \leq K \\ w(e), & c(e) > K \end{cases} \qquad (25)$$

is an optimal solution to problem $(\mathbf{SDIPT}_{BH})$.

Proof Clearly, $\bar{w}$ is a feasible solution to problem ($\mathbf{SDIPT}_{BH}$). Suppose that $\bar{w}$ is not the optimal solution, but w' is. Then, there must be at least one edge $e_i \in E$ such that $w'(e_i) > \bar{w}(e_i)$. If $c(e_i) \leq K$, we have $w'(e_i) \leq u(e) = \bar{w}(e_i)$, which contradicts $w'(e_i) > \bar{w}(e_i)$. If $c(e_i) > K$, the cost should be at least $c(e_i)$, which is greater than the cost limit K, contradicting the assumption. Therefore, the $\bar{w}$ defined in (25) is the optimal solution to problem ($\mathbf{SDIPT}_{BH}$). $\qquad\square$

Based on Theorem 10, for any given T, n, w, u, c, K, we can obtain an optimal solution $\bar{w}$ and its corresponding optimal value $\bar{w}(T)$ for the problem ($\mathbf{SDIPT}_{BH}$). For convenience, we denote this subroutine as:

$$[\bar{w}, \bar{w}(T)] = SDIPT_{BH}(T, n, w, u, c, K) \qquad (26)$$

Obviously, this problem only requires one traversal of $O(n)$, so we have the following conclusion.

Theorem 11 *The subroutine (26) can solve problem ($\mathbf{SDIPT}_{BH}$) in $O(n)$ time.*

Then we consider the original problem ($\mathbf{SDIPTC}_{BH}$). Similar to problem ($\mathbf{SDIPT}_{BH}$), we can partition E into two parts $\tilde{E}_{\leq} = \{e \in E \mid c(e) \leq K\}$ and $E \setminus \tilde{E}_{\leq}$. Then, we select elements in $\tilde{E}_{\leq}$ according to the value of $S(e) := |L(e)|(u(e) - w(e))$. Depending on the relationship between $|\tilde{E}_{\leq}|$ and the cardinality constraint N, we divide the problem into the following two cases and solve them. The first case is rather simple.

Lemma 6 *When $|\tilde{E}_{\leq}| \leq N$, the weight vector* $\tilde{w}(e) := \begin{cases} u(e), & e \in \tilde{E}_{\leq} \\ w(e), & e \in E \setminus \tilde{E}_{\leq} \end{cases}$ *is an optimal solution to problem ($\mathbf{SDIPTC}_{BH}$).*

When $|\tilde{E}_{\leq}| > N$, we first prioritize the edges with larger $S(e)$ values in the set $\tilde{E}_{\leq}$. We use the *Selection* algorithm $\tilde{S}_{\leq}^N := Selection(\tilde{S}_{\leq}, N)$ [7] to select the N-th largest element $\tilde{S}_{\leq}^N$ from $\tilde{S}_{\leq} = \left\{ S(e) | e \in \tilde{E}_{\leq} \right\}$. Based on this, we divide the set E into $\tilde{E}_N := \left\{ e \in \tilde{E}_{\leq} | S(e) \geq \tilde{S}_{\leq}^N \right\}$ and $E \setminus \tilde{E}_N$.

Theorem 12 *When $|\tilde{E}_{\leq}| > N$, the weight vector*

$$\tilde{w}(e) := \begin{cases} u(e), & e \in \tilde{E}_N \\ w(e), & e \in E \setminus \tilde{E}_N \end{cases} \qquad (27)$$

is an optimal solution to problem ($\mathbf{SDIPTC}_{BH}$).

Proof Obviously, $\tilde{w}$ is a feasible solution for problem ($\mathbf{SDIPTC}_{BH}$). Suppose $\tilde{w}$ is not optimal, while $\hat{w}$ is. Then we have $\hat{w}(T) > \tilde{w}(T)$ and there exists at least one edge e_k such that $\hat{w}(e_k) > \tilde{w}(e_k)$.

If $e_k \in \tilde{E}_N$, we have $\hat{w}(e_k) > \tilde{w}(e_k) = u(e_k)$, which contradicts the upper bound constraint.

If $e_k \in E \setminus \tilde{E}_N$ and $e_k \notin \tilde{E}_{\leq}$, we have $c(e_k) > K$, which contradicts the updated cost limit.

If $e_k \in E \setminus \tilde{E}_N$ and $e_k \in \tilde{E}_{\leq}$, since $|\tilde{E}_{\leq}| > N$, there must exist an edge $e_t \in \tilde{E}_N$ such that $\tilde{w}(e_t) > \hat{w}(e_t)$. Moreover, since $S(e) > S(e_k)$ for all $e \in \tilde{E}_N$, we have $S(e_t) > S(e_k)$. In this case, we can construct a new edge weight vector as follows:

$$w'(e) := \begin{cases} u(e), & e = e_t \\ w(e), & e = e_k \\ \hat{w}(e), & \text{otherwise} \end{cases}$$

and we have $\hat{w}(T) < w'(T)$, which contradicts the assumption. $\qquad\square$

Based on Theorem 12, we propose Algorithm 4 for problem ($\mathbf{SDIPTC}_{BH}$).

Algorithm 4 $[\tilde{w}, \tilde{w}(T)] = \mathrm{SDIPTC}_{BH}(T, n, w, u, c, K, N)$

Require: A rooted tree $T(V, E)$, the number n of edges, the weight vector w, an upper bound vector u, a cost vector c, and given values K and N.
Ensure: The optimal vector $\tilde{w}$ and the SRD $\tilde{w}(T)$.
1: $\tilde{E}_{\leq} := \{e \in E \mid c(e) \leq K\}$
2: **if** $|\tilde{E}_{\leq}| > N$ **then**
3: $\quad$ $S(e) := |L(e)|(w'(e) - w(e)), \tilde{S}_{\leq} := \left\{ S(e) | e \in \tilde{E}_{\leq} \right\}.$
4: $\quad$ $\tilde{S}_{\leq}^N := Selection(\tilde{S}_{\leq}, N), \tilde{E}_N := \left\{ e \in \tilde{E}_{\leq} | S(e) \geq \tilde{S}_{\leq}^N \right\}.$
5: $\quad$ Obtain $\tilde{w}$ by (27).
6: **else**
7: $\quad$ $[\tilde{w}, \tilde{w}(T)] = SDIPT_{BH}(T, n, w, u, c, K)$
8: **end if**
9: **return** $[\tilde{w}, \tilde{w}(T)]$.

Similar to the proof of Theorem 4, we have the following lemma.

Lemma 7 *Problem ($\mathbf{SDIPTC}_{BH}$) can be solved by Algorithm 4 in $O(n)$ time.*

3.2 An $O(n \log n)$ time algorithm to solve the problem ($\mathbf{MCSDIPTC}_{BH}$)

Concerning problem ($\mathbf{MCSDIPTC}_{BH}$), similarly, we first consider its subproblem ($\mathbf{MCSDIPT}_{BH}$) without cardinality constraint, which can be formulated as follows.

$$\min_{\bar{w}} C(\bar{w}) := \max_{e \in E} c(e) H(\bar{w}(e), w(e))$$

$$(\mathbf{MCSDIPT}_{BH}) \quad s.t. \ \sum_{t_k \in Y} \bar{w}(P_k) \geq D, \tag{28}$$

$$w(e) \leq \bar{w}(e) \leq u(e), \quad e \in E.$$

According to the relationship between $u(T)$ and D, as well as $w(T)$ and D, we discuss the following two cases of the problem separately.

Lemma 8 *When $u(T) < D$, problem (**MCSDIPT**$_{BH}$) is infeasible.*

Lemma 9 *If $w(T) \geq D$, then w is an optimal solution to (**MCSDIPT**$_{BH}$).*

When $u(T) \geq D > w(T)$, we first sort the edges in an ascending order by their $c(e)$ values and renumber them in the following sequence $\{e_{\alpha_i}\}$

$$c(e_{\alpha_1}) \leq c(e_{\alpha_2}) \leq \cdots \leq c(e_{\alpha_n}) \tag{29}$$

Let $S(e) = |L(e)|(u(e) - w(e))$, and we then construct an auxiliary function

$$g(k) = \sum_{i=1}^{k} S(e_{\alpha_i}) \tag{30}$$

which represents the SDR added when updating the first k edges in the sequence $\{e_{\alpha_i}\}$. We use a binary search method to iteratively determine k^* such that $g(k^*) < \Delta D \leq g(k^* + 1)$ and give the following conclusion.

Theorem 13 *If $u(T) \geq D > w(T)$, then the weight vector $\bar{w}$ defined by:*

$$\bar{w}(e) = \begin{cases} u(e), & e \in \bar{E}_{\leq} \\ w(e), & e \in E \backslash \bar{E}_{\leq} \end{cases} \tag{31}$$

*is an optimal solution to (**MCSDIPT**$_{BH}$). Here, $\bar{E}_{\leq} = \{e_{\alpha_i} \in E \mid i = 1, 2, \ldots, k^* + 1\}$.*

Proof It is obvious that $\bar{w}$ is feasible to the problem (**MCSDIPT**$_{BH}$). Suppose $\bar{w}$ is not optimal, but w' is. Then $C(w') < C(\bar{w})$. Let e_{α_j} and e_{α_i} be the edges in the sequence $\{e_{\alpha_k}\}$ with the maximum cost in w' and $\bar{w}$ respectively. Then $c(e_{\alpha_j}) = C(w') < C(\bar{w}) = c(e_{\alpha_i})$, so $g(j) < g(i)$. Since i is the smallest index satisfying $g(i) \geq \Delta D$, thus $g(j) < \Delta D$, which contradicts the optimality of w'. $\qquad\square$

Based on Theorem 13, we present the following Algorithm 5 to solve the problem (**MCSDIPT**$_{BH}$).

The following lemma follows from the time complexity of Theorem 6.

Lemma 10 *Problem (**MCSDIPT**$_{BH}$) can be solved by Algorithm 5 in $O(n \log n)$ time.*

Now we can consider solving the cardinality problem (**MCSDIPTC**$_{BH}$). First, we need to determine the infeasibility by Lemma 4 similarly.

Lemma 11 *When $S_N < \Delta D$, problem (**MCSDIPTC**$_{BH}$) is infeasible.*

Algorithm 5 $[\bar{w}, C(\bar{w})] := \text{MCSDIPT}_{BH}(T, n, w, u, c, D)$

Require: A rooted tree $T(V, E)$, the number n of edges, the weight vector w, an upper bound vector u, a cost vector c, and a given value D.
Ensure: The optimal vector $\bar{w}$ and the cost $C(\bar{w})$.
1: For each $e \in E$, determine the set $L(e)$, calculate $F(e) := c(e)(u(e) - w(e))$, and let $\Delta D := D - w(T)$.
2: $u(T) := \sum_{t_k \in Y} u(P_k)$.
3: **if** $u(T) < D$ **then**
4:    **return** "The problem is infeasible" and $\bar{w} = [\,], C(\bar{w}) = +\infty$.
5: **else if** $w(T) >= D$ **then**
6:    **return** $[w, 0]$.
7: **else**
8:    Sort the cost vector $c(e)$ in a non-decreasing order in (29).
9:    Set the initial values as $a := 1$, $b := n$, and $k^* := 0$.
10:    **while** $b - a > 1$ and $k^* = 0$ **do**
11:      $k := \left\lfloor \frac{a+b}{2} \right\rfloor$.
12:      Calculate $g(k)$ and $g(k+1)$ using (30).
13:      **if** $g(k) < \Delta D \leq g(k+1)$ **then**
14:        $k^* := k$.
15:      **else if** $g(k+1) < \Delta D$ **then**
16:        $a := k$.
17:      **else**
18:        $b := k$.
19:      **end if**
20:    **end while**
21:    Compute $\bar{w}$ by (31) and $C(\bar{w}) = c(e_{\alpha_{k^*+1}})$.
22:    **return** $[\bar{w}, C(\bar{w})]$.
23: **end if**

When $S_N \geq \Delta D$, we first run Algorithm 5:

$$[w', C(w')] = \text{MCSDIPT}_{BH}(T, n, w, u, c, D)$$

to obtain the optimal solution to its sub-problem $(\textbf{MCSDIPT}_{BH})$. This leads to the following lemma.

Lemma 12 *If $S_N \geq \Delta D$ and $\sum_{e \in E} H(w'(e), w(e)) \leq N$, then w' is the optimal solution to problem $(\textbf{MCSDIPTC}_{BH})$.*

If $\sum_{e \in E} H(w'(e), w(e)) > N$, sort the edges $e \in E$ in a non-decreasing order according to their $c(e)$ values by (29). For any $N \leq k \leq n$, let $E_k := \{e_{\alpha_1}, e_{\alpha_2}, \ldots, e_{\alpha_k}\}$, $S_{E_k} := \{S(e) | e \in E_k\}$, call $S_k^N := Selection(S_{E_k}, N)$ [7]. Let $E_k^N := \{e \in E_k | S(e) \geq S_k^N\}$, then

$$h(k) = \sum_{e \in E_k^N} S(e) \tag{32}$$

is the maximum SDR that can be increased by updating N edges among the first k edges in the sequence $\{e_{\alpha_i}\}$.

Next, we can use binary search for iteration to find the smallest k^* such that $h(k^*) \geq \Delta D$, and let the updated edge set at this point be $E_{k^*}^N$.

Theorem 14 *If $\sum_{e \in E} H(w'(e), w(e)) > N$, the weight vector*

$$\tilde{w}(e) = \begin{cases} u(e), & e \in E_{k^*}^N \\ w(e), & e \in E \setminus E_{k^*}^N \end{cases} \tag{33}$$

*is an optimal solution to problem (**MCSDIPTC**$_{BH}$).*

Proof It is obvious that $\tilde{w}$ is a feasible solution to problem (**MCSDIPTC**$_{BH}$). Assume that $\tilde{w}$ is not optimal, and $\hat{w}$ is an optimal solution with $C(\hat{w}) < C(\tilde{w})$. Suppose that the N updated edges in $\hat{w}$ correspond to the sequence $\{e_{\alpha_i}\}$ satisfying $c(e_{\alpha_{p_1}}) \le c(e_{\alpha_{p_2}}) \le \cdots \le c(e_{\alpha_{p_N}})$. It is clear that $h(p_N) \ge \Delta D$ and $p_N < k^*$ since $\hat{w}$ satisfies $c(e_{\alpha_{p_N}}) = C(\hat{w}) < C(\tilde{w}) = c(e_{\alpha_{k^*}})$. This contradicts the assumption that k^* is the smallest value satisfying $h(k^*) \ge \Delta D$. $\square$

We present the following Algorithm 6 for solving (**MCSDIPTC**$_{BH}$) based on Theorem 14.

Algorithm 6 $[\tilde{w}, C(\tilde{w})] := \text{MCSDIPTC}_{BH}(T, n, w, u, c, D, N)$

Require: A rooted tree $T(V, E)$, the number n of edges, the weight vector w, an upper bound vector u, a cost vector c, and two given values D and N.
Ensure: The optimal vector $\tilde{w}$ and the cost $C(\tilde{w})$.
1: For each e, determine the set $L(e)$, calculate $S(e) := |L(e)|(u(e) - w(e))$, and let $\Delta D := D - w(T)$.
2: Sort $S(e)$ in a non-increasing order by (13).
3: $S_N := \sum_{i=1}^N S(e_{\beta_i})$.
4: **if** $S_N < \Delta D$ **then**
5:     **return** "The problem is infeasible" and $\tilde{w} = [\ ], C(\tilde{w}) = +\infty$.
6: **end if**
7: $[w', C(w')] := \text{MCSDIPT}_{BH}(T, n, w, u, c, D)$.
8: **if** $\sum_{e \in E} H(w'(e), w(e)) \le N$ **then**
9:     **return** $[w', C(w')]$.
10: **else**
11:     Sort $c(e)$ in a non-increasing order by (29).
12:     $a := N, b := n, k^* := 0$
13:     **while** $a < b$ *and* $k^* = 0$ **do**
14:         $k := \left\lfloor \frac{a+b}{2} \right\rfloor$.
15:         Compute E_k^N, $h(k)$ and $h(k-1)$ by (32).
16:         **if** $h(k-1) < \Delta D \le h(k)$ **then**
17:             $k^* := k, E_{k^*}^N := E_k^N$.
18:         **else if** $h(k) < \Delta D$ **then**
19:             $a := k$.
20:         **else**
21:             $b := k$.
22:         **end if**
23:     **end while**
24:     Compute $\tilde{w}$ by (33) and $C(\tilde{w}) := c(e_{p_{k^*}})$.
25:     **return** $[\tilde{w}, C(\tilde{w})]$.
26: **end if**

Theorem 15 *Problem (**MCSDIPTC**$_{BH}$) can be solved by Algorithm 6 in $O(n \log n)$ time.*

Proof In Algorithm 6, we sort $S(e)$ and $c(e)$ in Lines 2 and 11, which takes $O(n \log n)$ time. Furthermore, in the loop from Line 13 to 23, we use the $Selection(E_k^N, N)$ algorithm to calculate $h(k)$ in Line 15, which takes $O(n)$ time [7], and the whole loop is searched using binary search, so it can be completed in $O(n \log n)$ time. Therefore, problem ($\mathbf{MCSDIPTC}_{BH}$) can be solved by Algorithm 6 in $O(n \log n)$ time.   □

4 Numerical Experiments

4.1 An example to show the process of Algorithm 3 and 6

For the better understanding of Algorithm 3 and 6, Example 1 is given to show the detailed computing process.

Example 1 As shown in Fig.2 and 3, let $V := \{s, v_1, \ldots, v_{19}\}$, $E := \{e_1, \ldots, e_{19}\}$, the corresponding c, w, L, u are shown on edges with different colors. Now we have $w(T) := 407$, $u(T) := 939$. Suppose the given values are $D := 460$ and $N := 4$.

4.1.1 Execution of Alg. 3

(1) By defining $S(e) := |L(e)|(u(e) - w(e))$, the values of $S(e)$ are sorted in a non-increasing order, resulting in $S_N := 296$. As $S_N > \Delta D := 53$, the problem is feasible and has a valid solution.

(2) The values of $\nu(e)$ are calculated and sorted in non-increasing order.

(3) The algorithm is initialized with $flg := 1$, $\tau := 0$, $\bar{E}^0 := \{e_1, e_3, e_4, e_{13}\}$.

(4) The cycle process is presented in the following table 2.

Table 2 Process of calling Algorithm 3.

τ	$C(\bar{w}^\tau)$	$\bar{E}^\tau$	$\bar{w}(\bar{E}^\tau)$
0	44.2580	$\{e_1, e_3, e_4, e_{13}\}$	(9.3294, 11.9505, 8.6882, 13.6034)
0	44.0455	$\{e_1, e_2, e_4, e_{13}\}$	(9.3182, 7.0000, 8.6705, 13.5909)
0	41.1018	$\{e_1, e_2, e_5, e_{13}\}$	(9.1633, 7.0000, 20.1102, 13.4178)
0	33.5776	$\{e_1, e_2, e_5, e_6\}$	(8.7672, 7.0000, 19.3578, 18.7155)
1	33.5776	$\{e_1, e_2, e_5, e_6\}$	(8.7672, 7.0000, 19.3578, 18.7155)

Finally, we get the optimal value 33.5776. It is worth noting that in the first cycle, we updated the edge set three times with different values $C(\bar{w}^\tau)$ on Table 2. Algorithm 2 was called only once at the beginning. We show the upgraded weights of 4 edges $\{e_1, e_2, e_5, e_6\}$ on the right figure in Fig. 2.

4.1.2 Execution of Alg. 6

(1) Similarly, sort $S(e)$ in non-increasing order, resulting in $S_N := 296$. As $S_N > \Delta D := 53$, the problem is feasible and has a valid solution.

Fig. 2 The left figure shows the weights $(e_i, c_i, w_i, L_i, u_i)$, where $L_i := |L(e_i)|$. The right figure shows the modified edges in Alg. 3.

(2) $[w', C(w')] :=\mathrm{MCSDIPT}_{BH}(T, n, w, u, c, D)$, and we get $k^* := 2$, so
$\bar{w}(e) = \begin{cases} u(e), & e \in \{e_6, e_{10}, e_{16}\} \\ w(e), & else \end{cases}$ and $w'(T) := 462 > D := 460$.

(3) $\sum_{e \in E} H(w'(e), w(e)) := 3 \le 4$, we get $C(\tilde{w}) = C(w') := 5$.

Fig. 3 The left figure shows the weights $(e_i, c_i, w_i, L_i, u_i)$. The right figure shows the modified edges in Alg. 6.

4.2 Computational experiments

We present the numerical experimental results for formulas (7), (26), and algorithms 1-6 in Table 3. These programs were coded in Matlab2021a and ran on an Intel(R) Core(TM) i7-10875H CPU @ 2.30GHz and 2.30 GHz machine running Windows 11. We tested these algorithms on six randomly generated trees with vertex numbers ranging from 1000 to 50000. We randomly generated

the vectors u, c and w such that $0 \leq w \leq u$ and $c > 0$. We randomly generated K, D and N for the problems (**SDIPT**), (**MCSDIPT**), (**SDIPTC**, **MCSDIPTC**), respectively. For each randomly generated tree T, vectors u, c, w, and values D and K, we tested formulas (7), (26), and algorithms 1-6, whose average, maximum and minimum CPU time are denoted by T_i, T_i^{max} and T_i^{min}, respectively, where $i = 1, \cdots, 8$.

From Table 3, we can see that Algorithm 3 is the most time-consuming due to the $O(Nn^2)$ complexity of its program and the uncertainty of its iteration number. Formulas (7) and (26) are relatively simple and require only one traversal of the vector, so they are less time-consuming.

Overall, these algorithms are all very effective and follow their respective time complexities well. When n is small, the time differences among the three algorithms are relatively small, but as n increases, the differences between the algorithms become more pronounced.

Table 3 Performance of Algorithms

Complexity	n	1000	5000	10000	30000	50000
$O(n)$	T_1	0.0002	0.0003	0.0005	0.0013	0.0019
	T_1^{max}	0.0005	0.0021	0.0038	0.0107	0.0186
	T_1^{min}	0.0001	0.0001	0.0002	0.0007	0.0014
$O(n)$	T_2	0.0008	0.0037	0.0084	0.0251	0.0393
	T_2^{max}	0.0022	0.0079	0.0095	0.0274	0.0428
	T_2^{min}	0.0003	0.0017	0.0029	0.0160	0.0251
$O(n \log n)$	T_3	0.0013	0.0062	0.0140	0.0679	0.1477
	T_3^{max}	0.0075	0.0122	0.0217	0.0805	0.1802
	T_3^{min}	0.0010	0.0036	0.0109	0.0561	0.1365
$O(Nn^2)$	T_4	0.0004	0.0612	0.3575	11.24	47.66
	T_4^{max}	0.0013	0.0682	0.4720	24.36	62.34
	T_4^{min}	0.0002	0.0446	0.2654	6.331	40.98
$O(n)$	T_5	0.0007	0.0037	0.0081	0.0246	0.0403
	T_5^{max}	0.0023	0.0071	0.0100	0.0267	0.0429
	T_5^{min}	0.0004	0.0031	0.0069	0.0221	0.0387
$O(n)$	T_6	0.0012	0.0057	0.0114	0.0320	0.0715
	T_6^{max}	0.0063	0.0098	0.0135	0.0493	0.0974
	T_6^{min}	0.0007	0.0042	0.0096	0.0262	0.0638
$O(n \log n)$	T_7	0.0016	0.0067	0.0150	0.0683	0.1578
	T_7^{max}	0.0064	0.0119	0.0413	0.0893	0.1917
	T_7^{min}	0.0008	0.0052	0.0120	0.0602	0.1424
$O(n \log n)$	T_8	0.0021	0.0097	0.0209	0.0781	0.1792
	T_8^{max}	0.0077	0.0141	0.0425	0.0914	0.2024
	T_8^{min}	0.0012	0.0085	0.0187	0.0774	0.1623

5 Conclusion and further research

This paper proposes two linear time greedy algorithms for problems (**SDIPT**$_\infty$) under weighted l_∞ norm and the problem (**SDIPT**$_{BH}$) under bottleneck Hamming distance, respectively. For their relevant minimum cost problems

($\mathbf{MCSDIPT}_\infty$) and ($\mathbf{MCSDIPT}_{BH}$), two $O(n \log n)$ time algorithms are proposed based on binary methods, respectively. In addition, this paper considers the sum of root-leaf distance interdiction problem with cardinality constraints by upgrading edges on trees($\mathbf{SDIPTC}$) and its relevant minimum cost problem ($\mathbf{MCSDIPTC}$). Then two binary search algorithms are proposed within $O(n \log n)$ time for the problems ($\mathbf{SDIPTC}$) under weighted l_∞ norm and weighted bottleneck Hamming distance, respectively. For problems ($\mathbf{MCSDIPTC}$), two binary search algorithms within $O(Nn^2)$ and $O(n \log n)$ under weighted l_∞ norm and weighted bottleneck Hamming distance are proposed, respectively.

We validate the effectiveness of the proposed algorithms through numerical experiments. As future work, we aim to extend our approach to interdiction problems on source-sink path length, maximum flow, and minimum spanning tree under different measurements for general graphs.

Data availability Data sharing is not applicable to this article as our datasets were generated randomly.

Declarations

Competing interests The authors declare that they have no competing interest.

References

1. Albert, R, Jeong, H, Barabási, A (2000) Error and attack tolerance of complex networks. Nature, 406(6794), 378-382.
2. 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
3. Ball MO, Golden BL, Vohra RV (1989) Finding the most vital arcs in a network. Oper Res Lett 8(2):73–76
4. 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
5. 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
6. Corley HW, Sha DY (1982) Most vital links and nodes in weighted networks. Oper Res Lett 1:157–161
7. Cormen TH, Leiserson CE, Rivest RL, et al. (2022) Introduction to algorithms. 4rd edn. MIT press.
8. Israeli E, Wood RK (2002). Shortest-path network interdiction. Networks, 40(2): 97-111.
9. 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
10. Lei Y, Shao H, Wu T, et al. (2023). An accelerating algorithm for maximum shortest path interdiction problem by upgrading edges on trees under unit Hamming distance. Optim Lett 17, 453–469.

11. 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
12. Zhang Q, Guan XC, Jia JH, et al. (2022). The sum of root-leaf distance interdiction problem by upgrading edges/nodes on trees. J Comb Optim, 44(1), 74-93.
13. Zhang Q, Guan XC, Pardalos PM (2021a) Maximum shortest path interdiction problem by upgrading edges on trees under weighted l1 norm. J Global Optim 79(4):959–987
14. 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