mathematics

MDPI

Article

Minimum-Cost Shortest-Path Interdiction Problem Involving Upgrading Edges on Trees with Weighted l_∞ Norm

Qiao Zhang [1] and Xiao Li [2,*]

[1] Aliyun School of Big Data, Changzhou University, Changzhou 213164, China; qiaozhang@cczu.edu.cn
[2] School of Industrial Engineering, Purdue University, West Lafayette, IN 47907, USA
* Correspondence: li5868@purdue.edu

Abstract

Network interdiction problems involving edge deletion on shortest paths have wide applications. However, in many practical scenarios, the complete removal of edges is infeasible. The minimum-cost shortest-path interdiction problem for trees with the weighted l_∞ norm (MCSPIT$_\infty$) is studied in this paper. The goal is to upgrade selected edges at minimum total cost such that the shortest root–leaf distance is bounded below by a given value. We designed an $O(n \log n)$ algorithm based on greedy techniques combined with a binary search method to solve this problem efficiently. We then extended the framework to the minimum-cost shortest-path double interdiction problem for trees with the weighted l_∞ norm, which imposes an additional requirement that the sum of root–leaf distances exceed a given threshold. Building upon the solution to (MCSPIT$_\infty$), we developed an equally efficient $O(n \log n)$ algorithm for this variant. Finally, numerical experiments are presented to demonstrate both the effectiveness and practical performance of the proposed algorithms.

Keywords: network interdiction problem; double interdiction problem; upgrading edges; shortest path; tree; weighted l_∞ norm

MSC: 90C27



Academic Editors: Andrea Scozzari and Iliya Bouyukliev

Received: 2 September 2025
Revised: 29 September 2025
Accepted: 4 October 2025
Published: 7 October 2025

Citation: Zhang, Q.; Li, X. Minimum-Cost Shortest-Path Interdiction Problem Involving Upgrading Edges on Trees with Weighted l_∞ Norm. *Mathematics* **2025**, *13*, 3219. https://doi.org/10.3390/math13193219

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Critical edge deletion in network interdiction problems typically involves two entities, a decision-maker (defender) and a disruptor (attacker), competing in a specific Stackelberg game. The decision-maker operates the network to optimize their objective function, such as shortest path, maximum flow, or minimum cost flow, whereas the disruptor attempts to degrade this performance by interdicting several arcs with limited resources [1].

The interdiction problem was first applied to the shortest-path problem by Corley and Sha in [2] in the shortest-path interdiction problem (denoted by (SPIP)). Ball et al. [3] proved that this problem is NP-hard. Israeli et al. [4] transformed this problem into a bilevel mixed integer programming problem. Khachiyan et al. [5] proved that this problem does not admit an approximation algorithm with a performance ratio of 2, which is the current best result. Chen et al. [6] provided an $O(m + n \log n)$ algorithm for the problem on undirected networks when $K = 1$. Nardelli et al. [7] improved the time complexity to $O(m\alpha(m, n))$, where α is the inverse Ackermann function, and m and n are the number of edges and nodes, respectively. Building upon classical deterministic frameworks, recent advances in SPIP have expanded into stochastic, temporal, and information-asymmetric settings while simultaneously enhancing our understanding of the problem's computational complexity

and solution methods. Henke et al. [8] established the NP-hardness of the bilevel SPIP, clearly delineating its intractability. Similarly, Boeckmann et al. [9] demonstrated that the temporal version remains NP-hard even when interdicting a single edge. To address uncertainty, Punla-Green et al. [10] and Azizi et al. [11] both developed models for SPIP with incomplete information; the former employed a max–min framework for asymmetric uncertainty, while the latter adopted a robust optimization approach. Nguyen et al. [12] further enriched the stochastic domain by incorporating the conditional value-at-risk (CVaR) metric to model asymmetric risk aversion in an uncertain environment. Beyond traditional mathematical programming, Huang et al. [13] pioneered a novel approach by solving the SPIP using reinforcement learning, demonstrating the potential of AI-based methods. Finally, previous research also clarifies strategic choices as Holzmann et al. [14] laid a key complexity foundation by proving that the problem with randomized interdiction strategies is generally NP-hard but can be solved in pseudo-polynomial time, while Borrero et al. [15] tackled sequential decision-making in a stochastic environment, proposing an approximate dynamic programming algorithm for optimal interdiction policies. Collectively, these studies significantly broaden the modeling landscape and clarify the hardness frontiers of SPIP under various complex conditions.

Currently, the majority of research in this area focuses on the SPIP under the assumption of critical edge deletion. However, factors such as rapid system recovery mechanisms and the presence of backup pathways often make complete edge deletion difficult or even infeasible in practical scenarios. For instance, in a military network scenario, available mobile forces may only be capable of interdicting support between K pairs of adjacent enemy positions. The objective is to choose which K pairs to disrupt so as to maximize the enemy's support time [16]. This represents a typical application of the shortest-path interdiction problem in military operations. Classical interdiction models focus on completely severing support between enemy positions, that is, deleting edges between the K selected pairs to make communication impossible. However, in reality, adjacent positions may maintain connectivity through multiple alternative means, such as sea, rail, or air transport. As a result, interdiction efforts may only partially disrupt certain support channels rather than entirely eliminating all connections. Another example is shown in Figure 1 [17]: Once the disruptor attacks node k_{10}, a fire will break out at that node. The disruptor aims to delay the fire truck's arrival by maximizing the shortest path from node k_1 to k_{10} with a limited interdiction budget. In this case, the disruptor can only increase the length of certain arcs rather than deleting them entirely. Correspondingly, the decision-maker must proactively identify vulnerable arcs and preemptively plan safer alternative routes to ensure the fire truck can reach the incident as quickly as possible. Recent work on the quickest path reliability problem underlines a similar focus on optimizing path performance under resource constraints [18]. This connection highlights the broad applicability of our interdiction and upgrade framework to diverse network optimization scenarios, such as ensuring reliable flow transmission.

Therefore, the classical SPIP with critical edge deletion has inherent limitations. To overcome these, refs. [19,20] introduced the concept of upgrading edges and investigated the maximum shortest-path interdiction problem involving upgrading edges on trees (denoted by MSPIT), which can be defined as follows. Let $T = (V, E, w)$ be an edge-weighted tree rooted at v_1, where $V := \{v_1, v_2, \cdots, v_n\}$, and $E := \{e_2, e_3, \cdots, e_n\}$ be the sets of nodes and edges, respectively. Let $Y := \{t_1, \cdots, t_r\}$ be the set of leaves. Let $w(e)$ and $u(e)$ be the original weight and upper bounds of the upgraded weight for an edge $e \in E$, respectively, where $w(e) \leq u(e)$. Let $c(e) > 0$ be a cost to upgrade the edge $e \in E$. $\Delta w(e) := u(e) - w(e)$ denotes the deviation between $u(e)$ and $w(e)$. P_{v_i,v_j} denotes the unique path from v_i to v_j on the tree T and $P_k := P_{v_1,t_k}$ the "root–leaf path" of the leaf node

$t_k \in Y$ in brief. Define $f(P_k) := \sum_{e \in P_k} f(e)$ as the "root–leaf distance" for a weight vector f and a root–leaf path P_k, where $t_k \in Y$. The problem MSPIT involves finding an upgrade scheme $\bar{w}$ to maximize the shortest root–leaf distance of the tree on the premise that the total upgrade cost with some norm is upper bounded by a given value M. The mathematical model can be stated as follows.

$$\text{(MSPIT)} \quad \begin{array}{ll} \max_{\bar{w}} & \min\limits_{t_k \in Y} \bar{w}(P_k) \\ s.t. & \|\bar{w}(e) - w(e)\| \leq M, \\ & w(e) \leq \bar{w}(e) \leq u(e), e \in E. \end{array} \tag{1}$$

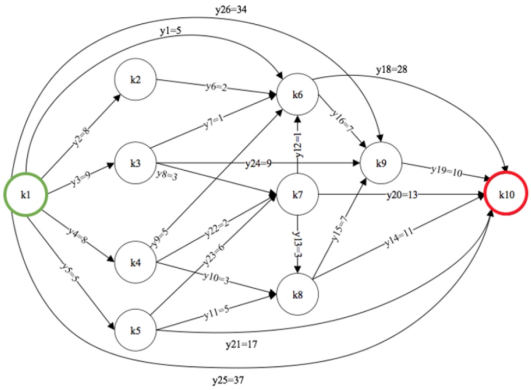


Figure 1. A terrorist network consisting of 10 nodes and 26 arcs, taken from [17], illustrating a shortest-path interdiction scenario. The disruptor aims to delay the fire truck's arrival by increasing the travel time (length) of arcs from node k_1 to the target node k_{10}, subject to a limited budget. The interdiction cost for each relevant arc is annotated. This scenario motivates the study of interdiction models based on partial edge degradation (weight increase) rather than complete edge deletion.

The corresponding minimum-cost shortest-path interdiction problem involving upgrading edges on trees (denoted by MCSPIT) entails upgrading some edges to minimize the total cost with measurement on the premise that the shortest root–leaf distance of the tree is bounded below by a given value D. The mathematical model can be stated as follows.

$$\text{(MCSPIT)} \quad \begin{array}{ll} \max_{\hat{w}} & \|\hat{w}(e) - w(e)\| \\ s.t. & \hat{w}(P_k) \geq D, t_k \in Y, \\ & w(e) \leq \hat{w}(e) \leq u(e), e \in E. \end{array} \tag{2}$$

Using the weighted l_1 norm, Zhang et al. [16] proposed primal–dual algorithms with time complexity $O(n^2)$ for both problems and with the unit l_1 norm and developed linear-time algorithms. Using the weighted Hamming distance, Zhang et al. [21] proved that both problems are NP-hard and designed dynamic programming algorithms with time complexities $O(n^4)$ and $O(n^4 \log n)$ for the problems MSPIT and MCSPIT with unit Hamming distance, respectively. Yi et al. [22] subsequently improved these algorithms with local optimizations for time complexities $O(n^3)$ and $O(n^3 \log n)$, respectively. Zhang et al. [23] also investigated the sum of the root–leaf distance interdiction problem involving upgrading edges on trees along with its corresponding minimum cost problem. Using the weighted l_1 norm, they provided linear-time algorithms for both problems. They showed the NP-hardness of the problems with the weighted Hamming distance and unit Hamming distance and designed a linear-time greedy algorithm and an $O(n \log n)$ time algorithm based on a binary search method for each problem. Furthermore, Li et al. [24] studied the problem with cardinality constraints and its corresponding minimum cost problem. Using the weighted l_∞ norm, they provided two algorithms with time complexity $O(n \log n)$ and

$O(Nn^2)$ for the problems, respectively, where N denotes the cardinality budget as the maximum number of edges that can be upgraded. They proposed two algorithms using the weighted bottleneck Hamming distance, both in $O(n \log n)$ time.

Research on extensions of the shortest-path interdiction problem involving upgrading edges remains relatively limited. Mohammadi et al. [25] studied the maximum capacity path interdiction problem involving upgrading edges with the sum-type Hamming distance. Unlike the traditional shortest-path problem, where the weight of a path is defined as the sum of the weights of all edges along it, this problem uses the maximum edge weight along the path as its capacity. They provided a strongly polynomial-time algorithm for the case where the cost of reducing each arc's capacity is fixed. Tayyebi et al. [26] studied the continuous maximum capacity path interdiction problem, where the capacity of each arc in the network can be reduced by any continuous amount. They proposed an efficient algorithm based on a binary search method and the discrete Newton method, solving the problem in polynomial time. Li et al. [27] introduced the minimum-cost shortest-path double interdiction problem involving upgrading edges on trees (MCDSPIT). They addressed the problem that measures the upgrade cost with both weighted l_∞ and weighted sum Hamming distances. They showed that the problem is NP-hard and proposed a pseudo-polynomial time algorithm. Research on the shortest-path interdiction problem involving upgrading edges is highly significant and greatly enriches the theory of network interdiction problems.

In this paper, we concentrate on the problems MSPIT and MCSPIT with the weighted l_∞ norm, denoted by MSPIT$_\infty$ and MCSPIT$_\infty$, respectively. They can be formulated as follows by applying the weighted l_∞ norm to the $\|\cdot\|$ in models (1) and (2), respectively.

$$\begin{aligned} & \max_{\bar{w}} && D(\bar{w}) := \min_{t_k \in Y} \bar{w}(P_k) \\ (\text{MSPIT}_\infty) \quad & s.t. && \max_{e \in E} c(e)(\bar{w}(e) - w(e)) \le M, \\ & && w(e) \le \bar{w}(e) \le u(e), e \in E. \end{aligned} \tag{3}$$

$$\begin{aligned} & \min_{\bar{w}} && C(\hat{w}) := \max_{e \in E} c(e)(\hat{w}(e) - w(e)) \\ (\text{MCSPIT}_\infty) \quad & s.t. && \min_{t_k \in Y} \hat{w}(P_k) \ge D, \\ & && w(e) \le \hat{w}(e) \le u(e), e \in E. \end{aligned} \tag{4}$$

We analyzed their models and properties and designed optimization algorithms with time complexity analysis. As summarized in Table 1, our study delineates the computational complexity of the MSPIT, MCSPIT, and MCDSPIT with various norms, including ℓ_1, Hamming distance, and ℓ_∞. A key contribution of this work is the establishment of efficient algorithms with complexities of $O(n)$ and $O(n \log n)$ for the weighted ℓ_∞ norm variants, thereby resolving previously open problems.

The paper is organized as follows. In Section 2, we propose an $O(n)$ time algorithm to solve the problem MSPIT$_\infty$. In Section 3, we research the problem MCSPIT$_\infty$ and propose a greedy algorithm in $O(n \log n)$ time based on a binary search method. In Section 4, based on the problem MCSPIT$_\infty$, the problem MCDSPIT$_\infty$ is studied, and an $O(n \log n)$ time algorithm is proposed. In Section 5, we present some computational experiments to show the effectiveness of the algorithms. In Section 6, we conclude and present directions for future research.

Table 1. Recent advances upgrading edges on trees in shortest-path interdiction problems.

Problem	Norms	Complexity	Reference
MSPIT	weighted ℓ_1	$O(n^2)$	[16]
	weighted Hamming	NP-hard	[21]
	unit Hamming	$O(n^4)$ & $O(n^3)$	[21,22]
	weighted ℓ_∞	$O(n)$	Section 2
MCSPIT	weighted ℓ_1	$O(n^2)$	[16]
	weighted Hamming	NP-hard	[21]
	unit Hamming	$O(n^4 \log n)$ & $O(n^3 \log n)$	[21,22]
	weighted ℓ_∞	$O(n \log n)$	Section 3
MCDSPIT	weighted ℓ_∞ + Hamming	NP-hard	[27]
	weighted ℓ_∞	$O(n \log n)$	Section 4

2. Solving the Problem MSPIT$_\infty$

In this section, we propose a linear time algorithm for the problem MSPIT$_\infty$ based on the analysis of properties.

According to model (3) for the MSPIT$_\infty$ problem, the optimal strategy for any edge $e \in E$ is to upgrade its weight to the maximum possible value. This upgrade is subject to the cost constraint M and the upper bound $u(e)$. The optimal increment for the edge weight can be determined with $\min\left\{\frac{M}{c(e)}, \Delta u(e)\right\}$, where $\Delta u(e) = u(e) - w(e)$. Thus an optimal solution is

$$\bar{w}(e) := w(e) + \min\left\{\frac{M}{c(e)}, \Delta u(e)\right\} \quad (e \in E) \tag{5}$$

and its relevant optimal objective value $D(\bar{w}) = \min_{t_k \in Y} \bar{w}(P_k)$.

Theorem 1. *The solution $\bar{w}$ defined as (5) is an optimal solution of the problem MSPIT$_\infty$.*

Proof. (1) For any edge $e \in E$, we have

$$w(e) \le \bar{w}(e) = w(e) + \min\left\{\frac{M}{c(e)}, u(e) - w(e)\right\} \le w(e) + u(e) - w(e) = u(e)$$

and

$$c(e)(\bar{w}(e) - w(e)) = c(e) \cdot \min\left\{\frac{M}{c(e)}, u(e) - w(e)\right\} \le c(e) \cdot \frac{M}{c(e)} = M.$$

Therefore, $\bar{w}$ is a feasible solution of the problem MSPIT$_\infty$.

(2) Suppose $\bar{w}$ is not optimal, but w^* is. Then, $D(w^*) = w^*(P_{t^*}) > D(\bar{w}) = \bar{w}(P_{\bar{t}})$.

(i) If $t^* = \bar{t}$, then there exists an edge $\hat{e} \in P_{t^*}$ satisfying $u(\hat{e}) \ge w^*(\hat{e}) > \bar{w}(\hat{e})$ as $w^*(P_{t^*}) > \bar{w}(P_{t^*})$. Then, we have

$$C(w^*) \ge c(\hat{e})(w^*(\hat{e}) - w(\hat{e})) > c(\hat{e})(\bar{w}(\hat{e}) - w(\hat{e})) = M,$$

which contradicts the feasibility of w^*.

(ii) If $t^* \ne \bar{t}$, then $w^*(P_{\bar{t}}) \ge w^*(P_{t^*}) > \bar{w}(P_{\bar{t}}) \le \bar{w}(P_{t^*})$, and there is an edge $\hat{e} \in P_{\bar{t}}$ satisfying $u(\hat{e}) \ge w^*(\hat{e}) > \bar{w}(\hat{e})$. Similarly, we have

$$C(w^*) \ge c(\hat{e})(w^*(\hat{e}) - w(\hat{e})) > c(\hat{e})(\bar{w}(\hat{e}) - w(\hat{e})) = M,$$

which contradicts the feasibility of w^*.

Hence, $\bar{w}$ is an optimal solution of the problem MSPIT$_\infty$. □

For a given tree $T(V, E)$ with a set Y of leaves, cost vector c, and two weight vectors w and u along with a given budget M, we can obtain an optimal solution $\bar{w}$ for the problem MSPIT$_\infty$ using Formula (5). This solution allows us to determine the corresponding optimal objective value $D(\bar{w}) = \min_{t \in Y} \bar{w}(P_t)$. For convenience, we represent the above process with (6):

$$[\bar{w}, D(\bar{w})] := \text{MSPIT}_\infty(T, Y, w, u, c, M). \tag{6}$$

Theorem 2. *The problem MSPIT$_\infty$ can be solved using (6) in $O(n)$ time.*

3. Solving the Problem MCSPIT$_\infty$

In this section, we address the problem MCSPIT$_\infty$. Through a series of property analyses and case discussions, we developed an efficient algorithm. The algorithm leverages greedy strategies and a binary search method to achieve a time complexity of $O(n \log n)$.

We begin by sorting $c(e)\Delta u(e)$ in a non-decreasing order and removing any duplicates to obtain the sequence $\{\alpha_{n^*}\}$:

$$c(e_{\alpha_1})\Delta u(e_{\alpha_1}) < c(e_{\alpha_2})\Delta u(e_{\alpha_2}) < \cdots < c(e_{\alpha_{n^*}})\Delta u(e_{\alpha_{n^*}}) \tag{7}$$

Define $M_j := c(e_{\alpha_j})\Delta u(e_{\alpha_j})$ for $j := 1, 2, \ldots, n^*$. We can then apply (6) using the assignment $[\hat{w}^j, D(\hat{w}^j)] := \text{MSPIT}_\infty(T, Y, w, u, c, M_j)$ to compute the value $\hat{w}^j$ and the corresponding shortest root–leaf distance $D_j = D(\hat{w}^j)$. Since M_j is increasing, the output D_j is non-decreasing.

Specifically, for $j = n^*$ and $j = 1$, the processes are defined as

$$[\hat{w}^{n^*}, D(\hat{w}^{n^*})] := \text{MSPIT}_\infty(T, Y, w, u, c, M_{n^*}) \text{ with } M_{n^*} = c(e_{\alpha_{n^*}})\Delta u(e_{\alpha_{n^*}})$$

and

$$[\hat{w}^1, D(\hat{w}^1)] := \text{MSPIT}_\infty(T, Y, w, u, c, M_1) \text{ with } M_1 = c(e_{\alpha_1})\Delta u(e_{\alpha_1}).$$

These computations yield $\hat{w}^{n^*}$ and $\hat{w}^1$ along with their respective distances D_{n^*} and D_1. Upon analysis, we derive the following two straightforward lemmas.

Lemma 1. *The optimal objective value of the problem MCSPIT$_\infty$ is in the range $[0, M_{n^*}]$.*

Lemma 2. *If $\hat{w}$ is an optimal solution of the problem MCSPIT$_\infty$, then its relevant length $D(\hat{w})$ of the shortest root–leaf distance satisfies $D(w) \le D(\hat{w}) \le D_{n^*}$.*

According to Lemma 2, the problem MCSPIT$_\infty$ is infeasible for $D > D_{n^*}$ as all edges have been upgraded to their maximum capacity. For $D \le D(w)$, the optimal solution is conversely w with an optimal value 0 since w already satisfies the feasibility conditions. We can consequently state the following theorems.

Theorem 3. *(1) If $D > D_{n^*}$, the (MCSPIT$_\infty$) is infeasible.*

(2) If $D \le D(w)$, the optimal solution is w, and the optimal value is 0.

We next consider the problem MCSPIT$_\infty$ when $D(w) < D \le D_1$. The optimal objective value is denoted by M^*. The optimal objective value M^* of problem MCSPIT$_\infty$ therefore satisfies $M^* \le M_1$. Consequently, the upgrade amount for any edge $e \in E$ is given by $\frac{M^*}{c(e)}$. Let $Y_< := \{t_k \in Y | w(P_k) < D\}$ be the set of leaves with root–leaf distances of less than

D. Then, for each path $P_k(t_k \in Y_<)$, $\sum_{e\in P_k} \frac{M^*}{c(e)} + w(P_k) \geq D$ holds. Hence, we have the following theorem.

Theorem 4. *When* $D(w) < D \leq D_1$, $M^* = \max\limits_{t_k \in Y_<} \frac{D - w(P_k)}{\sum_{e\in P_k} \frac{1}{c(e)}}$ *is the optimal objective value of problem MCSPIT*$_\infty$, *where* $Y_< = \{t_k \in Y | w(P_k) < D\}$.

Proof. Let $\bar{w}(e) = w(e) + \min\{\frac{M^*}{c(e)}, \Delta u(e)\} = w(e) + \frac{M^*}{c(e)}$ be the corresponding optimal solution of the objective value M^* as defined in the theorem.

(1) We show the feasibility of the solution $\bar{w}$. For any $t_k \in Y$,

$$\begin{aligned} \bar{w}(P_k) &= w(P_k) + \sum_{e\in P_k} \frac{M^*}{c(e)} \\ &= w(P_k) + \max_{t_j \in Y} \frac{D - w(P_j)}{\sum_{e\in P_j} \frac{1}{c(e)}} \cdot \sum_{e\in P_k} \frac{1}{c(e)} \\ &\geq w(P_k) + \frac{D - w(P_k)}{\sum_{e\in P_k} \frac{1}{c(e)}} \cdot \sum_{e\in P_k} \frac{1}{c(e)} = D \end{aligned}$$

Hence, $\bar{w}$ is feasible.

(2) Next, we show the optimality. Suppose $\bar{w}$ is not an optimal solution, but $\hat{w}(e) = w(e) + \frac{\hat{M}}{c(e)}$ (for $e \in E$) is, where $\hat{M} < M^* \leq M_1$ is its corresponding objective value. Let $P_{\hat{t}}$ be the shortest path with the weight $\hat{w}$.

(2.1) If $P_{\hat{t}}$ is also the shortest path with the weight $\bar{w}$, then $\bar{w}(P_{\hat{t}}) = D$ according to the definition of M^*. We then have

$$\hat{w}(P_{\hat{t}}) = w(P_{\hat{t}}) + \sum_{e\in P_{\hat{t}}} \frac{\hat{M}}{c(e)} < w(P_{\hat{t}}) + \sum_{e\in P_{\hat{t}}} \frac{M^*}{c(e)} = \bar{w}(P_{\hat{t}}) = D,$$

which contradicts the fact that $\hat{w}$ is feasible.

(2.2) If $P_{\hat{t}}$ is not the shortest path with the weight $\bar{w}$, but $P_{\bar{t}}$ is, and $\bar{w}(P_{\bar{t}}) < \bar{w}(P_{\hat{t}})$, then we have

$$\hat{w}(P_{\hat{t}}) \leq \hat{w}(P_{\bar{t}}) < \bar{w}(P_{\bar{t}}) = D,$$

which contradicts the fact that $\hat{w}$ is optimal.

In conclusion, $\bar{w}$ is an optimal solution to the problem with its relevant optimal objective value M^*. □

When $D_1 < D \leq D_{n^*}$, we employ a binary search method to determine the range of the optimal solution for the problem MCSPIT$_\infty$. We present the optimal solution by identifying the smallest cost among the infeasible paths and provide an $O(n \log n)$ time algorithm to solve the problem MCSPIT$_\infty$.

Specifically, we first use a binary search method based on the ascending sequence in (7) to determine the index k satisfying $D_k < D \leq D_{k+1}$. We call (6) to solve $[\hat{w}^k, D(\hat{w}^k)] :=$ MSPIT$_\infty(T, Y, w, u, c, M_k)$, obtaining $\hat{w}^k$ and its corresponding shortest root–leaf path length $D_k = D(\hat{w}^k)$. Next, we call (6) to solve $[\hat{w}^{k+1}, D(\hat{w}^{k+1})] :=$ MSPIT$_\infty(T, Y, w, u, c, M_{k+1})$, obtaining $\hat{w}^{k+1}$ and its corresponding shortest root–leaf path length $D_{k+1} = D(\hat{w}^{k+1})$. When $D_{k+1} = D$, the current cost M_{k+1} is just enough to satisfy the constraint, and the following theorem is obvious.

Theorem 5. *If* $D_{k+1} = D$, *then* M_{k+1} *is the optimal value to problem MCSPIT*$_\infty$, *and the corresponding optimal solution is* $\hat{w}^{k+1}$.

Then, we have the following theorem to give the optimal solution.

Theorem 6. *Suppose $D_k < D \leq D_{k+1}$ for an index k based on the ascending sequence in (7). The optimal objective value of the problem MCSPIT$_\infty$ is*

$$M^* = \max_{t_k \in Y_<} \frac{D - w(P_k) - \sum_{e \in P_k \setminus E_>} \Delta u(e)}{\sum_{e \in P_k \cap E_>} \frac{1}{c(e)}}, \tag{8}$$

where $Y_< := \{t_k \in Y \mid w(P_k) < D\}$ and $E_> := \{e \in E \mid c(e)\Delta u(e) > M_k\}$. Then, the corresponding optimal solution is given by

$$w^*(e) = w(e) + \min\left\{\frac{M^*}{c(e)}, \Delta u(e)\right\}. \tag{9}$$

Proof. We mainly show that w^* is a feasible solution of the problem MCSPIT$_\infty$. Notice that for any edge $e \in E_>$, we have $c(e)\Delta u(e) > M_k$, and $c(e)\Delta u(e) > M^*$ also holds. Otherwise, the case $M^* > c(e)\Delta u(e) > M_k$ occurs, which contradicts $D_k < M \leq D_{k+1}$, implying no values between M_k and M_{k+1} in sequence (7). Additionally, for any edge $e \in P_k \setminus E_>$, we obviously have $c(e)\Delta u(e) \leq M_k < M^*$.

For any $t_k \in Y_<$,

$$\begin{aligned}
w^*(P_k) = \sum_{e \in P_k} w^*(e) &= \sum_{e \in P_k} \left(w(e) + \min\left\{\frac{M^*}{c(e)}, \Delta u(e)\right\}\right) \\
&= w(P_k) + \sum_{e \in P_k \cap E_>} \frac{M^*}{c(e)} + \sum_{e \in P_k \setminus E_>} \Delta u(e) \qquad (10) \\
&\geq w(P_k) + \sum_{e \in P_k \cap E_>} \frac{\frac{D - w(P_k) - \sum_{e \in P_k \setminus E_>} \Delta u(e)}{\sum_{e \in P_k \cap E_>} \frac{1}{c(e)}}}{c(e)} + \sum_{e \in P_k \setminus E_>} \Delta u(e) \qquad (11) \\
&= w(P_k) + D - w(P_k) - \sum_{e \in P_k \setminus E_>} \Delta u(e) + \sum_{e \in P_k \setminus E_>} \Delta u(e) \\
&= D
\end{aligned}$$

For any $t_k \in Y \setminus Y_<$, we have $w^*(P_k) \geq w(P_k) \geq D$.

The proof for the optimality of M^* follows a similar argument to that in part (2) of Theorem 4, and we omit the details here. □

Based on the analysis above, we have the following Algorithm 1 to solve the problem MCSPIT$_\infty$.

Theorem 7. *The problem MCSPIT$_\infty$ can be solved with Algorithm 1 in $O(n \log n)$ time.*

Proof. We analyze the time complexity of Algorithm 1 step by step to establish the $O(n \log n)$ bound. The algorithm begins by calculating the weight difference vector $\Delta u = u - w$ in Line 1. This involves a simple subtraction operation for each of the $O(n)$ edges, which requires $O(n)$ time. In Line 2, sorting the edges based on the value of $c(e)\Delta u(e)$ is a fundamental operation. Since there are $O(n)$ edges, sorting them requires $O(n \log n)$ time.

In Lines 4–5, the algorithm calls the *MSPIT*$_\infty$ subroutine as (6) twice for the extreme values M_1 and M_{n^*} in $O(n)$ time. Lines 6–13 handle special cases regarding the feasibility and triviality of the problem. The checks $D > D_{n^*}$ and $D \leq D(w)$ involve simple comparisons. The calculation of M^* for the case $D(w) < D \leq D_1$ involves iterating over the paths to leaves in the set $Y_<$, which can be obtained from a breadth-first search in $O(n)$ time. Thus, this entire block runs in $O(n)$ time.

Algorithm 1 $[\bar{w}, M^*] := \text{MCSPIT}_\infty(T, Y, w, u, c, D)$

Input: The tree $T(V, E)$ rooted at a root node v_1,
The set Y of leaf nodes,
The cost vector c,
Two weight vectors w, u,
A given value D.
Output: An optimal solution $\bar{w}$,
Its relevant objective value M^*.

1: Calculate $\Delta u := u - w$.
2: Sort edges $e \in E$ by $c(e)\Delta u(e)$ in ascending order (remove duplicates).
3: Let the sorted distinct values be: $M_1 \leq M_2 \leq \cdots \leq M_{n^*}$.
4: Call (6) $[\hat{w}^{n^*}, D_{n^*}] := \text{MSPIT}_\infty(T, Y, w, u, c, M_{n^*})$.
5: Call (6) $[\hat{w}^1, D_1] := \text{MSPIT}_\infty(T, Y, w, u, c, M_1)$.
6: **if** $D > D_{n^*}$ **then**
7: **return** "The problem is infeasible."
8: **else if** $D \leq D(w)$ **then**
9: **return** $(w, 0)$.
10: **else if** $D(w) < D \leq D_1$ **then**
11: $M^* := \max_{t_k \in Y_<} \dfrac{D - w(P_k)}{\sum_{e \in P_k} \frac{1}{c(e)}}$, where $Y_< := \{t_k \in Y \mid w(P_k) < D\}$.
12: Call (6) $[\bar{w}, D(\bar{w})] := \text{MSPIT}_\infty(T, Y, w, u, c, M^*)$.
13: **return** $(\bar{w}, M^*)$.
14: **else**
15: Initialize $a := 1$, $b := n^*$, $k^* :=$ null
16: **if** $a = b$ **then**
17: $k^* := a$
18: **end if**
19: **while** $k^* =$ null **do**
20: $k := \lfloor \frac{a+b}{2} \rfloor$.
21: Call (6) $[\hat{w}^k, D_k] := \text{MSPIT}_\infty(T, Y, w, u, c, M_k)$.
22: Call (6) $[\hat{w}^{k+1}, D_{k+1}] := \text{MSPIT}_\infty(T, Y, w, u, c, M_{k+1})$.
23: **if** $D_{k+1} < D$ **then**
24: $a := k$.
25: **else if** $D_k \geq D$ **then**
26: $b := k$.
27: **else if** $D_k < D \leq D_{k+1}$ **then**
28: $k^* := k$.
29: **end if**
30: **end while**
31: $Y_< := \{t_k \in Y \mid w(P_k) < D\}$.
32: $E_> := \{e \in E \mid c(e)\Delta u(e) > M_{k^*}\}$.
33: Calculate M^* as (8) and w^* as (9).
34: **return** (w^*, M^*).
35: **end if**

The binary search loop in Lines 15–30 is the core component of the algorithm. The loop performs a binary search over the n^* (which is $O(n)$) sorted M_j values to find the critical interval $[M_k, M_{k+1}]$ such that $D_k < D \leq D_{k+1}$. The number of iterations of this binary search is $O(\log n)$. In each iteration, the algorithm makes two calls to the $MSPIT_\infty$ subroutine (Lines 21 and 22) to compute D_k and D_{k+1} for the current pivot M_k and its

successor M_{k+1}. Since each call to $MSPIT_\infty$ takes $O(n)$ time, the cost per iteration is $O(n)$. Therefore, the total time cost of the binary search loop is $O(\log n) \times O(n) = O(n \log n)$.

After identifying the critical index k^*, the final solution is constructed. Lines 31–32 define the sets $Y_<$ and $E_>$, which can be determined by checking each leaf and each edge, taking $O(n)$ time. Line 33 calculates M^* and the optimal weight vector w^* according to (8) and (9) in $O(n)$ time.

To sum up, the overall time complexity is dominated by the sorting operation and the binary search loop, both of which are $O(n \log n)$. Hence, Algorithm 1 solves the $MCSPIT_\infty$ problem in $O(n \log n)$ time. □

4. Minimum-Cost Shortest-Path Double Interdiction Problem Involving Upgrading Edges on Trees with Weighted l_∞ Norm

In this section, we consider an extended version of the problem MCSPIT$_\infty$, referred to as the minimum-cost shortest-path double interdiction problem involving upgrading edges on trees (MCDSPIT) with the weighted l_∞ norm (MCDSPIT$_\infty$). Li et al. [27] introduced the MCDSPIT problem, which has significant implications in diverse areas such as transportation networks, military strategies, and counter-terrorism efforts. The objective is to find a minimum-cost upgrade scheme such that two constraints are satisfied: the shortest root–leaf distance is at least a given value D, and the sum of all root–leaf distances is no less than a given value B. The mathematical model can be stated as follows.

$$\text{(MCDSPIT)} \quad \begin{aligned} \min_{\tilde{w}} \quad & \|\tilde{w}(e) - w(e)\| \\ s.t. \quad & \tilde{w}(P_i) \geq D, t_i \in Y, \\ & \sum_{t_i \in Y} \tilde{w}(P_i) \geq B, \\ & w(e) \leq \tilde{w}(e) \leq u(e), e \in E. \end{aligned} \tag{12}$$

When the l_∞ norm applies to $\|\cdot\|$ in model (12), we can obtain the problem MCDSPIT$_\infty$, which is formulated as follows.

$$\text{(MCDSPIT}_\infty\text{)} \quad \begin{aligned} \min_{\tilde{w}} \quad & C(\tilde{w}) := \max_{e \in E} c(e)(\tilde{w}(e) - w(e)) && (13) \\ s.t. \quad & \tilde{w}(P_i) \geq D, t_i \in Y, && (14) \\ & \sum_{t_i \in Y} \tilde{w}(P_i) \geq B, && (15) \\ & w(e) \leq \tilde{w}(e) \leq u(e), e \in E. && (16) \end{aligned}$$

It follows from the model of the problem MCDSPIT$_\infty$ that if upgrading all edges to their upper bounds still cannot satisfy the conditions (14) or (15), then the problem is infeasible.

Theorem 8. *If $\sum_{t_i \in Y} u(P_i) < B$ or $\min_{t_i \in Y} u(P_i) < D$, then the problem MCDSPIT$_\infty$ is infeasible.*

Notice that (13), (14), and (16) are the problem MCSPIT$_\infty$. An optimal solution of the problem MCDSPIT$_\infty$ is thus a feasible solution of the problem MCSPIT$_\infty$. Then, we have the following conclusion for the optimal objective values of the two problems. Note that for any $e \in E$, $\Delta u(e) = u(e) - w(e)$.

Lemma 3. *We have $M^* \leq K^* \leq \max_{e \in E} c(e)\Delta u(e)$, where K^* and M^* are the optimal values of the problems MCDSPIT$_\infty$ and MCSPIT$_\infty$, respectively.*

For any $K \geq 0$, define the weight vector w_K as follows:

$$w_K(e) := w(e) + \min\left\{\frac{K}{c(e)}, \Delta u(e)\right\}, e \in E. \tag{17}$$

Then, we have the following lemma to show the relationship between the optimal value K^* of the problem MCDSPIT$_\infty$ and any cost $K \geq 0$.

Lemma 4. *The vector w_K as (17) is feasible for MCDSPIT$_\infty$ if and only if $K \geq K^*$, where K^* is the optimal values of the problem MCDSPIT$_\infty$. Moreover, if $K = K^*$, then w_K is an optimal solution of the problem MCDSPIT$_\infty$.*

Proof. Necessity: Let w_K be defined as (17), a feasible solution to the problem MCDSPIT$_\infty$, and suppose $K < K^*$. This contradicts that K^* is the optimal value of the problem MCDSPIT$_\infty$.

Sufficiency: Suppose w^* is an optimal solution to problem MCDSPIT$_\infty$. We will prove that for any $e \in E$, $w_K(e) \geq w^*(e)$ holds. Suppose there exists $e_i \in E$, such that $w_K(e_i) = w(e_i) + \min\left\{\frac{K}{c(e_i)}, \Delta u(e_i)\right\} < w^*(e_i)$.

(1) If $\frac{K}{c(e_i)} > \Delta u(e_i)$, then $w_K(e_i) = w(e_i) + \Delta u(e_i) = u(e_i) < w^*(e_i)$, which contradicts $w^*(e_i) \leq u(e_i)$.

(2) If $\frac{K}{c(e_i)} \leq \Delta u(e_i)$, then $w_K(e_i) = w(e_i) + \frac{K}{c(e_i)} < w^*(e_i)$, which implies that $K^* \geq c(e_i)(w^*(e_i) - w(e_i)) > c(e_i)(w_K(e_i) - w(e_i)) = K$. This contradicts $K^* \leq K$.

Therefore, for any $e \in E$, $w_K(e) \geq w^*(e)$ holds so that constraints (14)–(16) hold for the vector w_K, confirming that w_K is indeed feasible. □

We can solve the problem MCDSPIT$_\infty$ from the analysis above based on an optimal solution $\hat{w}$ of MCSPIT$_\infty$ with its corresponding optimal cost $M^* := C(\hat{w})$. If the constraint for the sum of the root–leaf distance holds already, then $\hat{w}$ is also optimal for the problem MCDSPIT$_\infty$.

Theorem 9. *If $\sum_{t_i \in Y} \hat{w}(P_i) \geq B$, then $\tilde{w} := \hat{w}$ is an optimal solution of the problem MCDSPIT$_\infty$, where $\hat{w}$ is an optimal solution of the problem MCSPIT$_\infty$.*

Otherwise, we have the case when $\sum_{t_i \in Y} \hat{w}(P_i) < B$, which is the most significant and difficult part of solving the problem MCSPIT$_\infty$. To address this case, we first choose the sequence (7) by removing any duplicates and the values lower than $C(\bar{w})$ to obtain the sequence $\{\beta_{\hat{n}}\}$:

$$\hat{K} \leq c(e_{\beta_1})\Delta u(e_{\beta_1}) < c(e_{\beta_2})\Delta u(e_{\beta_2}) < \cdots < c(e_{\beta_{\hat{n}}})\Delta u(e_{\beta_{\hat{n}}}) \tag{18}$$

Define $K_0 := \hat{K}, K_j := c(e_{\beta_j})\Delta u(e_{\beta_j})$ for $j := 1, 2, \ldots, \hat{n}$. Then, the following lemma holds immediately.

Lemma 5. *The optimal value K^* of the problem MCDSPIT$_\infty$ lies in $[K_0, k_{\hat{n}}]$.*

We can consequently determine the index k through a binary search method such that $K_k < K^* \leq K_{k+1}$, that is, $\sum_{t_i \in Y} w_k(P_i) < B \leq \sum_{t_i \in Y} w_{k+1}(P_i)$, where w_k and w_{k+1} are constructed as (17) with K_k and K_{k+1}, respectively. For convenience, we introduce the following concept $Q(e)$ containing the leaf nodes controlled by an edge $e \in E$.

Definition 1 ([23]). *Define $Q(e) := \{t_k | e \in P_k, k = 1, \cdots, r\}$ as the set of leaf nodes with root–leaf paths passing through e. If $t_k \in Q(e)$, then the leaf node t_k is controlled by the edge $e \in E$.*

Based on the index k with $K_k < K^* \leq K_{k+1}$ in an non-decreasing sequence (18), we can then calculate the optimal value K^* as the following theorem.

Theorem 10. *The value*

$$K^* := \frac{B - \sum_{t_i \in Y} \sum_{e \in P_i} w(e) - \sum_{e \in E^k_{\leq}} |Q(e)| \Delta u(e)}{\sum_{e \in E^k_{>}} \frac{|Q(e)|}{c(e)}} \tag{19}$$

is the optimal objective value of the (MCDSPIT$_\infty$), where $E^k_{>} := \{e \in E | c(e)\Delta u(e) > K_k\}$ and $E^k_{\leq} := \{e \in E | c(e)\Delta u(e) \leq K_k\}$. Then, w_{K^*} as (17) is an optimal solution.

Proof. (1) We first prove that w_{K^*} is a feasible solution of the (MCDSPIT$_\infty$). It is obvious that $w(e) \leq w_{K^*}(e) \leq u(e)$, for $e \in E$ because it is constructed as (17). We then need to show that w_{K^*} satisfies the constraints (14) and (15).

For constraint (14), as $\hat{K} \leq K_k < K^* \leq K_{k+1}$, then for any $t_i \in Y$, we have $w_{K^*}(P_i) > w_{\hat{K}}(P_i) \geq D$.

For constraint (15), notice that for any edge $e \in E^k_{>}$, we have $c(e)\Delta u(e) > K_k$ and $c(e)\Delta u(e) > K^*$ also holds. Otherwise, the case $K^* > c(e)\Delta u(e) > K_k$ occurs, which contradicts $K_k < K^* \leq K_{k+1}$, implying no values between K_k and K_{k+1} in sequence (18). We also obviously have $c(e)\Delta u(e) \leq K_k < K^*$ for any edge $e \in E^k_{\leq}$. Then, we have

$$\begin{aligned}
\sum_{t_i \in Y} w_{K^*}(P_i) &= \sum_{t_i \in Y} \sum_{e \in P_i} w_{K^*}(e) \\
&= \sum_{t_i \in Y} \sum_{e \in P_i} \left(w(e) + \min \left\{ \frac{K^*}{c(e)}, \Delta u(e) \right\} \right) \\
&= \sum_{t_i \in Y} \sum_{e \in P_i} w(e) + \sum_{t_i \in Y} \sum_{e \in P_i} \min \left\{ \frac{K^*}{c(e)}, \Delta u(e) \right\} \\
&= \sum_{t_i \in Y} \sum_{e \in P_i} w(e) + \sum_{e \in E} \sum_{e \in P_i, t_i \in Y} \min \left\{ \frac{K^*}{c(e)}, \Delta u(e) \right\} \\
&= \sum_{t_i \in Y} \sum_{e \in P_i} w(e) + \sum_{e \in E} |Q(e)| \min \left\{ \frac{K^*}{c(e)}, \Delta u(e) \right\} \\
&= \sum_{t_i \in Y} \sum_{e \in P_i} w(e) + \sum_{e \in E^k_{>}} |Q(e)| \frac{K^*}{c(e)} + \sum_{e \in E^k_{\leq}} |Q(e)| \Delta u(e) \\
&= \sum_{t_i \in Y} \sum_{e \in P_i} w(e) + \left(B - \sum_{t_i \in Y} \sum_{e \in P_i} w(e) - \sum_{e \in E^k_{\leq}} |Q(e)| \Delta u(e) \right) + \sum_{e \in E^k_{\leq}} |Q(e)| \Delta u(e) \\
&= B
\end{aligned}$$

Therefore, w_{K^*} as (17) with K^* is a feasible solution of the problem MCDSPIT$_\infty$.

(2) We next prove that K^* is the optimal value. Suppose that $K' < K^*$ is the optimal value of the problem MCDSPIT$_\infty$ and $w_{K'}$ is a corresponding optimal solution; then, we have

$$\sum_{t_i \in Y} w_{K'}(P_i) = \sum_{t_i \in Y} \sum_{e \in P_i} w(e) + \sum_{e \in E^k_{>}} |Q(e)| \frac{K'}{c(e)} + \sum_{e \in E^k_{\leq}} |Q(e)| \Delta u(e) < B,$$

which contradicts the feasibility of $w_{K'}$. Hence, K^* is the optimal value of the problem MCDSPIT$_\infty$. □

From the above analysis, we have the following algorithm to solve the problem MCDSPIT$_\infty$.

Theorem 11. *The problem MCDSPIT$_\infty$ can be solved using Algorithm 2 in $O(n \log n)$ time.*

Algorithm 2 $[\tilde{w}, K^*] := \text{MCDSPIT}_\infty(T, Y, w, u, c, D, B)$

Input: The tree $T(V, E)$ rooted at a root node v_1,
The set Y of leaf nodes,
The cost vector c,
Two weight vectors w, u,
Given values D and B.
Output: An optimal solution $\tilde{w}$,
Its relevant objective value K^*.

1: **if** $\sum_{t_k \in Y} u(P_k) < B$ or $\min_{t_k \in Y} u(P_k) < D$ **then**
2: **return** "The problem is infeasible."
3: **else**
4: Call $[\hat{w}, M^*] := \text{MCSPIT}_\infty(T, Y, w, u, c, D)$.
5: **if** $\sum_{t_k \in Y} \bar{w}(P_k) \geq B$ **then**
6: **return** $(\hat{w}, M^*)$.
7: **end if**
8: Calculate $\Delta u := u - w$.
9: Construct the sequence $\{\beta_{\hat{n}}\}$ by removing duplicates and values lower than $C(\bar{w})$ from sequence (7):

$$\bar{K} \leq c(e_{\beta_1})\Delta u(e_{\beta_1}) < c(e_{\beta_2})\Delta u(e_{\beta_2}) < \cdots < c(e_{\beta_{\hat{n}}})\Delta u(e_{\beta_{\hat{n}}}).$$

10: Let $K_0 := \bar{K}$, $K_j := c(e_{\beta_j})\Delta u(e_{\beta_j})$ for $j := 1, 2, \ldots, \hat{n}$.
11: Initialize $a := 0$, $b := \hat{n}$.
12: **while** $b - a > 1$ **do**
13: $k := \lfloor \frac{a+b}{2} \rfloor$.
14: Construct w_{K_k} as Formula (17).
15: **if** $\sum_{t_i \in Y} w_{K_k}(P_i) < B$ **then**
16: $a := k$.
17: **else if** $\sum_{t_i \in Y} w_{K_k}(P_i) > B$ **then**
18: $b := k$.
19: **end if**
20: **end while**
21: Let $k := a$.
22: Determine $E_> := \{e \in E \mid c(e)\Delta u(e) > K_k\}$, $E_\leq := \{e \in E \mid c(e)\Delta u(e) \leq K_k\}$.
23: Calculate K^* as Formula (19).
24: Determine optimal value w_{K^*} as Formula (17).
25: **return** (w_{K^*}, K^*).
26: **end if**

Proof. We analyze the time complexity of Algorithm 2 step by step to establish the $O(n \log n)$ bound. The algorithm begins with a feasibility check in Lines 1–2. This involves computing the sum of path weights $\sum_{k \in Y} u(P_k)$ and finding the minimum path weight $\min_{k \in Y} u(P_k)$ over all leaf nodes, which can be obtained from a breadth-first search in $O(n)$ time. In Line 3, the algorithm calls Algorithm 1 *MCSPIT*$_\infty$. As established in Theorem 7, Algorithm 1 has a time complexity of $O(n \log n)$, which contributes significantly to the overall complexity.

Lines 4–5 perform a condition check $\sum_{k \in Y} \tilde{w}(P_k) \geq B$ and potentially return early. This operation involves summing path weights for all leaves, requiring $O(n)$ time. Line 6 calculates the weight difference vector $\Delta u = u - w$, which is a simple subtraction operation

over $O(n)$ edges, taking $O(n)$ time. The sequence construction in Lines 7–9 involves sorting and filtering operations. Constructing the sequence $\{\beta_{\hat{n}}\}$ from the original sequence requires removing duplicates and values below $C(\tilde{w})$. The sorting step dominates this process with $O(n \log n)$ time complexity, while the subsequent initialization of K_j values takes $O(n)$ time.

The core of Algorithm 2 is the binary search loop in Lines 10–20. This loop performs a binary search over the $\hat{n}$ (which is $O(n)$) sorted K_j values, resulting in $O(\log n)$ iterations. Within each iteration, the algorithm computes the midpoint ($O(1)$ time); constructs w_{K_k} using Formula (17) which processes $O(n)$ edges ($O(n)$ time); and evaluates the condition $\sum_{i_l \in Y} w_{K_k}(P_i)$ which requires $O(n)$ time. Therefore, each iteration takes $O(n)$ time, and the entire binary search loop requires $O(\log n) \times O(n) = O(n \log n)$ time.

Finally, Lines 21–24 handle the solution construction after the binary search. These steps include simple assignments ($O(1)$), set operations on edges ($O(n)$), and calculations using Formulas (19) and (17) ($O(n)$ each). The total time for these final steps is $O(n)$. To sum up, the overall time complexity is dominated by the $O(n \log n)$ operations. □

5. Computational Experiments

We used synthetic trees and enforced bit-level reproducibility within machine precision across typical environments to demonstrate the efficiency and reproducibility of our experimental results. All randomness was keyed with per-trial seeds, costs were median-normalized so that budget magnitudes do not drift with n, and all targets and path-sum budgets were calibrated on an instance-intrinsic, unit-free scale. These choices ensure that the results are comparable across sizes and that any observed differences reflect algorithmic behavior, not arbitrary units or sampling noise.

Instances were tested in the rooted recursive trees $T = (V, E)$: node i attaches uniformly to $\{1, \ldots, i-1\}$. Each edge e receives a base weight $w(e) \sim \text{Uniform}[0, 1]$ and an available slack $\Delta u(e) \sim \text{Uniform}[0, 1]$, with the upper bound $u(e) = w(e) + \Delta u(e)$. Raw costs $c_{\text{raw}}(e) \sim \text{Uniform}[0.5, 2]$ are converted to unit-free costs via median normalization: $c(e) = c_{\text{raw}}(e)/\text{median}(c_{\text{raw}})$. For a budget $M \geq 0$, upgrading along an edge follows

$$w_M(e) \;=\; w(e) + \min\{M/c(e),\, \Delta u(e)\},$$

which in turn determines the minimum root–leaf distance $D(M)$. Let $M_{\min} = \min_e c(e)\Delta u(e)$ and $M_{\max} = \max_e c(e)\Delta u(e)$. We set the target distance at the midpoint,

$$D \;=\; \tfrac{1}{2}\big(D(M_{\min}) + D(M_{\max})\big),$$

and quantify the instance difficulty with

$$\rho \;=\; \frac{D - D(M_{\min})}{D(M_{\max}) - D(M_{\min})} \in (0, 1],$$

so that ρ measures how far the target lies between "no upgrades" and "maximal upgrades" on that specific instance. In the double-constraint setting, we analogously work with the aggregate path sum $S(M)$. Let $M^\star$ be the MCSPIT$_\infty$ solution that attains the chosen D, and define

$$B \;=\; \tfrac{1}{2}\big(S(M^\star) + S(M_{\max})\big), \qquad \beta \;=\; \frac{B - S(M^\star)}{S(M_{\max}) - S(M^\star)} \in (0, 1],$$

which pairs (ρ, β) as difficulty indices for distance and path-sum goals, respectively. Rare degeneracies in which the reachable interval collapses are resolved by a minimal backoff

(e.g., $D \leftarrow D(M_{\max}) - 10^{-6}$). All feasibility checks and accept/reject outcomes were recorded so that third parties can replay the exact run list.

We evaluated the algorithms for the problems MSPIT$_\infty$, MCSPIT$_\infty$, and MCDSPIT$_\infty$. Unless stated otherwise, problem MSPIT$_\infty$ was parameterized by $M = \theta\, M_{\max}$ with $\theta = 0.5$ in the main results. Timing isolated the algorithms: instance generation and (D, B) calibration occurred once per trial, after which inputs were frozen. Each routine received a single untimed warm-up call and was then timed with `timeit`; timed functions contained no internal random draws, guaranteeing byte-identical repeats with a fixed seed. We swept $n \in \{10^3, 5 \times 10^3, 10^4, 2 \times 10^4, 3 \times 10^4, 5 \times 10^4\}$; ran 30 trials per size; and report the mean, maximum, and minimum wall-clock seconds in Table 2. To diagnose growth rates, we regressed time on n and on $n \log n$ and report R^2 in Table 3.

All artifacts followed a strict provenance protocol. A manifest `MANIFEST.csv` enumerated every run by $(n, \texttt{trial_id}, \texttt{seed}, M, D, B, \rho, \beta, D_1, D_{\text{all}}, S_0, S_1)$, and optional per-instance `.mat` files stored (T, w, u, c, D, B) for bit-level reproduction. Experiments were conducted in MATLAB R2025a on a laptop (Intel® Core™ i9-14900HX, 2.20 GHz; Windows 11). Absolute times naturally depend on hardware and BLAS/JIT settings, but our seeded, deterministic protocol ensures that relative comparisons and scaling trends are stable across environments. See the Supplementary Material ("Program_Result") for the full experiment manifest, fixed seed list, and results for all experiments and figures.

The results align with the intended complexity interpretations. Across a 50× increase in n, MSPIT$_\infty$ grows by $\sim 34\times$, MCSPIT$_\infty$ by $\sim 52\times$, and MCDSPIT$_\infty$ by $\sim 51\times$, indicative of near-linear behavior over our range. Least-squares fits achieve $R^2 \geq 0.996$ throughout; the log–log slope is $0.898 \approx 1$ for MSPIT$_\infty$ and $\approx$1.00 for MCSPIT$_\infty$ and MCDSPIT$_\infty$, with the latter showing a mild finite-size preference for $n \log n$ in the parametric regressions (Table 3). Together, these diagnostics suggest $O(n)$ versus $O(n \log n)$ behavior in practice for the studied algorithms.

Table 2. Empirical CPU time (in seconds) for the proposed algorithms on tree instances of varying sizes (n denotes the number of edges). T_1, T_2, and T_3 report the average time for algorithms MSPIT$_\infty$, MCSPIT$_\infty$, and MCDSPIT$_\infty$, respectively. For each algorithm, $T_{\max}$ and $T_{\min}$ denote the maximum and minimum observed times across test instances.

Complexity	n	1000	5000	10,000	20,000	30,000	50,000
$O(n)$	T_1	0.006	0.027	0.048	0.084	0.121	0.200
	$T_1^{\max}$	0.008	0.031	0.051	0.090	0.131	0.229
	$T_1^{\min}$	0.005	0.025	0.044	0.081	0.113	0.180
$O(n \log n)$	T_2	0.145	0.800	1.611	3.148	4.558	7.517
	$T_2^{\max}$	0.225	1.040	1.900	3.580	5.030	8.191
	$T_2^{\min}$	0.072	0.553	1.184	2.517	3.839	5.275
$O(n \log n)$	T_3	2.030	9.933	19.516	39.275	59.510	104.171
	$T_3^{\max}$	2.193	10.510	20.551	40.121	61.901	107.392
	$T_3^{\min}$	1.912	9.551	19.027	38.403	58.089	101.737

Table 3. Scaling diagnostics from least-squares fits and log–log slopes.

	R^2 for $t \sim an + b$	R^2 for $t \sim a(n \log n) + b$	log–log Slope
MSPIT$_\infty$	0.9987	0.9961	0.898
MCSPIT$_\infty$	0.9996	0.9971	1.007
MCDSPIT$_\infty$	0.9989	0.9998	1.001

The empirical scaling trends are further visualized in Figure 2, where each panel reports the mean CPU time (dots), the min–max range across instances (shaded band), a

least-squares linear fit (solid line with R^2 value), and a reference curve proportional to the theoretical complexity (dashed line). The figure clearly demonstrates the linear growth for MSPIT$_\infty$ and the $n \log n$ scaling for MCSPIT$_\infty$ and MCDSPIT$_\infty$.

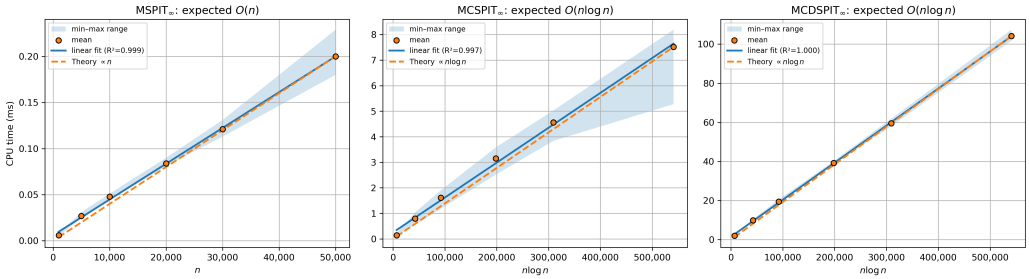


Figure 2. Empirical scaling of the algorithms confirms the theoretical time complexities. CPU time is plotted against instance size n. Dots represent mean execution time, the shaded band shows the min-max range, the solid line is a least-squares fit (with R^2 value), and the dashed line is a reference curve proportional to the theoretical complexity ($O(n)$ for MSPIT$_\infty$, $O(n \log n)$ for the others). The results validate the linear scaling of MSPIT$_\infty$ and the linearithmic scaling of MCSPIT$_\infty$ and MCDSPIT$_\infty$.

Recent work studied related problems for the l_1 norm and Hamming distance. We do not report cross-norm runtime plots because changing the norm (for example, from the infinity norm to the l_1 norm or to Hamming distance) changes the problem class itself and leads to different algorithmic structures and complexity profiles. As is typical in combinatorial optimization, each norm induces its own objective geometry and constraints, which in turn invite specialized methods—for instance, decision-diagram dynamic programming for bottleneck-style objectives versus flow-based, knapsack-like, or budgeted-edit formulations for additive or Hamming-style objectives. Direct wall-clock comparisons across different norms would therefore mix modeling choices with implementation details and hardware/software constants and would not yield a fair apples-to-apples assessment. Our experiments instead focus on validating the predicted scaling behavior for the studied infinity-norm model using a deterministic and reproducible protocol. Table 1 situates the related problem families and summarizes their typical algorithmic landscapes to make the scope of our comparisons explicit.

6. Conclusions and Further Research

Three interdiction problems on trees with the weighted l_∞ norm were investigated in this study: the maximum shortest-path interdiction problem MSPIT$_\infty$, its corresponding minimum-cost shortest-path interdiction problem MCSPIT$_\infty$, and the double-interdiction variant MCDSPIT$_\infty$. The problems have wide applications in cybersecurity, logistics, and infrastructure resilience. For MSPIT$_\infty$, we presented a linear-time greedy algorithm. For MCSPIT$_\infty$, we developed an efficient $O(n \log n)$-time algorithm based on greedy selection and binary search. Building on this framework, we extended the methodology to handle the additional constraint in MCDSPIT$_\infty$ and proposed an algorithm with the same asymptotic complexity. Numerical experiments on synthetic instances confirmed the scalability and practical effectiveness of the proposed methods.

Further research is needed to design more advanced approaches for extremely large-scale graphs with millions of edges and nodes, such as graph decomposition, to enhance scalability. It should also be noted that while our algorithms address the interdiction problems with the l_∞ norm, they may not directly extend to other norms—a common challenge in combinatorial optimization, where tailored methods are often necessary. Nevertheless, the methodology developed in this work provides a foundation that can be adapted to related interdiction problems in alternative infinite-norm settings.

In summary, a comprehensive and efficient algorithmic framework for solving a class of critical interdiction problems on tree networks was established in this work, providing both theoretical guarantees and practical computational tools. Future research directions include extending the proposed models and algorithms to more general graph classes (e.g., series–parallel graphs or general directed acyclic graphs), as well as developing stochastic and robust variants where edge upgrade costs or capacity bounds are uncertain. Moreover, interdiction problems involving node upgrades—rather than edge upgrades—remain largely unexplored and present both theoretical and computational challenges. Finally, applying these models to real-world domains such as cybersecurity, logistics, and infrastructure resilience represents a promising avenue for empirical validation and practical impact.

Supplementary Materials: The following supporting information can be downloaded at: https://www.mdpi.com/article/10.3390/math13193219/s1. The supplementary material "`Program_Result`" contains: (i) `MainProgram.m`—the MATLAB driver script to reproduce the numerical experiments; and (ii) `results/MANIFEST.csv`—the experiment manifest listing all runs and parameters (e.g., n, `trial_id`, `seed`, M, D, B, ρ, β, D_1, D_{all}, S_0, S_1). No other data files are included.

Author Contributions: Conceptualization, Q.Z. and X.L.; methodology, Q.Z. and X.L.; software, X.L.; validation, X.L.; formal analysis, Q.Z. and X.L.; investigation, Q.Z.; resources, Q.Z.; data curation, Q.Z.; writing—original draft preparation, Q.Z.; writing—review and editing, Q.Z. and X.L.; visualization, X.L.; supervision, X.L.; project administration, X.L.; funding acquisition, Q.Z. All authors have read and agreed to the published version of the manuscript.

Funding: The work of Q. Zhang was supported by National Natural Science Foundation of China (Grant No. 1230012046). None of the remaining authors received external funding. The APC was funded by the National Natural Science Foundation of China (Grant No. 1230012046).

Data Availability Statement: The original contributions presented in this study are included in the article/Supplementary Materials. Further inquiries can be directed to the corresponding authors.

Acknowledgments: The authors would like to thank the anonymous reviewers for their constructive comments that helped improve the quality of this manuscript.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

Symbol	Meaning
$T = (V, E, w)$	Edge-weighted tree rooted at v_1; set V of nodes, set E of edges; weight vector w.
$Y = \{t_1, \ldots, t_r\}$	Set of leaves.
P_{v_i,v_j}; P_k	Unique path from v_i to v_j; shorthand $P_k := P_{v_1,t_k}$ (root–leaf path).
$u(e)$	Upper bound of the upgraded weight on edge $e \in E$.
$\Delta u(e)$	Upgrade capacity on e: $\Delta u(e) := u(e) - w(e)$.
$c(e)$	(Weighted) cost coefficient on edge e.
$w(\cdot)$, $\bar{w}(\cdot)$, $\hat{w}(\cdot)$	Edge/path weights before and after upgrading (notations follow the algorithms).
$D(\bar{w})$	Shortest root–leaf distance with weights $\bar{w}$: $D(\bar{w}) := \min_{t_k \in Y} \bar{w}(P_k)$.
$C(\hat{w})$	Bottleneck (max) cost with weights $\hat{w}$: $C(\hat{w}) := \max_{e \in E} c(e)(\hat{w}(e) - w(e))$.
M, D, B	Budget/threshold parameters used in the models; see Problems (3)–(4), (12)–(16).
$Q(e)$	Set of leaves with root–leaf paths passing through e (Def. 1).

Acronym	Expanded Form
SPIP	Shortest path interdiction problem (background)
MSPIT	Maximum shortest path interdiction on trees
MSPIT∞	MSPIT with weighted ℓ_∞ norm
MCSPIT	Minimum-cost shortest-path interdiction on trees
MCSPIT∞	MCSPIT with weighted ℓ_∞ norm
MCDSPIT∞	minimum-cost shortest-path double interdiction on trees with weighted ℓ_∞ norm

References

1. Smith, J.C.; Song, Y.J. A survey of network interdiction models and algorithms. *Eur. J. Oper. Res.* **2020**, *283*, 797–811. [CrossRef]
2. Corley, H.W.; Sha, D.Y. Most vital links and nodes in weighted networks. *Oper. Res. Lett.* **1982**, *1*, 157–161. [CrossRef]
3. Ball, M.O.; Golden, B.L.; Vohra, R.V. Finding the most vital arcs in a network. *Oper. Res. Lett.* **1989**, *8*, 73–76. [CrossRef]
4. Israeli, E.; Wood, R.K. Shortest-path network interdiction. *Networks* **2002**, *40*, 97–111. [CrossRef]
5. Khachiyan, L.; Boros, E.; Borys, K.; Elbassioni, K.; Gurvich, V.; Rudolf, G.; Zhao, J.H. On short paths interdiction problems: Total and node-wise limited interdiction. *Theory Comput. Syst.* **2008**, *43*, 204–233. [CrossRef]
6. Chen, G.T. Finding the most vital edge of a shortest path in undirected networks. *J. Hangzhou Inst. Electron. Eng.* **2002**, *22*, 48–50.
7. Nardelli, E.; Proietti, G.; Widmayer, P. A faster computation of the most vital edge of a shortest path. *Inf. Process. Lett.* **2001**, *79*, 81–85. [CrossRef]
8. Henke, D.; Wulf, L. On the Complexity of the Bilevel Shortest Path Problem. *Networks* **2025**, 1–18.. [CrossRef]
9. Boeckmann, J.; Thielen, C.; Wittmann, A. Complexity of the temporal shortest path interdiction problem. In *Leibniz International Proceedings in Informatics (LIPIcs), SAND*; Schloss Dagstuhl—Leibniz-Zentrum für Informatik: Wadern, Germany, 2023; Volume 257, pp. 9:1–9:20.
10. Punla-Green, S.Z.; Mitchell, J.E.; Gearhart, J.L.; Hart, W.E.; Phillips, C.A. Shortest path network interdiction with asymmetric uncertainty. *Networks* **2024**, *83*, 605–623. [CrossRef]
11. Azizi, E.; Seifi, A. Shortest path network interdiction with incomplete information: A ro-bust optimization approach. *Ann. Oper. Res.* **2024**, *335*, 727–759. [CrossRef]
12. Nguyen, D.H.; Smith, J.C. Asymmetric stochastic shortest-path interdiction under condi-tional value-at-risk. *Iise Trans.* **2024**, *56*, 398–410. [CrossRef]
13. Huang, D.; Mao, Z.F.; Fang, K.; Chen, L. Solving the shortest path interdiction problem via reinforcement learning. *Int. J. Prod. Res.* **2023**, *61*, 31–48. [CrossRef]
14. Holzmann, T.; Smith, J.C. The Shortest Path Interdiction Problem with Randomized In-terdiction Strategies: Complexity and Algorithms. *Oper. Res.* **2021**, *69*, 82–99. [CrossRef]
15. Borrero, J.S.; Sauré, D.; Trigo, N. Optimal sequential stochastic shortest path interdiction. *Eur. J. Oper. Res.* **2025**, *326*, 641–655. [CrossRef]
16. Zhang, Q.; Guan, X.C.; Pardalos, P.M. Maximum shortest path interdiction problem by upgrading edges on trees under weighted l_1 norm. *J. Glob. Optim.* **2021**, *79*, 959–987. [CrossRef]
17. Ayyıldız, E.; Zelik, G.; Gencer, C.T. Determining the most vital arcs on the shortest path for fire trucks in terrorist actions that will cause fire. *Commun. Ser. A1 Math. Stat.* **2019**, *68*, 441–450. [CrossRef]
18. Forghani, E.M.; Alsalami, O.M. Using a node–child matrix to address the quickest path problem in multistate flow networks under transmission cost constraints. *Mathematics* **2023**, *11*, 4889. [CrossRef]
19. Zhang, Q.; Guan, X.C. Chapter 6: Shortest path interdiction problems on trees. In *Inverse Combinatorial Optimization Problems*; Springer Optimization and Its Applications 225; Guan, X., Pardalos, P.M., Zhang, B., Eds.; Springer: Berlin/Heidelberg, Germany, 2025. [CrossRef]
20. Li, X.; Zhang, Q.; Guan, X.C. Chapter 7: Sum of root-leaf distance interdiction problems on trees. In *Inverse Combinatorial Optimization Problems*; Springer Optimization and Its Applications 225; Guan, X., Pardalos, P.M., Zhang, B., Eds.; Springer: Berlin/Heidelberg, Germany, 2025. [CrossRef]
21. Zhang, Q.; Guan, X.C.; Wang, H.; Pardalos, P.M. Maximum shortest path interdiction problem by upgrading edges on trees under Hamming distance. *Optim. Lett.* **2021**, *15*, 2661–2680. [CrossRef]
22. Yi, L.; Shao, H.; Wu, T.; Liu, P.J. An accelerating algorithm for maximum shortest path interdiction problem by upgrading edges on trees under unit Hamming distance. *Optim. Lett.* **2023**, *17*, 453–469.
23. Zhang, Q.; Guan, X.C.; Jia, J.H.; Qian, X.Q. The sum of root-leaf distance interdiction problem by upgrading edges/nodes on trees. *J. Comb. Optim.* **2022**, *44*, 74–93. [CrossRef]

24. Li, X.; Zhang, Q.; Guan, X.C.; Pardalos, P.M. The sum of root-leaf distance interdiction problem with cardinality constraint by upgrading edges on trees. *J. Comb. Optim.* **2024**, *48*, 39. [CrossRef]
25. Mohammadi, A.; Tayyebi, J. Maximum capacity path interdiction problem with fixed costs. *Asia-Pac. J. Oper. Res.* **2019**, *36*, 1950019. [CrossRef]
26. Tayyebi, J.; Mitra, A.; Sefair, J.A. The continuous maximum capacity path interdiction problem. *Eur. J. Oper. Res.* **2023**, *305*, 38–52. [CrossRef]
27. Li, X.; Guan, X.C.; Jia, J.H.; Pardalos, P.M. Double interdiction problem on trees on the sum of root-leaf distances by upgrading edges. *J. Glob. Optim.* **2025**, *92*, 951–972. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.