Bulletin of the *Transilvania* University of Braşov
Series III: Mathematics and Computer Science, Vol. 3(65), No. 1 - 2023, 185-198
https://doi.org/10.31926/but.mif.2023.3.65.1.14

BOTTLENECK SPANNING TREE INTERDICTION PROBLEM WITH FIXED AND LINEAR COSTS

Abolfazl ABDOLAHZADEH[1], **Massoud AMAN**[*,2] **and Javad TAYYEBI**[3]

Abstract

This paper investigates a combinatorial optimization interdiction problem, called bottleneck spanning tree interdiction. This problem is a game containing two players with conflicting goals. The first player, called the user, wants to find a bottleneck (min-max or max-min) spanning tree in a weighted network. The other player, called the attacker, increases edge weights under a budget constraint as well as bound constraints so that the user does not achieve his/her goal. This game has a hierarchy structure. It means that the attacker first perturbates the network and then, the user chooses his/her strategy after observing the attacker's action. This paper considers the problem in two cases that there are fixed and linear costs for the attacker. Two divide-and-conquer algorithms are developed to solve the problem under both the costs in polynomial time.

2020 *Mathematics Subject Classification:* 90C23, 90C27, 91A68.
Key words: interdiction, spanning tree, divide-and-conquer, polynomial time

1 Introduction

For any optimization problem, an interdiction problem can be defined. Optimization problems often have one decision maker whereas the corresponding interdiction problems contain two decision makers having opposite goals. Similar to the terminology used in [20], a decision maker is referred to as the user and the other is called the attacker in this paper. However, several different names are

[1]Department of Mathematics, Faculty of Science, University of Birjand, Iran, e-mail: a.abdolahzadeh@birjand.ac.ir
[2*] *Corresponding author*, Department of Mathematics, Faculty of Science, University of Birjand, Iran, e-mail: mamann@birjand.ac.ir
[3]Department of Industrial Engineering, Birjand University of Technology, Birjand, Iran, e-mail: javadtayyebi@birjandut.ac.ir

used in different optimization interdiction problems due to their real-world applications (See [1, 7, 11] for instance). In an interdiction problem, the goal of the user is to optimize the objective function of a given optimization problem while the attacker would like to prevent that the user achieves his/her goal. Because of the conflict existence in the goals, interdiction problems can be regarded as zero-sum games, but with this difference that the strategy selection has a hierarchy structure. Namely, the attacker first chooses his/her strategy. Then, the user observes the opponent's strategy and chooses his/her strategy. Such problems are referred to as Stackelberg games in the game theory literature [18].

Among all interdiction problems, combinatorial optimization interdiction problems are interested widely by researchers because they arise in many real-world applications. The most prominent examples of these problems are matching interdiction problems [22], shortest path interdiction problems [9, 24], and maximum capacity path interdiction problems [14]. This paper focuses only on interdiction problems defined on spanning tree structures.

Two main classes of optimization problems are defined on spanning trees:

- minimum spanning tree (MST) problem,
- min-max spanning tree (MMST) problem.

The first is to find a spanning tree in a given weighted network $G(V, A, \mathbf{c})$ so that the summation of its weights is minimized. The second is to look for a spanning tree so as to minimize the maximum of its weights. It is well known that minimum spanning tree problems can be solved in $O(|A| + |V| \log |V|)$ time by Kruskal's algorithm and Prime's algorithm [2], and min-max spanning tree problems can be solved in $O(|A|)$ time by a recursive algorithm [5].

Interdiction problems can be defined for both the spanning tree optimization problems. The earliest and simplest form of minimum spanning tree interdiction problems is to find k edges whose removal results in the largest increase of the spanning tree weight. This problem is called the k-most vital edges with respect to minimum spanning tree [10, 16]. It is proved that the problem can be solved in polynomial time for $k = 1$, whereas it is strongly NP-hard for $k > 1$. The other interesting results can be found in [3, 4]. Various formulations of general minimum spanning tree interdiction problems are presented in [21]. An $O(1)$-approximation algorithm is presented to find a near-optimum solution [23].

In spite of the fact that the minimum spanning tree interdiction problems are widely investigated, the best of our knowledge, the min-max spanning tree interdiction problem is not studied until now. This paper focuses on this problem. It considers two distinct cases:

1. fixed interdiction costs;
2. linear interdiction costs.

It is shown that both the cases can be solved in polynomial time. This shows that the problem's behaviour is different from the minimum spanning tree interdiction problem which is NP-hard in general.

The reminder of this paper is organized as follows. Section 2 states some preliminaries which are used in the next sections. Section 3 formally introduces the problem and formulates it as a bilevel programming problem. Sections 4 and 5 considers the problem for fixed and linear costs, respectively. Finally, some concluding remarks are given in Section 6.

2 Preliminaries

This section states some notations used throughout the paper. Moreover, it recalls the min-max spanning tree problem and states some results on it.

Suppose that a graph $G(V, E)$ is given, where $V = \{1, 2, \dots, n\}$ is the node set and $E \subseteq \{\{i, j\} : i, j \in V \text{ and } i \neq j\}$ is the edge set. We say that an edge is incident to a node i if one of its endpoints is i. We apply the customary notation (i, j) to denote the edge whose two endpoints are i and j. A path from i to j is a sequence of edges so that any two consecutive edges intersect in one of their endpoints and additionally, the first and last edges are incident to i and j, respectively. A path without two repetitive nodes is called an elementary path. Since we only deal with elementary paths, let us use the simple term "path" instead of "elementary path" throughout the paper.

A graph $G'(V', E')$ is said to be a subgraph of G if $V' \subseteq V$ and $E' \subseteq E$. A subgraph $G'(V', A')$ is called spanning if it contains all node of G, i.e., $V' = V$. A (sub)graph is said to be connected if there is at least a path between any two nodes. Any maximal connected subgraph of G is referred to its connected component. So, G contains only one connected component if it is connected. Throughout this paper, we assume that G is connected. A cut is a set of edges whose removal exactly converts the graph into two connected components. A cut can be determined uniquely by the node set of the one of its connected components.

A path from a node to itself is called a cycle. A set of edges which does not contain any cycle is referred to as a forest. A forest of G with n nodes and $n-1$ edges is called a spanning tree. There is exactly a unique path between any two nodes in a spanning tree. In other words, a spanning tree of G is a spanning and connected subgraph which does not any cycle.

2.1 Bottleneck spanning tree problem

Now, suppose that a nonnegative weight w_{ij} is associated to each edge $(i, j) \in E$. The min-max (bottleneck) spanning tree (MMST) problem is to look for a spanning tree in G so that the maximum of its weights is minimized. This problem is formulated as a combinatorial optimization problem in the following form.

$$\min_{T \in \mathbb{T}} w(T), \tag{1}$$

in which $\mathbb{T}$ is the set of all spanning trees of G, and $w(T) = \max_{(i,j) \in T} w_{ij}$.

There are several formulations for the well-known minimum spanning tree (MST) problem (see [6, 15], for instance). Since the only difference between the MST and MMST problems is in their objective functions, one can use any formulation of MST problems to model MMST problems. Based on this fact, a flow-based formulation of MMST problems is stated below.

$$\min z \tag{2a}$$
$$w_{ij}(y_{ij}+y_{ji}) \le z \qquad \forall (i,j)\in E, \tag{2b}$$
$$\sum_{j:(i,j)\in E} x_{ij}-x_{ji} = \begin{cases} n-1 & i=1,\\ -1 & i\neq 1,\end{cases} \qquad \forall i\in V, \tag{2c}$$
$$y_{ij} \le x_{ij} \le (n-1)y_{ij} \qquad \forall (i,j)\in E, \tag{2d}$$
$$y_{ji} \le x_{ji} \le (n-1)y_{ji} \qquad \forall (i,j)\in E, \tag{2e}$$
$$y_{ij}+y_{ji} \le 1 \qquad \forall (i,j)\in E, \tag{2f}$$
$$y_{ij}, y_{ji} \in \{0,1\} \qquad \forall (i,j)\in E, \tag{2g}$$
$$x_{ij}, x_{ji} \ge 0 \qquad \forall (i,j)\in E, \tag{2h}$$

in which x_{ij} and x_{ji} are flow values on edge (i,j) from i to j and from j to i, respectively, and y_{ij} and y_{ji} are zero-one variables to determine whether any flow passes through edge (i,j), or not. Constraint (2b) guarantees that z is the maximum weight of the desired spanning tree. Constraint (2c) states that a single unit of flow has to be sent from node 1 to the other nodes. Constraints (2d) and (2e) ensure that $y_{ij}=1$ $(y_{ij}=1)$ if and only if at least one unit of flow passes through (i,j) $((j,i)$. Constraint (2f) is added due to sending flow only on a direction of any edge.

Liu and Yao [13] used an optimality condition of MMST problems to investigate inverse MMST problems. Let us mention it below.

Lemma 1. *(Lemma 2.1 in [13]) A spanning tree T of G is a min-max spanning tree under a weight vector $\mathbf{w}$ if and only if G becomes disconnected after deleting the edges whose costs are not less than $\max_{(i,j)\in T} w_{ij}$.*

Using the notion of cuts, one can simply state Lemma 1 in another fashion.

Lemma 2. *A spanning tree T is a min-max spanning tree with optimal value p if and only if there is a cut C so that*

$$\begin{cases} w_{ij} \le p & (i,j)\in T,\\ w_{ij} \ge p & (i,j)\in C,\end{cases} \qquad \forall (i,j)\in E. \tag{3}$$

Proof. For a proof, see [19]. □

Notice that, based on optimality conditions (3), all edges belonging to $C\cap T$ have the same weight equal to p.

Remark 1. *Although the min-max spanning tree problem is considered in this section, similar results can be obtained for the max-min spanning tree problems*

3 Problem statement

This section introduces the bottleneck (min-max) spanning tree interdiction problem and formulates it.

In the conventional min-max spanning tree problem, a decision maker, called the user, wants to find a spanning tree so that its maximum weight is minimized. So he/she requires to solve problem (2). Now, consider the situation in which another decision maker, called the attacker, wants to increase the objective value of problem (2) since he/she follows a conflicting goal. This can be achieved either by removing some edges or by increasing edge weights. This paper selects the latter because as early discussed, the former is a special case of the other. Two constraints restrict the attacker in increasing weights:

- The bound constraints: Each edge has an upper bound w_{ij}^U which is the maximum amount that its weight is allowed to be increased.
- The budget constraint: There is a total budget B which can be spent for increasing weights.

Based on this argument, the min-max spanning tree interdiction problem is formulated as follows:

$$\max z' \tag{4a}$$
$$\sum_{(i,j)\in E} b_{ij}(\bar{w}_{ij}) \le B, \tag{4b}$$
$$w_{ij} \le \bar{w}_{ij} \le w_{ij}^U \qquad (i,j)\in E, \tag{4c}$$
$$z' = \min\{z : \tag{4d}$$
$$\bar{w}_{ij}(y_{ij}+y_{ji}) \le z \qquad \forall (i,j)\in E, \tag{4e}$$
$$\sum_{j:(i,j)\in E} x_{ij} - x_{ji} = \begin{cases} n-1 & i=1, \\ -1 & i\ne 1, \end{cases} \qquad \forall i \in V, \tag{4f}$$
$$y_{ij} \le x_{ij} \le (n-1)y_{ij} \qquad \forall (i,j)\in E, \tag{4g}$$
$$y_{ji} \le x_{ji} \le (n-1)y_{ji} \qquad \forall (i,j)\in E, \tag{4h}$$
$$y_{ij} + y_{ji} \le 1 \qquad \forall (i,j)\in E, \tag{4i}$$
$$y_{ij}, y_{ji} \in \{0,1\} \qquad \forall (i,j)\in E, \tag{4j}$$
$$x_{ij}, x_{ji} \ge 0 \qquad \forall (i,j)\in E\}. \tag{4k}$$

Formulation (4) is a bilevel programming problem in which $\bar{w}_{ij}$'s are decision variables of the first level to be determined by the attacker and b_{ij} is a cost function of (i,j) which calculates the cost amount of increasing weight of (i,j) from w_{ij} to $\bar{w}_{ij}$. The second level is an instance of problem (2) with respect to new edge weights $\bar{w}_{ij}$.

This paper considers problem (4) for two distinct cases of cost functions $b_{ij}(\bar{w}_{ij})$. In the first case, $b_{ij}(\bar{w}_{ij})$ is a bi-valued function. It takes zero if $\bar{w}_{ij} = w_{ij}$ and a fixed value $r_{ij} > 0$ otherwise. In the second case, $b_{ij}(\bar{w}_{ij})$ is a linear function

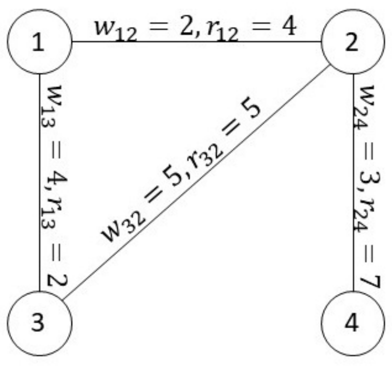

Figure 1: An instance of the min-max spanning tree interdiction problem with the total budget $B = 6$ and $w_{ij}^U = 6$ for every edge (i,j)

Table 1: The payoff matrix of players

Spanning trees\Cuts	$\{(1,2),(1,3)\}$	$\{(1,2),(2,3)\}$	$\{(1,3),(2,3)\}$	$\{(2,4)\}$
$\{(1,3),(1,2),(2,4)\}$	6	6	6	4
$\{((1,3),(3,2),(2,4)\}$	6	5	6	5
$\{(1,2),(2,3),(2,4)\}$	6	6	5	5

as $c_{ij}(\bar{w}_{ij} - w_{ij})$ in which c_{ij} is a fixed value equal to the cost per unit increment in (i,j). The first case is an extension of the problem in which the attacker's strategy is to remove edges because one can set upper bounds to a very large number so that if the attacker increases the weight of an edge to its upper bound, then the user never selects the edge. In the special case when all $r_{ij} = 1$, the problem is converted to finding the B-most vital edges [10, 16].

4 Fixed costs

This section considers problem (4) in the case that costs are fixed. It presents an efficient algorithm to solve the problem.

Based on Lemma 1, if the attacker increases weights of a cut, then the best payoff of the user is at most equal to the maximum weight of the cut. So it is reasonable that the attacker begins with increasing weights of the cut (from small to large) until his/her budget is insufficient. This gives us the idea that a pure strategy of the attacker is to choose a cut while the pure strategy of the user is to select a spanning tree. As an example, consider the instance shown in Figure 1. It contains 3 spanning trees and 4 cuts. Before the attacker's play, the user's optimal spanning tree is $\{((1,3),(1,2),(2,4)\}$ whose weight is equal to 4. If the attacker chooses the cut $\{(1,2),(1,3)\}$, then he/she first increases the weight of $(1,2)$ from 2 to 6 and then increases the weight of $(1,3)$ from 4 to 6. This causes that the optimal weight of the user becomes 6. One can simply check that the same solution is optimal. Table 1 displays the payoff matrix for the players. It is easy to see that the Nash-equilibrium strategy is the same optimal solution.

Although this idea can be extended to the general case, it cannot be applied to as an approach for solving problem (4) because the number of the players' pure strategies grows exponentially. However, it gives the worthwhile result that problem (4) can be seen to as a simultaneous zero-sum game which has always a pure equilibrium solution. Notice that if the attacker chooses a cut C, then he/she can solve the following bottleneck knapsack problem to understand how to increase edge weights:

$$\begin{aligned} \min \quad & \max_{(i,j)\in C}\{\bar{w}_{ij}\} \\ \text{s.t.} \quad & \sum_{(i,j)\in C} b_{ij}(\bar{w}_{ij}) \le B, \\ & w_{ij} \le \bar{w}_{ij} \le w^U_{ij} \qquad (i,j)\in C, \end{aligned}$$

where the parameters are defined similar to problem (2). It is easy to see that the optimal value of this problem is a upper bound on the payoff of the zero-sum game.

To develop an efficient algorithm, we introduce a new weight vector corresponding to a cut C and a value p. This vector is denoted by $\bar{w}^{(C,p)}_{ij}$ and is defined as

$$\bar{w}^{(C,p)}_{ij} = \begin{cases} p & w_{ij} < p \wedge (i,j)\in C, \\ w_{ij} & \text{otherwise}, \end{cases} \qquad \forall (i,j)\in E. \tag{5}$$

It is easy to see that the user's payoff is at least equal to p with respect to the new weight vector $\bar{\mathbf{w}}$.

Lemma 3. *If there is a feasible weight vector of problem (4) whose objective value is equal to p, then there is a cut so that $\bar{\mathbf{w}}^{(C,p)}$ is also feasible.*

Proof. The proof is straightforward. □

Based on Lemma 3, we can restrict ourselves to weight vectors defined as (5) to find an optimal solution. The proposed approach is to find the greatest value of p so that $\bar{\mathbf{w}}^{(C,p)}$ satisfies the bound and budget constraints. For this purpose, we introduce a new cost vector $\mathbf{b}^p$ as follows:

$$b^p_{ij} = \begin{cases} 0 & p \le w_{ij}, \\ r_{ij} & w_{ij} < p \le w^u_{ij}, \\ +\infty & w^u_{ij} < p, \end{cases} \qquad \forall (i,j)\in E. \tag{6}$$

The following lemma states the relationship between the new weight vector and the new cost vector.

Lemma 4. *The capacity of a cut C with respect to $\mathbf{b}^p$ is less than or equal to B if and only if $\bar{\mathbf{w}}^{(C,p)}$ satisfies the bound and budget constraints.*

Proof. By definition, the proof is immediate. □

We are ready to state our proposed algorithm in complete details. This algorithm uses a binary search on a set of possible objective values to look for the greatest value p so that $\bar{\mathbf{w}}^{(C,p)}$ satisfies the bound and budget constraints for some cut C. From Lemma 4, a minimum cut can be found with respect to $\mathbf{b}^p$. If the capacity of the minimum cut is less than or equal to B, then $\bar{\mathbf{w}}^{(C,p)}$ is feasible and otherwise, there is no feasible solution with the objective value greater than or equal to p. The following lemma establishes the search space for finding the optimal value.

Lemma 5. *The optimal value of problem (4) belongs to* $\bigcup_{(i,j)\in E}\{w_{ij}, w_{ij}^U\}$.

Proof. The proof is trivial. □

The algorithm is stated formally in Algorithm 1.

Algorithm 1 to solve problem (4) with fixed costs.

```
Input: An instance of problem (4) with fixed costs.
Output: An optimal solution $\bar{\mathbf{w}}^*$ with optimal value $z^*$.
Sort the elements of $\bigcup_{(i,j)\in E}\{w_{ij}, w_{ij}^U\}$ in nondecreasing order. Let $p_1 \le p_2 \le \dots p_{2m}$ be the ordered list.
Set $L = 1$ and $U = 2m$.
Find the minimum cut $C$ with respect to $\mathbf{b}^{p_U}$ defined by (6).
if the capacity of $C$ is greater than $B$ then
    Stop because the problem is infeasible.
end if
while $U - L \ge 1$ do
    Set $mid = \lceil \frac{L+U}{2} \rceil$.
    Find the minimum cut $C$ with respect to $\mathbf{b}^{p_{mid}}$ defined by (6).
    if the capacity of $C$ is greater than $B$ then
        Set $U = mid - 1$.
    else:
        Set $L = mid$, $\bar{\mathbf{w}}^* = \bar{\mathbf{w}}^{(C,p_{mid})}$ and $z^* = \max_{(i,j)\in C} w_{ij}^*$.
    end if
end while
```

Theorem 1. *Algorithm 1 solves problem* (4) *with fixed costs in* $O(\log(n)S(n,m))$ *where* $S(n,m)$ *is the required time for finding a minimum cut in the network* G *with* n *nodes and* m *edges.*

Proof. The correctness of Algorithm 1 is valid based on the above argument. Since the bottleneck operation of the algorithm is finding a minimum cut in While loop, and the number of iterations in the loop are $O(\log(2m)) = O(\log(n))$, it follows that the complexity of Algorithm 1 is $O(\log(n)S(n,m))$. □

Since a minimum cut can be iteratively constructed by $n-1$ maximum flow computations in $(n-1)O(mn\log n) = O(mn^2\log n)$ time [8, 17], it follows that Algorithm 1 solves problem (4) with fixed costs in polynomial time.

5 Linear costs

In this section, we consider problem (4) with linear costs. Hereafter, it is assumed that $b_{ij}(\bar{w}_{ij}) = c_{ij}(\bar{w}_{ij} - w_{ij})$ for every edge $(i,j) \in E$.

Since the only difference between the problems with linear and fixed costs is in the budget constraint, we can use the same notion of solution presented in (5) again. For this purpose, we recall that if the capacity of a minimum cut C with respect to $\mathbf{b}'^p$ defined as

$$b'^p_{ij} = \begin{cases} 0 & p \le w_{ij}, \\ c_{ij}(p - w_{ij}) & w_{ij} < p \le w^u_{ij}, \\ +\infty & w^u_{ij} < p, \end{cases} \tag{7}$$

is less than or equal to B, then the vector $\bar{w}^{(C,p)}$ defined by (5) is a feasible solution of problem (4) with linear costs. On the other hand, if the capacity of a minimum cut C with respect to $\mathbf{b}'^p$ is greater than B, then problem (4) does not contain any feasible solution with the objective value better than p. So, we restrict ourselves to the solution in the form of (5) to find an optimal solution. Hereafter, we denote the total cost corresponding to $\bar{w}^{(C,p)}$ by $c(p)$, i.e.,

$$c(p) = \sum_{(i,j)\in C} c_{ij}(\bar{w}^{(C,p)}_{ij} - w_{ij}).$$

As mentioned, Algorithm 1 searches the optimal value z^* among a finite number of elements (see Lemma 5). In the case of linear costs, the total cost is not discrete and Lemma 5 is not valid. So, we have to look for the optimal value in a continuous interval. Hence, by using the binary search, we can find a small interval containing the optimal value z^*. Nevertheless, we have to reply this question how to find the exact value z^* in a small interval. For this purpose, we must impose an additional assumption to the problem.

Assumption All the parameters of problem (4) are integer. It is obvious that if all parameters are rational, then we can transform them into integer numbers by multiplying them with the common denominator of all considered rational numbers. Moreover, if some parameters are irrational, then we have to convert them into rational numbers for storing them on a computer. Hence, Assumption **??** is not a restrictive assumption in practice.

Lemma 6. *Let z^* be the optimal value of problem* (4) *and $\bar{\mathbf{w}}^* = \mathbf{w}^{(C,z^*)}$ be its corresponding optimal solution for some minimum cut C. Either $z^* \in \bigcup_{(i,j)\in E}\{w^U_{ij}\}$ or the budget constraint is satisfied in the equality form, i.e.,*

$$c(z^*) = \sum_{(i,j)\in C} c_{ij}(\bar{w}^{(C,z^*)} - w_{ij}) = \sum_{(i,j)\in C} c_{ij} \max\{0, z^* - w_{ij}\} = B.$$

Proof. If $z^* \notin \bigcup_{(i,j)\in E}\{w^U_{ij}\}$, then any bound constraint is not binding. So, if the budget constraint is not satisfied in the equality form, then we can increase z^* by a small value ϵ satisfying all the constraints in the first level. This is a contradiction with the fact that z^* is the optimal value. □

Suppose that we have found an interval (p_L, p_U) containing the optimal value and its length is less than 1. Without loss of the generality, we can assume that

$$(p_L, p_U) \cap \bigcup_{(i,j)\in E} \{w_{ij}, w_{ij}^U\} = \emptyset$$

because based on Assumption **??**, they have at least one common element p, which can be removed in an additional step by assuming either $p_L = p$ or $p_U = p$. Suppose that $E' = \{(i,j) : w_{ij} < p_L \text{ and } p_U < w_{ij}^U\}$ is the set of edges which are potential to be modified. By this argument, we know that the optimal value satisfies the budget constraint in the equality form. So, the optimal value can be obtained explicitly as follows:

$$\begin{aligned} & \sum_{(i,j)\in C} c_{ij} \max\{0, z^* - w_{ij}\} = B, \\ \Longrightarrow \quad & \sum_{(i,j)\in C\cap E'} c_{ij} z^* - c_{ij} w_{ij} = B, \\ \Longrightarrow \quad & z^* = \frac{B + \sum_{(i,j)\in C\cap E'} c_{ij} w_{ij}}{\sum_{(i,j)\in C\cap E'} c_{ij}}. \end{aligned} \tag{8}$$

Hence, we have proved the following lemma.

Lemma 7. *If (p_L, p_U) is an interval which contains the optimal value and it has not any intersection with $\bigcup_{(i,j)\in E}\{w_{ij}, w_{ij}^U\}$, then finding the optimal value is equivalent to solving the following maximum ratio cut problem* (8)*.*

$$\max_{C\in\mathbb{C}} \frac{B + \sum_{(i,j)\in C\cap E'} c_{ij} w_{ij}}{\sum_{(i,j)\in C\cap E'} c_{ij}} \tag{9}$$

in which $E' = \{(i,j) : w_{ij} < p_L \wedge p_U < w_{ij}^U\}$ and $\mathbb{C}$ is the set of all cuts.

Since the set $\mathbb{C}$ is finite, then the number of objective values in problem (9) is also finite. If we can find a small interval (p_L, p_U) containing only one objective value, then it is the same optimal value. For this purpose, let $p_1 = \frac{B+\sum_{(i,j)\in C_1\cap E'} c_{ij} w_{ij}}{\sum_{(i,j)\in C_1\cap E'} c_{ij}}$ and $p_2 = \frac{B+\sum_{(i,j)\in C_2\cap E'} c_{ij} w_{ij}}{\sum_{(i,j)\in C_2\cap E'} c_{ij}}$ be two distinct objective values for some cuts $C_1, C_2 \in \mathbb{C}$. Then,

$$\begin{aligned} |p_2 - p_1| &= \left|\frac{B + \sum_{(i,j)\in C_1\cap E'} c_{ij} w_{ij}}{\sum_{(i,j)\in C_1\cap E'} c_{ij}} - \frac{B + \sum_{(i,j)\in C_2\cap E'} c_{ij} w_{ij}}{\sum_{(i,j)\in C_2\cap E'} c_{ij}}\right| && (10) \\ &\geq \frac{1}{\sum_{(i,j)\in C_1\cap E'} c_{ij} \sum_{(i,j)\in C_2\cap E'} c_{ij}} && (11) \\ &\geq \frac{1}{n^2 C^2}. && (12) \end{aligned}$$

where $C = \max_{(i,j)\in E}\{c_{ij}\}$. Here, the first inequality is derived from Assumption **??** and the fact that $p_1 \neq p_2$. So, if the length of (p_L, p_U) is less than $\frac{1}{n^2C^2}$, then we can assure that there is only one value $\frac{B+\sum_{(i,j)\in C\cap E'} c_{ij}w_{ij}}{\sum_{(i,j)\in C\cap E'} c_{ij}}$ at this interval which is the same optimal value. Hence, the optimal value can be computed by a linear search from either p_L or p_U.

Algorithm 2 describes our proposed approach for solving problem (4) with linear costs.

Theorem 2. *Algorithm 2 solves problem* (4) *in* $O(\log(nCW)S(n,m))$ *time where* $W = \max_{(i,j)\in E}\{w_{ij}^U\}$, $C = \max_{(i,j)\in E}\{c_{ij}\}$, *and* $S(n,m)$ *is the required time for finding a minimum cut in the network* G *with* n *nodes and* m *edges.*

Proof. The correctness of Algorithm 2 follows from the above argument. The length of the initial interval (p_L, p_U) is at most nW. On the other hand, the While loop decreases it by a factor 2 at each iteration until its length becomes less than $\frac{1}{n^2C^2}$. So, the number of iterations of While loop is at most $\log(nCW) = \log(n^3C^2W)$. Since the bottleneck operation of the algorithm is to find a minimum cut in While loop, it follows that the complexity of Algorithm 2 is $O(\log(nCW)$ $S(n,m))$. □

6 Conclusion

This paper studied the bottleneck spanning tree interdiction problem under two kinds of edge costs: fixed costs and linear costs. In both the cases, it presented the algorithms to solve the problem in polynomial time. Both the algorithms are designed based on the divide-and-conquer approach.

As the future works, it will be meaningful to consider other interdiction problems on spanning tree structures, such as min+sum spanning tree interdiction problem.

Algorithm 2 to solve problem (4) with linear costs.

```
Input: An instance of problem (4) with linear costs.
Output: An optimal solution $\bar{\mathbf{w}}^*$ with optimal value $z^*$.
Find a min-max spanning tree with respect to $\mathbf{w}^U$. Let $p_U$ be its optimal value.
Find the minimum cut $C$ with respect to $\mathbf{b}^{p_U}$ defined by (6).
if the capacity of $C$ is less than or equal to $B$ then
    Stop because the problem has the optimal solution $\bar{\mathbf{w}}^* = \bar{\mathbf{w}}^{(C,p_U)}$ and $z^* = p_U$.
end if
Set $p_L = 0$
while $p_U - p_L \geq \frac{1}{n^2C^2}$ do
    Set $p = [\frac{p_U+p_L}{2}]$.
    Find the minimum cut $C$ with respect to $\mathbf{b}^p$ defined by (6).
    if the capacity of $C$ is equal to $B$ then
        Stop because the problem has the optimal solution $\bar{\mathbf{w}}^* = \bar{\mathbf{w}}^{(C,p)}$ and $z^* = \max_{(i,j)\in C} w^*_{ij}$.
    else if the capacity of $C$ is greater than $B$ then
        Set $p_U = p$.
    else
        Set $p_L = p$.
    end if
end while
if $(p_L, p_U) \cap \bigcup_{(i,j)\in E}\{w_{ij}, w^U_{ij}\} \neq \emptyset$ then
    Let $p \in (p_L, p_U) \cap \bigcup_{(i,j)\in E}\{w_{ij}, w^U_{ij}\}$.
    Find the minimum cut $C$ with respect to $\mathbf{b}^p$ defined by (6).
    if the capacity of $C$ is equal to $B$ then
        Stop because the problem has the optimal solution $\bar{\mathbf{w}}^* = \bar{\mathbf{w}}^{(C,p)}$ and $z^* = p$.
    else if the capacity of $C$ is greater than $B$ then
        Set $p_U = p$.
    else
        Set $p_L = p$.
    end if
end if
Obtain the capacities $m_L$ and $m_U$ of minimum cuts with respect to $\mathbf{b}^{p_L}$ and $\mathbf{b}^{p_U}$, respectively.
Set $p_1 = p_L + \frac{B-c(p_L)}{m_L}$ and $p_2 = p_U + \frac{B-c(p_U)}{m_U}$.
Find the minimum cut $C$ with respect to $\mathbf{b}^{p_1}$ defined by (6).
if the capacity of $C$ is equal to $B$ then
    The problem has the optimal solution $\bar{\mathbf{w}}^* = \bar{\mathbf{w}}^{(C,p_1)}$ and $z^* = p_1$.
else
    Find the minimum cut $C$ with respect to $\mathbf{b}^{p_2}$ defined by (6).
    The problem has the optimal solution $\bar{\mathbf{w}}^* = \bar{\mathbf{w}}^{(C,p_2)}$ and $z^* = p_2$.
end if
```

References

[1] Abdolahzadeh, A., Aman, M. and Tayyebi, J., *Minimum st-cut interdiction problem*, Computers & Industrial Engineering, **148**, (2020), 106708.

[2] Ahuja, R. K., Magnanti, T. L. and Orlin, J. B., *Network Flows: Theory, Applications and Algorithms*, Prentice-Hall, Englewood Cliffs, New Jersey, USA Arrow, KJ, (1993).

[3] Bazgan, C., Toubaline, S. and Vanderpooten, D., *Critical edges/nodes for the minimum spanning tree problem: complexity and approximation*, Journal of Combinatorial Optimization, **26**, (2013), no. 1, 178-189.

[4] Bazgan, C., Toubaline, S. and Vanderpooten, D., *Efficient determination of the k most vital edges for the minimum spanning tree problem*, Computers and Operations Research, **39**, (2012), no. 11, 2888-2898.

[5] Camerini, P. M., *The min-max spanning tree problem and some extensions.* Information Processing Letters, **7**, (1978), no. 1, 10-14.

[6] Feremans, C., Labbé, M. and Laporte, G., *A comparative analysis of several formulations for the generalized minimum spanning tree problem*, Networks: An International Journal, **39**, (2002), no. 1, 29-34.

[7] Ghorbani-Renani, N., González, A. D. and Barker, K., *A decomposition approach for solving tri-level defender-attacker-defender problems*, Computers and Industrial Engineering, **153**, (2021), 107085.

[8] Hartmann, T. and Wagner, D., *Fast and simple fully-dynamic cut tree construction*, In International Symposium on Algorithms and Computation, Springer, Berlin, Heidelberg, (2012), 95-105.

[9] Holzmann, T. and Smith, J. C., *The shortest path interdiction problem with randomized interdiction strategies: Complexity and algorithms*, Operations Research, **69**, (2021), no. 1, 82-99.

[10] Hsu, L. H., Jan, R. H., Lee, Y. C., Hung, C. N. and Chern, M. S., *Finding the most vital edge with respect to minimum spanning tree in weighted graphs*, Information Processing Letters, **39**, (1991), no. 5, 277-281.

[11] Johnson, M. P., Gutfraind, A. and Ahmadizadeh, K., *Evader interdiction: algorithms, complexity and collateral damage*, Annals of operations research, **222**, (2014), no. 1, 341-359.

[12] Julien, L. A., *Stackelberg games*, In Handbook of Game Theory and Industrial Organization, Volume I, Edward Elgar Publishing, 2018.

[13] Liu, L. and Yao, E., *Inverse min-max spanning tree problem under the weighted sum-type Hamming distance*, In International Symposium on

Combinatorics, Algorithms, Probabilistic and Experimental Methodologies, Springer, Berlin, Heidelberg, (2007), 375-383.

[14] Mohammadi, A. and Tayyebi, J. (2019). *Maximum capacity path interdiction problem with fixed costs*. Asia-Pacific Journal of Operational Research, **36** (2019), no. 4, 1950018.

[15] Pop, P. C., *The generalized minimum spanning tree problem: An overview of formulations, solution procedures and latest advances*, European Journal of Operational Research, **283**, (2020), no. 1, 1-15.

[16] Shen, H., *Finding the k most vital edges with respect to minimum spanning tree*, Acta Informatica, **36**, (1999), no. 5, 405-424.

[17] Sleator, D. D. and Tarjan, R. E., *A data structure for dynamic trees*, Journal of computer and system sciences, **26**, (1983), no. 3, 362-391.

[18] Smith, J. C. and Song, Y., *A survey of network interdiction models and algorithms*, European Journal of Operational Research, **283** (2020), no. 3, 797-811.

[19] Tayyebi, J. and Sepasian, A. R., *Partial inverse min-max spanning tree problem*, Journal of Combinatorial Optimization, **40**, (2020), no. 4, 1075-1091.

[20] Washburn, A. and Wood, K., *Two-person zero-sum games for network interdiction*, Operations research, **43**, (1995), no. 2, 243-251.

[21] Wei, N., Walteros, J. L. and Pajouh, F. M., *Integer programming formulations for minimum spanning tree interdiction*, INFORMS Journal on Computing, **33**, (2021), no. 4, 1461-1480.

[22] Zenklusen, R., *Matching interdiction*, Discrete Applied Mathematics, **158** (2010), no. 15, 1676-1690.

[23] Zenklusen, R., *An $O(1)$-approximation for minimum spanning tree interdiction*, In 2015 IEEE 56th Annual Symposium on Foundations of Computer Science (2015), 709-728.

[24] Zhang, Q., Guan, X., Wang, H. and Pardalos, P. M., *Maximum shortest path interdiction problem by upgrading edges on trees under hamming distance*, Optimization Letters, **15** (2021), no. 8, 2661-2680.