Integer Programming Formulations for Minimum Spanning Tree Interdiction

Ningji Wei, Jose L. Walteros[*]
Department of Industrial and Systems Engineering
University at Buffalo
josewalt@buffalo.edu ningjiwe@buffalo.edu

Foad Mahdavi Pajouh
Department of Management Science and Information Systems
University of Massachusetts Boston
Foad.Mahdavi@umb.edu

Abstract

We consider a two-player interdiction problem staged over a graph where the leader's objective is to minimize the cost of removing edges from the graph so that the follower's objective, i.e., the weight of a minimum spanning tree in the residual graph, is increased up to a predefined level r. Standard approaches for graph interdiction frame this type of problems as bi-level formulations, which are commonly solved by replacing the inner problem by its dual to produce a single level reformulation. In this paper we propose an alternative integer program derived from the combinatorial structure of the follower's solution space and prove that this new formulation yields a stronger linear relaxation than the bi-level counterpart. We also study the convex hull of the feasible solutions of the problem and identify several families of facet-defining inequalities that can be used to strengthen the proposed integer program. We then proceed by introducing an alternative formulation defined by a set of so-called supervalid inequalities that may exclude feasible solutions, albeit solutions whose objective value is not better than that of an edge cut of minimum cost. We discuss several computational aspects required for an efficient implementation of the proposed approaches. Finally, we perform an extensive set of computational experiments to test the quality of our approaches by solving a large collection of real-life and randomly generated instances with various configurations, analyzing and comparing the benefits of each model, and also identifying further enhancements.

Keywords: Network Interdiction, Minimum Spanning Tree, Minimum Edge Blocker Problems, Bi-level Optimization.

1 Introduction and Motivation

We study an attacker-defender interdiction problem staged over a graph where two decision makers—an *attacker* who plays first and a *defender* who plays second—sequentially solve interrelated problems that optimize conflicting objective functions. In this particular context, the objective of the attacker is to minimize the cost of inflicting a given disruption level to the defender's objective by removing edges from the graph. In turn, the objective of the defender is to identify a minimum spanning tree that remains available in the residual graph.

Graph interdiction has been an ongoing research endeavor for both academicians and practitioners for many years (Corley and Sha 1982, Israeli and Wood 2002, Kennedy et al. 2011). Despite having their root in military applications (Ghare et al. 1971, Wood 1993), these models have recently gained traction in many other areas thanks to their ability of informing decision makers about vulnerabilities on infrastructure and/or

[*]Corresponding author. Phone: 716-645-8876; Fax: 716-645-3302; Address: 413 Bell Hall, Buffalo, NY 14260; Email: josewalt@buffalo.edu

operational networks. Therefore, it is common to find graph interdiction applications in areas as diverse as homeland security (Grubesic et al. 2008, Houck et al. 2004), evacuation planning (Matisziw and Murray 2009), immunization strategies (Tao et al. 2005), energy systems (Salmeron et al. 2004), communications (Dinh et al. 2012) and transportation (Jenelius et al. 2006), among others.

As attacker-defender problems keep permeating the applied literature, many researchers have focused on developing general models to study the disruption of different structures and operational properties of graphs that are amenable for modeling a wide variety of applications (Lozano and Smith 2016, Chestnut and Zenklusen 2017). Among all these models, the ones designed to disrupt optimal solutions of network flow problems, such as shortest paths, maximum flow, and minimum cost flow problems (Corley and Sha 1982, Israeli and Wood 2002, Kennedy et al. 2011, Lim and Smith 2007, Matisziw and Murray 2009, Wollmer 1964, Wood 1993) are the ones that have received the most attention. These problems have gained great notoriety because the wide range of applications that can be modeled with them, but also because most of such underlying network flow problems admit linear formulations that permit applying traditional dualize-and-combine solution strategies (Cormican et al. 1998).

A second area that has gained some traction lately deals with a specific type of problems often called *critical elements detection problems* that study vulnerabilities of graphs with respect to connectivity and cohesiveness properties, such as the number of connected vertex pairs, the size of the largest connected component, and the number of connected components (Addis et al. 2013, Arulselvan et al. 2009, Di Summa et al. 2012, Dinh et al. 2012, Oosten et al. 2007, Shen et al. 2012, Veremyev et al. 2014). Applications of these problems are often used to find vulnerable elements in communication networks, or—in social network analysis—to identify “key players” used to propagate ideas like advertising campaigns over social networks.

Lastly, other variations that are worth mentioning are interdiction problems aimed to disrupt the sizes or relative weight of some topological subsets of vertices or edges, like spanning trees, dominating sets, central vertices (e.g., the 1-median or 1-center vertices), matchings, independent sets, cliques, and vertex covers (Bazgan et al. 2010, 2011, Frederickson and Solis-Oba 1999, Furini et al. 2019, Mahdavi Pajouh et al. 2014, 2015, Zenklusen 2010).

Among the different structures mentioned above, we focus our attention on interdicting minimum spanning trees. In general, spanning trees are among the most widely studied graph structures having both theoretical and practical properties often used to derive insights in many problem settings (Gabow et al. 1986, Magnanti and Wolsey 1995). Given their ability to identify low-cost connected subgraphs on targeted networks, spanning trees are often used for designing computer, telecommunications, transportation and infrastructure networks (Graham and Hell 1985); for building optimal circuits (Ohlsson et al. 2004); for handwriting recognition (Tapia and Rojas 2004), image registration and feature extraction in computer vision (Suk and Song 1984); and for social networks analysis (Ströele A. Menezes et al. 2008). Modeling spanning tree interdiction games can therefore help to analyze graph resiliency properties in a wide variety of contexts.

Traditional attacker-defender interdiction problems are typically modeled from two different perspectives depending on the way in which the interaction between the attacker's and defender's objectives is defined. From the first perspective, the attacker is provided with an interdiction budget for attacking elements in the graph—either vertices or edges—and its objective is to design an optimal attack that maximizes the overall disruption on the defender's objective. Alternatively, from the second perspective, the attacker minimizes the cost of attacking the graph so as to guarantee that a minimum required disruption level is inflicted over the defender's objective. In this paper we develop solution methods to tackle the latter version.

1.1 Previous Work

Early approaches for attacking minimum spanning trees on graphs focused on finding a single edge whose deletion results in the largest increase of the spanning tree's weight. This variation is often called the most vital edge with respect to the minimum spanning tree problem and is proven to be solvable in polynomial time (Hsu et al. 1991). Several approaches, including sequential (Hsu et al. 1991, Iwano and Katoh 1993), randomized (Shen 1995), and parallelized (Suraweera et al. 1995) algorithms have been proposed over the years to tackle this particular variation. These algorithms are mainly adaptations of traditional methods for finding minimum spanning trees that are also able to identify the most vital edge while preserving their original complexity.

The natural extension that attempts to find k-most vital edges for $k > 1$ has received significantly more attention over the past few years and, contrary to the single edge version, it has been proven to be NP-hard in the strong sense (Lin and Chern 1993, Frederickson and Solis-Oba 1999). Among all the techniques that have been developed to solve this problem, algorithms with an approximation guarantee and exact approaches have been the preferred choices by most researchers.

Frederickson and Solis-Oba (1999) were the first to propose an approximation algorithm for this problem that yields a proven $O(\log k)$ approximation for any given $k > 1$. Furthermore, for the case in which there is a cost associated with the removal of each edge and a deletion budget, their algorithm produces a $O(\log m)$ approximation (here, n and m represent the number of vertices and edges of the graph, respectively). Almost two decades later, Zenklusen (2015) was the first to discover a $O(1)$-approximation algorithm for the general cost version of this problem. This 14-approximation algorithm was the first approach proven to yield a constant approximation. More recently, Linhares and Swamy (2017) improved this approximation ratio to 4, after introducing some further enhancements to Zenklusen's algorithm.

As for exact approaches, Liang and Shen (1997) were the first to develop an exact algorithm based on total enumeration for the k-most vital edges problem with an overall running time of $O(n^{k+1})$. Subsequently, Bazgan et al. (2012) proposed an improved combinatorial branch-and-bound algorithm that runs in $O(n^k \log \alpha(2(n-1), n))$ time for any fixed k, where $\alpha(\cdot,\cdot)$ is the inverse of the Ackerman function. They also introduced the first integer program inspired by similar formulations used to solve other interdiction problems (Cormican et al. 1998).

A closely related variation of the k-most vital edges for the spanning tree problems is called the minimum edge blocker spanning tree problem, which is the problem of finding the minimum number of edges whose removal from G results in a graph with spanning trees with a weight no less than a constant r. The relationship between this and the k-most vital edge problem has been studied by Bazgan et al. (2013). The attacker-defender game that we study in this paper is a generalization of this edge blocker problem.

In addition to the above discrete versions in which the attacker completely removes some edges, there is also another continuous variation where the attacker attempts to find a minimum cost increase of the edge weights so that the value of all minimum spanning trees is equal to some target value. Frederickson and Solis-Oba (1999) studied the case in which the cost of increasing the weights is linear and nondecreasing. They proved that this version of the problem is solvable in polynomial time and provided an algorithm that runs in $O(n^3 m^2 \log(n^2/m))$ time. Baïou and Barahona (2008) proposed a linear programming formulation for this problem that can be used to tackle the case in which the costs are nondecreasing piecewise-linear.

1.2 Our Contributions

Despite the wide interest in these problems, most of the available literature is focused on the k-vital edge case instead of the more general version that also accounts for edge removal costs. Similarly, with the exception of Bazgan et al. (2013), most efforts have been concentrated in the version that maximizes the increment of the weight of the minimum spanning tree given a predefined attacking budget, and very little attention has given to the version that minimizes the interdiction costs given a desired disruption level. Furthermore, there have been only a few approaches that use integer programming techniques and, to the best of our knowledge, there has not been any study that analyzes the convex hull of the corresponding solution space for this problem. In this paper, we are interested in addressing some of the aforementioned gaps by developing an integer programming framework aimed to optimally solve minimum spanning tree interdiction problems. A summary of the four main contributions of this paper follows:

1. We provide a detailed study of the problem's solution space including: a characterization of the set of feasible strategies for the attacker; an integer programming formulation derived directly from such a characterization; a polyhedral analysis of the convex hull of feasible solutions; a collection of facet-defining

inequalities used to strengthen the given integer program; and an analytical comparison of the strength of other traditional models with respect to the proposed integer program.

2. We provide an alternative formulation defined by a set of so-called supervalid inequalities that may exclude feasible solutions, albeit solutions whose objective value is not better than that of an edge cut of minimum cost. We show that this alternative formulation is not only stronger than the first formulation, but also requires fewer constraints to be defined.

3. We develop efficient algorithms to solve these formulations under a traditional branch-and-cut framework and provide the specific details required for the proper implementation of our methods. We also made our implementation publicly available to provide other researchers with a direct benchmark.

4. We test our models on a wide variety of networks arising from real-life applications as well as on randomly generated graphs to compare their performance and identify potential improvements.

The rest of the paper is organized as follows. In Section 2, we provide a general definition of the problem and discuss its computational complexity. In Section 3, we describe two mathematical formulations, the first derived from a dualization approach commonly used to formulate interdiction problems, and the second derived from the characterization of the set of feasible strategies for the attacker. In Section 4, we provide a polyhedral study of the convex hull of the feasible solutions. In Section 5, we exploit some fundamental properties of the feasible solutions to produce a strong formulation defined by a set of so-called supervalid inequalities. In Section 6, we study the computational performance of the proposed formulations. Finally, in Section 7, we provide our concluding remarks. To streamline the discussion of the paper, we compiled the proofs of all the non-trivial mathematical statements introduced hereafter in the Online Supplement, Appendix A.

2 Problem Definition

We work with a simple, nonempty, connected graph $G = (V, E)$, where V is a set of n vertices and $E \subseteq \binom{V}{2}$ a set of m edges. We will refer to each edge by its index $e = 1, \ldots, m$ or by its unordered pair of endpoints $\{i, j\}$. We associate edge set E with two rational vectors $\mathbf{w} = [w_e]_{e\in E}$ and $\mathbf{c} = [c_e]_{e\in E}$, each comprising what we denote the *weights* and *costs* of the edges, respectively. A weight function $w(\cdot) : 2^E \to \mathbb{Q}$ and cost function $c(\cdot) : 2^E \to \mathbb{Q}$ are defined accordingly, so that $w(S) = \sum_{e\in S} w_e$ and $c(S) = \sum_{e\in S} c_e$, for any given edge subset $S \subseteq E$. A subset of edges $S \subseteq E$ is called an *edge cut* of G if graph $G(V, E \setminus S)$ is disconnected. The size of the smallest edge cut of a graph G is known as its edge connectivity and is denoted by $\lambda(G)$. Similarly, the cost of a minimum-cost edge cut in G with respect to the cost vector $\mathbf{c}$ is denoted by $\lambda_{\mathbf{c}}(G)$. The value of $\lambda_{\mathbf{c}}(G)$, and consequently of $\lambda(G)$, can be found in polynomial time for any graph G by solving a sequence of max-flow problems between the different pair of vertices in G (Gomory and Hu 1961).

We say that H is a subgraph of G, if the vertices and edges in H, denoted by $V(H)$ and $E(H)$, are subsets of V and E, respectively. A subgraph T of G is called a *tree* if every pair of vertices in $V(T)$ is connected by a unique path in T. Furthermore, T is called a *spanning tree* if it includes all the vertices in G, i.e., $V(T) = V$. We use Ω_G to denote the set of all spanning trees in G. For a given edge set $S \subseteq E$, let $\Omega_G[S] := \{T \in \Omega_G | S \subseteq E(T)\}$ be the set of all the spanning trees in G that contain all the edges in S. Whenever set S is a singleton (i.e., $S = \{e\}$ for a given $e \in E$), we simply use $\Omega_G[e]$. The weight of a spanning tree $T \in \Omega_G$ is denoted by $w(E(T))$, or simply $w(T)$. We call the spanning trees of minimum and maximum weight *minimum spanning trees* and *maximum spanning trees*, and denote their weight as $\underline{w}_G$ and $\overline{w}_G$, respectively. Both minimum and maximum spanning trees of G can be computed in polynomial time (Ahuja et al. 1993). If graph G is disconnected, we assume that $\Omega_G = \emptyset$, and consequently say that both $\underline{w}_G$ and $\overline{w}_G$ are infinity.

We provide a summary of the notation used throughout the paper as a reference table in the Online Supplement, Appendix B.

2.1 Attacking Spanning Trees

Formally, the spanning tree interdiction problem that we study in this paper is defined as follows. Given a graph $G = (V, E)$, cost and weight vectors $\mathbf{c}$ and $\mathbf{w}$ in $\mathbb{Q}^m$, respectively, and a scalar $r \in \mathbb{Q}$, the problem is aimed to find a subset of edges $S \subseteq E$ such that the weight of any spanning tree in $G(V, E \setminus S)$ is at least r and the cost $c(S)$ is minimum. That is,

$$\min_{S \subseteq E} \quad c(S) \tag{1}$$

$$\text{s.t.} \quad \min_{T \in \Omega_{G(V,E\setminus S)}} \{w(T)\} \geq r. \tag{2}$$

Importantly, other interdiction problems with a similar structure for which the attacking cost $\mathbf{c}$ is equal to $\mathbf{1}$ are often called in the literature *Edge Blocker Problems* (Bazgan et al. 2013, Mahdavi Pajouh et al. 2014, 2015). We will resort however to the term spanning tree interdiction (as in Zenklusen (2015)) given the general association with many other attacker-defender games of similar nature, where non-unitary vertex/edge removal costs are introduced.

Before addressing its computational complexity, we will discuss some basic characteristics of the problem. First, we will assume without loss of generality that the edge weights are nonnegative (i.e., $\mathbf{w} \geq \mathbf{0}$). Note that any instance with negative weights can be trivially transformed into an instance with nonegative weights; one just need add a sufficiently large constant M to all the weights to make them positive and then set $r + (n-1)M$ as the interdiction bound. Furthermore, we will also assume without loss of generality that the edge costs are positive (i.e., $\mathbf{c} > \mathbf{0}$). Notice that any edge with a negative deletion cost will be part of any optimal solution and since removing any edge with deletion cost zero does not affect the objective function, one might as well directly remove it from G. Thus, we can set any edge with non-positive cost as part of the attacker's strategy and solve the resulting problem in which all the edges have a positive cost.

Second, for a given edge $e \in E$, notice that if the minimum weight among all the spanning trees in $\Omega_G[e]$ is greater than or equal to r, edge e is in fact *irrelevant* for the attacker and will never be selected as part of any optimal solution. Then, one may equivalently solve the problem over $G(V, E \setminus \{e\})$ instead. Furthermore, checking whether there exists an irrelevant edge can be trivially done in polynomial time by performing m calls to a minimum spanning tree algorithm. We will assume then that set E contains no irrelevant edges.

Third, independently of the value of r, we note that the problem is always feasible because removing any subset of edges that disconnects the graph (i.e., an edge cut) is sufficient to raise the weight of the spanning trees in the resulting graph to infinity. As a consequence, the optimal objective function value of the problem is bounded above by $\lambda_{\mathbf{c}}(G)$. Furthermore, if the value of r is greater than the largest weight of a spanning tree in Ω_G, the optimal solution of the problem is $\lambda_{\mathbf{c}}(G)$ because the set of feasible solutions will simply be the set of edge cuts of G. Similarly, if the value of r is less than or equal to the weight of a minimum spanning tree in Ω_G, the empty set is a feasible solution to this problem. To avoid the aforementioned trivial cases, we will henceforth assume that the required disruption level r is strictly sandwiched between $\underline{w}_G$ and $\overline{w}_G$.

2.2 Computational Complexity

In this section, we discuss the complexity of spanning tree interdiction problems and briefly address the relationship with other well-studied variations. We begin by defining the decision (or recognition) version of this problem.

minimum spanning tree interdiction (Decision)

Instance: A graph $G = (V, E)$, a non-negative rational vector of edge weights $\mathbf{w}$, a positive rational vector of edge costs $\mathbf{c}$, and two rational numbers r and b (denoted by $\langle G(V, E), \mathbf{w}, \mathbf{c}, r, b\rangle$).

Question: Is there a subset of edges $S \subseteq E$ with edge cost $c(S)$ less than or equal to b such that the weight of a minimum spanning tree in graph $G(V, E \setminus S)$ is greater than or equal to r?

Theorem 1 *The minimum spanning tree interdiction problem is strongly NP-complete (Bazgan et al. 2013, Frederickson and Solis-Oba 1999).*

The NP-completeness of this problem should come at no surprise because it is clearly a generalization of the decision version of the k-most vital edges for the minimum spanning tree (Frederickson and Solis-Oba 1999) and the edge blocker spanning tree problems (Bazgan et al. 2013), which both assume $\mathbf{c} = \mathbf{1}$. Since both problems are known to be NP-hard, even for graphs with weights restricted to 0 or 1, the decision version of the minimum spanning tree interdiction problem is NP-complete too. Furthermore, the fact that this problem is NP-complete directly renders the two optimization versions we discussed previously NP-hard as well.

When the costs of removing the edges are all equal to one (i.e., $\mathbf{c} = \mathbf{1}$), Bazgan et al. (2013) also proved that both of these optimization problems are polynomial-time equivalent, which is expected given that both problems arise from the same decision problem. Naturally, this observation can be easily extended for the case in which $\mathbf{c} \in \mathbb{Q}^m$ by noting that if an efficient algorithm exists for solving any of the two variations, one can immediately solve instances of the other via binary search.

3 Mathematical Formulations

In this section we study two formulations for the proposed spanning tree interdiction problem. We begin our discussion by presenting a formulation that is derived directly from (1)–(2). This formulation is an adaptation from one originally introduced by Bazgan et al. (2012) to model the k-most vital edges variation of the problem, and is solved via a reformulation technique commonly used to tackle interdiction problems in which the defender's decisions are modeled as linear programs (Cormican et al. 1998, Lim and Smith 2007, Wood 1993).

3.1 Bi-level Mixed-Integer Programming Formulation

The minimum spanning tree problem is known to admit several compact, linear extended formulations that can be used directly to model the defender's problem in constraint (2). Here, we consider the following network design type of formulation (Magnanti and Wolsey 1995) that is commonly used to characterize the spanning trees in a graph G:

$$\min \quad \sum_{e \in E} w_e y_e \tag{3}$$

$$\sum_{\{j:\{i,j\}\in E\}} f_{ij}^l - \sum_{\{j:\{i,j\}\in E\}} f_{ji}^l = b_i^l \quad \forall i \in V,\, l \in V \setminus \{k\} \tag{4}$$

$$y_e \geq f_{ij}^l + f_{ji}^l \qquad \forall e = \{i,j\} \in E,\, l \in V \setminus \{k\} \tag{5}$$

$$\sum_{e \in E} y_e = n - 1 \tag{6}$$

$$f_{ij}^l, f_{ji}^l \geq 0 \qquad \forall \{i,j\} \in E,\, l \in V \setminus \{k\} \tag{7}$$

$$y_e \geq 0 \qquad \forall e \in E, \tag{8}$$

where $\mathbf{y} = [y_e]_{e \in E}$ represents the incidence vector of the spanning tree to be selected. In this setting, a vertex $k \in V$ is arbitrarily chosen to be the root of the spanning tree and a commodity l is defined for every vertex $l \in V \setminus \{k\}$. One unit of each commodity l, which is assumed to emanate from the root vertex k, is required to be delivered to vertex l. This is modeled by a parameter b_i^l that takes the values of $1, -1$, and 0 whenever $i = k$, $i = l$, and $i \in V \setminus \{k, l\}$, respectively. The commodities are allowed to flow through edge $e = \{i,j\} \in E$ in both directions—i to j and j to i—provided that edge e belongs to the selected spanning tree (i.e., $y_e = 1$). For a given edge $\{i,j\} \in E$, the flow decision variable f_{ij}^l indicates the flow of commodity l from i to j and the balance flow constraints (4) ensure that all the commodities reach their corresponding destination. Constraints (6) limit the number of edges to be selected to $n-1$, constraints (5) restrict the flow being pushed through the edges that do not belong to the selected spanning tree, and constraints (7) and (8) define the nonegative nature of the variables. Notice that the balance flow constraints guarantee that a path exists from vertex k to every other vertex $i \in V \setminus \{k\}$ and the requirement of having exactly $n-1$ edges selected guarantees that $\mathbf{y}$ indeed represents a spanning tree of G. Therefore, since the objective function (3) minimizes the sum of the weights of the selected spanning tree edges, the resulting formulation yields a minimum spanning tree in G.

Remark 1 *Formulation (4)–(8) is known to be integral, which implies that the* $\mathbf{y}$ *variables will always take the value of either 0 or 1 without the need of enforcing integrality constraints (Magnanti and Wolsey 1995). Moreover, the set of extreme points of this formulation corresponds to solutions* $(\mathbf{y}, \mathbf{f})$ *where* $\mathbf{y}$ *is the incidence vector of a spanning tree* $T \in \Omega_G$ *and* $\mathbf{f}$ *is a set of flows over* T *satisfying balance flow constraints (4).*

Given an interdiction strategy $S \subseteq E$ for the attacker, let $\mathbf{x} = [x_e]_{e \in E}$ be the incidence vector of edge set S. Since the defender edge selection is also limited by the attacker's strategy, the formulation must then be adapted to ensure the defender does not select any of the edges in S. A common practice (Cormican et al. 1998) to model this requirement is to alter the objective function (3) by adding a sufficiently large weight M_e to every edge $e \in E$ that is interdicted by the attacker. Then, the resulting minimum spanning tree formulation for the defender becomes:

$$\begin{aligned} \min \quad & \sum_{e \in E} (w_e + M_e x_e) y_e && (9) \\ \text{s.t} \quad & (6)\text{–}(8), \end{aligned}$$

whose corresponding dual is:

$$\begin{aligned} \max \quad & \sum_{i \in V} \sum_{l \in V \setminus \{k\}} b_i^l \alpha_i^l + (n-1)\gamma && (10) \\ \text{s.t.} \quad & \sum_{l \in V \setminus \{k\}} \beta_e^l + \gamma \leq w_e + M_e x_e \quad \forall e \in E && (11) \\ & \alpha_i^l - \alpha_j^l - \beta_e^l \leq 0 \qquad \forall e \in E,\, l \in V \setminus \{k\} && (12) \\ & \alpha_j^l - \alpha_i^l - \beta_e^l \leq 0 \qquad \forall e \in E,\, l \in V \setminus \{k\} && (13) \\ & \beta_e^l \geq 0 \qquad \forall e \in E,\, l \in V \setminus \{k\}. && (14) \end{aligned}$$

Here, α_i^l, β_e^l, and γ represent the dual variables associated with constraints (4), (5), and (6), respectively, and (11), (12), and (13) are the dual constraints associated with variables y_e, f_{ij}^l and f_{ji}^l, respectively, for all $e \in E$, $l \in V \setminus \{k\}$. It should be noted that the feasible space of this dual formulation is not empty, as the solution where all the dual variables are set to zero is feasible. Furthermore, since by the weak duality theorem any feasible solution to (11)–(14) yields a lower bound on the objective function of the primal problem (i.e., the weight of a minimum spanning tree in the residual graph given $\mathbf{x}$), by enforcing $\sum_{i \in V} \sum_{l \in V \setminus \{k\}} b_i^l \alpha_i^l + (n-1)\gamma \geq r$, one can then replace the defender's problem in (2) with a constraint set derived from its dual counterpart obtaining the single-level reformulation:

$$\begin{aligned} \text{(EXT):} \quad \min \quad & \sum_{e \in E} c_e x_e && (15) \\ \text{s.t.} \quad & (11)\text{–}(14) \\ & \sum_{i \in V} \sum_{l \in V \setminus \{k\}} b_i^l \alpha_i^l + (n-1)\gamma \geq r && (16) \\ & x_e \in \{0, 1\} \qquad \forall e \in E, && (17) \end{aligned}$$

which has an $O(nm)$ number of variables and constraints. Since this formulation is defined over a higher dimensional space given by dual variables α, β, and γ, we will also refer to this formulation as the *mixed-integer extended formulation* (EXT).

Importantly, a careful choice of the weights M_e, for all $e \in E$ has a strong impact on the quality of this formulation. Ideally, one should aim for the smallest possible value for such weights to obtain a tight representation. Since the interdiction criteria requires an optimal solution of (9) to be at least r, it is possible to identify a set of lower bounds on the values of M_e, for $e \in E$ that guarantee the validity of the resulting formulation.

Proposition 1 *Setting the values of M_e, for all $e \in E$ so that*

$$M_e \geq \max \left\{ 0, \max_{T \in \Omega_G[e]} \left\{ r - \sum_{e' \in E(T)} w_{e'} \right\} \right\}, \forall e \in E, \tag{18}$$

results in a valid EXT formulation for the minimum spanning tree interdiction problem.

Finding a tight set of values for M_e for $e \in E$ according to Proposition 1 can then be achieved in $O(n^2 \log n)$ time by repeatedly applying Prim's algorithm (Ahuja et al. 1993), each time fixing edge $e \in E$ to be part of the selected spanning tree.

3.2 Critical Spanning Trees Formulation

We begin this section by characterizing the set of feasible solutions of the spanning tree interdiction problem and then describe an integer programming formulation that results directly from it.

Definition 1 (Critical Spanning Tree) *Given a rational value r, so that $\underline{w}_G < r < \overline{w}_G$, a spanning tree T of graph G is said to be* critical *if $w(T) < r$. We use Ω^r_G to denote the set of all critical spanning trees of graph G.*

The set of critical spanning trees given the interdiction threshold r is naturally the collection of targets the attacker must focus the interdiction efforts on. This observation yields the following direct characterization of the feasible solutions to the minimum spanning tree interdiction problem, whose proof is omitted as it follows directly from the definition.

Theorem 2 *A given edge set $S \subseteq E$ is a feasible strategy for the attacker if and only if $S \cap T \neq \emptyset$, for all $T \in \Omega^r_G$.*

This characterization yields the following set-covering type of model for which $\mathbf{x} = [x_e]_{e \in E}$ is the incidence vector of edge set S. In this paper we refer to this model as the *critical spanning tree formulation* (CST), and consequently will refer to the constraints in set (20) as *critical spanning tree constraints.*

$$\text{(CST):} \quad \min \sum_{e \in E} c_e x_e \tag{19}$$

$$\text{s.t} \quad \sum_{e \in E(T)} x_e \geq 1, \quad \forall T \in \Omega^r_G \tag{20}$$

$$x_e \in \{0, 1\}, \quad \forall e \in E. \tag{21}$$

As stated in Theorem 2, constraint set (20) ensures that each critical spanning tree in the graph is effectively interdicted by the attacker; otherwise, the defender can then freely select any of such critical spanning trees left intact therein.

Set covering formulations, like (19)–(21), that are designed to restrict the existence of some type of graph structures from a given solution space appear to be quite common in the literature. See for instance the work by Chvátal and Cook (1990) who use a similar set covering formulation to block a specific type of edge structures called whips to raise the domination number of a graph; the set covering formulation by Ben-Ameur et al. (2015) that blocks connected components with more than k vertices to solve the k-separator problem; or the cut-set inequalities that are often used to break subtours when solving traveling salesman problems (Nemhauser and Wolsey 1988).

For the particular case of graph interdiction, the critical spanning tree constrains (20) are analogous to the *supervalid* type-I inequalities first introduced in the seminal paper by Israeli and Wood (2002) for interdicting shortest paths. One important difference however is that for the particular type of graph interdiction problem that we solve here (i.e., we minimize the cost of achieving a given disruption level over the defender's objective), constraints (20) are in fact valid as they naturally describe the attacker's feasible space.

3.3 Comparing the Strength of the Linear Relaxations

A common approach to analyze the quality of an integer formulation is by studying the strength of its linear relaxation (Vielma 2015). In this section we compare the linear relaxations of the extended formulation (EXT) and the critical spanning tree formulation (CST). Performing a direct comparison between these formulations raises some challenges because the extended formulation is defined over a higher dimensional space due to the additional variables that are introduced by the dual of the inner spanning tree model. Interestingly, we will show that it is possible to obtain a full description of the projection of the extended formulation into the space of variables $\mathbf{x}$. We will then use such a projection as the base for our comparison.

Theorem 3 *Let $\mathcal{P}_{ext} = \{(\mathbf{x}, \alpha, \beta, \gamma) \in [0,1]^m \times \mathbb{R}^{n^2} \times \mathbb{R}^{nm} \times \mathbb{R},$ satisfying (11)–(14),(16)$\}$ be the polytope described by the linear relaxation of the extended formulation, with a proper value for $M_e, \forall e \in E$ satisfying (18). Then, the projection $\mathrm{proj}_{\mathbf{x}}(\mathcal{P}_{ext})$ onto the $\mathbf{x}$-space corresponds to the polyhedron described by the following inequalities:*

$$\sum_{e \in E(T)} M_e x_e \geq r - \sum_{e \in E(T)} w_e, \quad \forall T \in \Omega_G \tag{22}$$

$$0 \leq x_e \leq 1, \quad \forall e \in E. \tag{23}$$

An interesting observation derived from Theorem 3 is summarized by the following corollary.

Corollary 1 *Let $\mathcal{P}_{cst} = \{\mathbf{x} \in [0,1]^m,\ \mathbf{x}$ satisfying (20)$\}$ be the polytope described by the linear relaxation of the critical spanning tree formulation, then*

1. *$\mathcal{P}_{cst} \subseteq \mathrm{proj}_{\mathbf{x}}(\mathcal{P}_{ext})$.*
2. *$\mathcal{P}_{cst} = \mathrm{proj}_{\mathbf{x}}(\mathcal{P}_{ext})$ if and only if all critical spanning trees have the same weight.*

In conclusion, the critical spanning tree formulation always yields a tighter relaxation for the non-trivial instances of the problem.

3.4 Separating the Critical Spanning Tree Constraints

The size of the critical spanning tree in formulation (19)–(21) is directly given by the critical spanning trees in Ω_G^r. Depending on the graph's density and the weight vector $\mathbf{w}$, it is possible for this set to grow exponentially large. Thus, to solve this formulation, some critical spanning tree constraints must be generated sequentially when needed. In this section, we discuss the complexity of separating such constraints. We observe that separation is polynomial-time solvable for integer values of $\mathbf{x}$, but show that separation is at least weakly NP-hard for the general case.

The separation problem for the critical spanning tree formulation can be stated as follows.

Critical spanning tree constraint separation

Instance: A graph $G = (V, E)$, a non-negative integer vector of rational edge weights $\mathbf{w}$, a vector $\mathbf{x} \in [0,1]^m$, and a rational number r.

Question: Is there a spanning tree $T \in \Omega_G^r$ such that $\sum_{e \in T} x_e < 1$?

Theorem 4 *Let $\mathbf{x}$ be a candidate solution for the attacker. Then, separating the critical spanning tree constraints is:*

1. *NP-hard if $\mathbf{x} \in [0,1]^m$*
2. *solvable in $O(m \log n)$ if $\mathbf{x} \in \{0,1\}^m$*

Interestingly, the NP-hardness of the separation algorithm also implies the following corollary, whose proof follows directly from Theorem 1 in Carr and Lancia (2002) because the critical spanning tree formulation does not admit a compact separation.

Corollary 2 *Unless P=NP, there is no polynomial-time constructible extended formulation for the polytope $\mathcal{P}_{cst}$ described by the critical spanning tree constraints.*

In Section 4, we will show that most of the critical spanning trees constraints induce facets for the minimum spanning tree interdiction polytope. This result and the one of Corollary 2 provide some additional evidence of the strength of the proposed formulation.

In practice, given that the size of set Ω_G^r may grow exponentially with respect to the size of the graph, we solve formulation (19)–(21) via branch and cut, generating critical spanning tree constraints as lazy constraints (i.e., we call the separation subroutine whenever there is an integer candidate solution for the attacker). We notice however that the constrained minimum spanning tree problem admits a polynomial-time approximation scheme (PTAS) (Ravi and Goemans 1996) that can be adapted to potentially separate fractional solutions as well. In our computational experiments, we limit our implementation and exclusively separate integer solutions.

Importantly, each round of the separation subroutine may yield several violated critical spanning tree constraints. This is particularly crucial at early stages of the branch and bound scheme, when the algorithm has yet to add enough constraints to yield a good lower bound. In fact, during the separation stage one may identify a set of k minimum spanning trees, for any given k (Katoh et al. 1981) and add the constraints associated with spanning trees among those that are deemed critical.

A typical implementation can benefit from a simpler greedy approach to potentially find several critical spanning trees whose corresponding inequality is violated. Given a candidate integer attacker solution $\mathbf{x}$, for which S is the associated set of edges being targeted, on can first find the minimum spanning tree T left in $G(V, E \setminus S)$. Second, among the edges not in T or S, find the edge e of minimum weight and add it to T creating a cycle. Third, find the edge e' of maximum weight in the cycle such that if e' is replaced by e in T the resulting new spanning tree is also critical. This process can be repeated until no further critical spanning trees can be found.

4 Polyhedral Analysis

In this section, we study the convex hull of the incidence vectors of all edge subsets $S \subseteq E$ that yield feasible solutions for the attacker and discuss some important properties of such polytope.

4.1 Basic Properties of the Convex Hull

Let $\mathcal{P}$ denote the convex hull of the incidence vectors of all feasible solutions, for any given $r > 0$. We begin our discussion by proving that $\mathcal{P}$ is a full-dimensional polytope and that the upper-bound constraint on each decision variable ($x_e \leq 1$, for each edge $e \in E$) induces a trivial facet of $\mathcal{P}$. We then identify necessary and sufficient conditions under which the non-negativity constraints ($x_e \geq 0$, for each edge $e \in E$) and the critical spanning tree constraints are facet inducing for $\mathcal{P}$. These results are presented in the following theorem.

Theorem 5 *Given a simple, nonempty, connected graph $G = (V, E)$, a weight vector $\mathbf{w} \in \mathbb{Q}^m$ so that there are no irrelevant edges in E (i.e., $\min\{w(T) | T \in \Omega_G[e]\} < r, \forall e \in E$), and a scalar $r \in \mathbb{Q}$, so that $\underline{w}_G < r < \overline{w}_G$, the following statements are true.*

(1) $\mathcal{P}$ is a full-dimensional polytope.

(2) Given an edge $e \in E$, inequality $x_e \leq 1$ induces a facet of $\mathcal{P}$.

(3) Given an edge $e \in E$, inequality $x_e \geq 0$ induces a facet of $\mathcal{P}$ if and only if $n \geq 4$.

(4) Given a critical spanning tree $T \in \Omega^r_G$, the corresponding critical spanning tree inequality $\sum_{e \in E(T)} x_e \geq 1$ induces a facet of $\mathcal{P}$ if and only if:

(a) there is no edge $e \in E$ such that if added to T, it completes a Hamiltonian cycle in G, or

(b) if such an edge exists, at least one of the spanning trees contained in the resulting Hamiltonian cycle has a weight greater than r.

4.2 Other Sets of Facet-Defining Inequalities

In this section, we study a broad family of facet defining inequalities of $\mathcal{P}$ that could be used to strengthen formulation (19)–(21). We analyze the complexity of the corresponding separation algorithm and provide a brief discussion about how to use some of these inequalities in practice.

Definition 2 (Cactus) *A* cactus *is defined as a connected graph in which every edge belongs to at most one cycle. For a given graph $G = (V, E)$, a cactus H that contains all the vertices in V is called a* spanning cactus *of G. The* rank *of a given cactus H is defined as the number of cycles contained in H; we will refer to a cactus of rank k, simply as a k-cactus.*

In what follows, we deal exclusively with spanning cacti of a graph G; thus, we will drop the spanning categorization when referring to those, unless further clarifications are required.

Given a k-cactus H of graph G, let $E(H)$ be the set of all edges in H; clearly, $|E(H)| = n + k - 1$. Furthermore, for any order given to the k cycles of H, let $C_l(H), \forall l = 1, \ldots, k$ be the set of edges comprising the l^{th} cycle of H, $\mathcal{C}(H) = \bigcup_{l=1}^k C_l(H)$, and $F(H) = E(H) \setminus \mathcal{C}(H)$ be the set (possibly empty) composed of the edges that do not belong to any cycle in H. Notice that removing one edge from cycle $C_l(H)$, for any $l = 1, \ldots, k$, reduces H into a $(k-1)$-cactus; removing any two edges from $C_l(H)$ disconnects H; as well as removing any edge from set $F(H)$. It is easy to see that H is in fact the union of $\prod_{l=1}^k |C_l(H)|$ spanning trees of G, each of those created by removing one of the edges from each subset $C_l(H), \forall l = 1, \ldots, k$.

Figure 1 provides an example of a 3-cactus H with 18 vertices and 20 edges. In this example, there are three cycles, $C_1(H) = \{e_3, e_6, e_7\}$, $C_2(H) = \{e_8, e_9, e_{10}, e_{11}\}$, and $C_3(H) = \{e_{15}, e_{16}, e_{17}, e_{18}, e_{19}, e_{20}\}$, hence, the cactus is of rank 3. The set $\mathcal{C}(H) = \{e_3, e_6, e_7, e_8, e_9, e_{10}, e_{11}, e_{15}, e_{16}, e_{17}, e_{18}, e_{19}, e_{20}\}$ is simply the union of cycles $C_1(H)$, $C_2(H)$ and $C_3(H)$, and set $F(H) = \{e_1, e_2, e_4, e_5, e_{12}, e_{13}, e_{14}\}$ is its complement. It is also easy to verify that H contains a total of $\prod_{l=1}^k |C_l(H)| = 3 \times 4 \times 6 = 72$ spanning trees.

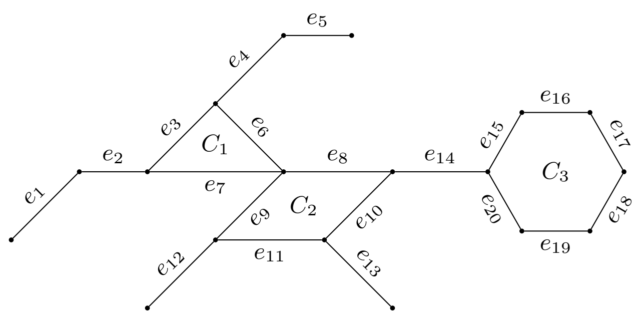

Figure 1: An example of a 3-cactus.

Consider a k-cactus H of a given graph $G = (V, E)$. For each edge set $C_l(H)$, with $l = 1, \ldots, k$, let $\hat{C}_l(H) \subseteq C_l(H)$ be a set that includes each edge $e \in C_l(H)$ for which $\Omega^r_{G(V, E(H) \setminus \{e\})}$ is not empty. That is, edge $e \in C_l(H)$ belongs to $\hat{C}_l(H)$ if, after removing edge e from H, at least one critical spanning tree T remains in the resulting $(k-1)$-cactus. We also denote $\hat{\mathcal{C}}(H) = \bigcup_{l=1}^k \hat{C}_l(H)$ to be the set of all these edges.

Theorem 6 *Given a simple, nonempty, connected graph $G = (V, E)$, a weight vector $\mathbf{w} \in \mathbb{Q}^m$, a scalar $r \in \mathbb{Q}$ so that $\underline{w}_G < r < \overline{w}_G$, and a k-cactus H of G, with $k \geq 1$, the inequality*

$$\sum_{e \in \hat{\mathcal{C}}(H)} x_e + 2 \sum_{e \in \mathcal{C}(H) \setminus \hat{\mathcal{C}}(H)} x_e + 2 \sum_{e \in F(H)} x_e \geq 2 \tag{24}$$

induces a facet of $\mathcal{P}$, if and only if $|\hat{C}_l(H)| \geq 3, \forall l = 1, \dots, k$.

In what follows, we will refer to these cuts as *cactus inequalities.*

Cactus graphs are essentially a generalization of other well-known structures. Indeed, notice that a tree is a cactus of rank 0, and a 1-tree (i.e., a connected graph containing exactly one cycle) (Held and Karp 1970), also called a pseudotree, is a cactus of rank 1. In our computational experiments we give special attention to the cactus inequalities associated with 1-trees, for those are easier to separate compared to inequalities for cacti of higher ranks.

We now discuss the separation problem for the cactus inequalities, which is formally stated below. Theorem 4 provided a proof that separating the critical spanning tree constraints is NP-hard. Since a tree is a cactus of rank 0, separating cactus inequalities is in general NP-hard. The proof of the following theorem (see Appendix A in the Online Supplement) goes a step further and presents a simple construction that is used to show that separating inequalities associated with 1-cacti is NP-hard too. It is easy to see that such a construction can be extended for cactus inequalities of any given rank k.

cactus inequality separation

Instance: A graph $G = (V, E)$, a non-negative integer vector of rational edge weights $\mathbf{w}$, a vector $\mathbf{x} \in [0, 1]^m$, and a rational number r.

Question: Is there a k-cactus H of G with $k \geq 1$ and $\hat{C}_l(H) \geq 3, \forall l = 1, \dots, k$ for which inequality (24) is violated?

Theorem 7 *Let $\mathbf{x} \in [0, 1]^m$ be a candidate solution for the attacker. Then, the separation problem for the cactus inequalities is NP-hard.*

Despite the fact that the separation algorithm for the cactus inequalities is NP-hard, in our implementation we tested the following greedy integer separation approach tailored for cacti of rank one (i.e., 1-trees). Given a candidate integer attacker solution $\mathbf{x}$, for which S is the associated set of edges being targeted, we first find a minimum spanning tree T left in $G(V, E \setminus S)$. Among the edges not in T or S, we find the edge e of minimum weight and add it to T creating a 1-tree H. Then, for each edge $e' \in C_1(H)$, we add e to $\hat{C}_1(H)$ if $w(T) + w_e + w_{e'} < r$. If $|\hat{C}_1(H)| \geq 3$, we add the resulting cactus inequality.

As will be described in Section 6, we observed in our experiments that separating 1-tree inequalities too often as described above can be relatively slow, potentially hindering the efficacy of the algorithm, particularly when applied over dense graphs. Thus, our implementation was adapted so that the search for this type of inequalities is only conducted with a predefined probability.

5 Critical Edge Sets Formulation

In this section, we exploit some fundamental properties of the the problem's feasible solutions to produce a stronger variation of the critical spanning tree formulation. This alternative formulation is defined by a new type of structures referred to as critical edge sets, which can be used to produce a set of so-called supervalid inequalities (cf. Israeli and Wood (2002)) that exclude feasible solutions for the attacker, albeit solutions whose objective value is not better than the minimum cost edge cut $\lambda_{\mathbf{c}}(G)$. We show that the linear relaxation of this formulation is not only contained by the linear relaxation of (20)–(21), but also requires fewer constraints to be defined.

We will discuss several properties of this formulation including details about the complexity of the separation algorithm for its constraints. Interestingly, despite requiring an additional computational effort, the separation algorithm can take advantage of an efficient enumeration scheme that exploits some hereditary properties of the critical edge sets. Based on the results of our computational experiments (see Section 6), this approach is the one that produced the best results.

It is important to mention that the supervalid inequalities that we will describe in this section are by design different to the ones firstly introduced in Israeli and Wood (2002). As mentioned in Section 3, the inequalities developed in their paper (particularly the inequalities of type I) are analogous to the critical spanning tree constraints we discuss here, albeit defined over a set of *st*-paths instead of spanning trees. As we will explain in the following section, the set of supervalid inequalities that we propose come from the fact that there exists a natural bipartition of the feasible solutions for the attacker into solutions that are edge cuts and solutions that are not. Our inequalities are somehow aimed to restrict the search for optimal solutions over the latter set.

Definition 3 (Critical edge set) *Given an integer value r, so that $\underline{w}_G < r < \overline{w}_G$, a subset of edges $K \subseteq E$ is called a* critical edge set *if $\Omega_G[K] \subseteq \Omega^r_G$ and $\Omega_G[K] \neq \emptyset$. We use Λ^r_G and $\tilde{\Lambda}^r_G$ to denote the collection of all critical edge sets and the set of all* minimal *critical edge sets under inclusion, respectively.*

Clearly, a critical edge set $K \in \Lambda^r_G$ contains no cycles because otherwise no spanning tree would contain all the edges in K. Also, the edge set $E(T)$ of any critical spanning tree $T \in \Omega^r_G$ is a critical edge set too, for $\Omega_G[E(T)] = \{T\}$. In fact, it is a maximal critical edge sets under inclusion because any proper superset of $E(T)$ contains cycles. Moreover, notice that for all $T \in \Omega^r_G$, there is a $K \in \tilde{\Lambda}^r_G$ such that $K \subseteq E(T)$, as either set $E(T)$ or a subset of it belongs to $\tilde{\Lambda}^r_G$.

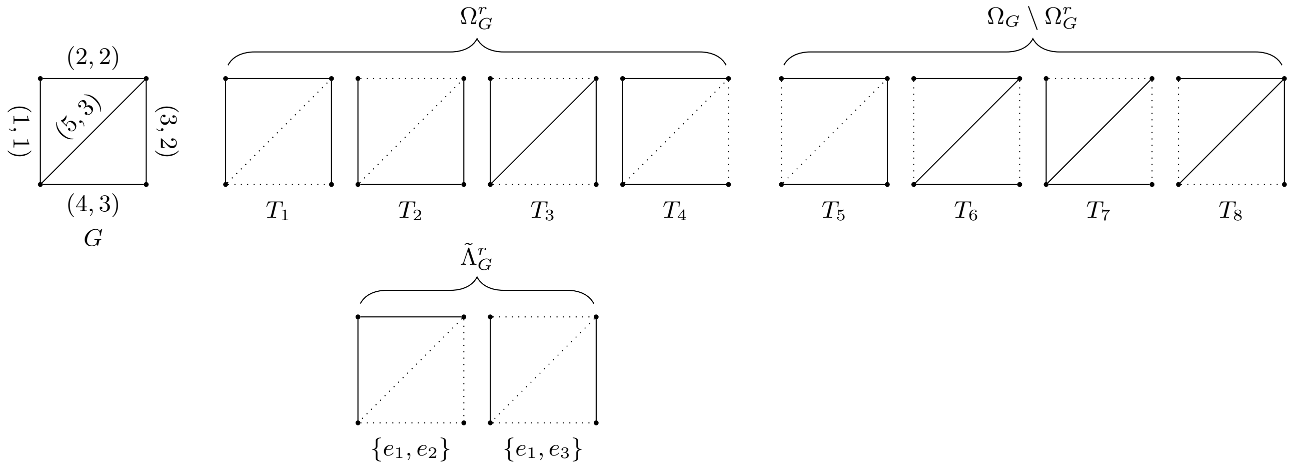

Figure 2: An example of critical spanning trees and critical edge sets.

For illustrative purposes, consider the example presented in Figure 2, where we depict a graph G composed of $n = 4$ vertices and $m = 5$ edges. Next to each edge, we provide the pair (e, w_e) that indicates the index and weight of the corresponding edge, respectively. We assume for this example that $r = 7$. The graph contains eight spanning trees, $\Omega_G = \{T_i\}_{i=1}^{8}$, among which four are critical, i.e., $\Omega^r_G = \{T_1, T_2, T_3, T_4\}$. The set of critical edge sets is $\Lambda^r_G = \{E(T_1), E(T_2), E(T_3), E(T_4), \{e_1, e_2\}, \{e_1, e_3\}\}$, and the set of minimal critical edge sets is then $\tilde{\Lambda}^r_G = \{\{e_1, e_2\}, \{e_1, e_3\}\}$. Notice for instance that edge set $\{e_3\}$ is not critical because $\Omega_G[e_3] = \{T_1, T_2, T_3, T_5, T_8\}$ and both T_5 and T_8 are not critical. Similarly, edge set $\{e_3, e_4\}$ is not critical because $\Omega_G[\{e_3, e_4\}] = \{T_2, T_5\}$, but T_5 is not critical. In contrast, edge set $\{e_1, e_2\}$ is critical because all trees in $\Omega_G[\{e_1, e_2\}] = \{T_1, T_4\}$ are critical.

The following theorem provides an interesting characterization of the set of feasible solutions of the interdiction problem with respect the the critical edge sets.

Theorem 8 *An edge set $S \subseteq E$ is a feasible solution for the attacker if and only if S is an edge cut or $S \cap K \neq \emptyset$, for all $K \in \tilde{\Lambda}^r_G$.*

Interestingly, Theorem 8 defines a partition of the feasible strategies for the attacker into two sets: the solutions that are edge cuts and the ones that are not. As mentioned in Section 2, since the objective value of any edge cut is bounded below by $\lambda_{\mathbf{c}}(G)$, to find the optimal solution of the interdiction problem, one may find first a minimum cost edge cut (which can be done in polynomial time), and then focus on searching for the best non-edge-cut feasible strategy. Based on this observation, we use the result from Theorem 8 to define the following formulation, named the *critical edge set formulation* (CES). We will show that we can use this integer program to find a minimum cost non-edge-cut solution for the attacker.

$$\text{(CES):} \quad \min \sum_{e \in E} c_e x_e \tag{25}$$

$$\text{s.t} \quad \sum_{e \in K} x_e \geq 1, \quad \forall K \in \tilde{\Lambda}^r_G \tag{26}$$

$$x_e \in \{0,1\}, \quad \forall e \in E. \tag{27}$$

Importantly, the critical edge set formulation is not valid for the original problem because its solution space may omit some feasible solutions that are edge cuts. Nevertheless, an optimal solution of the minimum spanning tree interdiction problem, based on Theorem 8, corresponds to either an optimal solution of (25)–(27) or an edge cut of minimum cost. For this reason, if while solving the critical edge set formulation, one finds that a lower bound for the formulation is greater than or equal to $\lambda_{\mathbf{c}}(G)$, then one can stop the search and use any minimum cost edge cut as the optimal strategy for the attacker.

Consider the example shown in Figure 3, which presents a side-by-side comparison of the critical spanning tree and the critical edge set formulations for the instance depicted in Figure 2. Notice that solution $S = \{e_3, e_4\}$ is a feasible solution for the attacker, as it is an edge cut, but it is not feasible for the critical edge set formulation.

CST Formulation:

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e x_e \\ \text{s.t.} \quad & x_1 + x_2 + x_3 \geq 1 \\ & x_1 + x_3 + x_4 \geq 1 \\ & x_1 + x_3 + x_5 \geq 1 \\ & x_1 + x_2 + x_4 \geq 1 \\ & x_e \in \{0,1\}, \ \forall e \in E \end{aligned}$$

CES Formulation:

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e x_e \\ \text{s.t.} \quad & x_1 + x_3 \geq 1 \\ & x_1 + x_2 \geq 1 \\ & x_e \in \{0,1\}, \quad \forall e \in E \end{aligned}$$

Figure 3: A comparison of the critical spanning tree (CST) and the critical edge set (CES) formulations for the example depicted in Figure 2.

Remark 2 *The quality of the critical edge set formulation strongly depends on whether there exist irrelevant edges in graph G. We recall from Section 2.1 that an edge $e \in E$ is called irrelevant if the weight of all the spanning trees in $\Omega_G[e]$ (i.e., all the spanning trees that contain e) is greater than r. Having irrelevant edges in the graph may prevent some edge sets to be critical, which will in turn affect the overall strength of the critical edge set formulation. Therefore, one should guarantee that all the irrelevant edges are removed before solving the interdiction problem.*

For an illustration, consider again the example depicted in Figure 2. Suppose that we create a new graph G' by adding the second diagonal edge e_6 to G (i.e., the edge that is left for the graph to be complete) so that $w_6 = 7$. Since the weight of e_6 is equal to r, no spanning tree that contains this edge is critical; therefore, e_6 is irrelevant. Clearly, solving the minimum spanning tree interdiction problem in both G and G' yields the same solution. However, notice that for graph G', the edge sets $\{e_1.e_2\}$ and $\{e_1, e_3\}$ are no longer critical because the spanning trees induced by the edges $\{e_1.e_2, e_6\}$ and $\{e_1, e_3, e_6\}$ are not critical, which implies that $\tilde{\Lambda}^r_{G'} = \Omega^r_{G'}$. In conclusion, the critical edge set formulation gets weakened in the presence of e_6.

5.1 A Characterization of the Solution Space of the Critical Edge Set Formulation

The result given by Theorem 8 guarantees that solving the CES formulation coupled with a global minimum cost edge-cut incumbent solution is a valid approach for tackling the proposed minimum spanning tree interdiction problem. However, this theorem provides no further information about the properties of this new formulation and its relation to the CST formulation. In what follows, we will study these by analyzing the solution space of the CES formulation and provide an analytical description of why this formulation is in general stronger. We consider this analysis one of the key results of this paper.

First, the following theorem provides an exact characterization of this solution space.

Theorem 9 *An edge set $S \subseteq E$ is a feasible solution for the critical edge set formulation if and only if S is a superset (not necessarily proper) of any non-edge-cut feasible strategy for the attacker.*

This theorem implies that the CES formulation is tight with regards to excluding edge-cut solutions. That is, most of the edge-cut solutions are removed by the CES constraints, and those remaining in the solution space are all dominated by some non-edge-cut solutions. In essence, the CES formulation takes advantage of the global min-edge-cut incumbent, and focus the solution search procedure in a smaller space that consists of only non-edge-cut strategies. A direct consequence of this is that the solution space of CES formulation is contained within the solution space of CST formulation. Interestingly, this also offers another perspective about the origin of critical edge sets: they are induced by the bipartition of the solution space resulting from the edge-cut property.

Next, the following theorem establishes a relation between the minimal critical edge sets and the critical spanning trees.

Theorem 10 *For any minimal critical edge set $K \in \tilde{\Lambda}^r_G$, there exists a spanning tree $T \in \Omega_G[K]$ that does not contain any other minimal critical edge set $K' \in \tilde{\Lambda}^r$, where $K \neq K'$.*

This relation allows us to build an injective function from the set of minimal critical edge sets $\tilde{\Lambda}^r_G$ to the set of critical spanning trees Ω^r_G. Hence, we have the following corollary.

Corollary 3 $|\tilde{\Lambda}^r_G| \leq |\Omega^r_G|$.

Notice that $\tilde{\Lambda}^r_G$ and Ω^r_G are the sets of constraints for the CES and CST formulations, respectively. Therefore, the solution space of CES formulation is not just contained within the CST formulation, it also requires fewer inequalities to be described. This is quite interesting—and perhaps counter-intuitive—because, in general, formulations that are stronger often require more variables/constraints to be represented. Moreover, in practice, the number of minimal critical edge sets needed in $\tilde{\Lambda}^r_G$ for solving a problem can be much smaller than $|\Omega^r_G|$, since usually a small-sized minimal critical edge set can be contained by many critical spanning trees. This is of great importance for practical purposes, for it implies that the CES formulation does not just produce stronger inequalities, but also requires fewer of them to solve the problem.

5.2 Separating the Critical Edge Set Constraints

It should be clear that separating a critical edge set constraint from a fractional solution is NP-Hard, for a separation in polynomial time implies that the critical spanning tree constraints can also be separated in polynomial time, which contradicts Theorem 4. For an integer solution, we will show that the separation complexity is $O(\alpha(n, m)n \log n)$.

To design an efficient algorithm to separate the critical edge set constraints, we first consider the partial ordered set (poset) defined by $(\downarrow \Omega_G^r, \subseteq)$, where $\downarrow \Omega_G^r$ represents the lower closure of Ω_G^r, i.e., $\downarrow \Omega_G^r = \{K \subseteq E(T) \mid T \in \Omega_G^r\}$, and $\subseteq$ indicates that the poset is ordered by inclusion. Since any subset of elements of this poset has a greatest lower bound (i.e., a *meet*) the poset is also a meet-semilattice, which means that the poset is closed under intersection. We will show that this meet-semilattice structure yields a natural enumeration scheme for all the critical edge sets. The following theorem provides some important properties of this meet-semilattice.

Theorem 11 *The following statements about the poset $(\downarrow \Omega_G^r, \subseteq)$ are true:*

1. *The least element in $(\downarrow \Omega_G^r, \subseteq)$ is the empty set.*
2. *The maximal elements in $(\downarrow \Omega_G^r, \subseteq)$ are all the critical spanning trees.*
3. *$(\Lambda_G^r, \subseteq)$ is a subposet embedded in $(\downarrow \Omega_G^r, \subseteq)$.*

In this meet-semilattice, we call the empty set the root, each maximal element a leaf, and each maximal chain that emanates from $\emptyset$ and reaches a critical spanning tree a branch. For example, the meet-semilattice of $(\downarrow \Omega_G^r, \subseteq)$ for the graph G of Figure 2 is illustrated in Figure 4. Notice, that the bold nodes and arcs compose the subposet $(\Lambda_G^r,\subseteq)$ that is embedded in this meet-semilattice.

$$\begin{tikzcd}
 & & & \emptyset \arrow[dll] \arrow[dl] \arrow[d] \arrow[dr] \arrow[drr] & & & \\
 & \{e_1\} \arrow[dl] \arrow[d] \arrow[dr] \arrow[drr] & \{e_2\} \arrow[dll] \arrow[drr] & \{e_3\} \arrow[dll] \arrow[dr] \arrow[drr] \arrow[drrr] & \{e_4\} \arrow[dll] \arrow[dr] & \{e_5\} \arrow[dll] \arrow[dr] & \\
\{\mathbf{e_1}, \mathbf{e_2}\} \arrow[drr, line width=1.5pt] & \{\mathbf{e_1}, \mathbf{e_3}\} \arrow[dr, line width=1.5pt] \arrow[drr, line width=1.5pt] \arrow[drrr, line width=1.5pt] & \{e_1, e_4\} \arrow[dr] & \{e_1, e_5\} \arrow[dr] & \{e_2, e_3\} \arrow[dll] & \{e_3, e_4\} \arrow[dll] & \{e_3, e_5\} \arrow[dll] \\
 & & \{\mathbf{e_1}, \mathbf{e_2}, \mathbf{e_3}\} & \{\mathbf{e_1}, \mathbf{e_3}, \mathbf{e_4}\} & \{\mathbf{e_1}, \mathbf{e_3}, \mathbf{e_5}\} & &
\end{tikzcd}$$

Figure 4: The meet-semilattice of $(\downarrow \Omega_G^r, \subseteq)$ for the example depicted in Figure 2.

Before the construction, the algorithm sorts all the edges so that at each level of the meet-semilattice, the weights of the nodes of the meet-semilattice are increasing from left to right. Clearly, the size of this meet-semilattice is very large even for small sized instances; therefore, in our implementation, we construct portions of it, when needed. At the beginning of the algorithm, we begin with the root. Once the formulation returns an integer candidate solution associated with edge set S, the algorithm creates the leftmost unexplored branch that does not intersect S. Since all the branches are sorted from left to right, this branch corresponds to one of a critical spanning tree of minimum weight that is not blocked by S. If S is not a feasible solution, then there exists a critical edge set on this branch, and the separation algorithm proceeds to search for the minimal set on this branch. Once such edge set is found, the corresponding critical edge set constraint is added to the constraint pool of the formulation.

At each node of the branch, with edge set K, the algorithm needs to compute a maximum spanning tree restricted on K to check whether it is a critical edge set or not. We use a variant of the Kruskal's algorithm, which is of complexity $O(m \log n)$. However, the dominant part of Kruskal's algorithm is the edge sorting; once the edges are sorted, a maximum spanning tree can be found with time complexity $O(n\alpha(n,m))$ using a sophisticated disjoint-set data structure (Tarjan and Van Leeuwen 1984, Tarjan 1979). Therefore, the verification complexity at each node is $O(n\alpha(n,m))$, since all the edges have been sorted in advance.

If we execute the aforementioned branching method naively, the overall complexity of separating a critical edge set is $O(\alpha(n,m)n^2)$. However, the following proposition ensures that a binary search based branching scheme can be applied.

Proposition 2 *Let β be the function from the ordered set $(\downarrow \Omega_G, \subseteq)$ to $(\mathbb{R}, \leq)$ defined as $\beta(K) = \max_{T \in \Omega_G[K]} w(T)$, then β is order reversing.*

Given a branch, this proposition helps the binary search to determine which half to proceed. Suppose the algorithm is at a node of an edge set K with a search scope from K_0 to K_1 in a given branch. Clearly, $K_0 \subseteq K \subseteq K_1$. If $\beta(K) < r$, then K is a critical edge set, so we update the search scope to the subchain between K_0 and K in order to find a smaller one; otherwise, K is not a critical edge set, so the new search scope is the subchain between K and K_1. With this branching method, the complexity of separating a critical edge set which is minimal on the branch is $O(\alpha(n,m)n \log n)$.

The last thing to notice is that the critical edge set, which is minimal on each given branch produced by the above method, is not necessarily an element in $\tilde{\Lambda}_G^r$. But, since all the minimal critical edge sets would be eventually enumerated from this semilattice, the correctness of our implementation still holds.

6 Computational Experiments

The computational experiments reported in this section were conducted on a computing cluster equipped with twelve nodes, each having a 12-core Intel Xeon E5-2620 v3 2.4GHz processor, 128 GB of RAM, and running Linux x86_64, CentOS 7.2. All the formulations and algorithms were implemented in C++, compiled with GCC 6.3.0, and solved using the commercial Optimizer Gurobi 8.0. Each instance was solved by a single node of the cluster under a time limit of 7,200 seconds. The data sets are provided in the online supplements.

We compare the performance of all the formulations studied in this paper: (1) the mixed-integer extended formulation derived from the bi-level representation of the problem (see Section 3.1), which we refer to as EXT; (2) the critical spanning tree formulation (see Section 3.2), which we call CST; (3) the critical spanning tree formulation strengthened with cactus inequalities of rank 1 (see Section 4.2), which we named CST-1; and (4) the critical edge set formulation (see Section 5), referred to as CES. The CST, CST-1, and CES formulations are all implemented using the proposed sequential cut-generation method, due to the large number of constraints. In our implementation, both CST and CST-1 are initialized with a minimum spanning tree constraint, and the CES is initialized with a constraint of critical edge set, which is contained in a minimum spanning tree. Notice that one may warm-start the CST and CST-1 formulations by introducing as many constraints associated with critical spanning trees as one may desire. For instance, one can solve an instance for the k-minimum spanning trees (Katoh et al. 1981) for any given k and add the constraints associated with spanning trees among those that are deemed critical.

We use three main criteria for our tests: the percentage of instances solved to optimality, the average runtime required to solve the problem, and the average optimality gap. Furthermore, we also compare the cut generation subroutines for the CST and CES formulations.

6.1 Test Instances

We generated a total of 155 connected graphs of different sizes and topologies using four types of random graph generator models: small-world (Watts and Strogatz 1998), Erdös-Renyi (Erdös and Rényi 1959), binomial

(Gilbert 1959), and Barabási-Albert (Albert and Barabási 2002). For the size of the graphs, we set $n = 40, 80, 160, 200$. Furthermore, we adjusted the parameters of each graph generator model using three different settings in order to produce three sets of graphs of similar densities for each value of n (except for $n = 40$, for which we only used two). In the following tables, we use the average number of edges of the graphs ($\bar{m}$) to represent the densities. For each combination of n, graph density, and graph generator model, we created three different graphs and use a uniform distribution from 20 to 80 to generate the weights ($\mathbf{w}$) and costs ($\mathbf{c}$) of the edges. The data sets associated with all the 155 graphs are available as online supplemental material.

In addition to the density of the graph, the complexity of the instances also depends on the values chosen for r, as this parameter dictates the cardinality of Ω^r_G. We solve all instances with four different levels for r. More specifically, we pick $r = (\overline{w}_G - \underline{w}_G) \times \gamma$, for $\gamma = \{0.05,\ 0.3,\ 0.7,\ 0.95\}$, where γ is referred to as the *interdiction requirement*.

Notice that since the weights of the graphs are randomly generated, varying the interdiction requirement as a percentage of the difference between the weights of the minimum and maximum spanning trees gives us the possibility of aggregating instances that are expected to be of similar difficulty, rather than by directly changing the values for r. In general, for a fixed value of γ, the value of r increases with respect to the number of vertices and the minimum degree of the graphs. Thus, larger instances tend to be more challenging for most formulations, not only because of their size, but also because of the interdiction requirement. For the different instances that we generated, we ensured that, even when taking the lowest value of $\gamma = 0.05$, the resulting values of r avoid trivial cases where the set of the critical spanning trees Ω^r_G is empty or simply too small, thus providing a good spectrum for the performance comparisons. In summary, each of the four formulations was tested over a set of 155 (graphs) $\times$ 4 (interdiction requirements) = 620 instances.

To present the computational results, we aggregate all the instances that are expected to be of similar complexity based on the size of the graph in terms of n and $\bar{m}$, as well as the interdiction requirement γ. We call such a triplet $(n, \bar{m}, \gamma)$ a graph configuration. All the tables provided in the following sections showcase the averages over all the instances of the same graph configuration for the different metrics. Interestingly, we observed no major difference in the difficulty of the different instances with respect to the type of graph generator model used. Thus, the averages shown in the tables are not disaggregated by such factor.

Whenever we present the average of the optimality gaps, we evaluated them as $(UB - LB)/LB \times 100$, where UB and LB stand for the best upper and lower bounds identified by the formulations, respectively. Furthermore, if a formulation was not able to find the optimal solution of an instance within the time limit, we set the corresponding running time as 7,200 seconds.

In addition to the randomly generated graphs, for the purpose of demonstrating the performance of our algorithms on real-life graphs, we also collected 12 instances from Network Repository (2019) comprising infrastructure, power, and road networks. The data sets of these instances are also available as online supplemental material. As mentioned before, because of their sparsity, this type of instances are significantly easier to solve; therefore, we only tested two of our formulations on these instances.

Finally, since any minimum-cost edge cut is a feasible solution for the attacker, we first identified one in polynomial time (Gomory and Hu 1961) and feed it to all the formulations as an initial incumbent solution. Thus, all the approaches started with the same upper bound. For the particular case of the CES formulation, we did not add it as an incumbent because such a solution is likely to be infeasible for that formulation (see Theorem 9); however, we used its objective value as a stopping criterion, as described in Section 5. Identifying a minimum-cost edge cut solution is significantly faster than solving the interdiction problem. Even for the largest and densest instances, it takes less than 0.1 seconds to compute. Since all the formulations use $\lambda_{\mathbf{c}}(G)$ as a valid upper bound, we report the time required to compute it separately from the overall running times.

6.2 Results for the CST and CST-1 Formulations

We begin our discussion by analyzing the effect that introducing cactus inequalities of rank 1 has on the efficiency of the critical spanning tree formulation. Among all the four formulations we compare in our experiments, the CST and the CST-1 are clearly the most similar ones, as the latter is simply the CST formulation strengthened with some extra facet-defining inequalities. Since the CST-1 formulation may have a stronger lower bound thanks to the extra cuts, one might expect it to obtain better results than the plain CST formulation.

However, we note that in practice the implementation of the separation algorithm for these cuts has a strong impact on the formulation's performance. In general, when applied too often, the overhead required to separate and add the cactus inequalities may overcome the benefit that they produce. Thus, instead of naively separating them at every branch-and-bound node, we generate them randomly with a probability of ρ. That is, at each time after separating a critical spanning tree constraint, there is a ρ chance that a cactus inequality is separated. Tuning this probability is crucial for the performance of the CST-1 algorithm, as adding too few inequalities would not make much of a difference, but adding too many would make the linear relaxation grow significantly, thus taking longer to solve. In our experiments, we observed a good performance when selecting $\rho = 1/3$. This method is commonly used in the literature to improve the performance of cutting plane algorithms (Fischetti et al. 2017).

Table 1 reports the results for these two formulations; we highlight in bold the best results obtained for each of the metrics. As expected, for most of the graph configurations, CST-1 outperforms CST in all the three criteria.

An interesting observation is that the total number of cuts added with both formulations is relatively similar, but the impact of adding the cactus inequalities on the overall performance is notable and often results in a much faster execution. See for example the cases for the graph configurations where $n = 80$ and $\bar{m} = 160$; $n = 160$ and $\bar{m} = 318$; $n = 200$ and $\bar{m} = 396$, for which the CST-1 performs much better.

Since the CST formulation was outperformed in most of the instances by the CST-1 formulation, we exclude it from all further comparisons.

6.3 EXT, CST-1, and CES Performance Comparison

In this section, we continue our discussion by analyzing the computational results of the three remaining formulations. Table 2 reports the averages of the three metrics we use for our analysis. As before, we highlight in bold the best results obtained for each graph configuration.

A glance at Table 2 shows that the CES formulation outperforms the other two in all three metrics. The percentage of instances solved by CES is close to 90%, which is almost 20% higher than that of the other two formulations. The average running time of CES is also significantly shorter—about 70 % faster than the times of the other two formulations. In fact, in several occasions, it takes only a few seconds for CES to solve several instances compared to the several minutes it takes EXT and CST-1. Also, the average gap is noticeable smaller, exceeding 15% in just a few cases, whereas gaps larger than 30% are common for the other two formulations.

A further analysis of this data helps us draw the following conclusions. First, in terms of the percentage of instances solved, EXT performance is similar to that of CST-1, as both have solved around 68% of the instances. However, with respect to the average running times and optimality gaps, EXT is the worst among all three formulations. The fact that EXT has a similar solving percentage as CST-1 can be directly explained by the fact that EXT performs better on small instances. In fact, the CST-1 formulation outperforms EXT in all occasions except for the cases where the graphs are small and both $\bar{m}$ and γ are large. This behavior can be partially explained by the fact that the CST-1 formulation yields a tighter linear relaxation than EXT (See Corollary 1) and also by the fact that the number of variables comprising EXT is $O(nm)$, which can become quite large for some of the big instances.

Second, the CES formulation drastically outperforms CST-1, except for the cases where γ is small. An interesting explanation for this difference in performance is that the CES formulation removes a large subset of fractional solutions that are feasible for the linear relaxation of the CST-1 formulation. This effect is exacerbated when γ is large, in which case the critical edge set constraints tends to become significantly stronger than the critical spanning tree constraints (see Corollary 3). This effect can be observed from for the different values of γ in Table 2. Notice that both the average running times and gaps decrease significantly for CES whenever γ increases. Conversely, when γ is very small, the number of critical spanning trees $|\Omega^r_G|$ tends to be also small, so CST-1 occasionally performs better because its separation subroutine is faster.

Table 1: Computational results for the CST and CST-1 formulations. For each graph configuration, we show the time required to identify a minimum-cost edge cut ($\lambda_{\mathbf{c}}(G)$), the percentage of instances solved to optimality (Opt), the average run time in seconds (time), the average optimality gap (Gap), the number of cuts generated (Num. cuts), and the total time in seconds spent separating inequalities (Sep. time).

Configuration				Opt (%)		Time		Gap (%)		Num. Cuts		Sep. Time	
n	$\bar{m}$	γ	$\lambda_{\mathbf{c}}(G)$	CST	CST-1	CST	CST-1	CST	CST-1	CST	CST-1	CST	CST-1
40	78	0.05	0.00	**100**	**100**	**0.02**	0.04	**0.0**	**0.0**	39	51	0.00	0.01
		0.30	0.00	**100**	**100**	0.30	**0.24**	**0.0**	**0.0**	616	549	0.07	0.06
		0.70	0.00	**100**	**100**	0.31	**0.24**	**0.0**	**0.0**	616	549	0.07	0.06
		0.95	0.00	**100**	**100**	0.32	**0.24**	**0.0**	**0.0**	616	549	0.07	0.06
	118	0.05	0.00	**100**	**100**	0.50	**0.26**	**0.0**	**0.0**	579	459	0.08	0.05
		0.30	0.00	79	**96**	1,521.26	**1,163.71**	1.3	**0.6**	33,557	32,419	32.36	22.19
		0.70	0.00	79	**96**	1,520.60	**1,166.55**	1.3	**0.6**	33,591	32,401	32.17	22.00
		0.95	0.00	79	**96**	1,520.42	**1,156.39**	1.3	**0.6**	33,530	32,539	31.83	22.38
80	160	0.05	0.01	**100**	**100**	**0.41**	0.49	**0.0**	**0.0**	460	513	0.10	0.09
		0.30	0.01	**100**	**100**	417.60	**87.04**	**0.0**	**0.0**	10,151	7,774	7.77	2.37
		0.70	0.01	**100**	**100**	429.02	**81.89**	**0.0**	**0.0**	10,151	7,774	7.87	2.37
		0.95	0.01	**100**	**100**	427.80	**83.81**	**0.0**	**0.0**	10,151	7,774	8.11	2.35
	240	0.05	0.02	71	**75**	2,183.99	**1,943.23**	10.1	**9.8**	35,350	41,248	39.01	26.64
		0.30	0.02	50	**54**	3,879.27	**3,682.90**	15.0	**14.7**	63,256	76,053	65.18	46.34
		0.70	0.02	50	**54**	3,877.28	**3,683.06**	15.0	**14.7**	62,657	76,121	64.99	46.59
		0.95	0.02	50	**54**	3,880.59	**3,680.04**	15.0	**14.7**	62,990	76,196	64.79	46.36
	305	0.05	0.04	**83**	**83**	1,753.58	**1,469.49**	4.0	**2.9**	38,062	40,842	31.56	18.93
		0.30	0.04	**50**	**50**	4,460.88	**3,864.57**	18.1	**17.2**	76,276	77,701	78.58	48.50
		0.70	0.04	**50**	**50**	4,490.00	**3,870.72**	18.1	**17.2**	76,217	78,267	79.64	48.66
		0.95	0.04	**50**	**50**	4,497.59	**3,851.17**	18.1	**17.2**	76,190	77,742	79.78	48.07
160	318	0.05	0.05	**100**	**100**	482.46	**97.28**	**0.0**	**0.0**	10,843	11,076	11.24	6.15
		0.30	0.05	**90**	**90**	1,184.63	**789.86**	**4.7**	4.8	17,259	18,296	22.61	16.58
		0.70	0.05	**90**	**90**	1,188.60	**790.20**	**4.7**	4.8	17,202	18,181	22.85	16.49
		0.95	0.05	**90**	**90**	1,193.56	**790.62**	**4.7**	4.8	17,218	18,287	23.25	16.45
	476	0.05	0.06	40	**48**	4,554.39	**4,021.91**	24.1	**22.5**	54,125	64,388	80.05	62.79
		0.30	0.06	40	**48**	4,559.88	**4,019.90**	24.1	**23.4**	54,282	63,315	78.73	62.93
		0.70	0.06	40	**48**	4,547.75	**4,029.75**	24.1	**23.4**	54,169	63,153	78.81	63.10
		0.95	0.06	40	**48**	4,551.50	**4,025.95**	24.1	**23.4**	53,996	63,254	79.33	63.12
	624	0.05	0.06	**0**	**0**	**7,200.00**	**7,200.00**	**42.8**	44.2	78,794	92,249	116.04	86.68
		0.30	0.06	**0**	**0**	**7,200.00**	**7,200.00**	**42.8**	44.1	79,571	92,678	116.18	86.07
		0.70	0.06	**0**	**0**	**7,200.00**	**7,200.00**	**42.9**	44.1	78,600	93,037	116.05	86.94
		0.95	0.06	**0**	**0**	**7,200.00**	**7,200.00**	**42.9**	44.1	78,782	92,626	114.94	85.85
200	398	0.05	0.08	90	**100**	777.55	**280.76**	0.9	**0.0**	10,719	14,264	12.99	11.17
		0.30	0.08	90	**100**	773.93	**273.86**	0.9	**0.0**	10,779	14,264	12.61	10.87
		0.70	0.08	90	**100**	771.35	**276.13**	0.9	**0.0**	10,858	14,264	12.41	10.77
		0.95	0.08	90	**100**	774.78	**273.65**	0.9	**0.0**	10,688	14,264	12.55	11.15
	596	0.05	0.09	**48**	**48**	**3,858.18**	3,904.13	**30.2**	31.0	42,346	47,730	65.39	55.38
		0.30	0.09	**48**	**48**	**3,857.94**	3,906.96	**30.2**	31.0	42,244	47,816	64.90	55.44
		0.70	0.09	**48**	**48**	**3,854.05**	3,911.56	**30.2**	31.1	42,179	47,784	64.78	55.56
		0.95	0.09	**48**	**48**	**3,855.45**	3,903.94	**30.2**	31.0	42,347	47,849	64.63	55.32
	784	0.05	0.09	**0**	**0**	**7,200.00**	**7,200.00**	**63.2**	64.0	71,315	82,216	122.47	112.42
		0.30	0.09	**0**	**0**	**7,200.00**	**7,200.00**	**63.2**	64.0	71,608	82,111	120.90	112.43
		0.70	0.09	**0**	**0**	**7,200.00**	**7,200.00**	**63.2**	64.0	71,476	82,241	120.26	113.03
		0.95	0.09	**0**	**0**	**7,200.00**	**7,200.00**	**63.2**	64.0	71,090	82,118	122.40	112.14

Table 2: Computational results for the EXT, CST-1, and CES formulations. For each graph configuration, we show the time required to identify a minimum-cost edge cut ($\lambda_{\mathbf{c}}(G)$), the percentage of instances solved to optimality (Opt), the average run time in seconds (time), and the average optimality gap (Gap).

Configration				Opt (%)			Time			Gap (%)		
n	$\bar{m}$	γ	$\lambda_{\mathbf{c}}(G)$	EXT	CST-1	CES	EXT	CST-1	CES	EXT	CST-1	CES
40	78	0.05	0	**100**	**100**	**100**	3.41	**0.04**	0.08	**0**	**0**	**0**
		0.30	0	**100**	**100**	**100**	5.33	0.24	**0.13**	**0**	**0**	**0**
		0.70	0	**100**	**100**	**100**	5.29	0.24	**0.04**	**0**	**0**	**0**
		0.95	0	**100**	**100**	**100**	5.34	0.24	**0.01**	**0**	**0**	**0**
	118	0.05	0	**100**	**100**	**100**	33.01	**0.26**	250.12	**0**	**0**	**0**
		0.30	0	**100**	96	**100**	**38.19**	1,163.71	89.29	**0**	1	**0**
		0.70	0	**100**	96	**100**	21.28	1,166.55	**0.38**	**0**	1	**0**
		0.95	0	**100**	96	**100**	20.86	1,156.39	**0.01**	**0**	1	**0**
80	160	0.05	0.01	**100**	**100**	**100**	430.38	**0.49**	2.74	**0**	**0**	**0**
		0.30	0.01	**100**	**100**	**100**	108.36	87.04	**4.39**	**0**	**0**	**0**
		0.70	0.01	**100**	**100**	**100**	53.92	81.89	**0.36**	**0**	**0**	**0**
		0.95	0.01	**100**	**100**	**100**	39.48	83.81	**0.01**	**0**	**0**	**0**
	240	0.05	0.02	67	**75**	63	3,663.23	**1,943.23**	2,821.78	22	**10**	12
		0.30	0.02	**100**	54	79	**648.94**	3,682.90	1,566.82	**0**	15	5
		0.70	0.02	**100**	54	**100**	477.78	3,683.06	**30.72**	**0**	15	**0**
		0.95	0.02	**100**	54	**100**	373.60	3,680.04	**0.02**	**0**	15	**0**
	305	0.05	0.04	**83**	**83**	67	4,425.12	**1,469.49**	3,463.12	5	**3**	13
		0.30	0.04	**100**	50	83	1,936.39	3,864.57	**1,818.04**	**0**	17	4
		0.70	0.04	**100**	50	**100**	2,328.33	3,870.72	**12.19**	**0**	17	**0**
		0.95	0.04	**100**	50	**100**	2,241.41	3,851.17	**0.02**	**0**	17	**0**
160	318	0.05	0.05	80	**100**	90	3,845.33	**97.28**	848.19	18	**0**	3
		0.30	0.05	**100**	90	**100**	2,295.34	789.86	**25.62**	**0**	5	**0**
		0.70	0.05	90	90	**100**	2,174.50	790.20	**0.70**	9	5	**0**
		0.95	0.05	90	90	**100**	1,973.65	790.62	**0.02**	8	5	**0**
	476	0.05	0.06	24	**48**	**48**	5,876.53	4,021.91	**3,829.95**	64	22	**18**
		0.30	0.06	32	48	**68**	5,479.02	4,019.90	**2,448.93**	52	23	**9**
		0.70	0.06	40	48	**100**	5,512.48	4,029.75	**64.36**	44	23	**0**
		0.95	0.06	40	48	**100**	5,009.95	4,025.95	**0.04**	39	23	**0**
	624	0.05	0.06	0	0	**20**	7,200.00	7,200.00	**6,066.43**	95	44	**35**
		0.30	0.06	0	0	**40**	7,200.00	7,200.00	**4,409.31**	94	44	**21**
		0.70	0.06	0	0	**100**	7,200.00	7,200.00	**274.48**	90	44	**0**
		0.95	0.06	0	0	**100**	7,200.00	7,200.00	**0.06**	100	44	**0**
200	398	0.05	0.08	60	**100**	**100**	4,377.91	**280.76**	645.62	38	**0**	**0**
		0.30	0.08	50	**100**	**100**	4,222.71	273.86	**12.33**	36	**0**	**0**
		0.70	0.08	60	**100**	**100**	3,493.65	276.13	**2.66**	21	**0**	**0**
		0.95	0.08	60	**100**	**100**	4,136.79	273.65	**0.02**	20	**0**	**0**
	596	0.05	0.09	28	**48**	**48**	5,597.36	3,904.13	**3,757.74**	67	31	**26**
		0.30	0.09	44	48	**56**	4,706.76	3,906.96	**3,270.92**	51	31	**18**
		0.70	0.09	36	48	**92**	5,462.50	3,911.56	**1,109.46**	59	31	**2**
		0.95	0.09	40	48	**100**	5,762.72	3,903.94	**0.08**	53	31	**0**
	784	0.05	0.09	**0**	**0**	**0**	**7,200.00**	**7,200.00**	**7,200.00**	97	64	**53**
		0.30	0.09	0	0	**20**	7,200.00	7,200.00	**5,789.52**	100	64	**37**
		0.70	0.09	0	0	**75**	7,200.00	7,200.00	**2,124.91**	100	64	**1**
		0.95	0.09	0	0	**100**	7,200.00	7,200.00	**0.16**	100	64	**0**
Average			0.03	68	68	**87**	2,964.13	2,569.97	**1,051.13**	27	15	**5**

Third, one additional explanation of why the CES formulation performs significantly better for larger values of γ is that whenever the interdiction requirement becomes relatively large, it is likely for the optimal solutions of the instances to be minimum-cost edge cuts (i.e., the interdiction requirement is so high that the best option for the attacker is to disconnect the graph). Despite all formulations starting with a minimum-cost edge cut as an incumbent solution, it takes significantly less time for the CES formulation to close the gap thanks to its stronger constraints.

6.4 CST and CES Constraint Separation Comparison

We now study the performance of separation subroutines for the CST and CES formulations in terms of the total number of cuts generated and the strength of such cuts. To this end, we analyze three different metrics: the number of cuts generated by each separation subroutine; the average size of the cuts generated (i.e., the number of variables with nonzero coefficient); and the separation time ratio, which we calculate as the time spent by the separation subroutine over the total running time taken by the optimizer to solve each instance. We summarize these results in Table 3.

Table 3: Computational results of the cut generation subroutines for the CST and CES formulations. For each graph configuration, we show the average number of constraints added by the separation subroutine (Constraints), the average size of the constraints (size), and the average separation time ratio (Sep. Ratio).

Configuration			Constraints		Size		Sep. Ratio	
n	$\underline{\delta}$	γ	CST	CES	CST	CES	CST	CES
40	78	0.05	39	109	39	36	6	55
		0.30	616	251	39	28	28	59
		0.70	616	54	39	16	26	75
		0.95	616	16	39	2	27	10
	118	0.05	579	11,453	39	37	21	45
		0.30	33,557	8,872	39	28	13	53
		0.70	33,591	627	39	18	13	80
		0.95	33,530	15	39	2	12	13
80	160	0.05	460	2,111	79	73	34	61
		0.30	10,151	1,923	79	55	21	79
		0.70	10,151	233	79	32	21	90
		0.95	10,151	10	79	4	21	91
	240	0.05	35,350	60,714	79	74	19	27
		0.30	63,256	39,271	79	56	13	45
		0.70	62,657	6,386	79	38	13	85
		0.95	62,990	17	79	4	13	86
	305	0.05	38,062	76,541	79	73	4	16
		0.30	76,276	49,249	79	54	3	38
		0.70	76,217	7,838	79	41	3	88
		0.95	76,190	12	79	4	3	67
160	318	0.05	10,843	21,253	159	143	22	68
		0.30	17,259	3,401	159	112	21	86
160	318	0.70	17,202	217	159	59	22	98
		0.95	17,218	7	159	7	21	85
	476	0.05	54,125	69,188	159	146	9	30
		0.30	54,282	48,798	159	109	9	49
		0.70	54,169	11,702	159	75	9	86
		0.95	53,996	20	159	7	9	85
	624	0.05	78,794	102,209	159	145	2	6
		0.30	79,571	81,787	159	106	2	20
		0.70	78,600	41,180	159	85	2	70
		0.95	78,782	30	159	7	2	72
200	398	0.05	10,719	14,803	199	179	24	79
		0.30	10,779	2,312	199	142	25	89
		0.70	10,858	663	199	78	25	99
		0.95	10,688	8	199	9	24	97
	596	0.05	42,346	53,423	199	182	16	42
		0.30	42,244	54,065	199	136	16	48
		0.70	42,179	31,639	199	83	17	75
		0.95	42,347	36	199	9	16	93
	784	0.05	71,315	99,195	199	181	2	5
		0.30	71,608	91,791	199	132	2	11
		0.70	71,476	78,270	199	102	2	58
		0.95	71,090	73	199	9	2	82

As one might expect, by observing the total number of cuts added and separation time ratio of both approaches, it is easy to conclude that the separation subroutine for the CST formulation is much faster, as it only requires finding spanning trees of small weight, whereas the CES separation algorithm requires exploring the branches of the semilattice structure described in Section 5. Indeed, the separation time ratio for CES is in average about four times the one of CST.

Furthermore, the results shown in Table 3 also provide a clear explanation why the CES formulation typically outperforms the CST formulation in almost all the graph configurations, despite having to compute a more complex separation subroutine. First, the experiments indicate that the size of the constraints—measured by the number of non-zeros—of the CES formulation is always smaller than those of CST, often by a large margin. As expected, this trend becomes significantly more pronounced for larger values of γ. Notice for example that when $\gamma = 0.70$, the size of the critical edge set constraints is about half the size of the critical spanning tree constraints; and, when $\gamma = 0.95$, the size of the critical edge set constraints becomes less than 6% the size of the critical spanning tree constraints, making the former significantly stronger. Second, it can be observed that the average number of constraints added to the CES formulation is much smaller than the ones added to CST, which is also expected given the result from Corollary 3. The number of cuts generated by the CES separation subroutine is only larger than the one for CST when γ is small, but decreases rapidly as γ becomes larger.

With constraints of such smaller sizes, it is natural to expect the CES's feasible set to be significantly smaller than the one of CST. A natural observation derived from this trend is that whenever the value of γ grows, a large portion of the feasible solutions for the attacker tend to be edge cuts, which are part of the solution space of CST, but by design are excluded form the one of CES.

6.5 Results on Real-Life Networks

All the experiments presented so far were conducted on randomly generated graphs. In general, real-life infrastructure graphs have interesting properties that are worth studying. In Table 4, we show the results of running the CST-1 and CES algorithms over 12 real-life networks collected from the Network Repository (2019). These are instances that have more vertices than the random graphs we used before, but in general are sparser. For these instances, we also generate the weight on each edge using the same uniform distribution ranging from 20 to 80, but we assume the interdiction costs are one. Also, we set the interdiction percentage γ at the lowest level 0.05. The results are presented in Table 4. Interestingly, due to the sparsity of the graphs, despite their size, both formulations are able to solve these instances significantly faster than the randomly generated graphs. We notice that CST-1 is slightly faster than CES over all the instances. This is consistent with the previous analysis, since when γ is small, CST-1 tends to have a better performance.

Table 4: Computational results for the CES formulation for solving several real-life instances (time in seconds).

Network type	**Instance**	n	m	**Avg. Degree**	γ	r	**Time CST-1**	**Time CES**
Infrastructure	inf-euroroad	1,174	1,442	2.46	0.05	50,542	0.16	0.83
	inf-openflights	2,939	15,687	10.68	0.05	88,446	0.07	0.36
	inf-power	4,936	6,584	2.67	0.05	204,261	0.06	0.49
	inf-USAir97	327	2,113	12.92	0.05	7,946	0.01	0.06
Power	power-1138-bus	1,138	1,458	2.56	0.05	49,115	0.03	0.12
	power-494-bus	494	586	2.37	0.05	21,840	0.01	0.03
	power-662-bus	662	906	2.74	0.05	26,525	0.02	0.05
	power-bcspwr10	5,300	8,271	3.12	0.05	192,378	0.12	0.66
	power-US-Grid	4,937	6,587	2.67	0.05	204,529	0.06	0.48
Road	road-euroroad	1,174	1,442	2.46	0.05	48,906	0.02	0.10
	road-minnesota	2,641	3,303	2.50	0.05	109,363	0.12	0.92

7 Concluding Remarks

In this paper, we discussed several integer programs for solving a minimum spanning tree interdiction problem where the objective of the attacker is to minimize the cost of raising the weight of the minimum spanning tree in the residual graph to a given threshold by removing a set of edges.

We performed a polyhedral analysis of the convex hull of feasible solutions for the problem, identifying several families of facet defining inequalities, and designing the corresponding algorithms to separate them. We also developed a third formulation composed by a family of so-called supervalid inequalities that may remove several feasible solutions, but retain at least one optimal solution. The constraints of this third formulation result from a new class of structures we called critical edge sets. We proved several fundamental properties of such structures and of the resulting formulation.

Our experiments showed that a systematic generation of cactus inequalities of rank one can help to improve the performance of the critical spanning tree formulation. Furthermore, they also evidenced that the critical edge set formulation outperforms the other two in most of the instances we tested.

Finally, we believe that identifying other families of valid inequalities, as well as developing efficient separation algorithms of cactus inequalities of higher ranks can be beneficial in practice. All the developments of this paper are aimed towards exact solution approaches for the spanning tree interdiction problem. We believe, however, that exploring heuristic algorithms can further complement our approaches and may help to potentially solve other large and challenging instances. Furthermore, it may be interesting to test whether other types of critical structures, like the critical edge sets, can be found in other graph interdiction problems to produce similar types of supervalid inequalities.

A Mathematical Proofs

Proposition 1 *Setting the values of M_e, for all $e \in E$ so that*

$$M_e \geq \max\left\{0, \max_{T\in\Omega_G[e]}\left\{r - \sum_{e'\in E(T)} w_{e'}\right\}\right\}, \forall e \in E, \tag{18}$$

results in a valid EXT formulation for the minimum spanning tree interdiction problem.

Proof. By the weak duality theorem, constraint (16) ensures that for any spanning tree $T \in \Omega_G$ whose weight is less than r, at least one edge $e \in E(T)$ must be removed by the attacker (i.e., there is at least one edge $e \in E(T)$ for which $x_e = 1$). Thus, to guarantee the validity of (15)–(17), it suffices to show that for the given bounds, any spanning tree T in G that includes an edge e for which $x_e = 1$ has an objective value in (9) of no less than r. To this end, assume for a contradiction that for a given feasible attacking strategy $S \subseteq E$, there is a spanning tree $T \in \Omega_G$ so that $E(T) \cap S \neq \emptyset$ and $\sum_{e\in E(T)} w_e + \sum_{e\in E(T)\cap S} M_e < r$. Then, for any edge $e \in E(T) \cap S$,

$$M_e \leq \sum_{e'\in E(T)\cap S} M_{e'} < r - \sum_{e'\in E(T)} w_{e'} \leq \max_{T'\in\Omega_G[e]}\left\{r - \sum_{e'\in E(T')} w_{e'}\right\},$$

which contradicts (18). Furthermore, this bound is tight for edge $e \in E$, if there is a spanning tree $T \in \arg\max_{T'\in\Omega_G[e]}\{r - \sum_{e'\in E(T')} w_{e'}\}$, so that $w(T) < r$ and $S \cap T = \{e\}$. □

Theorem 3 *Let $\mathcal{P}_{ext} = \{(\mathbf{x}, \alpha, \beta, \gamma) \in [0,1]^m \times \mathbb{R}^{n^2} \times \mathbb{R}^{nm} \times \mathbb{R}$, satisfying (11)–(14),(16)$\}$ be the polytope described by the linear relaxation of the extended formulation, with a proper value for $M_e, \forall e \in E$ satisfying (18). Then, the projection $\text{proj}_{\mathbf{x}}(\mathcal{P}_{ext})$ onto the $\mathbf{x}$-space corresponds to the polyhedron described by the following inequalities:*

$$\sum_{e\in E(T)} M_e x_e \geq r - \sum_{e\in E(T)} w_e, \quad \forall T \in \Omega_G \tag{22}$$

$$0 \leq x_e \leq 1, \quad \forall e \in E. \tag{23}$$

Proof. Let $S \subseteq E$ be an interdiction strategy for the attacker, with $\mathbf{x} = [x_e]_{e\in E}$ being the incidence vector of edge set S. From (10)–(14) we have that S is a feasible solution for the attacker if the following system is not empty:

$$\begin{aligned}
&-\sum_{i\in V}\sum_{l\in V\setminus\{k\}} b_i^l \alpha_i^l - (n-1)\gamma \leq -r \\
&\sum_{l\in V\setminus\{k\}} \beta_e^l + \gamma \leq w_e + M_e x_e \quad \forall e \in E \\
&\alpha_i^l - \alpha_j^l - \beta_e^l \leq 0 \qquad \forall e \in E, l \in V\setminus\{k\} \\
&\alpha_j^l - \alpha_i^l - \beta_e^l \leq 0 \qquad \forall e \in E, l \in V\setminus\{k\} \\
&\beta_e^l \geq 0 \qquad \forall e \in E, l \in V\setminus\{k\}.
\end{aligned}$$

By the Farkas' theorem of alternatives, the above system has a solution if and only if there is no solution $(z, \mathbf{y}, \mathbf{f})$, where $z \in \mathbb{R}$, $\mathbf{y} \in \mathbb{R}^n$ and $\mathbf{f} \in \mathbb{R}^{(2m)\times(n-1)}$, that satisfies the following system:

$$\sum_{e\in E}(w_e + M_e x_e)y_e - rz < 0 \tag{28}$$

$$\sum_{\{j:\{i,j\}\in E\}} f_{ij}^l - \sum_{\{j:\{i,j\}\in E\}} f_{ji}^l - b_i^l z = 0 \quad \forall i \in V, l \in V\setminus\{k\} \tag{29}$$

$$y_e - f_{ij}^l - f_{ji}^l \geq 0 \qquad \forall e = \{i,j\} \in E, l \in V\setminus\{k\} \tag{30}$$

$$\sum_{e \in E} y_e - (n-1)z = 0 \tag{31}$$

$$z \geq 0 \tag{32}$$

$$f_{ij}^l, f_{ji}^l \geq 0 \qquad \forall \{i,j\} \in E,\, l \in V \setminus \{k\} \tag{33}$$

$$y_e \geq 0 \qquad \forall e \in E, \tag{34}$$

Claim 1 *The set of extreme rays of the cone defined by (29)–(34) is characterized by the spanning trees of graph G.*

Proof. First, it should be clear that the cone is centered at the origin $(z, \mathbf{y}, \mathbf{f}) = (0, \mathbf{0}, \mathbf{0})$ and from (29) and (31), having $z = 0$ results in $\mathbf{y} = \mathbf{0}$ and $\mathbf{f} = \mathbf{0}$, which implies that, for all the rays of the given cone, $z > 0$. Second, note that setting $z = 1$ results in the network design formulation (6)–(8), which is known to be integral and whose extreme points characterize the set of spanning trees in G (see Remark 1). Finally, the value of z can be seen as a positive multiplier for the right hand side of constraints (6) and (7). Thus, any ray satisfying (29)–(34) with $z = \hat{z} > 0$ is essentially the ray $(1, \mathbf{y}, \mathbf{f})$ that characterizes a spanning tree $T \in \Omega_G$ after being scaled by $\hat{z}$.

By the Minkowski-Weyl representation theorem, all the solutions of cone (29)–(34) can be produced as conic combinations of its extreme rays. Thus, system (28)–(34) has no solution if

$$\sum_{e \in T} (w_e + M_e x_e) \geq r \qquad \forall T \in \Omega_G,$$

proving the result. □

Corollary 1 *Let $\mathcal{P}_{cst} = \{\mathbf{x} \in [0,1]^m,\ \mathbf{x}$ satisfying (20)$\}$ be the polytope described by the linear relaxation of the critical spanning tree formulation, then*

1. *$\mathcal{P}_{cst} \subseteq \operatorname{proj}_{\mathbf{x}}(\mathcal{P}_{ext})$.*
2. *$\mathcal{P}_{cst} = \operatorname{proj}_{\mathbf{x}}(\mathcal{P}_{ext})$ if and only if all critical spanning trees have the same weight.*

Proof. For part 1, we argue that given any $\mathbf{x} \in \mathcal{P}_{cst}$, the inequality $\sum_{e \in E(T)} M_e x_e \geq r - w(T)$ holds for all $T \in \Omega_G$. First, it is clear that the corresponding constraint is trivially satisfied for any tree $T \notin \Omega_G^r$, as for those trees, $r < w(T)$. Now, if $T \in \Omega_G^r$, then we have

$$\sum_{e \in E(T)} M_e x_e \geq \min_{e \in T} M_e \geq r - w(T),$$

where the first inequality stands from the fact that $\mathbf{x} \in \mathcal{P}_{cst}$ and the second inequality is derived directly from the definition of M_e in (18).

For part 2, it is enough to show that $\mathcal{P}_{cst} \supseteq \operatorname{proj}_{\mathbf{x}}(\mathcal{P}_{ext})$ if and only if all critical spanning trees have the same weight. To prove sufficiency, we note that if all the critical spanning trees have the same weight (i.e., $w(T) = \underline{w}_G, \forall T \in \Omega_G^r$), then from (18) we obtain

$$M_e = r - \min_{T' \in \Omega_G[e]} w(T') = r - \underline{w}_G,$$

for all $e \in E$. Now, for any $\mathbf{x} \in \operatorname{proj}_{\mathbf{x}}(\mathcal{P}_{ext})$ we have

$$\sum_{e \in E(T)} M_e x_e \geq r - w(T) = r - \underline{w}_G), \forall T \in \Omega_G^r.$$

Dividing by M_e on both sides yields $\sum_{e\in T} x_e \geq 1, \forall T \in \Omega_G^r$, which implies that $\mathbf{x} \in \mathcal{P}_{cst}$.

To prove necessity, we begin by showing that $\mathcal{P}_{cst} \supseteq \text{proj}_{\mathbf{x}}(\mathcal{P}_{ext})$ implies that

$$M_e = r - w(T), \forall T \in \Omega_G^r,\ e \in E(T). \tag{35}$$

To prove this statement, assume for a contradiction that $\mathcal{P}_{cst} \supseteq \text{proj}_{\mathbf{x}}(\mathcal{P}_{ext})$ and that there exists a $T_0 \in \Omega_G^r$ and an edge $e_0 \in E(T_0)$ for which $M_{e_0} \neq r - w(T_0)$. From (18), this implies that $M_{e_0} > r - w(T_0)$. We now show that in such a case, it is possible to construct a solution $\bar{\mathbf{x}} \in \text{proj}_{\mathbf{x}}(\mathcal{P}_{ext})$, so that $\bar{\mathbf{x}} \notin \mathcal{P}_{cst}$. The proposed construction is as follows:

$$\bar{x}_e = \begin{cases} (r - w(T_0))\,/M_{e_0} & e = e_0, \\ 0 & e \in E(T_0) \setminus \{e_0\}, \\ 1 & e \in E \setminus E(T_0). \end{cases}$$

First, to see that $\bar{\mathbf{x}} \in \text{proj}_{\mathbf{x}}(\mathcal{P}_{ext})$ note that for any $T \in \Omega_G^r$, if $T \neq T_0$, there is some edge $e_1 \in E(T)$ for which $\bar{x}_{e_1} = 1$, then from (18) we obtain

$$\sum_{e\in E(T)} M_e \bar{x}_e \geq M_{e_1}\bar{x}_{e_1} = M_{e_1} \geq r - w(T).$$

On the other hand, if $T = T_0$, we have

$$\sum_{e\in E(T_0)} M_e x_e = M_{e_0} x_{e_0} = M_{e_0}(r - w(T_0))/M_{e_0} = r - w(T_0),$$

which proves that $\bar{\mathbf{x}}$ satisfies (22)–(23). Furthermore, observe that for this solution, $\sum_{e\in T_0} \bar{x}_e = (r-w(T_0))/M_{e_0} < 1$ implying that $\bar{\mathbf{x}} \notin \mathcal{P}_{cst}$. Thus, $\mathcal{P}_{cst} \not\supseteq \text{proj}_{\mathbf{x}}(\mathcal{P}_{ext})$, a contradiction.

We continue the proof by considering any pair of critical spanning trees $T_1, T_2 \in \Omega_G^r$. Notice that that if there exists some edge $\hat{e} \in E$ for which $\hat{e} \in E(T_1) \cap E(T_2)$, then from (35) we have that $M_{\hat{e}} = r - w(T_1) = r - w(T_2)$, implying that $w(T_1) = w(T_2)$. Now, suppose $E(T_1) \cap E(T_2) = \emptyset$; then, since the spanning trees are the basis of the graph matroid, by the *Strong Basis Exchange Property* of matroids (Brualdi 1969), there exists a pair of edges $e_1 \in E(T_1)$ and $e_2 \in E(T_2)$ such that the subgraphs T_1' and T_2' induced by the edge sets $E(T_1)\setminus\{e_1\}\cup\{e_2\}$ and $E(T_2)\setminus\{e_2\}\cup\{e_1\}$, respectively, are spanning trees. Without loss of generality, assume that $w(T_1') \leq w(T_1)$ (otherwise $w(T_2') \leq w(T_2)$ and the following result applies for T_2'); then, T_1' must also be a critical spanning tree and since T_1' shares at least one edge with T_1 and T_2, form (35) we have that $w(T_1') = w(T_1) = w(T_2)$, proving the desired result. $\square$

Theorem 4 *Let* $\mathbf{x}$ *be a candidate solution for the attacker. Then, separating the critical spanning tree constraints is:*

1. *NP-hard if* $\mathbf{x} \in [0,1]^m$
2. *solvable in* $O(m \log n)$ *if* $\mathbf{x} \in \{0,1\}^m$

Proof. The membership in NP should be obvious. The first statement follows by noting that this separation problem is closely related to the *constrained minimum spanning tree problem*, which is known to be weakly NP-hard (Aggarwal et al. 1982, Ravi and Goemans 1996). The decision version of this problem is given below.

Constrained minimum spanning tree (Decision)

Instance: A graph $G = (V, E)$, a non-negative integer vector of edge weights $\mathbf{v} = [v_e]_{e\in E}$, a non-negative integer vector of lengths $\boldsymbol{\ell} = [l_e]_{e\in E}$, and two integer positive numbers R and L.

Question: Is there a spanning tree T in G such that $\sum_{e\in T} v_e \leq R$ and $\sum_{e\in T} l_e < L$?

Given an instance of the constrained minimum spanning tree problem, one can easily construct a critical spanning tree separation instance by setting $x_e = \min\{1, l_e/L\}$, $\forall e \in E$, $r = R - 1$, and $\mathbf{w} = \mathbf{v}$, which is possible given that constant $L > 0$. This yields the separation problem NP-hard.

To prove that the integral separation is polynomial-time solvable, it suffices to see that finding a minimum spanning tree, according to the weight vector w in graph $G(V, E \setminus S)$, where $S = \{e \in E, x_e = 1\}$ yields a most violated critical spanning tree constraint, in case the weight of such spanning tree is less than r. Generating $G(V, E \setminus S)$ can be done in $O(m)$ and solving the spanning tree problem can be done in $O(m \log n)$ (Ahuja et al. 1993). $\square$

Theorem 5 *Given a simple, nonempty, connected graph $G = (V, E)$, a weight vector $\mathbf{w} \in \mathbb{Q}^m$ so that there are no irrelevant edges in E (i.e., $\min\{w(T) | T \in \Omega_G[e]\} < r, \forall e \in E$), and a scalar $r \in \mathbb{Q}$, so that $\underline{w}_G < r < \overline{w}_G$, the following statements are true.*

(1) $\mathcal{P}$ is a full-dimensional polytope.

(2) Given an edge $e \in E$, inequality $x_e \leq 1$ induces a facet of $\mathcal{P}$.

(3) Given an edge $e \in E$, inequality $x_e \geq 0$ induces a facet of $\mathcal{P}$ if and only if $n \geq 4$.

(4) Given a critical spanning tree $T \in \Omega_G^r$, the corresponding critical spanning tree inequality $\sum_{e \in E(T)} x_e \geq 1$ induces a facet of $\mathcal{P}$ if and only if:

(a) there is no edge $e \in E$ such that if added to T, it completes a Hamiltonian cycle in G, or

(b) if such an edge exists, at least one of the spanning trees contained in the resulting Hamiltonian cycle has a weight greater than r.

Proof.

(1) To show that $\mathcal{P}$ is a full-dimensional polytope, it suffices to prove that there are $m+1$ affinely independent points in $\mathcal{P}$. Notice that edge sets $E \setminus \{e\}, \forall e \in E$, along with edge set E are feasible solutions for the attacker because removing each of those edge sets disconnects graph G. It can be easily verified that the incidence vectors of these feasible solutions are affinely independent; therefore, $\mathcal{P}$ is a full-dimensional polytope.

(2) For any given edge $e \in E$, notice that the face of $\mathcal{P}$ that is induced by inequality $x_e \leq 1$ contains the following m feasible solutions, the complete edge set E and the edge sets $E \setminus \{e'\}, \forall e' \in E \setminus \{e\}$. Since the corresponding incidence vectors of these solutions are affinely independent, the dimension of such a face is $m - 1$; therefore, it is a facet of $\mathcal{P}$.

(3) To show sufficiency, we prove that for any given edge $e \in E$, the face induced by inequality $x_e \geq 0$ is of dimension $m - 1$, whenever $n \geq 4$. To this end, notice that sets $E \setminus \{e\}$ and $E \setminus \{e, e'\}, \forall e' \in E \setminus \{e\}$ are feasible solutions for the attacker because removing the those edge sets leaves at the most two edges in the graph, making the resulting graph disconnected. Observe that for the graph to remain connected, there must exists a spanning tree with $n - 1 \geq 3$ edges, since by assumption $n \geq 4$. Furthermore, it is easy to see that such m solutions belong to the face and are affinely independent.

To show necessity, it is trivial to see by inspection that $x_e \geq 0$ only induces a facet of $\mathcal{P}$ for a connected graph G with $n \leq 3$ in cases where $\underline{w}_G = \overline{w}_G \geq r$, or whenever there is an irrelevant edge in G, which are both trivial instances ruled out by definition.

(4) To show sufficiency, it is enough to prove that given a critical spanning tree $T \in \Omega_G^r$, satisfying either (a) or (b), the dimension of the face induced by the critical spanning tree inequality $\sum_{e \in E(T)} x_e \geq 1$ is $m - 1$. We will show that it is possible to construct m affinely independent solutions contained in such a face. The proposed construction is as follows. First, consider the $n - 1$ solutions, given by one edge $e \in E(T)$ and all the edges in $E \setminus E(T)$. Clearly, such solutions lie in the given face and are feasible alternatives for the attacker, as

those disconnect the graph. Second, consider the solution composed of all the edges in $E \setminus E(T)$ except for one edge e' and one edge $e \in E(T)$ that is selected depending on e' as follows: since adding edge e' to spanning tree T produces a cycle, if T satisfies condition (a), then select as edge e one of the edges in $E(T)$ that is not in the cycle. Alternatively, if T satisfies condition (b), then select as edge e one of the edges in $E(T)$ so that the tree induced by $(E(T) \setminus \{e\}) \cup \{e'\}$ has a weight greater than r. This procedure is then repeated for every edge $e' \in E \setminus E(T)$, thus generating the other $m - n + 1$ solutions. Notice that all of them lie on the given face and are feasible for the attacker, as the residual graph becomes disconnected whenever T satisfies (a), or the residual graph left is a spanning tree of weight greater than r whenever T satisfies (b). It can be easily verified that the incidence vectors associated with such solutions are affinely independent.

To show necessity, assume for a contradiction that there is a spanning tree $T \in \Omega^r_G$ and an edge $e' \in E \setminus E(T)$ so that $E(T) \cup \{e'\}$ yields a Hamiltonian cycle in G for which all the n spanning trees contained in it are in Ω^r_G. First, notice that we can take all the n critical spanning tree inequalities associated with the spanning trees contained in the Hamiltonian cycle and apply the Chvátal-Gomory rounding procedure (Chvátal 1973, Gomory 1963) to them in order to generate a new valid inequality. To this end, we simply add such inequities and since each edge in the Hamiltonian cycle appears in $n - 1$ spanning trees, after dividing by $n - 1$ and rounding up the resulting values, we obtain the following valid inequality

$$\sum_{e \in E(T)} x_e + x_{e'} \geq \left\lceil \frac{n}{n-1} \right\rceil = 2. \tag{36}$$

Then, notice that adding the upper bound inequality $-x_{e'} \geq -1$ to (36) produces the original critical spanning tree inequality, which implies that such a constraint can be generated by a linear combination of other valid inequalities for the problem and therefore does not induce a facet of $\mathcal{P}$. This concludes the proof. □

Theorem 6 *Given a simple, nonempty, connected graph $G = (V, E)$, a weight vector $\mathbf{w} \in \mathbb{Q}^m$, a scalar $r \in \mathbb{Q}$ so that $\underline{w}_G < r < \overline{w}_G$, and a k-cactus H of G, with $k \geq 1$, the inequality*

$$\sum_{e \in \hat{\mathcal{C}}(H)} x_e + 2 \sum_{e \in \mathcal{C}(H) \setminus \hat{\mathcal{C}}(H)} x_e + 2 \sum_{e \in F(H)} x_e \geq 2 \tag{24}$$

induces a facet of $\mathcal{P}$, if and only if $|\hat{C}_l(H)| \geq 3, \forall l = 1, \ldots, k$.

Proof. We begin by proving that the cactus inequality (24) is valid for $\mathcal{P}$. To this end, we apply the Chvátal-Gomory rounding procedure for all the critical spanning tree inequalities associated with the critical spanning trees in H. For each edge $e \in E(H)$, let π_e and $\bar{\pi}_e$ be the number of critical spanning trees in H that contain and not contain edge e, respectively. Since the total number of critical spanning trees in H is $|\Omega^r_H|$, we have that $\pi_e + \bar{\pi}_e = |\Omega^r_H|$, for every $e \in E(H)$. Furthermore, notice that $\pi_e = |\Omega^r_H|$, for all $e \in F(H)$ as well as for all $e \in \mathcal{C}(H) \setminus \hat{\mathcal{C}}(H)$. To see this, observe that each edge $e \in F(H)$ must be part of any spanning tree of H because removing e disconnects H. Similarly, each $e \in \mathcal{C}(H) \setminus \hat{\mathcal{C}}(H)$ must exist in all the critical spanning trees of H because those trees are generated by removing one edge from each $\hat{C}_l(H), \forall l = 1, \ldots, k$. If one of the edges in $\mathcal{C}(H) \setminus \hat{\mathcal{C}}(H)$ were to be missing from one of the critical spanning trees, that would imply that the tree does not span V because in such a case there must be a cycle $\hat{C}_l(H)$ for some $l \in \{1, \ldots, k\}$ for which two edges are not in the tree.

Now, after adding all the inequalities associated with the critical spanning trees in H, dividing by $\boldsymbol{\pi} = \max_{e \in \hat{\mathcal{C}}(H)} \{\pi_e\}$, and rounding up the resulting coefficients, we obtain the following expression:

$$\sum_{e \in \hat{\mathcal{C}}(H)} \left\lceil \frac{\pi_e}{\boldsymbol{\pi}} \right\rceil x_e + \sum_{e \in \mathcal{C}(H) \setminus \hat{\mathcal{C}}(H)} \left\lceil \frac{|\Omega^r_H|}{\boldsymbol{\pi}} \right\rceil x_e + \sum_{e \in F(H)} \left\lceil \frac{|\Omega^r_H|}{\boldsymbol{\pi}} \right\rceil x_e \geq \left\lceil \frac{|\Omega^r_H|}{\boldsymbol{\pi}} \right\rceil . \tag{37}$$

To complete the proof of validity, it suffices to show that $1 < \left\lceil \frac{|\Omega^r_H|}{\boldsymbol{\pi}} \right\rceil \leq 2$. Let $e' \in \arg\max_{e \in \hat{\mathcal{C}}(H)} \{\pi_e\}$. First, note that

$$\frac{|\Omega^r_H|}{\boldsymbol{\pi}} = \frac{\pi_{e'} + \bar{\pi}_{e'}}{\pi_{e'}} = 1 + \frac{\bar{\pi}_{e'}}{\pi_{e'}} > 1.$$

Then, assume that $e' \in \hat{C}_l(H)$ and, since by definition $|\hat{C}_l(H)| \geq 3$, let $e'' \in \hat{C}_l(H)$, so that $e'' \neq e'$. It should be clear that $\pi_{e''} \geq \bar{\pi}_{e'}$ because both edges cannot be missing at the same time from any spanning tree, so if e' is not in a spanning tree, e'' must be in such a tree. Furthermore, observe that $\boldsymbol{\pi} = \pi_{e'} \geq \pi_{e''} \geq \bar{\pi}_{e'}$, proving that $\frac{\bar{\pi}_{e'}}{\pi_{e'}} \leq 1$.

We now proceed to prove the main result of the theorem. To show sufficiency, it is enough to prove that for the given k-cactus H, the dimension of the face induced by (24) is $m-1$. For this purpose, we will show that it is possible to construct m affinely independent, feasible solutions contained in such a face. The proposed construction is divided into the following three steps:

First, let $e_{(1)}, e_{(2)}$, and $e_{(3)}$ be the first three edges of $\hat{C}_l(E)$, based on the index given to those edges in E. For each edge $e \in \hat{C}_l(H) \setminus \{e_{(1)}\}$ and each cycle $l \in 1, \ldots, k$, construct the solution given by the edge set $\{e_{(1)}, e\} \cup (E \setminus E(H))$. Also, consider the solution associated with edge set $\{e_{(2)}, e_{(3)}\} \cup (E \setminus E(H))$. This procedure results in a total of $|\hat{\mathcal{C}}(H)|$ solutions that lie on the given face. Note that all of these solutions are feasible for the attacker because they disconnect G.

Second, consider all the $|E(H) \setminus \hat{\mathcal{C}}(H)|$ solutions composed of one edge e from either $F(H)$ or $\mathcal{C}(H) \setminus \hat{\mathcal{C}}(H)$ and all the edges in $E \setminus E(H)$. Notice that if $e \in F(H)$, then the resulting solution disconnects G and, if $e \in C(H) \setminus \hat{C}(H)$, then the corresponding solution leaves a subgraph in which all the spanning trees left have a weight at least r. Therefore, all of these solutions are feasible for the attacker and belong to the given face.

Third, consider the solutions composed of all the edges in $E \setminus E(H)$, except for one edge e, and two edges e' and e'' from a cycle $\hat{C}_l(H)$, so that l, e' and e'' are selected depending on edge e as follows: (1) if edge e shares both endpoints with edges in $F(H)$, then select as e' and e'' any two edges in $\hat{C}_l(H)$, for any $l \in \{1, \ldots, k\}$. (2) If edge e shares both endpoints, say vertices i and j, with edges of cycle $C_l(H)$, for a given $l \in \{1, \ldots, k\}$ (i.e., if e is a cord of cycle $C_l(H)$), then it is easy to see that cycle $C_l(H)$ is composed of two paths connecting vertices i and j and, since $|\hat{C}_l(H)| \geq 3$, one of these two paths must contain at least two edges in $\hat{C}_l(H)$. For this case then, select as e' and e'' any two edges in $\hat{C}_l(H)$ that belong to the same path. (3) If edge e shares exactly one endpoint, say i, with edges in $F(H)$ and the other endpoint, say j, with edges in $C_l(H)$, for a given $l \in \{1, \ldots, k\}$, then there must be a path in H connecting vertices i and j that passes through a vertex p that is the endpoint of both an edge in $F(H)$ and an edge in $C_l(H)$ (i.e., such a path goes from i to p via edges in $F(H)$ and from p to j via edges in $C_l(H)$). Furthermore, since vertices j and p are both endpoints of edges in cycle $C_l(H)$, there are two paths connecting such vertices in the cycle. For this case then, similarly as in (2), select as e' and e'' any two edges in $\hat{C}_l(H)$ that belong to the same path. (4) If edge e shares exactly one endpoint, say i, with edges in $C_l(H)$ and the other endpoint, say j, with edges in $C_{l'}(H)$, then there must be a path in H connecting vertices i and j that passes through a vertex $p \neq i$ that is the endpoint of an edge in $C_l(H)$ (i.e., such a path goes from i to p via edges in $C_l(H)$ and from p to j via edges in $F(H) \cup C_{l'}(H)$. Furthermore, since vertices j and p are both endpoints of edges in cycle $C_l(H)$, there are two paths connecting such vertices in the cycle. For this case then, similarly as in (2) and (3), select as e' and e'' any two edges in $\hat{C}_l(H)$ that belong to the same path. Notice that this third step produces $|E \setminus E(H)|$ solutions that belong to the face and are specifically constructed to disconnect graph G; thus, are feasible solutions for the attacker.

For any given graph G having a k-cactus H in which $\hat{C}_l(H) \geq 3, \forall l = 1, \ldots, k$, the above procedure generates $|\hat{\mathcal{C}}(H)| + |E(H) \setminus \hat{\mathcal{C}}(H)| + |E \setminus E(H)| = m$ feasible solutions for the attacker. Moreover, from the way they are constructed, it can be easily verified that the incidence vectors associated with such solutions are affinely independent; therefore, cactus inequality (24) induces a facet of $\mathcal{P}$.

Finally, to show necessity, we consider the two remaining trivial cases. First, notice that if $|\hat{C}_l(H) = 1|$ for some $l \in \{1, \ldots, k\}$, then the corresponding cactus inequality (24) for H is a linear combination of the cactus inequality associated with the $(k-1)$-cactus that results from removing the only edge $e \in \hat{C}_l(H)$ from H and the lower bound $x_e \geq 0$; thus, it does not induce a facet of $\mathcal{P}$. Second, if $|\hat{C}_l(H) = 2|$ for some $l = 1, \ldots, k|$, then the corresponding cactus inequality (24) for H is a linear combination of the two cactus inequalities for the two $(k-1)$-cacti that result from removing each of the two edges in $\hat{C}_l(H)$, respectively, and the two lower bounds of such edges. $\square$

Theorem 7 *Let $\mathbf{x} \in [0,1]^m$ be a candidate solution for the attacker. Then, the separation problem for the cactus inequalities is NP-hard.*

Proof. The membership in NP should be obvious. Now, the following is a trivial transformation from the critical spanning tree constraint separation problem, stated in Section 3, to the cactus inequality separation problem. Given an instance for the critical spanning tree constraint separation problem defined by a graph $G = (V,E)$, $\mathbf{w}$, $\mathbf{x}$, and r, we construct a graph $\hat{G}$ by adding two dummy vertices v' and v'' to G, one edge between v' and a vertex $v \in V$, one edge between v'' and v, and one edge between v' and v''. Hence, $\hat{G} = (V \cup \{v', v''\}, E \cup \{\{v,v'\}, \{v,v''\}, \{v',v''\}\})$. We create a weight vector $\hat{\mathbf{w}} \in \mathbb{Q}^{m+3}$ by letting $\hat{w}_e = w_e, \forall e \in E$ and $\hat{w}_{\{v,v'\}} = \hat{w}_{\{v,v''\}} = \hat{w}_{\{v',v''\}} = 0$. Similarly, we construct a vector $\hat{\mathbf{x}} \in [0,1]^{m+3}$ by letting $\hat{x}_e = x_e, \forall e \in E$ and $\hat{x}_{\{v,v'\}} = \hat{x}_{\{v,v''\}} = \hat{x}_{\{v',v''\}} = 0$. Finally, we say that $\hat{r} = r$. The procedure then results on an instance for the cactus inequality separation problem defined by $\hat{G}, \hat{\mathbf{w}}, \hat{\mathbf{x}}$, and $\hat{r}$.

It is easy to see that every spanning tree T in G is associated with a 1-cactus H in $\hat{G}$ whose cycle $C_1(H)$ is composed of edges $\{v,v'\}, \{v,v''\}, \{v',v''\}$ and whose set $F(H) = E(T)$. Clearly, if $T \in \Omega^r_G$, then H is a 1-cactus for which $\hat{C}_l(H) = C_l(H)$ and $|\hat{C}_l(H)| = 3$. Moreover, since $\hat{x}_e = 0, \forall e \in C(H)$, then the resulting cactus inequality is violated (i.e., $2\sum_{e \in F(H)} x_e < 2$), if and only if $\sum_{e \in E(T)} x_e < 1$. □

Theorem 8 *An edge set $S \subseteq E$ is a feasible solution for the attacker if and only if S is an edge cut or $S \cap K \neq \emptyset$, for all $K \in \tilde{\Lambda}^r_G$.*

Proof. The result is trivial for the case where S is an edge cut because removing S leaves the graph disconnected; thus, we will focus on the alternative. To prove sufficiency, we recall from Theorem 2 that a solution $S \subseteq E$ is feasible for the attacker if it contains at least one edge from each critical spanning tree $T \in \Omega^r_G$. Let T be any of such trees. Since $E(T)$ contains some $K \in \tilde{\Lambda}^r_G$, then $S \cap E(T) \supseteq S \cap K \neq \emptyset$, proving the result.

To show necessity, we prove the contrapositive. Suppose that a strategy S is not an edge cut and there exists some $K \in \tilde{\Lambda}^r_G$, whose intersection with S is empty. Clearly, graph $G(V, E \setminus S)$ is connected since S is not an edge cut. Moreover, since $K \subseteq (E \setminus S)$ and the fact that spanning trees are the bases of the graphic matroids of connected graphs, by the augmentation property of matroids (Oxley 2006), there must exist a spanning tree T in $G(V; E \setminus S)$ so that $E(T)$ contains K (i.e., $T \in \Omega_G[K]$). By definition all the spanning trees in $\Omega_G[K]$ are critical; therefore, T must be critical, which implies that S is not a feasible strategy. □

The Definition 4 and Proposition 3 below are for the proofs of Theorem 9 and 10.

Definition 4 (Dependent Product) *Given an indexed family of sets $\{A_i\}_{i \in I}$, the dependent product $\prod_{i \in I} A_i$ is defined as the collection of all the indexed sets $\{a_i \in A_i\}_{i \in I}$. Equivalently, it is the space of all the choice functions that select one element from each set A_i. Since the sets in $\{A_i\}_{i \in I}$ are not necessarily disjoint, the ordered sets in $\prod_{i \in I} A_i$ may have repeated elements.*

Proposition 3 *Given a spanning tree T and a cycle C such that $E(T) \cap C \neq \emptyset$, then, for any $e \in E(T) \cap C$ there exists an edge $e' \in C \setminus E(T)$ such that the cycle that is formed by adding e' to T contains e.*

Proof. Let $e = \{u,v\}$ be an arbitrary edge in $E(T) \cap C$, then the rest edges in the cycle form a path $P = C \setminus \{e\}$ that connects u and v. Removing e from $E(T)$ disconnects T into two trees T_u and T_v that contain the endpoints u and v, respectively. Let $V(T_u)$ and $V(T_v)$ be their sets of vertices; then, all the edges in E across these two vertex sets compose an edge cut of G. Since the path P connects $u \in V(T_u)$ and $v \in V(T_v)$, there must exist an edge $e' = \{u', v'\} \in P$ that is also in the cut. Now, we show that this choice of e' has the desired properties. First, $e' \neq e$ because e is not an edge in P by definition. Since both u and u' are in $V(T_u)$, and T_u is a tree, there is an unique path in T_u that connects u and u'. Similarly, there is an unique path in T_v that connects v and v'. Then, these two paths along with e and e' form a cycle in $E(T) \cup \{e'\}$ that contains e. □

Theorem 9 *An edge set $S \subseteq E$ is a feasible solution for the critical edge set formulation if and only if S is a superset (not necessarily proper) of any non-edge-cut feasible strategy for the attacker.*

Proof. To prove sufficiency, suppose the edge set S is the superset of a non-edge-cut feasible strategy for the attacker S'. Then, by Theorem 8, $S' \cap K \neq \emptyset$, for all $K \in \tilde{\Lambda}^r_G$; therefore, both S and S' satisfy (26).

Now, let $\{X_i\}_{i \in I}$ be the family of all minimal non-edge-cut feasible strategies for the attacker, in no particular order. To prove necessity, assume for a contradiction that S is a feasible solution for the critical edge set formulation, but is not a superset of any non-edge-cut feasible strategy for the attacker. Then, $X_i \setminus S \neq \emptyset$ for all $i \in I$, which implies that the dependent product $\prod_{i \in I}(X_i \setminus S)$ is not empty. Clearly, every set $L \in \prod_{i \in I}(X_i \setminus S)$ has a non-empty intersection with all non-edge-cut feasible solutions and $L \cap S = \emptyset$, by construction.

Claim 2 *If S is a feasible solution for the critical edge set formulation, but it is not a superset of any non-edge-cut feasible strategy for the attacker, then every set $L \in \prod_{i \in I}(X_i \setminus S)$ has cycles.*

Proof. First, note that $\Omega_G[L] \subseteq \Omega^r_G$; otherwise, there is a spanning tree $T \in \Omega_G[L]$ which is non-critical and therefore, $E \setminus E[T]$ is a non-edge-cut feasible strategy for the attacker, which is a contradiction since $L \cap (E \setminus E[T]) = \emptyset$. Also, notice that $L \notin \Lambda^r_G$, i.e., L cannot be a critical edge set, otherwise $S \cap L \neq \emptyset$ because S is a feasible solution for the formulation. This forces $\Omega_G[L] = \emptyset$ by the definition of the critical edge sets, which implies that some edges in L must form cycles.

We proceed to prove that the result of this claim leads to a contradiction by induction on the number of cycles in any arbitrary set $L \in \prod_{i \in I}(X_i \setminus S)$. Here, the induction basis states that there is an $L \in \prod_{i \in I}(X_i \setminus S)$ that contains no cycles, which directly leads to a contradiction. Now for the inductive step, suppose that there is a set $L \in \prod_{i \in I}(X_i \setminus S)$ that contains $n + 1$ cycles, we will show that in such a case, there exists another set $L' \in \prod_{i \in I}(X_i \setminus S)$ that contains at the most n cycles. First, since set X_i is minimal, for all $i \in I$, for each edge $e \in X_i$, there must exists a critical spanning tree $T_i \in \Omega^r_G$ such that $E[T_i] \cap X_i = \{e\}$; otherwise, if all the critical spanning trees that contain $\{e\}$ also contain more edges from X_i, then $X_i \setminus \{e\}$ is still a feasible strategy, which contradicts the minimality of X_i. Now, for an arbitrary cycle $C \subseteq L$, and an edge $e \in \arg\max_{e' \in C}\{w_{e'}\}$, there a spanning tree $T \in \Omega^r_G$, such that $E[T] \cap L = \{e\}$. Then, by Proposition 3, there is another edge $e' \in C \setminus E[T]$ that forms a cycle with T that contains e. Since $w(e) \leq w(e')$ by our choice of e, the new tree T' for which $E[T'] = E[T] \setminus \{e\} \cup \{e'\}$ has a smaller weight than the original tree T; thus, is critical too. Let $X_i \setminus S$ be the set where e is selected from to compose L, then either $e' \in X_i$ or not. For the latter case, since $E[T] \cap X_i = \{e\}$ by our choice of T, and $e' \notin X_i$, then $E(T') \cap X_i = \emptyset$, which is a contradiction because it implies X_i is not a feasible solution. For the former case, since $e' \in X_i$, we can construct another set $L' \in \prod_{i \in I}(X_i \setminus S)$ by replacing e with e' from set $X_i \setminus S$ (clearly e' is not in S since it has been selected from $(X_j \setminus S)$) and keeping other elements the same. Now, either e is not in L', or it is still there because some other $X_k \setminus S$ $(k \neq i, k \neq j)$ also select e_i to compose L. In the former case, the cycle C has been removed along with the edge e, so there are fewer cycles. Then, by induction, it leads to a contradiction. For the latter, we can repeat the previous argument for any set X_k that also selected edge e until we reach the desired contradiction. This procedure eventually terminates since L only contains a finite number of elements. □

Theorem 10 *For any minimal critical edge set $K \in \tilde{\Lambda}^r_G$, there exists a spanning tree $T \in \Omega_G[K]$ that does not contain any other minimal critical edge set $K' \in \tilde{\Lambda}^r$, where $K \neq K'$.*

Proof. We prove the theorem by contradiction. Suppose there is a minimal critical edge set $K \in \tilde{\Lambda}^r_G$ such that all trees in $\Omega_G[K]$ contain at least one critical edge set from $\tilde{\Lambda}^r_G \setminus \{K\}$. First, suppose that the spanning trees in $\Omega_G[K]$ are listed in no particular order according to an index set I, so that $\Omega_G[K] = \{T_i\}_{i \in I}$. Then, by assumption, we can directly create a list $\{K_i\}_{i \in I}$ so that K_i is a minimal critical edge set in $\tilde{\Lambda}^r_G \setminus \{K\}$ contained by spanning tree T_i, for all $i \in I$. The elements of list $\{K_i\}_{i \in I}$ may not necessarily unique, but, since each K_i is minimal and different from K, the dependent product set $\prod_{i \in I}(K_i \setminus K)$ is nonempty. Moreover, any element $L \in \prod_{i \in I}(K_i \setminus K)$ is an edge cut. For otherwise, $G(V, E \setminus L)$ would still be connected and contain K, then there must exist a spanning tree T in $G(V, E \setminus L)$ that contains K. Note that $E(T) \cap L = \emptyset$, so it does not contain any K_i by the construction of L, which is a contradiction.

Now, we show that the non-emptiness of $\prod_{i \in I}(K_i \setminus K)$ leads to a contradiction by induction on the number of minimal edge cuts that are contained in L. Here, the induction basis states that there are no minimal edge cuts in L, which is a constradiction from the last paragraph. Now, for the inductive step, suppose L contains $n+1$ minimal edge cuts. Take any minimal edge cut $D \subseteq L$, and pick an edge $e \in D$ with the highest weight, and let $K_i \setminus K$ be the set where e is picked from to produce L. Since K_i is a minimal critical edge set, there must exist some non-critical spanning tree T that contains $K_i \setminus \{e\}$, so that $e \notin E(T)$. Clearly, the intersection of $E(T)$ with D contains some other edge e' since D is a minimal edge cut. Now, either e' belongs to $K_i \setminus K$ or not. For the latter case, $E(T) \cup \{e\}$ forms a cycle that contains both e and e', and by our choice, $w(e) \geq w(e')$, so the tree formed by the edge set $E(T') = E(T) \setminus \{e'\} \cup \{e\}$ is still non-critical. However, this contradicts the fact that T' contains the critical edge set K_i. For the former case, we can construct a new $L' \in \prod_{i \in I}(K_i \setminus K)$ by replacing e with e' in the set $K_i \setminus K$ and keep other choices the same. Now, either L' still contains e because some other $K_j \setminus K$ also selects e to compose L, or it does not. In the latter case, the minimal edge cut D is not a subset of L', so there are fewer minimal edge cuts, then by induction, it leads to a contradiction. In the former case, we repeat the previous argument for any set K_j that also contains edge e until we reach the desired contradiction. This procedure terminates since L only contains a finite number of edges. $\square$

Theorem 11 *The following statements about the poset $(\downarrow \Omega^r_G, \subseteq)$ are true:*

1. *The least element in $(\downarrow \Omega^r_G, \subseteq)$ is the empty set.*
2. *The maximal elements in $(\downarrow \Omega^r_G, \subseteq)$ are all the critical spanning trees.*
3. *$(\Lambda^r_G, \subseteq)$ is a subposet embedded in $(\downarrow \Omega^r_G, \subseteq)$.*

Proof. For part 1, the empty set is a least element since it is contained by all other sets; and it is known that in any poset, the least element is unique given it exists. Part 2 is trivial, since $\downarrow \Omega^r_G$ consists all the subsets of critical spanning trees. For part 3, every critical edge set by definition is a subset of some critical spanning tree, so it is embedded in the meet-semilattice $(\downarrow \Omega^r_G, \subseteq)$ and the ordering on the set Λ^r_G is naturally inherited from the poset. $\square$

Proposition 2 *Let β be the function from the ordered set $(\downarrow \Omega_G, \subseteq)$ to $(\mathbb{R}, \leq)$ defined as $\beta(K) = \max_{T \in \Omega_G[K]} w(T)$, then β is order reversing.*

Proof. If $K \subseteq K'$, then by definition $\Omega_G[K] \supseteq \Omega_G[K']$, so the maximum in a subset must be less or equal to the original set. $\square$

B Notation Table

Symbol	Definition
$G = (V, E)$	graph G with vertex set V and edge set E
n	number of vertices of given graph
m	number of edges of given graph
$V(H)$	vertices in subgraph H
$E(H)$	edges in subgraph H
$w(e)$	weight of edge e
$w(K)$	total weight of edges in edge set K
$w(H)$	total weight of edges in subgraph H
$\lambda(G)$	edge connectivity of graph G
$\lambda_{\mathbf{c}}(G)$	value of a min-cost edge cut of graph G with respect to cost vector $\mathbf{c}$
$\overline{w}(G)$	weight of a maximum spanning tree in G
$\underline{w}(G)$	weight of a minimum spanning tree in G
Ω_G	all spanning trees in graph G
$\Omega_G[e]$	all spanning trees in graph G that contains edge e
$\Omega_G[K]$	all spanning trees in graph G that contains edge set K
Ω_G^r	all spanning trees in graph G that have weight less than r, i.e., the set of all critical spanning trees
Λ_G^r	all critical edge sets in graph G with respect to r
$\tilde{\Lambda}_G^r$	all minimal (with respect to set inclusion) critical edge sets in Λ_G^r
$\downarrow \mathcal{S}$	the lower closure of $\mathcal{S}$, i.e., $\downarrow \mathcal{S} = \{K \subseteq S \mid \forall S \in \mathcal{S}\}$

References

Addis B, Summa MD, Grosso A (2013) Identifying critical nodes in undirected graphs: Complexity results and polynomial algorithms for the case of bounded treewidth. *Discrete Applied Mathematics* 161(16–17):2349–2360.

Aggarwal V, Aneja Y, Nair K (1982) Minimal spanning tree subject to a side constraint. *Computers & Operations Research* 9(4):287 – 296.

Ahuja RK, Magnanti TL, Orlin JB (1993) *Network Flows: Theory, Algorithms, and Applications* (New Jersey, USA: Prentice Hall).

Albert R, Barabási AL (2002) Statistical mechanics of complex networks. *Rev. Mod. Phys.* 74:47–97.

Arulselvan A, Commander CW, Elefteriadou L, Pardalos PM (2009) Detecting critical nodes in sparse graphs. *Computers and Operations Research* 36(7):2193–2200.

Baïou M, Barahona F (2008) A linear programming approach to increasing the weight of all minimum spanning trees. *Networks: An International Journal* 52(4):227–234.

Bazgan C, Toubaline S, Tuza Z (2011) The most vital nodes with respect to independent set and vertex cover. *Discrete Applied Mathematics* 159(17):1933–1946.

Bazgan C, Toubaline S, Vanderpooten D (2010) Complexity of determining the most vital elements for the 1-median and 1-center location problems. Wu W, Daescu O, eds., *Combinatorial Optimization and Applications*, volume 6508, 237–251 (Springer Berlin Heidelberg).

Bazgan C, Toubaline S, Vanderpooten D (2012) Efficient determination of the k most vital edges for the minimum spanning tree problem. *Computers & Operations Research* 39(11):2888–2898.

Bazgan C, Toubaline S, Vanderpooten D (2013) Critical edges/nodes for the minimum spanning tree problem: complexity and approximation. *Journal of Combinatorial Optimization* 26(1):178–189.

Ben-Ameur W, Mohamed-Sidi MA, Neto J (2015) The k-separator problem: polyhedra, complexity and approximation results. *Journal of Combinatorial Optimization* 29(1):276–307.

Brualdi RA (1969) Comments on bases in dependence structures. *Bulletin of the Australian Mathematical Society* 1(2):161–167.

Carr RD, Lancia G (2002) Compact vs. exponential-size lp relaxations. *Operations Research Letters* 30(1):57 – 65.

Chestnut SR, Zenklusen R (2017) Interdicting structured combinatorial optimization problems with 0, 1-objectives. *Mathematics of Operations Research* 42(1):144–166.

Chvátal V (1973) Edmonds polytopes and a hierarchy of combinatorial problems. *Discrete Mathematics* 4(4):305 – 337.

Chvátal V, Cook W (1990) The discipline number of a graph. *Discrete mathematics* 86(1-3):191–198.

Corley H, Sha DY (1982) Most vital links and nodes in weighted networks. *Operations Research Letters* 1(4):157–160.

Cormican KJ, Morton DP, Wood RK (1998) Stochastic network interdiction. *Operations Research* 46(2):184–197.

Di Summa M, Grosso A, Locatelli M (2012) Branch and cut algorithms for detecting critical nodes in undirected graphs. *Computational Optimization and Applications* 53(3):649–680.

Dinh TN, Xuan Y, Thai MT, Pardalos PM, Znati T (2012) On new approaches of assessing network vulnerability: Hardness and approximation. *IEEE/ACM Transactions on Networking* 20(2):609–619.

Erdös P, Rényi A (1959) On random graphs, I. *Publicationes Mathematicae (Debrecen)* 6:290–297.

Fischetti M, Ljubić I, Sinnl M (2017) Redesigning benders decomposition for large-scale facility location. *Management Science* 63(7):2146–2162.

Frederickson GN, Solis-Oba R (1999) Increasing the weight of minimum spanning trees. *Journal of Algorithms* 33(2):244–266.

Furini F, Ljubić I, Martin S, San Segundo P (2019) The maximum clique interdiction problem. *European Journal of Operational Research* 277(1):112 – 127.

Gabow HN, Galil Z, Spencer T, Tarjan RE (1986) Efficient algorithms for finding minimum spanning trees in undirected and directed graphs. *Combinatorica* 6(2):109–122.

Ghare P, Montgomery DC, Turner W (1971) Optimal interdiction policy for a flow network. *Naval Research Logistics Quarterly* 18(1):37–45.

Gilbert EN (1959) Random graphs. *The Annals of Mathematical Statistics* 30(4):1141–1144.

Gomory RE (1963) An algorithm for integer solutions to linear programs. Graves R, Wolfe P, eds., *Recent advances in mathematical programming*, volume 6508, 269–302 (McGrawHill, New York).

Gomory RE, Hu TC (1961) Multi-terminal network flows. *Journal of the Society for Industrial and Applied Mathematics* 9(4):551–570.

Graham RL, Hell P (1985) On the history of the minimum spanning tree problem. *Annals of the History of Computing* 7(1):43–57.

Grubesic TH, Matisziw TC, Murray AT, Snediker D (2008) Comparative approaches for assessing network vulnerability. *International Regional Science Review* 31(1):88–112.

Held M, Karp RM (1970) The traveling-salesman problem and minimum spanning trees. *Operations Research* 18(6):1138–1162.

Houck DJ, Kim E, O'Reilly GP, Picklesimer DD, Uzunalioglu H (2004) A network survivability model for critical national infrastructures. *Bell Labs Technical Journal* 8(4):153–172.

Hsu LH, Jan RH, Lee YC, Hung CN, Chern MS (1991) Finding the most vital edge with respect to minimum spanning tree in weighted graphs. *Information Processing Letters* 39(5):277–281.

Israeli E, Wood RK (2002) Shortest-path network interdiction. *Networks* 40(2):97–111.

Iwano K, Katoh N (1993) Efficient algorithms for finding the most vital edge of a minimum spanning tree. *Information Processing Letters* 48(5):211–213.

Jenelius E, Petersen T, Mattsson LG (2006) Importance and exposure in road network vulnerability analysis. *Transportation Research Part A: Policy and Practice* 40(7):537–560.

Katoh N, Ibaraki T, Mine H (1981) An algorithm for finding k minimum spanning trees. *SIAM Journal on Computing* 10(2):247–255.

Kennedy KT, Deckro RF, Moore JT, Hopkinson KM (2011) Nodal interdiction. *Mathematical and Computer Modelling* 54(11-12):3116–3125.

Liang W, Shen X (1997) Finding the k most vital edges in the minimum spanning tree problem. *Parallel Computing* 23(13):1889–1907.

Lim C, Smith JC (2007) Algorithms for discrete and continuous multicommodity flow network interdiction problems. *IIE Transactions* 39(1):15–26.

Lin KC, Chern MS (1993) The most vital edges in the minimum spanning tree problem. *Information Processing Letters* 45(1):25–31.

Linhares A, Swamy C (2017) Improved algorithms for MST and metric-tsp interdiction `http://arxiv.org/abs/1706.00034`. (Accessed Apr 03, 2019).

Lozano L, Smith JC (2016) A backward sampling framework for interdiction problems with fortification. *INFORMS Journal on Computing* 29(1):123–139.

Magnanti TL, Wolsey LA (1995) Optimal trees. *Handbooks in operations research and management science* 7:503–615.

Mahdavi Pajouh F, Boginski V, Pasiliao EL (2014) Minimum vertex blocker clique problem. *Networks* 64(1):48–64.

Mahdavi Pajouh F, Walteros JL, Boginski V, Pasiliao EL (2015) Minimum edge blocker dominating set problem. *European Journal of Operational Research* 247(1):16–26.

Matisziw TC, Murray AT (2009) Modeling *s*-*t* path availability to support disaster vulnerability assessment of network infrastructure. *Computers and Operations Research.* 36:16–26.

Nemhauser GL, Wolsey LA (1988) *Integer and Combinatorial Optimization* (New York, NY, USA: Wiley-Interscience).

Network Repository (2019) Network repository. `https://networkrepository.com/` (Accessed Apr 03, 2019).

Ohlsson H, Gustafsson O, Wanhammar L (2004) Implementation of low complexity fir filters using a minimum spanning tree. *Proceedings of the 12th IEEE Mediterranean Electrotechnical Conference*, volume 1, 261–264.

Oosten M, Rutten JHGC, Spieksma FCR (2007) Disconnecting graphs by removing vertices: a polyhedral approach. *Statistica Neerlandica* 61(1):35–60.

Oxley JG (2006) *Matroid theory*, volume 3 (Oxford University Press, USA).

Ravi R, Goemans MX (1996) The constrained minimum spanning tree problem. Karlsson R, Lingas A, eds., *Algorithm Theory — SWAT'96*, 66–75 (Berlin, Heidelberg: Springer Berlin Heidelberg).

Salmeron J, Wood KR, Baldick R (2004) Analysis of electric grid security under terrorist threat. *Power Systems, IEEE Transactions on* 19(2):905–912.

Shen H (1995) Improved parallel algorithms for the most vital edge of a graph with respect to minimum spanning tree. *Technical Report* (Dept. of Computer Sci., Abo Akademi Univ. Finland).

Shen S, Smith JC, Goli R (2012) Exact interdiction models and algorithms for disconnecting networks via node deletions. *Discrete Optimization* 9(3):172–188.

Ströele A Menezes V, Tadeu da Silva R, Ferreira de Souza M, Oliveira J, de Mello CER, Moreira de Souza J, Zimbrão G (2008) Mining and analyzing organizational social networks using minimum spanning tree. Meersman R, Tari Z, Herrero P, eds., *On the Move to Meaningful Internet Systems: OTM 2008 Workshops*, 18–19 (Berlin, Heidelberg: Springer Berlin Heidelberg).

Suk M, Song O (1984) Curvilinear feature extraction using minimum spanning trees. *Computer Vision, Graphics, and Image Processing* 26(3):400 – 411.

Suraweera F, Maheshwari P, Bhattacharya P (1995) Optimal algorithms to find the most vital edge of a minimum spanning tree. *Technical Report CIT-95-21* (School of Computing and Information Technology, Griffith University).

Tao Z, Zhongqian F, Binghong W (2005) Epidemic dynamics on complex networks. *Progress in Natural Science* 16(5):452–457.

Tapia E, Rojas R (2004) Recognition of on-line handwritten mathematical expressions using a minimum spanning tree construction and symbol dominance. Lladós J, Kwon YB, eds., *Graphics Recognition. Recent Advances and Perspectives*, 329–340 (Berlin, Heidelberg: Springer Berlin Heidelberg).

Tarjan RE (1979) A class of algorithms which require nonlinear time to maintain disjoint sets. *Journal of computer and system sciences* 18(2):110–127.

Tarjan RE, Van Leeuwen J (1984) Worst-case analysis of set union algorithms. *Journal of the ACM (JACM)* 31(2):245–281.

Veremyev A, Prokopyev OA, Pasiliao EL (2014) An integer programming framework for critical elements detection in graphs. *Journal of Combinatorial Optimization* 28(1):233–273.

Vielma JP (2015) Mixed integer linear programming formulation techniques. *Siam Review* 57(1):3–57.

Watts DJ, Strogatz SH (1998) Collective dynamics of 'small-world'networks. *nature* 393(6684):440.

Wollmer R (1964) Removing arcs from a network. *Operations Research* 12(6):934–940.

Wood RK (1993) Deterministic network interdiction. *Mathematical and Computer Modeling* 17(2):1–18.

Zenklusen R (2010) Matching interdiction. *Discrete Appl. Math.* 158(15):1676–1690.

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