Operations Research

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Multilevel Approaches for the Critical Node Problem

Andrea Baggio, Margarida Carvalho, Andrea Lodi, Andrea Tramontani

http://pubsonline.informs.org/journal/opre

OPERATIONS RESEARCH
Vol. 69, No. 2, March–April 2021, pp. 486–508
ISSN 0030-364X (print), ISSN 1526-5463 (online)

Contextual Areas

Multilevel Approaches for the Critical Node Problem

Andrea Baggio,[a] Margarida Carvalho,[b] Andrea Lodi,[c] Andrea Tramontani[d]

[a] Quant Research, swissQuant Group AG, CH-8001 Zurich, Switzerland; [b] CIRRELT and DIRO, Université de Montréal, Montréal, Québec H3T 1J4, Canada; [c] École Polytechnique de Montréal, Montreal, Québec H3T 1J4, Canada; [d] CPLEX Optimization, IBM, 40132 Bologna, Italy

Contact: baggio@swissquant.com (AB); carvalho@iro.umontreal.ca, https://orcid.org/0000-0002-2344-0960 (MC); andrea.lodi@polymtl.ca, https://orcid.org/0000-0001-9269-633X (AL); andrea.tramontani@it.ibm.com (AT)

Published Online in Articles in Advance:
January 19, 2021

Subject Classifications: games/group decisions; mathematics: combinatorics; networks/graphs; programming: integer: algorithms
Area of Review: Military and Homeland Security

https://doi.org/10.1287/opre.2020.2014

Copyright: © 2021 INFORMS

Abstract. In recent years, a lot of effort has been dedicated to develop strategies to defend networks against possible cascade failures or malicious viral attacks. On the one hand, network safety is investigated from a preventive perspective. On the other hand, blocking models have been proposed for scenarios in which the attack has already taken place causing a harmful spreading throughout the network. In this work, we combine these two perspectives. More precisely, following the framework defender–attacker–defender, we consider a model of prevention, attack, and damage containment using a three-stage, zero-sum game. The defender is not only able to adopt preventive strategies, but also to defend the network after an attack takes place. Assuming that the attacker acts optimally, we compute a defensive strategy for the first stage that minimizes the total damage to the network in the end of the third stage. Our contribution consists of considering this problem as a trilevel mixed-integer program and designing an exact algorithm for it based on tools developed for multilevel programming.

Funding: A. Baggio is thankful for support from the European Union [Grant FP7-PEOPLE-2012-ITN 316647] "Mixed-Integer Nonlinear Optimization" project. M. Carvalho is thankful for the support of the Institut de valorisation des données (IVADO) and Fonds de recherche du Québec (FRQ) through the FRQ-IVADO Research Chair, the ERDF European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation–COMPETE 2020 Programme [Project POCI-01-0145-FEDER-006961], and National Funds through the FCT (Portuguese Foundation for Science and Technology) [Project UID/EEA/50014/2013].

Keywords: multilevel programming • critical node problem • firefighter problem • mixed-integer programming • defender–attacker–defender

1. Introduction

Many important connectivity systems, such as communication or social networks, transportation systems, electrical grids, mobility networks, computer networks, etc., are susceptible to attacks, breakdowns, or epidemic phenomena, which may trigger a cascade of failures that propagate through the entire system. In the last decades, a lot of effort has been dedicated to develop strategies to defend networks against possible cascade failures or malicious viral attacks. The problem we consider in this paper is of the latter type. In particular, it aims at designing defensive strategies for networks that are threatened by well-planned attacks that trigger a spreading of failures. Moreover, we consider that the network operator can act preventively before an attack has taken place and also implement a protective reaction against the spread after the attack has been localized. The combination of these two strategic defenses results in a problem that can be interpreted as a defender–attacker–defender model as described by Brown et al. (2006). We call this problem the *multilevel critical node* problem (MCN), which we model by trilevel optimization.

Among the many contexts of application mentioned, we stress those in social networks and cybersecurity. In particular, in social networks, the spread of fake news has received a relevant amount of attention because it is associated with significant societal risks. This is the case of dangerous health decisions (about the use of vaccines), political misperceptions that might lead to influencing elections, or deceiving advertisement (Markines et al. 2009, Ratkiewicz et al. 2011), just to mention a few. Efforts to minimize the impact of misinformation are flourishing; see, for example, Shu et al. (2017) for a recent survey. The trilevel programming approach we propose can model the decision process of minimizing the number of nodes receiving the false information. More precisely, (1) the vaccination step, which can be viewed as topology design, concerns the selection (introduction) of a subset of nodes that are informed civilians; (2) the infection corresponds to the injection of fake (health, political, financial, etc.) news in a distinct subset of nodes; and (3) the protection is a direct information campaign targeted at a subset of nodes that does not share/spread the news. Our model is a preventive one: It assumes that any node has the potential to be the source of the fake news or to spread it.

The other context of application that we target is that of cybersecurity in which, in particular, we mention the protection of a botnet against malware injections. A botnet is a network of bots, that is, software applications that run automated tasks (scripts) over the internet. These networks are very attractive for the introduction of malware, that is, software intentionally designed to cause damage. An infected ("poisoned") network generally reduces its effectiveness, resulting in service denials, and the trilevel programming approach we propose minimizes the effect of such poisoning on the botnet. More precisely, (1) possible preventive (vaccination) strategies are the introduction of duplicated memory (backup) and/or addition of more cryptography, (2) the infection corresponds to the introduction of malware in (some of) the bots, and (3) a protection phase can be performed by disconnecting some nodes and, thus, stopping the spread of the malware.

1.1. Literature Review

Because we focus on identifying critical components in a network for a potential attack, this work falls into the framework of network interdiction problems. Both paradigms we combine in our model are special cases of this class of problems: by acting preventatively in parts of the network, they become immune to an attack, and by attacking parts of the network, they become interdict to protection. Network interdiction problems have been vastly investigated for decades and have found applications in many real-world problems, including controlling the diffusion of infections in hospitals (Assimakopoulos 1987), coordinating military strikes (Ghare et al. 1971), combating the traffic of illegal drugs (Wood 1993), protection of infrastructure systems (Church et al. 2004, Salmeron et al. 2009), flood control (Ratliff et al. 1975), and contrasting nuclear smuggling (Morton et al. 2007). Interdiction problems have been widely investigated for well-known combinatorial optimization problems as well. This includes natural interdiction versions of problems, such as the knapsack problem (Caprara et al. 2016), minimum spanning tree (Frederickson and Solis-Oba 1999, Zenklusen 2015), connectivity of a graph (Zenklusen 2014), packing (Dinitz and Gupta 2013), maximum matching (Zenklusen 2010a, Dinitz and Gupta 2013), independent set and vertex cover (Bazgan et al. 2011), maximum s-t flow (Phillips 1993, Wood 1993, Zenklusen 2010b; often known as *network flow interdiction*), shortest path (Ball et al. 1989, Khachiyan et al. 2008), and facility location (Church et al. 2004).

Interdiction problems are often modeled as mixed-integer bilevel programs (MIBPs). Analogously, we model the MCN problem as a mixed-integer trilevel program. In particular, the algorithm we devise is based on tools developed for bilevel programming. Solving MIBPs to optimality is generally a challenging task. Because of this and especially because MIBPs model many important real-world problems, a lot of effort has been dedicated to developing efficient algorithms for solving generic MIBPs in recent years. Common approaches try to exploit consolidated techniques known for solving mixed-integer linear programs (MILPs). These algorithms use as subroutines MILP models, and their implementations use modern MILP solvers. The first approach for solving MIBPs is presented in Moore and Bard (1990), in which the authors apply a branch-and-bound algorithm to a relaxed formulation. Later, DeNegre and Ralphs (2009) and DeNegre (2011) improve on these techniques by adding cuts based on integrality and feasibility properties. This approach is further improved by Caramia and Mari (2015) and Fischetti et al. (2016), in which the authors provide significantly better performances by introducing stronger cuts. Very recently, Fischetti et al. (2017) considerably extend the latter branch-and-bound through new families of cuts and effective preprocessing. Moreover, Hemmati and Smith (2016) propose a cutting plane for integer bilevel programming problems. A lot of effort is also dedicated to devising tailored approaches to solve particular special cases as it generally allows faster running times. For instance, Smith et al. (2007) tackle a bilevel program in which the defender designs a network and transmits a set of multicommodity flows in the network while the attacker decreases the capacity of some arcs; another bilevel program modeling a game between a defender and an attacker, the r-interdiction median problem with fortification, is addressed in Scaparra and Church (2008a, b); Hemmati et al. (2014) consider a bilevel influence interdiction problem in networks; and DeNegre (2011), Brotcorne et al. (2013), and Caprara et al. (2016) investigate interdiction versions of the knapsack problem. Finally, Fischetti et al. (2019) build a branch-and-cut algorithm for general interdiction problems and report computational results on some benchmark instances that show their approach to be faster than any previous method from the literature.

Our approach requires solving bilevel programs, attacker–defender, and thus, as literature findings suggest, we propose a model tailored to our specific interdiction problem.

There are also studies on trilevel programs. For example, a three-stage sequential game designed for modeling infrastructure defense is presented in Alderson et al. (2011). In particular, the authors describe a defend–attack–defend model for system resilience against an attack. They present a realistic formulation of this model for the case of a transportation network and design a decomposition algorithm for solving the resulting model. In Cappanera and Scaparra (2011), a trilevel interdiction program models a shortest-path interdiction problem with fortification; the defender's aim is to harden a subset of arcs by taking into account that a subset of unprotected arcs are interdicted; the lower level problem concerns the computation of the shortest path among the available arcs. The approach of Cappanera and Scaparra (2011) is outperformed by a more general framework proposed in Lozano and Smith (2017), in which a trilevel optimization problem is considered in which a defender fortifies a set of assets, the attacker interdicts a subset of unprotected assets, and finally the defender solves a recourse problem. The authors propose a backward sampling framework in which the key idea is the iterative sampling of recourse solutions so that only a restricted version of the problem, defined over a smaller search space, needs to be solved at each iteration. In contrast, our approach tackles a trilevel program by solving a relaxation of the problem that is iteratively refined by adding attacking scenarios. In other words, our approach iteratively increases the set of feasible attacks, and in Lozano and Smith (2017), the set of the defender's lower level feasible solutions is iteratively increased.

1.2. Our Contributions and Paper Organization

In Section 2, we define the multilevel critical node problem, and a trilevel mixed-integer program is proposed modeling a three-stage defender–attacker–defender zero-sum game. We also show that it can be seen as the combination of two previously known paradigms from network defense. The model is motivated by the real-world hypothesis that the network operator has two stages in which to act to protect the network, namely before and after the harmful propagation starts. In Section 3, we design an exact algorithm for solving the problem. The algorithm is based on recent tools developed for bilevel programming and on a simultaneous column- and row-generation approach. Our algorithm invokes as subroutine an algorithm that we devise for solving the bilevel problem in Section 3.1 defined by the second and third stages. Moreover, both the algorithm and the subroutine invoke a solver for MILPs as a black box. In particular, for the implementation of our algorithm, we use CPLEX 12.7.1 (IBM 2017) in order to solve MILPs. In Section 4, we perform computational experiments to analyze the performances of the developed algorithm. The results show that the optimal action, when we consider all three stages of the problem simultaneously, has a vastly better objective function value than the value obtained by sequentially optimizing the first and third steps separately. Such a gap, that we call *relative gain*, is also illustrated by our numerical experiments. Finally, we draw some conclusions and present new possible directions for future research in Section 5.

2. The Multilevel Critical Node Problem

In this section, we formally define the trilevel, sequential, zero-sum game that we consider in this paper. We start in the next section by stating few preliminary definitions, and we continue, in Section 2.2, with the problem definition. In Section 2.3, we introduce the mathematical formulations we propose for the problem. Sections 2.4 and 2.5 discuss the connections between the problem and its bilevel special cases and review the associated literature. Finally, Section 2.6 introduces the notion of relative gain, which highlights the advantages of solving our trilevel problem over a simpler approach consisting of solving its two bilevel special cases separately.

2.1. Preliminaries

We are given a directed graph $G = (V, A)$ in which V and A denote the nodes and the arcs, respectively. Nodes V are susceptible to viral attacks, which trigger a cascade of infections that propagate through arcs from node to node. Moreover, we assume the presence of two agents that act on G. An *attacker*, which can choose a subset of nodes to *attack*, and a *defender*, whose goal consists of limiting the spread of infection by choosing subsets of nodes to *vaccinate* and *protect* before and after the attack takes place, respectively. For any action of vaccination, attack, and protection, we consider a budget limiting the number of nodes that the respective agent can choose.

Our main interest consists of developing a new network protection model combining, in a unique, three-stage, sequential, zero-sum game, two different well-studied paradigms. The first paradigm investigates strategies that prevent against possible viral attacks; the second paradigm isolates the viral propagation after an attack.

In this work, we assume *perfect information* and *rationality* for the game. In other words, both agents have perfect knowledge of the graph topology, viral dynamics, and adversary's actions already taken. Moreover, the defender and attacker act optimally, according to their respective goals and to the complete knowledge of the system.

2.2. Problem Definition

Next, we introduce each of the three stages (levels) of the problem tackled in this paper. The problem is described in a nonchronological order starting with the last optimization and going backward until the beginning.

In the third stage, for a given set of attacked nodes I, the goal of the defender is to find a *protection strategy* $P \subseteq V \setminus I$

that maximizes the weight $w(S(I,P))$ of the set $S(I,P)$ of saved nodes. This value strongly depends on the set of nodes I that the attacker chooses to infect. Given the attacker's full knowledge of the graph topology and of the defender's budget $\Lambda \in \mathbb{Z}_{\geq 0}$ for protection in the third stage, the attacker, thus, chooses the nodes that mostly affect the network after the best blocking reaction is used, that is, after the defender's protection strategy. Moreover, the attack's intensity is limited by a budget constraint. Let $\Phi \in \mathbb{Z}_{\geq 0}$ be the attacker's budget, let $h(v) \in \mathbb{Z}_{\geq 0}$ for every $v \in V$ be the cost incurred for infecting v, and let $c(P) \in \mathbb{Z}_{\geq 0}$ be the cost incurred by the defender to protect P. The attacker target is modeled by the following bilevel problem:

$$\min_{\substack{I \subseteq V \\ h(I) \leq \Phi}} \max_{\substack{P \subseteq V \setminus I \\ c(P) \leq \Lambda}} w(S(I,P)). \tag{AP}$$

In the first stage, instead, the defender precedes the attacker by vaccinating a subset of nodes $D \subseteq V$, whose cost does not exceed a budget $\Omega \in \mathbb{N}$. In this stage, the goal of the defender is to vaccinate a subset of nodes D such that, in the worst scenario, considering every possible attack together with its best protective response, the total weight of the saved network is maximized. Formally, the goal of the defender is to find the set $D \subseteq V$ that solves the following problem:

$$\max_{\substack{D \subseteq V \\ c(D) \leq \Omega}} \min_{\substack{I \subseteq V \setminus D \\ h(I) \leq \Phi}} \max_{\substack{P \subseteq V \setminus (I \cup D) \\ c(P) \leq \Lambda}} w(S(I, D \cup P)). \tag{MCN}$$

We indicate the preceding problem as the MCN problem. Notice that, once a node is selected in any of the three stages, it becomes interdicted to the forthcoming stages. Thus, (MCN) has two players: a defender and an attacker. The defender plays in the first and last (third) stages by vaccinating and protecting subsets of nodes, respectively, and the attacker plays in the second stage by infecting a subset of nonvaccinated nodes.

Although this definition refers to general budget constraints, in the following we concentrate on the cardinality case; that is, each player has a budget in terms of number of nodes on which the player can act. Therefore, the objective function is also a cardinality one.

2.3. Mathematical Models for the MCN

We start by modeling the problem as a trilevel program. Let us consider the following indicator variables that model all the possible different states for a node $v \in V$:

$$z_v = \begin{cases} 1 & \text{if } v \text{ is vaccinated} \\ 0 & \text{otherwise} \end{cases}$$

$$x_v = \begin{cases} 1 & \text{if } v \text{ is protected} \\ 0 & \text{otherwise} \end{cases}$$

$$y_v = \begin{cases} 1 & \text{if } v \text{ is attacked} \\ 0 & \text{otherwise} \end{cases}$$

$$\alpha_v = \begin{cases} 1 & \text{if } v \text{ is saved} \\ 0 & \text{otherwise} \end{cases}$$

Note that a node $v \in V \setminus I$ is saved if it has been vaccinated, protected, or all of its adjacent nodes are saved. In this way, it is possible to determine if a node is saved by only restricting it to constraints that take into account the state of adjacent nodes. We can then formulate the MCN problem as the following mixed-integer trilevel program:

$$\begin{aligned} \max_{\substack{z \in \{0,1\}^V \\ \sum_{v \in V} z_v \leq \Omega}} \min_{\substack{y \in \{0,1\}^V \\ \sum_{v \in V} y_v \leq \Phi}} \max_{\substack{x \in \{0,1\}^V \\ \alpha \in [0,1]^V}} & \sum_{v \in V} \alpha_v \\ & \sum_{v \in V} x_v \leq \Lambda \\ & \alpha_v \leq 1 + z_v - y_v \quad \forall v \in V \\ & \alpha_v \leq \alpha_u + x_v + z_v \, \forall \, (u,v) \in A. \end{aligned} \tag{3lvMIP}$$

It is not difficult to see that (3lvMIP) models exactly the MCN problem. In the first stage, a subset of nodes of size at most Ω is vaccinated; in the second stage, a subset of nodes of size at most Φ is attacked; and in the third stage, a subset of nodes of size at most Λ is protected. Note that the third stage properly determines the saved nodes α: a node is saved if it is vaccinated, protected, or all of its adjacent nodes are saved. We remark that, in the second stage, there is no need to make a vaccinated node v unavailable for the attacker because, if $z_v = 1$, the third-stage constraint $\alpha_v \leq 1 + z_v - y_v$, together with the objective function, enforces α_v to be one, that is, saved. Furthermore, we also remark that requiring α to be binary is not necessary because this is already enforced by the model.

Bilevel and trilevel programs are, in general, very difficult problems. The feasible region for the optimization problem at the first level in (3lvMIP) is determined by a budget constraint and by an inner bilevel problem, which makes its feasible region extremely difficult to describe. For the particular case we are studying, notice that the three levels share the same objective function up to the direction of optimization. This allows (3lvMIP) to be equivalently reformulated as a single-level program as follows. Let $\mathcal{U} = \{y \in \{0,1\}^V \mid \sum_{v \in V} y_v \leq \Phi\}$ be the set of all possible feasible attack strategies. With respect to the given definition of x and α, we denote by $x(y)$ and $\alpha(y)$, respectively, the binary variables for the protected and saved nodes in response to the attack scenario $y \in \mathcal{U}$. For any feasible attack $y \in \mathcal{U}$, we have the two indicator vectors $x(y), \alpha(y) \in \{0,1\}^V$. Thus, (3lvMIP) is equivalent to the following MILP:

$$\begin{aligned}
\max \quad & \Delta \\
& \Delta \le \sum_{v\in V} \alpha_v(y) && \forall y \in \mathcal{U} \\
& \sum_{v\in V} z_v \le \Omega \\
& \sum_{v\in V} x_v(y) \le \Lambda && \forall y \in \mathcal{U} \\
& \alpha_v(y) \le 1 + z_v - y_v && \forall v \in V,\ \forall y \in \mathcal{U} \\
& \alpha_v(y) \le \alpha_u(y) + x_v(y) + z_v && \forall\ (u,v) \in A,\ \forall y \in \mathcal{U} \\
& 0 \le \alpha_v(y) \le 1 && \forall v \in V,\ \forall y \in \mathcal{U} \\
& z \in \{0,1\}^V \\
& x(y) \in \{0,1\}^V && \forall y \in \mathcal{U}. \qquad \text{(1lvMIP)}
\end{aligned}$$

Clearly, every element of $\mathcal{U}$ contributes to the size of the model $\Theta(|V|)$ variables and $\Theta(|V| + |A|)$ constraints. Because the size of $\mathcal{U}$ is $\Omega(|V|^{\Phi})$, the size of (1lvMIP) gets prohibitively large for efficient computations on any nontrivial graph. We show in Section 3 that only a subset of the scenarios $\mathcal{U}$ have to be algorithmically selected to effectively solve (1lvMIP).

As anticipated, model (3lvMIP) can be seen as the combination of two well-studied paradigms aiming at preventing and protecting from attacks, respectively. The next two sections review those paradigms so as to describe and put into perspective the associated literature.

2.4. Preventive Strategies (Vaccination)

An agent defends a network $G = (V, A)$ against a possible viral attack on nodes chosen by an attacker. The agent has resources to vaccinate some subset of nodes in V. A vaccinated node is immune from the attack and infection. Because we assume the attacker to be an intelligent agent, the defender has to identify a subset of nodes whose removal transforms the original graph into one whose resilience against an attack is maximized. For a given network, measuring its resilience against a possible attack depends on the graph topology, the dynamics of propagation of the virus, and the resources available for the attacker. For these reasons, there might not be a clear measure of how to evaluate network survivability against attacks. In the literature, many different approaches are suggested in order to understand how to identify those structures—both nodes and arcs—whose removal reduces attack effectiveness. In general, these approaches ask for removing *critical components* so as to decrease some given graph metric, which measures the resilience of the network against attacks, which depend on the attacker's resources and virus dynamics.

All problems that address identification of a subset of nodes whose removal maximally decreases some given graph metric fall under the category of critical node problems (CNPs). In a fairly general formulation, they can be described as follows. Let $\mathcal{M} : G \to \mathbb{Z}_{\ge 0}$ be a function that maps any graph to a real value, which captures the graph's resilience against viral attacks. For example, $\mathcal{M}(G)$ might represent the pairwise connectivity of G (see Di Summa et al. 2011, Addis et al. 2013), the number of disconnected components, or the size of the biggest connected component (see Shen and Smith 2012, Shen et al. 2012), and so on. Let $c(v) \in \mathbb{Z}_{\ge 0}$, for every $v \in V$, be the vaccination cost associated with v. The critical node problem asks to identify a subset of nodes $D \subseteq V$, the cost of which does not exceed the budget Ω, that is, $c(D) \le \Omega$, for which the subgraph obtained after removing D, which is $G[V \setminus D]$, minimizes $\mathcal{M}(G[V \setminus D])$. In compact form, the problem reads

$$\begin{aligned}
\min / \max\ & \mathcal{M}(G[V \setminus D]) \\
& c(D) \le \Omega \\
& D \subseteq V, \qquad \text{(CNP)}
\end{aligned}$$

where the direction of optimization is chosen according to the given metric.

These definitions state the prevention paradigm in its general form in an attempt at reviewing the literature in the area. As already discussed in Sections 2.2 and 2.3, in this paper, we consider the special case of unitary costs; that is, we have a budget on the cardinality of D. Furthermore, for the specific CNP modeled by (3lvMIP), $\mathcal{M}$ is a parametric function representing the number of saved nodes (to be maximized) after a trilevel sequence defend–attack–defend.

2.5. Blocking Strategies (Protection)

Another important type of network protection is based on the assumption that it is possible to detect viral attacks, and the dynamics of propagation allow for a prompt reaction, which limits their spread. In this case, we assume that the attack has already taken place. We denote by $I \subseteq V$ the subset of nodes that have been attacked. According to the viral propagation model and the resources available for the defender, the goal consists of finding a feasible blocking strategy for which, at the end of the spreading process, the impact of the infection is minimized.

The firefighter problem is a natural example of this class of problems (Hartnell 1995, Finbow and MacGillivray 2009). In this case, the infection propagates through the network from a node to any adjacent node at every time step. The defender knows which node or set of nodes has been attacked at an initial time step, and at any following time point, the defender is allowed to protect $B \in \mathbb{Z}_{>0}$ not yet infected nodes so as to maximize the number of not-infected nodes at the end of the process.

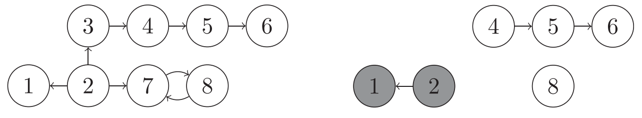
Another example of network protection against detected attacks can be found in the context of the *unbalanced cuts* problem and, most relevantly to our framework, the vertex-separator version. In general, an s-t cut can be seen as a tool to design a protection strategy to defend a sink t from a viral source s by deleting a subset of arcs. In Hayrapetyan et al. (2005), the authors introduce the *minimum-size, bounded-capacity, vertex-cut* problem, which asks for finding a subset of nodes whose total weight is at most $\Lambda \in \mathbb{Z}_{\geq 0}$, whose removal disconnect s and t, and the size of the connected component containing s is minimized. The same problem has also been described and addressed in Fomin et al. (2013). The problem we consider for the protection stage is a generalization of the minimum-size, bounded-capacity, vertex-cut problem in which more than one node can be infected. This generalization is also known as the *fence* problem, see Klein et al. (2014).

Recall that I represents the set of nodes that have been infected, and let again $c(v) \in \mathbb{Z}_{\geq 0}$ for every $v \in V$ be the cost incurred by protecting v. We assume the defender has a budget $\Lambda \in \mathbb{Z}_{\geq 0}$ for protecting not-yet-infected nodes. Given a set of *protected* nodes $P \subseteq V \setminus I$, we say that $v \in V$ is *saved* from I by P if either v is in P or there is no path in $G[V \setminus P]$ from any of the nodes in I to v. Thus, given a set of infected nodes I and a set of protected nodes P, the set of saved nodes is calculated as the union of P and of all the connected components in graph $G[V \setminus P]$ that do not contain nodes of I. Denote by $S(I,P) \subseteq V$ the set of nodes saved from I by P, and let $w(v) \in \mathbb{Z}_{\geq 0}$ be the reward obtained for saving each node $v \in V$. Given the set of attacked nodes I, the fence problem reads as follows (see Figure 1 for an illustration):

$$\max_{\substack{P \subseteq V \setminus I \\ c(P) \leq \Lambda}} \; w(S(I,P)). \tag{Fe}$$

Notice that the fence problem can be seen as a variant of the firefighter problem in which the budget Λ is given fully at time 1, and no protection is allowed during the next time steps. The dynamics of propagation indeed make the saved nodes at the end of the process coincide with the saved set of nodes, leading to the same objective.

Figure 1. Fence Problem Example



Notes. Consider the instance on the left. If $I = \{2\}$, $\Lambda = 2$, $c(v)$, and $w(v)$ unitary for all $v = 1,\ldots,8$, the optimal fence solution has $P = \{3,7\}$. The connected components of $G[V \setminus P]$ are represented on the right. Thus, six nodes are saved with $S(I,P) = P \cup \{4,5,6,8\}$.

2.6. Measuring Three-Stage Approach Effectiveness

In order to check the effectiveness of our trilevel model with respect to an approach that considers preventive and blocking strategies separately, we introduce the notion of relative gain. Concretely, for any given instance, we define the relative gain as

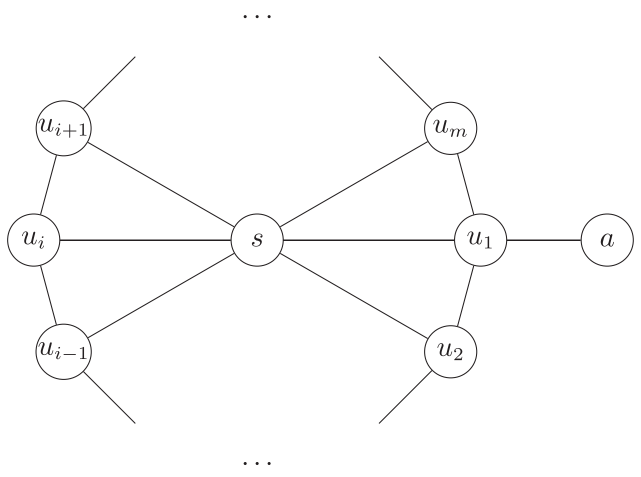
$$relGain = \frac{OPT - VA\_AP}{OPT}, \tag{1}$$

where OPT is the optimal value determined for MCN and VA_AP is the value of the solution obtained by separating the three stages into two steps (vaccinate–attack and attack–protect) and solving them to optimality sequentially. More precisely, for the evaluation of VA_AP, we first determine the best vaccination strategy for the three-stage problem in the case that no budget for protection is given, that is, $\Lambda = 0$, and then we look for an optimal solution for the resulting (AP) problem obtained by removing the vaccinated set of nodes found before. Notice that, for the computation of VA_AP, the infected sets in the two steps do not need to coincide. Clearly, $OPT \geq VA\_AP$ always holds, so *relGain* indeed represents the relative gain obtained by computing all three stages.

Next, we show that *relGain* can be arbitrarily close to one, revealing the advantage of considering the trilevel model. Let $m \in \mathbb{Z}_{\geq 5}$, and consider an undirected graph $G = (V,E)$ as illustrated in Figure 2, in which the edges represent the existence of arcs in both directions. Assume unitary cost and weight for each node, that is, $c(v) = h(v) = w(v) = 1$ $\forall v \in V$, and $\Omega = 1$, $\Phi = 1$, and $\Lambda = 2$.

On the one hand, the optimal solution for MCN is characterized by the vaccination strategy $\{s\}$ whose corresponding value OPT is m. On the other hand, assuming $\Lambda = 0$, MCN can be reduced to the problem

Figure 2. MCN Example



Notes. An Example for Constructing an Instance Whose *relGain* Is Arbitrarily Close to One.

that asks for finding a node whose removal minimizes the size of the biggest connected component. The optimal solution corresponds to select u_1 because it is the only option that disconnects the graph into two components. Once u_1 gets vaccinated, the best possible attacking strategy for (AP) corresponds to strike $\{s\}$. Then, any selection of two nodes among $\{u_2, \dots, u_m\}$ is equivalent, leading to a value $VA\_AP = 4$. Finally, notice that $relGain = \frac{m-4}{m}$, which can be arbitrarily close to one for a large enough m.

3. Solving the Multilevel Critical Node Problem

In this section, we describe an approach to solve the MCN problem based on tools developed for bilevel programming. Specifically, we work with model (1lvMIP), whose size, as already observed in Section 2.3, gets prohibitively large for efficient computations on any nontrivial graph. Therefore, it is necessary to find a subset of attack strategies $\overline{\mathcal{Q}} \subseteq \mathcal{U}$ for which (1lvMIP$_{\overline{\mathcal{Q}}}$), that is, (1lvMIP) restricted to $\overline{\mathcal{Q}}$, is solvable in practice but still returns an optimal solution for (1lvMIP). To this end, note that, $\forall \mathcal{Q} \subseteq \mathcal{U}$, (1lvMIP$_{\mathcal{Q}}$) is a relaxation of (1lvMIP) (see Lemma A.1). Thus, a viable approach to construct $\overline{\mathcal{Q}}$ and implicitly solve (1lvMIP) to optimality consists of repetitively solving (1lvMIP$_{\mathcal{Q}}$), starting from the empty set $\mathcal{Q} = \emptyset$ and adding a new attack strategy $y \in \mathcal{U}$ to $\mathcal{Q}$ each time until a guarantee of optimality is reached. This can be interpreted as a simultaneous column- and row-generation approach, and it is formalized in the high-level algorithm RowColGen, which is guaranteed to return an optimal solution for (1lvMIP) (see proof in the appendix).

RowColGen (Solving (1lvMIP) by Row and Column Generation)

1. Initialize an empty set of scenarios $\mathcal{Q} = \emptyset$.
2. Solve (1lvMIP$_{\mathcal{Q}}$) (this gives an upper bound). Let *best* and z^{best} be the optimal value and the z-part of an optimal solution, respectively.
3. If there is any $y \in \mathcal{U}$ such that solving the third stage (protection) with initial vaccination z^{best} and attack y yields a number of saved nodes strictly smaller than *best*, then add y to $\mathcal{Q}$ and go to step 2; otherwise, return *best* and z^{best}.

Column- and row-generation approaches have been extensively studied in the literature (see, for example, Frangioni and Gendron 2009, Muter et al. 2013), and they are commonly used for trilevel programming problems of the type defense–attack–defense (Alderson et al. 2011). However, the way step 3 is executed varies in the literature. The vaccination of a subset of nodes $V^{z^{best}} = \{v \in V \mid z_v^{best} = 1\}$ at the first level corresponds to the removal of these nodes from the original graph for the next levels. For instance, once a node is saved in the first stage, it represents a disconnection through which the attack cannot propagate. Thus, step 3 requires solving the attack–protect problem (AP) restricted to the subgraph $G[V \setminus V^{z^{best}}]$, which is a bilevel program consisting of the second and third levels in (3lvMIP). By relaxing integrality requirements on the third level variables x, the corresponding relaxation of (AP) can be reformulated as a MILP by applying strong duality and exploiting the McCormick convex relaxation (see McCormick 1976) to linearize the bilinear terms. An optimal value for this MILP provides an upper bound on the best value for the original attack–protect problem. Duality-based reformulations for general bilevel problems in which the follower's problem is linear have been explored in the literature (see Bard and Moore 1990, Hansen et al. 1992) as well as in particular contexts, such as the shortest-path intediction in Israeli and Wood (2002) and the flow network interdiction in Lim and Smith (2007). In the next section, we see how to solve the attack–protect problem and, in particular, how to obtain the aforementioned MILP, which we call rlxAP.

At this point, it is important to mention that, because an optimal solution to rlxAP returns an upper bound on the (AP) problem, in the event that this upper bound is smaller than *best*, the corresponding attack gives a scenario that can be added to $\mathcal{Q}$. On the other hand, if this upper bound is greater than *best*, no conclusion can be drawn, and it is necessary to proceed further in the exploration of the possible attack strategies for (AP) restricted to $G[V \setminus V^{z^{best}}]$. In particular, we are looking for an attack strategy for the (AP) problem such that either its value is smaller than *best* and, thus, it can be added to $\mathcal{Q}$ or it is optimal for (AP) and its value is equal to *best*; thus, $V^{z^{best}}$ is optimal for MCN. This implies that, at least once in the whole algorithm to solve (1lvMIP), there exists an (AP) instance restricted to some $G[V \setminus V^{z^{best}}]$ that has to be solved to optimality, and such $V^{z^{best}}$ corresponds to an optimal strategy for MCN. For all the other strategies that are not optimal for MCN, we only require finding a strategy for (AP) whose value is smaller than *best*.

We conclude this part by describing the full algorithm for solving the MCN problem, and we proceed in the following section by presenting the subroutine used for the (AP) problem, AP$(V, A, \Phi, \Lambda, target)$. We anticipate that AP$(V, A, \Phi, \Lambda, target)$ receives as input, together with the budget variables and the value *target*, the induced subgraph $G[V \setminus D]$ in which the vaccinated nodes are removed and returns an attacking strategy I and a status flag for that solution that depends on *target*. The value *target* represents the quality that an attacking strategy should outmatch in order to be output without proof of optimality. In particular, *target* represents the value *best*. However, because (AP) works on the induced subgraph $G[V \setminus D]$, *target* cannot include the already saved nodes D, namely $target = best - |D|$. In case *target* is smaller than or equal to the optimal value for (AP) and, hence, *target* is a lower bound of the optimal value, $\mathrm{AP}(V, A, \Phi, \Lambda, target)$ runs until it finds an optimal strategy I, and it guarantees optimality for it. The status flag stores the information about these two possible states for the returned solution I. In case the value of the strategy is strictly smaller than *target*, the corresponding status is "goal." Otherwise, if the value of the strategy is greater than or equal to *target*, the corresponding status is "opt" because it has to be optimal as already discussed. In the following, let $V_D = V \setminus D$ and $A_D = A[G[V_D]]$ be the set of nodes and arcs of the subgraph induced by the removal of D, respectively.

```
MCN($V, A, \Omega, \Phi, \Lambda$) Solving the Multilevel Critical Node Problem
  $\mathcal{Q} \leftarrow \emptyset$, $best \leftarrow |V|$, $D \leftarrow \emptyset$. % start with no attacking
        scenario, all nodes are saved.
  While True: % iteratively generate $y \in \mathcal{U} \setminus \mathcal{Q}$.
    $target \leftarrow best - |D|$. % remove vaccinated nodes in
        the attack–protect problem.
    $(I, status) \leftarrow \mathrm{AP}(V_D, A_D, \Phi, \Lambda, target)$. % find an
        attack that results in less than target nodes saved.
    If status = "opt": % if there is no attack that
        results in less than target nodes saved,
        then D is optimal.
          Return $(D, best)$.
    If status = "goal": % if there is an attack that
        results in less than target nodes saved, then
        add it to $\mathcal{Q}$.
      $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{I\}$.
      $(D = V^{z^{best}}, best) \leftarrow$ solve ($1lvMIP_{\mathcal{Q}}$).
```

3.1. Solving the Attack–Protect Problem

A bilevel formulation for the attack–protect problem (AP) is modeled by the second and third levels in 3lvMIP:

$$\begin{aligned}
\min_{\substack{y \in \{0,1\}^V \\ \sum_{v \in V} y_v \le \Phi}} \quad \max_{\substack{x \in \{0,1\}^V \\ \alpha \in [0,1]^V}} \quad & \sum_{v \in V} \alpha_v \\
& \sum_{v \in V} x_v \le \Lambda \\
& \alpha_v \le 1 - y_v \qquad \forall v \in V \\
& \alpha_v \le \alpha_u + x_v \qquad \forall\, (u,v) \in A.
\end{aligned} \tag{2lvAP}$$

This problem can be solved using an approach similar to the one seen in Caprara et al. (2016). More precisely, by relaxing the integrality requirement of the third-level variables, we can substitute the third level, which corresponds to a relaxed fence problem

$$\begin{aligned}
\max \quad & \sum_{v \in V} \alpha_v \\
& \sum_{v \in V} x_v \le \Lambda \\
& \alpha_v \le 1 - y_v \qquad \forall v \in V \\
& \alpha_v - \alpha_u - x_v \le 0 \qquad \forall\, (u,v) \in A \\
& \alpha_v, x_v \ge 0 \qquad \forall v \in V,
\end{aligned} \tag{Primal}$$

with its corresponding dual problem (see Dempe and Zemkoho 2013)

$$\begin{aligned}
\min \quad & \Lambda p + \sum_{v \in V} (1 - y_v) h_v \\
& h_v + \sum_{(u,v) \in A} q_{(u,v)} - \sum_{(v,u) \in A} q_{(v,u)} \ge 1 \qquad \forall v \in V \\
& p - \sum_{(u,v) \in A} q_{(u,v)} \ge 0 \qquad \forall v \in V \\
& p,\, h_v,\, q_{(u,v)} \ge 0 \qquad \forall v \in V, \\
& \qquad \forall\, (u,v) \in A.
\end{aligned} \tag{Dual}$$

Because, by strong duality, for a primal–dual optimal solution, $\sum_{v \in V} \alpha_v = \Lambda p + \sum_{v \in V} (1 - y_v) h_v$ holds, after relaxing the third-level variables, (2lvAP) can be rewritten as

$$\begin{aligned}
\min \quad & \Lambda p + \sum_{v \in V} (1 - y_v) h_v \\
& \sum_{v \in V} y_v \le \Phi \\
& h_v + \sum_{(u,v) \in A} q_{(u,v)} - \sum_{(v,u) \in A} q_{(v,u)} \ge 1 \qquad \forall v \in V \\
& p - \sum_{(u,v) \in A} q_{(u,v)} \ge 0 \qquad \forall v \in V \\
& p,\, h_v,\, q_{(u,v)} \ge 0 \qquad \forall v \in V, \forall\, (u,v) \in A \\
& y_v \in \{0,1\} \qquad \forall v \in V.
\end{aligned}$$

This mathematical program can be linearized to a MILP exploiting the McCormick envelope (see McCormick 1976) for the bilinear terms $(1 - y_v) h_v$. More precisely, we introduce the auxiliary variables γ_v so that $\gamma_v = (1 - y_v) h_v$ for every $v \in V$. Because we deal with a minimization problem, it suffices to consider the two underestimators $\gamma_v \ge 0$ and $\gamma_v \ge h_v - |V| y_v$, for which we use the valid upper bound $h_v \le |V|$. The latter upper bound follows by summing all the inequalities $h_v + \sum_{(u,v) \in A} q_{(u,v)} - \sum_{(v,u) \in A} q_{(v,u)} = 1$ for all nodes $v \in V$ obtaining $\sum_{v \in V} h_v = |V|$ because variables $q_{(u,v)}$ cancel out for all $(u, v) \in A$. Finally, applying the latter linearization, we obtain the following MILP:

$$\begin{aligned}
\min \quad & \Lambda p + \sum_{v \in V} \gamma_v \\
& \sum_{v \in V} y_v \le \Phi \\
h_v + \sum_{(u,v)\in A} q_{(u,v)} - \sum_{(v,u)\in A} q_{(v,u)} &\ge 1 \qquad \forall v \in V \\
p - \sum_{(u,v)\in A} q_{(u,v)} &\ge 0 \qquad \forall v \in V \\
\gamma_v + |V| y_v - h_v &\ge 0 \qquad \forall v \in V \\
p,\ h_v,\ \gamma_v,\ q_{(u,v)} &\ge 0 \qquad \forall v \in V, \\
& \qquad\qquad \forall\ (u,v) \in A \\
y_v &\in \{0,1\} \quad \forall v \in V.
\end{aligned} \tag{rlxAP}$$

The optimal objective value of (rlxAP) provides only an upper bound to (2lvAP). This is because the relaxation of the third level makes the defense response more effective against any possible attack. Moreover, even in the case that the solution is integral, there is no guarantee that it is also an optimal solution for (2lvAP) as already noted in Caprara et al. (2016). The actual quality of the optimal solution y given by (rlxAP) can be estimated by solving the third level (fence problem) once the attack strategy V^y is fixed. The optimal saved region obtained by the defensive strategy, that is, the set of saved nodes, characterizes all the possible nodes that a potential better attack should take into account. More precisely, given any set of attacked nodes $I \subseteq V$ and a corresponding best defensive strategy $P \subseteq V$ corresponding to an optimal solution to the fence problem with infected set I, consider the set of saved nodes $S \subseteq V$. Notice that $P \subseteq S$ and $I \cap S = \emptyset$. Any potentially more effective attack I_+ has to satisfy $I_+ \cap S \neq \emptyset$. This is because, if $I_+ \cap S = \emptyset$, the strategy $P \subseteq S$ would save at least all the nodes in S for the given attack I_+.

The previous consideration guarantees that, if an optimal solution y for (rlxAP) does not coincide with an optimal attack y_o for (2lvAP), the cut

$$\sum_{v \in S} y_v \ge 1, \tag{cutAP}$$

does not eliminate y_o from the feasible region when added to (rlxAP), and on the other hand, it cuts off the current solution y. By the preceding arguments, the following procedure represents a possible approach to finding an optimal solution to (2lvAP).

1. Solve (rlxAP). Let y be the indicator vector for the attack's strategy in an optimal solution.
2. Given the attack y, solve the corresponding fence problem and find the optimal saved set S_y.
3. Add (cutAP) corresponding to S_y to (rlxAP). If (rlxAP) is feasible, go to step 1; otherwise, return V^y and the corresponding protection strategy for the solution obtained in step 2.

When we say that we add a cut (cutAP) to (rlxAP), we implicitly assume we are modifying (rlxAP). In particular, the number of constraints in (rlxAP) increases after any iteration of the procedure.

As mentioned before, the algorithm we are developing for the MCN problem does not always require finding an optimal solution from the subroutine that solves (AP). Instead, the value of a returned solution for (2lvAP) may just need to be strictly smaller than a given bound. Recall that this bound *best* is evaluated by the main algorithm MCN$(V, A, \Omega, \Phi, \Lambda)$ and given as input $target = best - |D|$ to the subroutine solving (2lvAP). It represents the critical threshold that the value of an attack has to overtake in order to be added to $\mathcal{Q}$. Thus, we include two control flow statements in steps 1 and 2 of the high-level procedure described that interrupt the algorithm when the quality of the strategy found fulfills this condition and return the strategy. Moreover, MCN$(V, A, \Omega, \Phi, \Lambda)$ requires knowing whether the strategy returned by the subroutine has been output because of optimality or because it attains the required quality. As we say in the previous section, we store this information as a status flag whether optimal (opt) or outmatching the required bound (goal).

We conclude this section by showing the subroutine for (AP) in algorithm AP$(V, A, \Phi, \Lambda, target)$. Moreover, we denote by P(V, A, I, Λ) the subroutine that, given a graph (V, A), the set of infected nodes I, and the protection budget Λ, returns the set of nodes that are saved by the optimal protection strategy for the corresponding fence problem. Concretely, P(V, A, I, Λ) is the MILP modeled in the lower level of (2lvAP) with y fixed accordingly to the set I, and it can be solved by any MILP solver.

AP$(V, A, \Phi, \Lambda, target)$ Subroutine for (AP)

```
$best \leftarrow |V|$, $I^{best} \leftarrow \emptyset$.
While (rlxAP) is feasible:
    $(I = \chi^y, value) \leftarrow$ solve (rlxAP).
    If $value \le target - 1$:
        Return $(I, \text{"goal"})$.
    $S \leftarrow P(V, A, I, \Lambda)$.
    If $|S| \le target - 1$:
        Return $(I, \text{"goal"})$.
    If $|S| < best$:
        $best \leftarrow |S|$, $I^{best} \leftarrow I$.
    Add $\sum_{v \in S} y_v \ge 1$ to (rlxAP).
Return $(I^{best}, \text{"opt"})$.
```

4. Computational Results

In this section, we present some computational results for the algorithm we designed for the MCN problem. We focus on undirected graphs $G = (V, E)$, in which an edge represents the existence of arcs in both directions because the undirected case seems to be more relevant to our problem and represents the worst case scenario because it makes the spread of the infection easier. The experiments investigate both the performance of our framework in terms of running time and iterations and the relative gain as defined in (1), which measures the extent to which the trilevel approach impacts the quality of an optimal solution compared with the solution obtained by solving to optimality the different levels sequentially. Algorithm MCN(V, A, Ω, Φ, Λ) was implemented in Python 2.7.6, and all MILPs have been solved with IBM CPLEX 12.7.1. The experiments were conducted on an Intel Xeon E5-2637 processor clocked at 3.50 GHz and equipped with 8 GB RAM, using a single core. Each run was performed with a time limit of two hours. All data associated with our computational experiments (code, instances, results) are available at https://github.com/mxmmargarida/Critical-Node-Problem.

4.1. Results on Random Instances

Our primary test set consists of randomly generated instances with the following two graph topologies.

• Randomly generated trees: An instance is built by starting from a node and adding in each step a new node together with an edge that connects this node to a randomly chosen node already in the tree.

• Random undirected graphs: An instance is generated by adding all the nodes at once and then choosing a uniformly random set of edges of a given size. This size is determined by the parameter *density*, which represents the fraction of chosen edges with respect to the total number of edges in a complete graph.

For each graph topology, we consider several types of instances by varying the number of nodes, the density, and the values of the different budgets Ω, Φ, and Λ for the three stages of the problem. In addition, for each instance type, we generate 20 different random instances.

Three variants of our framework for solving the MCN problem are compared:

• MCN: Algorithm MCN($V, A, \Omega, \Phi, \Lambda$) illustrated in Section 3.

• MCN^{MIX++}: MCN framework with algorithm AP replaced by the bilevel code MIX ++ (available at https://msinnl.github.io/pages/bilevel.html) proposed in Fischetti et al. (2017), which means that the attack–protect problem is always solved to optimality.

• HIB: MCN framework with our algorithm AP restricted to a maximum of 10 iterations and then followed by MIX ++ in case 10 iterations are not sufficient to attain the *target*.

For each instance type and each tested algorithm, we report average results on the subset of the 20 instances of the given type that are solved to optimality by the given algorithm within the two-hour time limit.

The following list summarizes the notation adopted for the results given in the tables:

• Type: The type of the graph: *tree* for random trees and *rndgraph*(d) for random graphs of density d.

• Size: The number of nodes in the graph.

• **Ω-Φ-Λ**: The budgets available for the three levels.

• Sol%: Percentage of instances optimally solved within the time limit (two hours).

• Time: Average total running time (seconds) for the solved instances.

• AP%: Average time spent by the last call to AP given as percentage of the total time.

• It: Average number of iterations in MCN of the solved instances.

• RG%: Percentage of instances with a nonzero relative gain among the solved instances.

• Avg: Average percentage of relative gain for instances with nonzero relative gain.

• Max: Maximum percentage of relative gain observed for the solved instances.

To complement the tables, performance profile plots for the three algorithms are also given. The plots are presented twice: once with a linear scale, which enables seeing the percentage of instances that each algorithm solves within the time limit and again with a logarithmic scale, which allows determining the percentage of instances solved in less time by each method.

4.1.1. Performance Analysis. Our computational analysis starts by comparing different instances with identical graph topology but with different size and vector of budgets. Table 1 and Figure 3 summarize our results for random trees. For all graph classes, we considered the sizes (number of nodes) 20, 40, 60, 80, and 100. The budgets in our computations range in the set $\{1, 2, 3\}$. As expected, for all three algorithms, instances get harder to solve as the number of nodes and budgets increase. In particular, a large budget for the second level Φ seems to slow the performance down more than increasing the budget for the first and third levels, Ω and Λ. This is not surprising: the vaccination and protection optimize in the same direction in opposition to the attacker. Moreover, for trees of size 100 with $\Phi = 3$, MCN and MCN^{MIX++} are only capable of solving a small fraction of the instances although HIB performs significantly better. Another observation is that the more difficult the instance, the bigger the impact that the last call to solve the attack–protect problem has on the total time for MCN and HIB. Recall that, for MCN, solving the last call to AP is the one that has to return a solution to a corresponding (AP) problem with guarantee of optimality. For all the other calls, the solutions are only required to match the quality of some given

Table 1. Detailed Results on Tree Instances

size	Ω-Φ-Λ	sol%	time(s)	AP%	it	RG%	avg	max
20	1-1-1	100.0	0.3	66.7	3.2	15.0	9.2	13.3
20	1-1-1	100.0	1.1	27.3	3.1	15.0	9.2	13.3
20	1-1-1	100.0	0.4	50.0	3.2	15.0	9.2	13.3
20	3-1-3	100.0	0.4	75.0	5.0	0.0	-	-
20	3-1-3	100.0	2.5	12.0	5.0	0.0	-	-
20	3-1-3	100.0	0.7	42.9	5.0	0.0	-	-
20	2-2-2	100.0	5.0	92.0	5.9	25.0	8.0	13.3
20	2-2-2	100.0	7.6	19.7	5.9	25.0	8.0	13.3
20	2-2-2	100.0	2.0	70.0	5.9	25.0	8.0	13.3
20	3-3-1	100.0	8.9	94.4	9.2	10.0	8.0	8.3
20	3-3-1	100.0	15.6	15.4	8.7	10.0	8.0	8.3
20	3-3-1	100.0	2.9	75.9	9.2	10.0	8.0	8.3
20	1-3-3	100.0	17.9	98.9	4.2	30.0	11.4	16.7
20	1-3-3	100.0	14.7	25.9	4.4	30.0	11.4	16.7
20	1-3-3	100.0	4.1	85.4	4.2	30.0	11.4	16.7
20	3-3-3	100.0	37.2	98.1	10.9	35.0	7.9	14.3
20	3-3-3	100.0	40.7	11.5	11.1	35.0	7.9	14.3
20	3-3-3	100.0	5.5	83.6	10.9	35.0	7.9	14.3
40	1-1-1	100.0	0.5	60.0	3.2	10.0	6.7	10.0
40	1-1-1	100.0	3.1	29.0	3.2	10.0	6.7	10.0
40	1-1-1	100.0	1.0	60.0	3.2	10.0	6.7	10.0
40	3-1-3	100.0	1.6	81.2	5.0	45.0	3.2	7.7
40	3-1-3	100.0	7.7	16.9	5.0	45.0	3.2	7.7
40	3-1-3	100.0	2.1	57.1	5.0	45.0	3.2	7.7
40	2-2-2	100.0	23.6	95.8	6.8	35.0	5.8	17.2
40	2-2-2	100.0	60.6	19.0	6.8	35.0	5.8	17.2
40	2-2-2	100.0	12.2	85.2	6.8	35.0	5.8	17.2
40	3-3-1	100.0	208.0	99.1	9.8	10.0	6.0	8.3
40	3-3-1	100.0	310.5	16.1	9.7	10.0	6.0	8.3
40	3-3-1	100.0	48.7	94.5	9.8	10.0	6.0	8.3
40	1-3-3	100.0	372.2	99.8	4.5	40.0	6.5	13.0
40	1-3-3	100.0	187.1	27.6	4.6	40.0	6.5	13.0
40	1-3-3	100.0	49.5	95.4	4.5	40.0	6.5	13.0
40	3-3-3	75.0	3179.3	99.9	13.3	13.3	5.0	6.7
40	3-3-3	100.0	870.9	11.0	12.7	10.0	5.0	6.7
40	3-3-3	100.0	91.3	96.2	12.4	10.0	5.0	6.7
60	1-1-1	100.0	1.5	53.3	3.4	10.0	3.4	4.5
60	1-1-1	100.0	8.0	36.2	3.3	10.0	3.4	4.5
60	1-1-1	100.0	2.7	70.4	3.4	10.0	3.4	4.5
60	3-1-3	100.0	4.3	86.0	5.2	40.0	2.3	3.4
60	3-1-3	100.0	19.4	14.9	5.2	40.0	2.3	3.4
60	3-1-3	100.0	5.1	66.7	5.2	40.0	2.3	3.4
60	2-2-2	100.0	85.4	97.0	6.8	25.0	5.9	11.4
60	2-2-2	100.0	224.1	20.0	6.5	25.0	5.9	11.4
60	2-2-2	100.0	44.8	88.8	6.8	25.0	5.9	11.4
60	3-3-1	100.0	934.2	99.2	10.9	45.0	3.4	7.7

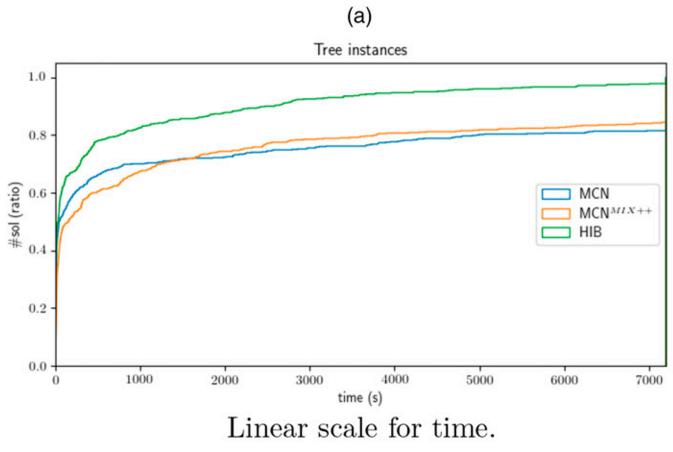
Table 1. (Continued)

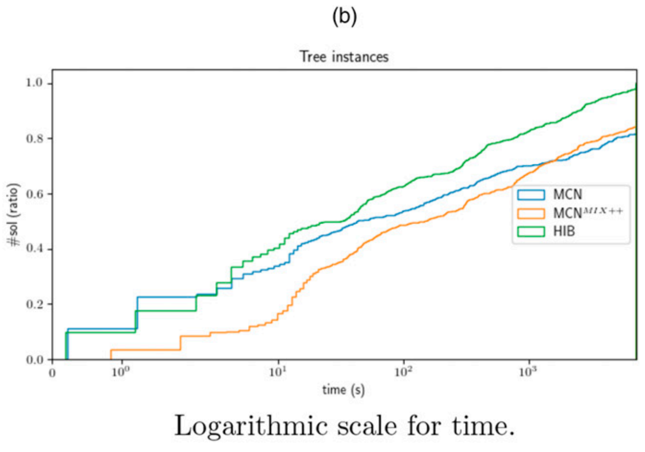
size	Ω-Φ-Λ	sol%	time(s)	AP%	it	RG%	avg	max
60	3-3-1	100.0	2314.8	12.9	12.2	45.0	3.4	7.7
60	3-3-1	100.0	307.8	95.9	10.9	45.0	3.4	7.7
60	1-3-3	100.0	2211.5	99.9	4.2	45.0	9.6	17.1
60	1-3-3	100.0	931.2	31.5	4.2	45.0	9.6	17.1
60	1-3-3	100.0	310.1	97.2	4.2	45.0	9.6	17.1
60	3-3-3	5.0	6264.4	99.7	16.0	100.0	2.3	2.3
60	3-3-3	90.0	4948.5	11.9	12.6	55.6	2.9	4.7
60	3-3-3	100.0	536.3	97.8	13.2	55.0	2.8	4.7
80	1-1-1	100.0	1.7	58.8	3.1	10.0	3.5	3.6
80	1-1-1	100.0	20.7	38.6	3.1	10.0	3.5	3.6
80	1-1-1	100.0	6.0	81.7	3.1	10.0	3.5	3.6
80	3-1-3	100.0	12.0	86.7	5.3	60.0	1.5	2.6
80	3-1-3	100.0	49.7	17.1	5.3	60.0	1.5	2.6
80	3-1-3	100.0	10.4	75.0	5.3	60.0	1.5	2.6
80	2-2-2	100.0	200.4	97.4	6.5	25.0	4.2	9.5
80	2-2-2	100.0	768.7	19.3	6.5	25.0	4.2	9.5
80	2-2-2	100.0	125.0	91.1	6.5	25.0	4.2	9.5
80	3-3-1	80.0	3046.0	99.6	10.9	25.0	5.6	8.2
80	3-3-1	30.0	6587.9	14.7	10.0	16.7	6.2	6.2
80	3-3-1	100.0	1352.2	98.6	10.2	20.0	5.6	8.2
80	1-3-3	45.0	3896.2	99.9	4.2	44.4	10.8	20.0
80	1-3-3	100.0	3497.6	29.4	4.5	35.0	9.3	20.0
80	1-3-3	100.0	1078.2	98.4	4.5	35.0	9.3	20.0
80	3-3-3	0.0	-	-	-	-	-	-
80	3-3-3	0.0	-	-	-	-	-	-
80	3-3-3	100.0	2138.5	98.8	13.3	40.0	2.9	6.7
100	1-1-1	100.0	1.7	52.9	3.6	10.0	3.4	5.3
100	1-1-1	100.0	29.5	35.3	3.5	10.0	3.4	5.3
100	1-1-1	100.0	8.5	81.2	3.6	10.0	3.4	5.3
100	3-1-3	100.0	13.2	90.9	5.2	35.0	1.6	4.1
100	3-1-3	100.0	104.9	14.9	5.2	35.0	1.6	4.1
100	3-1-3	100.0	19.3	82.9	5.2	35.0	1.6	4.1
100	2-2-2	100.0	379.7	98.0	6.5	30.0	3.2	6.8
100	2-2-2	100.0	1559.1	21.1	6.5	30.0	3.2	6.8
100	2-2-2	100.0	400.8	92.5	6.5	30.0	3.2	6.8
100	3-3-1	35.0	4935.3	99.7	9.7	0.0	-	-
100	3-3-1	0.0	-	-	-	-	-	-
100	3-3-1	90.0	3600.5	99.3	10.0	0.0	-	-
100	1-3-3	5.0	4345.4	99.8	5.0	100.0	3.6	3.6
100	1-3-3	10.0	6716.6	30.4	5.0	50.0	13.0	13.0
100	1-3-3	95.0	3259.3	99.2	4.5	47.4	7.6	13.0
100	3-3-3	0.0	-	-	-	-	-	-
100	3-3-3	0.0	-	-	-	-	-	-
100	3-3-3	50.0	6079.5	99.3	15.2	70.0	4.1	6.9

Legend

□	MCN
░	MCN^{MIX++}
▓	HIB

Figure 3. (Color online) Performance Profile on Tree Instances





bound. Similarly, for the last (AP) problem that HIB has to solve, if algorithm AP does not solve it to optimality within 10 iterations, then MIX ++ is called. The fact that, for MCN^{MIX++}, all (AP) problems are solved to optimality can explain why the last call to MIX ++ to solve the (AP) problem does not consume most of the total computational time. Figure 3 suggests that HIB is the more suitable algorithm to solve these instances. The reason is because the balance between solving (AP) problems to optimality with MIX ++ or simply computing good quality attacker's solutions with algorithm AP. Note that each iteration of MCN, except the last one, efficiently computes a good quality attacker's solution; however, algorithm AP struggles to prove optimality, which is required in MCN's last iteration. In turn, MCN^{MIX++} does not seem to provide a significant decrease in the number of iterations by solving all the (AP) problems to optimality; moreover, solving these bilevel problems to optimality is revealed to be slower than using algorithm AP when only a *target* has to be attained. For what concerns the relative gain, it seems that, for sufficiently large trees, there is a high probability that the instance has a nonzero relative gain if the protection budget Λ is high.

Table 2 and Figure 4 show computational results on random graphs of density 5% with sizes and budget vectors identical to those used for the tree case. Based on the number of instances solved, it seems that, also in this case, the algorithm's performance declines when the attack budget Φ increases although the increase of Ω and Λ has a milder effect. Moreover, for sizes smaller than 40, we notice that, analogously to the tree case, the last call to algorithm AP has a very high impact on the running time of MCN and HIB. However, for sizes bigger than 60, this phenomenon disappears, and it seems that the performance of MCN and HIB are not dependent on the time required for algorithm AP to obtain a certified optimal solution. In order to understand the reason behind this result, we explore in detail these instances. For example, consider the instance of size 60 with budgets $3-3-1$; the 75% instances solved by MCN are also solved by MCN^{MIX++} and HIB. For these 75% instances, the average number of iterations for MCN, MCN^{MIX++}, and HIB are 25.7, 13.2, and 23.4, respectively. The significantly smaller number of MCN^{MIX++} iterations is explained by the fact that the cuts (cutAP) are built on optimal (AP) solutions determined by MIX ++; on the other hand, the non-necessary optimality of the (AP) solutions used to build the cuts (cutAP) for intermediate iterations of MCN and HIB increases their total number of iterations; therefore, more time is spent in MCN and HIB intermediate iterations, reducing the impact of the last iteration computational cost over the total running time of MCN and HIB.

Figure 4 shows that, for general graphs, the comparison between HIB and MCN^{MIX++} is overall more mixed with respect to the case of instances defined on trees. Indeed, as long as we are concerned with instances that can be solved reasonably fast (i.e., within 100 seconds), HIB seems to be more efficient than MCN^{MIX++}, and as shown in Figure 4(b), it can solve roughly half of the instances in less than 100 seconds. However, on harder instances, MCN^{MIX++} appears to be more effective than HIB, and as shown in Figure 4(a), it is the approach that can solve more instances within the time limit of two hours. As previously observed, it seems that MCN^{MIX++} benefits from the fact that it always finds the attacker's optimal solutions, which results in fewer iterations of the overall framework. Apart from small sizes, random graphs of density 5% show much higher relative gains compared with tree instances. In general, computations show many instances with nonzero relative gains, and on average, they are significant.

Table 2. Detailed Results on Random Graphs of Density 5%

size	Ω-Φ-Λ	sol%	time(s)	AP%	it	RG%	avg	max
20	1-1-1	100.0	0.3	66.7	3.1	5.0	5.6	5.6
20	1-1-1	100.0	2.0	20.0	3.1	5.0	5.6	5.6
20	1-1-1	100.0	0.6	50.0	3.1	5.0	5.6	5.6
20	3-1-3	100.0	0.4	50.0	5.0	0.0	-	-
20	3-1-3	100.0	3.3	12.1	5.0	0.0	-	-
20	3-1-3	100.0	0.7	42.9	5.0	0.0	-	-
20	2-2-2	100.0	3.6	94.4	5.6	5.0	5.6	5.6
20	2-2-2	100.0	13.9	18.0	5.3	5.0	5.6	5.6
20	2-2-2	100.0	2.0	80.0	5.6	5.0	5.6	5.6
20	3-3-1	100.0	22.4	98.7	10.4	0.0	-	-
20	3-3-1	100.0	37.6	9.6	10.4	0.0	-	-
20	3-3-1	100.0	2.9	79.3	10.4	0.0	-	-
20	1-3-3	100.0	65.8	99.8	4.1	25.0	7.7	12.5
20	1-3-3	100.0	45.9	25.5	4.2	25.0	7.7	12.5
20	1-3-3	100.0	9.2	95.7	4.1	25.0	7.7	12.5
20	3-3-3	100.0	25.0	99.2	7.9	0.0	-	-
20	3-3-3	100.0	68.7	13.0	8.2	0.0	-	-
20	3-3-3	100.0	6.7	92.5	7.9	0.0	-	-
40	1-1-1	100.0	0.6	50.0	3.5	45.0	13.1	27.3
40	1-1-1	100.0	4.3	23.3	3.9	45.0	13.1	27.3
40	1-1-1	100.0	1.2	50.0	3.5	45.0	13.1	27.3
40	3-1-3	100.0	3.0	80.0	5.3	55.0	3.3	5.1
40	3-1-3	100.0	11.6	14.7	5.3	55.0	3.3	5.1
40	3-1-3	100.0	2.3	52.2	5.3	55.0	3.3	5.1
40	2-2-2	100.0	24.6	95.1	6.4	70.0	7.8	17.9
40	2-2-2	100.0	70.3	17.2	7.0	70.0	7.8	17.9
40	2-2-2	100.0	12.4	80.6	6.4	70.0	7.8	17.9
40	3-3-1	100.0	180.2	96.8	10.6	60.0	6.3	14.3
40	3-3-1	100.0	233.8	15.3	11.8	60.0	6.3	14.3
40	3-3-1	100.0	38.9	81.0	10.6	60.0	6.3	14.3
40	1-3-3	95.0	1093.9	99.8	4.9	47.4	8.0	14.3
40	1-3-3	100.0	207.3	24.3	4.7	55.0	7.9	18.2
40	1-3-3	100.0	58.2	85.4	4.9	55.0	7.9	18.2
40	3-3-3	50.0	2730.5	99.8	13.7	50.0	5.1	7.4
40	3-3-3	100.0	874.8	9.4	13.7	55.0	6.7	14.8
40	3-3-3	100.0	81.4	93.1	12.4	55.0	6.7	14.8
60	1-1-1	100.0	1.7	17.6	4.4	10.0	21.1	31.2
60	1-1-1	100.0	3.6	19.4	3.6	10.0	21.1	31.2
60	1-1-1	100.0	1.8	16.7	4.4	10.0	21.1	31.2
60	3-1-3	100.0	35.8	2.0	7.0	95.0	16.1	37.8
60	3-1-3	100.0	138.9	2.6	7.8	95.0	16.1	37.8
60	3-1-3	100.0	40.7	7.6	7.0	95.0	16.1	37.8
60	2-2-2	95.0	1374.1	0.4	14.3	42.1	9.5	15.4
60	2-2-2	100.0	126.1	10.4	7.7	45.0	9.4	15.4
60	2-2-2	100.0	1217.3	1.1	14.3	45.0	9.4	15.4

Table 2. (Continued)

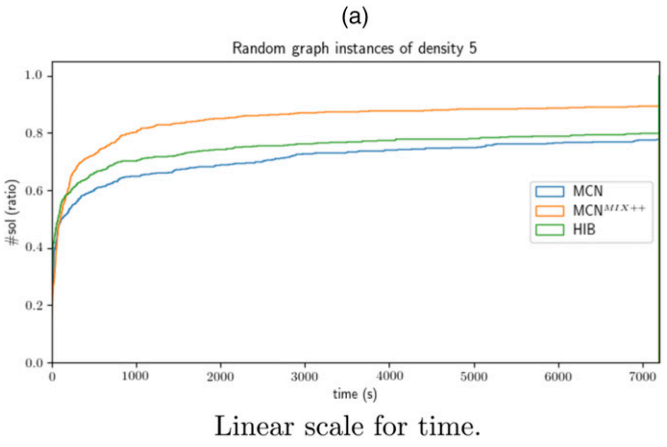
size	Ω-Φ-Λ	sol%	time(s)	AP%	it	RG%	avg	max
60	3-3-1	75.0	1704.6	2.7	25.7	53.3	12.0	20.0
60	3-3-1	100.0	442.1	1.4	14.2	55.0	11.0	20.0
60	3-3-1	85.0	1567.1	0.4	23.9	52.9	11.7	20.0
60	1-3-3	90.0	347.7	13.1	11.0	61.1	14.5	23.1
60	1-3-3	95.0	285.6	23.9	4.8	63.2	14.2	23.1
60	1-3-3	90.0	400.5	19.5	9.3	61.1	14.5	23.1
60	3-3-3	20.0	2509.2	14.3	15.8	100.0	10.2	15.0
60	3-3-3	55.0	2264.9	5.8	11.1	90.9	8.3	16.7
60	3-3-3	30.0	3022.1	5.5	16.8	100.0	8.6	15.0
80	1-1-1	100.0	6.4	9.4	4.5	0.0	-	-
80	1-1-1	100.0	6.0	13.3	3.9	0.0	-	-
80	1-1-1	100.0	6.4	9.4	4.5	0.0	-	-
80	3-1-3	100.0	291.7	0.2	7.5	85.0	8.0	14.3
80	3-1-3	100.0	476.4	1.4	7.5	85.0	8.0	14.3
80	3-1-3	100.0	308.5	2.9	7.5	85.0	8.0	14.3
80	2-2-2	70.0	1506.8	0.3	10.6	35.7	11.6	20.0
80	2-2-2	100.0	675.6	1.6	6.6	25.0	11.6	20.0
80	2-2-2	65.0	1023.0	1.0	9.9	38.5	11.6	20.0
80	3-3-1	35.0	1564.1	1.8	13.9	14.3	12.5	12.5
80	3-3-1	80.0	919.8	0.4	11.2	25.0	11.5	12.5
80	3-3-1	45.0	2197.2	0.2	14.4	33.3	11.7	12.5
80	1-3-3	65.0	2314.3	0.8	11.0	53.8	12.3	14.3
80	1-3-3	85.0	188.7	20.2	4.5	41.2	12.3	14.3
80	1-3-3	85.0	1562.9	2.2	9.6	41.2	12.3	14.3
80	3-3-3	0.0	-	-	-	-	-	-
80	3-3-3	40.0	2889.7	2.6	8.5	87.5	7.6	8.3
80	3-3-3	0.0	-	-	-	-	-	-
100	1-1-1	100.0	3.6	13.9	3.4	0.0	-	-
100	1-1-1	100.0	2.6	19.2	3.0	0.0	-	-
100	1-1-1	100.0	3.6	13.9	3.4	0.0	-	-
100	3-1-3	100.0	1192.4	0.1	6.8	35.0	9.7	15.4
100	3-1-3	90.0	857.5	1.3	6.1	27.8	8.7	10.0
100	3-1-3	100.0	1188.1	0.1	6.8	35.0	9.7	15.4
100	2-2-2	60.0	1313.6	0.5	7.5	8.3	14.3	14.3
100	2-2-2	90.0	368.7	2.2	5.2	5.6	14.3	14.3
100	2-2-2	60.0	1297.9	0.6	7.5	8.3	14.3	14.3
100	3-3-1	15.0	3308.0	0.8	11.3	33.3	12.5	12.5
100	3-3-1	45.0	2838.6	0.1	11.8	22.2	13.4	14.3
100	3-3-1	10.0	1692.1	0.2	11.5	0.0	-	-
100	1-3-3	25.0	4299.4	0.7	8.8	60.0	15.9	16.7
100	1-3-3	70.0	1194.8	2.7	4.6	50.0	15.7	16.7
100	1-3-3	25.0	3892.5	0.9	8.0	80.0	15.5	16.7
100	3-3-3	0.0	-	-	-	-	-	-
100	3-3-3	30.0	1279.9	4.5	6.3	66.7	9.1	9.1
100	3-3-3	0.0	-	-	-	-	-	-

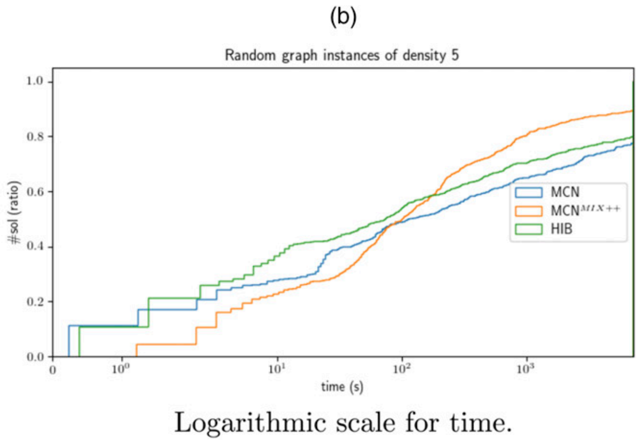
Legend

□	MCN
▒	MCN^{MIX++}
▓	HIB

Finally, Table 3 and Figure 5 show results for random graphs with different densities ranging from 6% to 15% for a fixed size of 40 nodes. Overall, it is hard to draw any general conclusion on the impact of the graph's density on the different performance indicators reported in Table 3, such as running time and relative gain. However, if we restrict our investigation to particular fixed budget vectors, we can

Figure 4. (Color online) Performance Profile on Random Graphs of 5% Density





point out some interesting behavior. For example, for budgets $\Omega = 1, \Phi = 3, \Lambda = 3$, we notice a significant decrease in running times when the density increases from 6% to 15%. However, for $\Omega = 3, \Phi = 3, \Lambda = 3$, the algorithms perform better if the density is around 7%–9%. Moreover, over all computations we perform, it seems that the largest relative gains are attained for budgets $\Omega = 3, \Phi = 1, \Lambda = 3$ with density around 7%.

We conclude this section by remarking that, for all three variants of our framework, the number of iterations needed to solve the MCN problem is much smaller than $\Omega(|V|^{\Phi})$, which is the size $|\mathcal{U}|$ of all feasible attacks. Thus, all three algorithms converge to proven optimality by generating a final set of feasible attacks $\overline{\mathcal{Q}} \subseteq \mathcal{U}$ with $|\overline{\mathcal{Q}}| \ll |\mathcal{U}|$. In other words, the final restricted model (1lvMIP$_{\overline{\mathcal{Q}}}$) that needs to be solved is much smaller than the complete model (1lvMIP). Given the relatively small values of the attacker budget Φ that we considered in the random instances, we also tested with solving the complete model (1lvMIP). We do not report detailed results, but the experiments confirm that directly solving (1lvMIP) is completely outperformed by our iterative approach. In particular, none of the instances with $\Phi = 3$ can be solved within the time limit as well as most of the instances with $\Phi = 2$ (including the ones defined on trees) and some of the largest instances with $\Phi = 1$.

4.2. Results on Graph Instances from a Public Repository

In order to better evaluate the performance of our framework for the MCN problem, we performed additional experiments on graph instances arising from different real-world applications. We focused on algorithm *HIB*, which appears to be, on average, the fastest approach on the random instances considered in Section 4.1, and we consider some instances from the network repository proposed by Rossi and Ahmed (2015). Specifically, we selected 24 graphs with up to 113 nodes and 2,196 edges, which have a density d varying from 2.6% up to 55.6% and with an average value of 25.0%. For each graph, we tested six different budget vectors (Ω, Φ, Λ), which give a total of 144 instances. In particular, because the difficulty of the problem seems to increase with the attacker budget, we consider budgets for the attacker up to $\Phi = 5$. The results reported in Tables 4–6 show that 137 out of 144 instances can be solved to optimality within the two-hour time limit.

On the 137 instances that can be solved, there is a positive relative gain in 45 cases, and on those 45 instances, we observe a maximum relative gain of 43.8% with an average value of 16.0% and a standard deviation of 9.7, which indicates that the results are quite heterogeneous. Looking at the computing times, the results are also heterogeneous:

- The difficulty of the problem does not depend only on the size of the instance, and there are some relatively small graphs, such as lap_25 and Hamrle1, for which some values of the budgets result in large computing times.
- Differently from the random instances tested in the previous section, in most of the cases, the (AP) problem seems not to be the bottleneck as the time spent in the final (AP) solved is only a small fraction of the whole computing time. However, there are several remarkable exceptions, such as ca-sandi_auths and rt-retweet, in which, for most of the budget vectors, more than 90% of the time is spent in the final (AP) solve.
- On the same graph instance, the difficulty of the problem generally increases with the values of

Table 3. Detailed Results on Random Graphs of Size 40

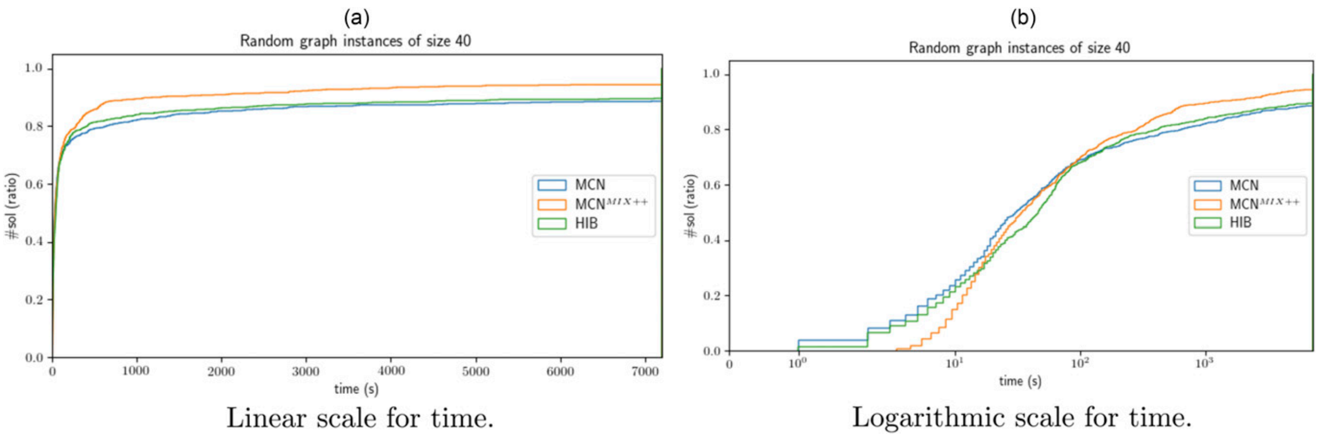
type	Ω-Φ-Λ	sol%	time(s)	AP%	it	RG%	avg	max
rndgraph06	3-1-3	100.0	2.1	76.2	5.8	75.0	7.2	21.1
rndgraph06	3-1-3	100.0	7.5	16.0	5.8	75.0	7.2	21.1
rndgraph06	3-1-3	100.0	2.4	45.8	5.8	75.0	7.2	21.1
rndgraph07	3-1-3	100.0	2.3	65.2	6.3	100.0	27.0	48.6
rndgraph07	3-1-3	100.0	10.8	13.9	6.8	100.0	27.0	48.6
rndgraph07	3-1-3	100.0	3.0	46.7	6.3	100.0	27.0	48.6
rndgraph08	3-1-3	100.0	8.0	7.5	6.9	100.0	23.9	44.1
rndgraph08	3-1-3	100.0	17.8	7.3	7.7	100.0	23.9	44.1
rndgraph08	3-1-3	100.0	9.0	12.2	6.9	100.0	23.9	44.1
rndgraph09	3-1-3	100.0	9.4	4.3	7.0	95.0	23.1	38.1
rndgraph09	3-1-3	100.0	30.2	4.3	7.8	95.0	23.1	38.1
rndgraph09	3-1-3	100.0	10.0	7.0	7.0	95.0	23.1	38.1
rndgraph10	3-1-3	100.0	11.3	1.8	7.0	75.0	11.5	36.4
rndgraph10	3-1-3	100.0	27.9	6.1	7.3	75.0	11.5	36.4
rndgraph10	3-1-3	100.0	11.7	4.3	7.0	75.0	11.5	36.4
rndgraph11	3-1-3	100.0	30.2	0.7	7.5	60.0	9.4	15.4
rndgraph11	3-1-3	100.0	43.1	4.4	7.5	60.0	9.4	15.4
rndgraph11	3-1-3	100.0	30.5	1.3	7.5	60.0	9.4	15.4
rndgraph12	3-1-3	100.0	24.1	0.8	7.5	65.0	10.1	16.7
rndgraph12	3-1-3	100.0	20.6	9.2	6.4	65.0	10.1	16.7
rndgraph12	3-1-3	100.0	24.1	1.2	7.5	65.0	10.1	16.7
rndgraph13	3-1-3	100.0	68.6	0.3	8.4	70.0	10.5	18.2
rndgraph13	3-1-3	100.0	58.5	3.4	7.5	70.0	10.5	18.2
rndgraph13	3-1-3	100.0	68.6	0.3	8.4	70.0	10.5	18.2
rndgraph14	3-1-3	100.0	41.4	0.5	7.8	50.0	12.4	20.0
rndgraph14	3-1-3	100.0	32.7	6.4	6.7	50.0	12.4	20.0
rndgraph14	3-1-3	100.0	41.5	1.0	7.8	50.0	12.4	20.0
rndgraph15	3-1-3	100.0	27.1	0.7	7.1	55.0	10.8	12.5
rndgraph15	3-1-3	100.0	48.2	5.4	6.5	55.0	10.8	12.5
rndgraph15	3-1-3	100.0	27.0	0.7	7.1	55.0	10.8	12.5
rndgraph06	2-2-2	100.0	10.6	65.1	7.3	80.0	15.9	27.8
rndgraph06	2-2-2	100.0	39.1	15.6	8.0	80.0	15.9	27.8
rndgraph06	2-2-2	100.0	10.8	53.7	7.3	80.0	15.9	27.8
rndgraph07	2-2-2	100.0	27.8	11.9	10.1	80.0	13.3	29.4
rndgraph07	2-2-2	100.0	25.9	12.7	7.2	80.0	13.3	29.4
rndgraph07	2-2-2	100.0	42.6	8.5	10.1	80.0	13.3	29.4
rndgraph08	2-2-2	100.0	51.0	3.9	10.9	70.0	11.3	18.2
rndgraph08	2-2-2	100.0	33.5	9.3	7.2	70.0	11.3	18.2
rndgraph08	2-2-2	100.0	54.9	5.5	10.9	70.0	11.3	18.2
rndgraph09	2-2-2	95.0	154.3	0.8	12.1	36.8	13.4	25.0
rndgraph09	2-2-2	100.0	41.5	7.5	7.4	35.0	13.4	25.0
rndgraph09	2-2-2	95.0	158.2	1.8	12.1	36.8	13.4	25.0
rndgraph10	2-2-2	100.0	119.9	0.8	10.8	45.0	13.4	16.7
rndgraph10	2-2-2	100.0	20.9	9.6	6.7	45.0	13.4	16.7
rndgraph10	2-2-2	100.0	121.4	1.6	10.8	45.0	13.4	16.7
rndgraph11	2-2-2	100.0	258.3	0.4	10.1	45.0	16.6	28.6
rndgraph11	2-2-2	100.0	44.9	4.2	7.3	45.0	16.6	28.6
rndgraph11	2-2-2	100.0	258.5	0.6	10.1	45.0	16.6	28.6
rndgraph12	2-2-2	100.0	149.5	0.7	11.4	15.0	15.1	16.7
rndgraph12	2-2-2	100.0	37.7	3.7	7.5	15.0	15.1	16.7
rndgraph12	2-2-2	100.0	148.8	0.9	11.4	15.0	15.1	16.7
rndgraph13	2-2-2	95.0	269.6	0.4	10.9	52.6	16.4	20.0
rndgraph13	2-2-2	100.0	52.5	2.3	7.8	50.0	16.4	20.0
rndgraph13	2-2-2	95.0	269.5	0.4	10.9	52.6	16.4	20.0
rndgraph14	2-2-2	100.0	262.2	0.5	10.8	45.0	19.3	20.0
rndgraph14	2-2-2	100.0	34.5	2.9	7.8	45.0	19.3	20.0
rndgraph14	2-2-2	100.0	262.7	0.4	10.8	45.0	19.3	20.0
rndgraph15	2-2-2	95.0	33.2	3.9	8.2	73.7	20.0	20.0
rndgraph15	2-2-2	100.0	17.3	3.5	6.7	70.0	20.0	20.0
rndgraph15	2-2-2	95.0	33.2	3.9	8.2	73.7	20.0	20.0

Table 3. (Continued)

type	Ω-Φ-Λ	sol%	time(s)	AP%	it	RG%	avg	max
rndgraph06	1-3-3	100.0	73.7	93.8	5.1	45.0	9.1	15.8
rndgraph06	1-3-3	100.0	102.6	25.6	4.5	45.0	9.1	15.8
rndgraph06	1-3-3	100.0	78.0	64.1	5.2	45.0	9.1	15.8
rndgraph07	1-3-3	100.0	29.8	46.0	7.7	50.0	11.7	25.0
rndgraph07	1-3-3	100.0	81.5	25.9	4.7	50.0	11.7	25.0
rndgraph07	1-3-3	100.0	114.0	36.2	7.5	50.0	11.7	25.0
rndgraph08	1-3-3	100.0	39.1	22.5	8.6	65.0	10.4	11.1
rndgraph08	1-3-3	100.0	81.3	20.0	5.5	65.0	10.4	11.1
rndgraph08	1-3-3	100.0	137.3	21.2	8.6	65.0	10.4	11.1
rndgraph09	1-3-3	100.0	94.9	5.7	8.3	30.0	11.5	14.3
rndgraph09	1-3-3	100.0	57.8	19.6	4.8	30.0	11.5	14.3
rndgraph09	1-3-3	100.0	143.1	13.0	8.0	30.0	11.5	14.3
rndgraph10	1-3-3	100.0	32.3	12.1	7.8	60.0	14.4	16.7
rndgraph10	1-3-3	100.0	40.0	22.2	4.7	60.0	14.4	16.7
rndgraph10	1-3-3	100.0	107.3	11.2	7.7	60.0	14.4	16.7
rndgraph11	1-3-3	100.0	39.7	10.1	7.8	25.0	16.2	16.7
rndgraph11	1-3-3	100.0	29.0	21.0	4.6	25.0	16.2	16.7
rndgraph11	1-3-3	100.0	76.3	13.1	7.7	25.0	16.2	16.7
rndgraph12	1-3-3	100.0	152.1	2.3	7.1	35.0	16.4	16.7
rndgraph12	1-3-3	100.0	128.4	4.0	5.4	35.0	16.4	16.7
rndgraph12	1-3-3	95.0	169.8	2.7	7.1	31.6	16.3	16.7
rndgraph13	1-3-3	100.0	34.7	11.0	6.2	15.0	16.7	16.7
rndgraph13	1-3-3	100.0	36.8	10.1	4.9	15.0	16.7	16.7
rndgraph13	1-3-3	100.0	37.4	9.1	6.0	15.0	16.7	16.7
rndgraph14	1-3-3	100.0	28.0	13.2	6.5	10.0	16.7	16.7
rndgraph14	1-3-3	100.0	25.5	14.1	5.3	10.0	16.7	16.7
rndgraph14	1-3-3	100.0	38.6	9.1	6.6	10.0	16.7	16.7
rndgraph15	1-3-3	100.0	19.3	21.2	4.8	15.0	20.0	20.0
rndgraph15	1-3-3	100.0	12.8	21.1	3.9	15.0	20.0	20.0
rndgraph15	1-3-3	100.0	19.4	19.1	4.8	15.0	20.0	20.0
rndgraph06	3-3-3	100.0	1302.3	99.4	11.5	60.0	8.8	18.2
rndgraph06	3-3-3	100.0	490.4	10.2	13.1	60.0	8.8	18.2
rndgraph06	3-3-3	100.0	64.1	80.2	11.5	60.0	8.8	18.2
rndgraph07	3-3-3	95.0	177.9	62.7	13.2	94.7	13.2	22.7
rndgraph07	3-3-3	95.0	737.6	4.8	14.0	94.7	12.9	22.7
rndgraph07	3-3-3	100.0	224.7	15.1	13.9	95.0	12.8	22.7
rndgraph08	3-3-3	80.0	765.2	3.6	15.9	81.2	9.0	14.3
rndgraph08	3-3-3	90.0	1055.6	2.5	14.2	83.3	9.3	15.4
rndgraph08	3-3-3	75.0	505.0	4.7	14.5	93.3	8.8	14.3
rndgraph09	3-3-3	55.0	1395.9	1.0	17.8	63.6	9.0	14.3
rndgraph09	3-3-3	75.0	670.4	2.6	13.0	66.7	8.7	14.3
rndgraph09	3-3-3	65.0	1146.3	1.7	18.0	61.5	8.8	14.3
rndgraph10	3-3-3	55.0	1693.6	1.2	15.8	54.5	13.2	23.1
rndgraph10	3-3-3	80.0	695.4	2.1	10.7	50.0	12.2	23.1
rndgraph10	3-3-3	65.0	1852.6	0.9	16.6	46.2	13.0	23.1
rndgraph11	3-3-3	30.0	1564.3	0.8	17.3	33.3	9.1	9.1
rndgraph11	3-3-3	80.0	1002.3	1.4	10.8	43.8	11.6	20.0
rndgraph11	3-3-3	35.0	2422.0	0.8	17.3	42.9	12.7	20.0
rndgraph12	3-3-3	25.0	2331.1	0.4	17.2	40.0	16.1	22.2
rndgraph12	3-3-3	55.0	1547.6	0.6	11.0	54.5	11.2	12.5
rndgraph12	3-3-3	35.0	2506.7	0.6	16.1	42.9	14.4	22.2
rndgraph13	3-3-3	30.0	1849.1	0.2	18.3	66.7	16.6	22.2
rndgraph13	3-3-3	80.0	1354.8	0.7	12.2	50.0	14.2	22.2
rndgraph13	3-3-3	45.0	1632.0	0.5	17.1	66.7	15.0	22.2
rndgraph14	3-3-3	45.0	2374.8	0.2	16.9	88.9	12.3	12.5
rndgraph14	3-3-3	65.0	872.6	0.9	10.8	84.6	12.4	12.5
rndgraph14	3-3-3	50.0	1990.8	0.3	16.1	90.0	12.3	12.5
rndgraph15	3-3-3	45.0	1913.2	0.2	16.8	100.0	12.3	12.5
rndgraph15	3-3-3	60.0	2627.8	0.4	10.6	91.7	12.5	14.3
rndgraph15	3-3-3	35.0	1851.1	0.4	14.3	100.0	12.3	12.5

Legend □ MCN ▨ MCN^{MIX++} ■ HIB

Figure 5. (Color online) Performance Profile on Random Graphs of Size 40



the budgets for the three stages. However, even in this case, there are remarkable exceptions, such as can_24, in which the instance with $(\Omega, \Phi, \Lambda) = (2, 2, 2)$ is much harder than all the other cases, or lap_25, in which the instances with $(\Omega, \Phi, \Lambda) = (4, 4, 4)$ and $(\Omega, \Phi, \Lambda) = (5, 5, 5)$ are solved, respectively, in 37.6 and 174.3 seconds although the case $(\Omega, \Phi, \Lambda) = (2, 2, 2)$ takes 3,411.8 seconds to be solved and the case $(\Omega, \Phi, \Lambda) = (3, 3, 3)$ hits the time limit.

5. Conclusions

In this work, we introduce a new three-stage model for protecting a network from a harmful spreading agent based on a combination of the critical node and the fence problems, which we call the multilevel critical node problem. We devise an exact algorithm for this problem and test its performance on different types of randomly generated graph structures and on some graph instances arising from practical applications.

Table 4. Detailed Results of Algorithm *HIB* on Graph Instances from a Public Repository

Name	|V|	|E|	Ω-Φ-Λ	Time, s	AP time, s	It	*relGain*
ia-southernwomen	18	64	3-1-3	0.4	0.1	5	8.3
			2-2-2	0.4	0.2	4	0.0
			1-3-3	0.3	0.2	3	0.0
			3-3-3	2.1	0.6	8	0.0
			4-4-4	2.3	1.0	8	0.0
			5-5-5	4.4	2.7	11	9.1
n3c4-b2	20	50	3-1-3	3.9	0.1	8	0.0
			2-2-2	0.8	0.2	4	0.0
			1-3-3	0.7	0.5	2	0.0
			3-3-3	5.6	0.4	9	0.0
			4-4-4	12.7	0.7	20	11.1
			5-5-5	22.8	2.4	30	9.1
n3c4-b3	20	50	3-1-3	6.7	0.1	9	0.0
			2-2-2	0.5	0.2	3	0.0
			1-3-3	0.9	0.4	3	0.0
			3-3-3	5.6	0.4	10	0.0
			4-4-4	12.7	0.6	19	0.0
			5-5-5	11.3	2.5	21	9.1
Trefethen_20	20	69	3-1-3	31.1	0.1	10	14.3
			2-2-2	0.5	0.2	3	0.0
			1-3-3	1.3	0.2	2	0.0
			3-3-3	5.7	0.2	7	0.0
			4-4-4	15.9	0.2	11	0.0
			5-5-5	50.9	1.1	21	0.0
Ragusa18	23	51	3-1-3	1.0	0.3	5	26.3
			2-2-2	0.8	0.3	5	0.0
			1-3-3	2.9	2.0	4	14.3
			3-3-3	3.9	2.7	6	0.0
			4-4-4	9.4	7.9	9	7.1
			5-5-5	33.6	31.7	13	0.0

Table 4. (Continued)

Name	|V|	|E|	Ω-Φ-Λ	Time, s	AP time, s	It	*relGain*
GD02_a	23	59	3-1-3	1.0	0.4	5	15.0
			2-2-2	1.4	0.6	3	0.0
			1-3-3	1.8	0.9	3	0.0
			3-3-3	3.8	2.6	6	0.0
			4-4-4	12.1	11.0	8	6.7
			5-5-5	58.0	56.5	12	0.0
can_24	24	68	3-1-3	17.5	0.2	13	40.0
			2-2-2	1,681.9	0.3	32	0.0
			1-3-3	1.7	0.5	5	0.0
			3-3-3	148.1	0.4	30	0.0
			4-4-4	22.5	2.0	15	0.0
			5-5-5	26.0	3.2	18	0.0
lap_25	25	72	3-1-3	1.8	0.1	8	33.3
			2-2-2	3,411.8	0.3	31	0.0
			1-3-3	4.1	0.5	8	20.0
			3-3-3	—	—	—	—
			4-4-4	37.6	3.3	16	20.0
			5-5-5	174.3	2.5	31	0.0

Table 5. Detailed Results of Algorithm *HIB* on Graph Instances from a Public Repository

Name	|V|	|E|	Ω-Φ-Λ	Time, s	AP time, s	It	*relGain*
rel4	28	95	3-1-3	0.5	0.1	5	43.8
			2-2-2	1.0	0.3	5	0.0
			1-3-3	0.9	0.2	3	0.0
			3-3-3	3.1	0.4	9	0.0
			4-4-4	31.7	31.0	6	0.0
			5-5-5	84.1	83.2	6	0.0
johnson8-2-4	28	210	3-1-3	1.2	0.1	5	0.0
			2-2-2	0.6	0.3	3	0.0
			1-3-3	1.1	0.2	2	0.0
			3-3-3	1.1	0.1	3	0.0
			4-4-4	1.8	0.4	3	0.0
			5-5-5	2.8	1.2	3	0.0
Hamrle1	32	90	3-1-3	12.6	0.2	6	25.0
			2-2-2	109.8	0.4	13	0.0
			1-3-3	32.3	2.8	8	0.0
			3-3-3	41.3	2.3	9	14.3
			4-4-4	7,038.2	5.3	28	0.0
			5-5-5	624.5	15.4	30	16.7
ibm32	32	90	3-1-3	3.5	0.2	7	10.0
			2-2-2	8.2	0.3	7	20.0
			1-3-3	10.7	2.1	5	20.0
			3-3-3	464.3	1.8	18	0.0
			4-4-4	590.4	4.5	23	10.0
			5-5-5	870.0	10.7	35	8.3
Maragal_1	32	207	3-1-3	0.9	0.1	5	0.0
			2-2-2	5.3	0.2	10	0.0
			1-3-3	5.3	0.6	8	0.0
			3-3-3	10.3	0.8	12	12.5
			4-4-4	10.6	7.8	5	0.0
			5-5-5	20.6	13.1	7	0.0

Table 5. (Continued)

Name	|V|	|E|	Ω-Φ-Λ	Time, s	AP time, s	It	*relGain*
GD01_c	33	135	3-1-3	28.6	0.2	12	0.0
			2-2-2	3.5	1.3	4	0.0
			1-3-3	15.2	9.4	3	16.7
			3-3-3	51.6	9.2	10	0.0
			4-4-4	413.4	7.5	17	9.1
			5-5-5	1,164.8	21.7	29	15.4
karate	34	78	3-1-3	1.6	0.6	5	0.0
			2-2-2	3.3	1.1	7	8.3
			1-3-3	2.6	0.8	4	0.0
			3-3-3	7.7	6.0	6	0.0
			4-4-4	66.0	64.3	7	0.0
			5-5-5	257.8	254.6	11	0.0
soc-karate	34	78	3-1-3	1.6	0.6	5	0.0
			2-2-2	3.4	1.1	7	8.3
			1-3-3	2.9	0.8	4	0.0
			3-3-3	7.8	6.1	6	0.0
			4-4-4	65.2	63.2	7	0.0
			5-5-5	257.2	254.3	11	0.0

Interestingly, the computational experiments show that this model often yields solutions that are significantly better than the ones obtained by solving the critical node problem and the fence prob-lem sequentially.

In this line of work, devising faster algorithms for MCN is an interesting direction for future research. Moreover, replacing the fence problem with the firefighter problem might be a way to describe more sophisticated models, which better capture some specific dynamics of network defense. Additionally, simultaneous vertex and edge removal or only edge removal

Table 6. Detailed Results of Algorithm *HIB* on Graph Instances from a Public Repository

Name	$\|V\|$	$\|E\|$	Ω-Φ-Λ	time, s	AP time, s	It	*relGain*
road-chesapeake	39	170	3-1-3	2.9	0.3	7	30.0
			2-2-2	4.7	0.6	7	0.0
			1-3-3	1.8	0.9	3	0.0
			3-3-3	10.6	0.6	7	25.0
			4-4-4	38.9	16.2	9	27.3
			5-5-5	76.2	39.2	10	21.4
eco-stmarks	54	350	3-1-3	470.7	0.3	12	14.3
			2-2-2	5.2	1.7	4	0.0
			1-3-3	4.3	1.2	2	0.0
			3-3-3	140.1	8.1	11	14.3
			4-4-4	341.8	12.2	12	0.0
			5-5-5	2,694.6	51.2	17	16.7
soc-dolphins	62	159	3-1-3	14.8	0.4	9	13.8
			2-2-2	77.7	1.5	9	0.0
			1-3-3	30.4	10.9	4	22.2
			3-3-3	966.0	23.0	16	0.0
			4-4-4	5,064.2	152.1	24	0.0
			5-5-5	—	—	—	—
eco-everglades	69	879	3-1-3	733.3	0.8	9	0.0
			2-2-2	7.4	1.2	4	0.0
			1-3-3	12.6	1.5	3	0.0
			3-3-3	18.8	5.4	4	0.0
			4-4-4	281.8	3.6	5	0.0
			5-5-5	3,146.3	22.4	8	0.0
ca-sandi_auths	86	124	3-1-3	13.9	11.1	5	0.0
			2-2-2	90.1	81.5	6	2.0
			1-3-3	396.1	373.3	4	0.0
			3-3-3	1,767.8	1,745.5	8	0.0
			4-4-4	—	—	—	—
			5-5-5	—	—	—	—
rt-retweet	96	117	3-1-3	33.6	31.1	5	9.0
			2-2-2	103.3	97.1	5	17.1
			1-3-3	1,006.8	489.1	6	3.2
			3-3-3	2,226.9	2,200.8	10	5.5
			4-4-4	—	—	—	—
			5-5-5	—	—	—	—
eco-mangwet	97	1,446	3-1-3	437.3	1.4	8	14.3
			2-2-2	9.7	2.9	3	0.0
			1-3-3	10.4	7.3	2	0.0
			3-3-3	81.8	15.6	6	0.0
			4-4-4	3,178.7	36.6	10	0.0
			5-5-5	—	—	—	—
ia-infect-hyper	113	2,196	3-1-3	51.8	4.0	5	0.0
			2-2-2	42.7	12.8	4	0.0
			1-3-3	163.7	73.4	3	0.0
			3-3-3	265.9	89.1	4	0.0
			4-4-4	443.3	8.8	4	0.0
			5-5-5	3,709.4	65.3	4	0.0

can be considered instead. Finally, it is clear that considering a static model, that is, disregarding the timescale of the propagation of an attack, is a simplification. This simplification is currently necessary because of the mathematical and algorithmic difficulties in dealing with the problems presented in this paper, but future work in this direction is of paramount importance.

Acknowledgments

The authors are indebted to the associate editor and three anonymous reviewers for an impressively deep reading and very constructive comments that helped strengthen the paper. The first author's contribution was performed while he was at IFOR in the Department of Mathematics at ETH Zurich and at the IBM-UniBo Center of Excellence on Mathematical Optimization in Bologna. Part of this work was performed while the second author was in INESC TEC.

Appendix. Convergence of Algorithm RowColGen

In the following, we prove that algorithm RowColGen converges to an optimal solution of (1lvMIP) in a finite number of iterations. First, Lemma A.1 shows that (1lvMIP$_{\mathcal{Q}}$) is a relaxation of (1lvMIP) for any subset of feasible attacks $\mathcal{Q} \subseteq \mathcal{U}$. Then, the proof of convergence is given in Lemma A.2 and Theorem A.1.

Lemma A.1. *For any two subsets of feasible attacks $\mathcal{Q}_1$, $\mathcal{Q}_2 \subseteq \mathcal{U}$, if $\mathcal{Q}_1 \subset \mathcal{Q}_2$, then* (1lvMIP$_{\mathcal{Q}_1}$) *is a relaxation of* (1lvMIP$_{\mathcal{Q}_2}$).

Proof (1lvMIP$_{\mathcal{Q}_1}$). can be constructed from (1lvMIP$_{\mathcal{Q}_2}$) by removing variables $x(y)$ and $\alpha(y)$ $\forall y \in \mathcal{Q}_2 \setminus \mathcal{Q}_1$ together with all the constraints in which they appear. Therefore, for any feasible solution $(\bar{\Delta}, \bar{z}, \bar{x}(y) \forall y \in \mathcal{Q}_2, \bar{\alpha}(y) \forall y \in \mathcal{Q}_2)$ of (1lvMIP$_{\mathcal{Q}_2}$), the solution restricted to $\mathcal{Q}_1$, that is, $(\bar{\Delta}, \bar{z}, \bar{x}(y) \forall y \in \mathcal{Q}_1, \bar{\alpha}(y)$ $\forall y \in \mathcal{Q}_1)$, is feasible for (1lvMIP$_{\mathcal{Q}_1}$). Because the removed variables do not appear in the objective function, the two solutions have the same value. □

Lemma A.2. *Given a subset of feasible attacks $\mathcal{Q} \subseteq \mathcal{U}$, let best and z^{best} be the optimal value and the z-part of an optimal solution of* (1lvMIP$_{\mathcal{Q}}$), *respectively. Then,*

i. *If there exists $\bar{y} \in \mathcal{U}$ such that solving the third stage (protection) with initial vaccination z^{best} and attack $\bar{y}$ yields a number of saved nodes strictly smaller than best, then $\bar{y} \notin \mathcal{Q}$.*

ii. *Otherwise, z^{best} is optimal to* (1lvMIP) *with optimal value best.*

Proof. First, we prove (i). Because z^{best} is optimal to (1lvMIP$_{\mathcal{Q}}$) with value *best* for any $y \in \mathcal{Q}$, there exists a protection $x(y)$ such that the number of saved nodes is at least *best*. Therefore, $\bar{y} \notin \mathcal{Q}$.

Next, we prove (ii). If, with initial vaccination z^{best}, there is no attack $y \in \mathcal{U}$ for which solving the third stage yields a number of saved nodes strictly smaller than *best*, then the optimal solution value of (1lvMIP$_{\mathcal{Q} \cup \{y\}}$) is at least *best* for every $y \in \mathcal{U}$. By induction, also the optimal solution value of (1lvMIP) is at least *best*. Given that (1lvMIP$_{\mathcal{Q}}$) is a relaxation of (1lvMIP) by Lemma A.1, it follows that the optimal solution value of (1lvMIP) is exactly *best*, and z^{best} is an optimal vaccination strategy for (1lvMIP). □

Theorem A.1. *Algorithm RowColGen converges to an optimal solution of (1lvMIP) in a finite number of iterations.*

Proof. The proof directly follows from Lemma A.2. In step 3 of RowColGen, either the algorithm stops with an optimal solution of (1lvMIP), or a new attack scenario $y \in \mathcal{U} \setminus \mathcal{Q}$ is generated. Because the set of feasible attacks $\mathcal{U}$ is finite, in the worst case, the algorithm generates all the attacks and terminates after $|\mathcal{U}|$ iterations. □

References

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

Alderson DL, Brown GG, Carlyle WM, Wood RK (2011) Solving defender-attacker-defender models for infrastructure defense. *12th INFORMS Comput. Soc. Conf. (Monterey, California).*

Assimakopoulos N (1987) A network interdiction model for hospital infection control. *Comput. Biol. Medicine* 17(6):413–422.

Ball MO, Golden BL, Vohra RV (1989) Finding the most vital arcs in a network. *Oper. Res. Lett.* 8(2):73–76.

Bard JF, Moore JT (1990) A branch and bound algorithm for the bilevel programming problem. *SIAM J. Sci. Statist. Comput.* 11(2):281–292.

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

Brotcorne L, Hanafi S, Mansi R (2013) One-level reformulation of the bilevel knapsack problem using dynamic programming. *Discrete Optim.* 10(1):1–10.

Brown G, Carlyle M, Salmerón J, Wood K (2006) Defending critical infrastructure. *Interfaces* 36(6):530–544.

Cappanera P, Scaparra MP (2011) Optimal allocation of protective resources in shortest-path networks. *Transportation Sci.* 45(1): 64–80.

Caprara A, Carvalho M, Lodi A, Woeginger G (2016) Bilevel knapsack with interdiction constraints. *INFORMS J. Comput.* 28(2): 319–333.

Caramia M, Mari R (2015) Enhanced exact algorithms for discrete bilevel linear problems. *Optim. Lett.* 9(7):1447–1468.

Church RL, Scaparra MP, Middleton RS (2004) Identifying critical infrastructure: The median and covering facility interdiction problems. *Ann. Assoc. Amer. Geographers* 94(3):491–502.

Dempe S, Zemkoho AB (2013) The bilevel programming problem: Reformulations, constraint qualifications and optimality conditions. *Math. Program.* 138:447–473.

DeNegre S (2011) Interdiction and discrete bilevel linear programming. Unpublished doctoral thesis, Lehigh University, Bethlehem, PA.

DeNegre S, Ralphs TK (2009) A branch-and-cut algorithm for integer bilevel linear programs. Chinneck JW, Kristjansson B, Saltzman MJ, eds. *Operations Research and Cyber-Infrastructure*, vol. 47 (Springer, Boston), 65–78.

Dinitz M, Gupta A (2013) Packing interdiction and partial covering problems. Goemans M, Correa J, eds. *Internat. Conf. Integer Programming Combin. Optim.*, vol. 7801 (Springer, Berlin, Heidelberg), 157–168.

Di Summa M, Grosso A, Locatelli M (2011) Complexity of the critical node problem over trees. *Comput. Oper. Res.* 38(12):1766–1774.

Finbow S, MacGillivray G (2009) The firefighter problem: A survey of results, directions and questions. *Australasian J. Combinatorics* 43:57–77.

Fischetti M, Ljubić I, Monaci M, Sinnl M (2016) Intersection cuts for bilevel optimization. Louveaux Q, Skutella M, eds. *Internat. Conf. Integer Programming Combin. Optim.*, vol. 9682 (Springer, Cham, Switzerland), 77–88.

Fischetti M, Ljubić I, Monaci M, Sinnl M (2017) A new general-purpose algorithm for mixed-integer bilevel linear programs. *Oper. Res.* 65(6):1–23.

Fischetti M, Ljubić I, Monaci M, Sinnl M (2019) Interdiction games and monotonicity, with application to knapsack problems. *INFORMS J. Comput.* 31(2):390–410.

Fomin FV, Golovach PA, Korhonen JH (2013) On the parameterized complexity of cutting a few vertices from a graph. Chatterjee K, Sgall J, eds. *Mathematical Foundations of Computer Science*, Lecture Notes in Computer Science, vol. 8087 (Springer, Berlin, Heidelberg), 421–432.

Frangioni A, Gendron B (2009) 0–1 reformulations of the multicommodity capacitated network design problem. *Discrete Appl. Math.* 157(6):1229–1241.

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

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

Hansen P, Jaumard B, Savard G (1992) New branch-and-bound rules for linear bilevel programming. *SIAM J. Sci. Statist. Comput.* 13(5):1194–1217.

Hartnell B (1995) Firefighter! An application of domination. *24th Manitoba Conf. Combin. Math. Comput. (University of Manitoba, Winnipeg).*

Hayrapetyan A, Kempe D, Pál M, Svitkina Z (2005) Unbalanced graph cuts. Brodal GS, Leonardi S, eds. *Algorithms ESA 2005*, Lecture Notes in Computer Science, vol. 3669 (Springer, Berlin, Heidelberg), 191–202.

Hemmati M, Smith JC (2016) A mixed-integer bilevel programming approach for a competitive prioritized set covering problem. *Discrete Optim.* 20:105–134.

Hemmati M, Smith JC, Thai MT (2014) A cutting-plane algorithm for solving a weighted influence interdiction problem. *Comput. Optim. Appl.* 57(1):71–104.

IBM (2017) CPLEX Optimizer. Accessed October 8, 2020, https://www.ibm.com/analytics/cplex-optimizer.

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

Khachiyan L, Boros E, Borys K, Elbassioni K, Gurvich V, Rudolf G, Zhao J (2008) On short paths interdiction problems: Total and node-wise limited interdiction. *Theory Comput. Systems* 43(2):204–233.

Klein R, Levcopoulos C, Lingas A (2014) Approximation algorithms for the geometric firefighter and budget fence problems. Pardo A, Viola A, eds. *LATIN 2014: Theoretical Informatics*, Lecture Notes in Computer Science, vol. 8392 (Springer, Berlin, Heidelberg), 261–272.

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

Lozano L, Smith JC (2017) A backward sampling framework for interdiction problems with fortification. *INFORMS J. Comput.* 29(1):123–139.

Markines B, Cattuto C, Menczer F (2009) Social spam detection. *Proc. 5th Internat. Workshop Adversarial Inform. Retrieval Web* (ACM, New York), 41–48.

McCormick GP (1976) Computability of global solutions to factorable nonconvex programs: Part I—Convex underestimating problems. *Math. Programming* 10(1):147–175.

Moore JT, Bard JF (1990) The mixed integer linear bilevel programming problem. *Oper. Res.* 38(5):911–921.

Morton DP, Pan F, Saeger KJ (2007) Models for nuclear smuggling interdiction. *IIE Trans.* 39(1):3–14.

Muter I, Birbil Şİ, Bülbül K (2013) Simultaneous column-and-row generation for large-scale linear programs with column-dependent-rows. *Math. Programming* 142(1–2):47–82.

Phillips CA (1993) The network inhibition problem. *Proc. 25th Annual ACM Sympos. Theory Comput.* (ACM, New York), 776–785.

Ratkiewicz J, Conover MD, Meiss M, Flammini A, Menczer F (2011) Detecting and tracking political abuse in social media. *Proc. 5th AAAI Internat. Conf. Weblogs Social Media* (AAAI, Palo Alto, CA), 297–304.

Ratliff HD, Sicilia GT, Lubore S (1975) Finding the *n* most vital links in flow networks. *Management Sci.* 21(5):531–539.

Rossi RA, Ahmed NK (2015) The network data repository with interactive graph analytics and visualization. *Proc. 29th AAAI Conf. Artificial Intelligence.* (AAAI Press, Boston), 4292–4293.

Salmeron J, Wood K, Baldick R (2009) Worst-case interdiction analysis of large-scale electric power grids. *IEEE Trans. Power Systems* 24(1):96–104.

Scaparra MP, Church RL (2008a) A bilevel mixed-integer program for critical infrastructure protection planning. *Comput. Oper. Res.* 35(6):1905–1923.

Scaparra MP, Church RL (2008b) An exact solution approach for the interdiction median problem with fortification. *Eur. J. Oper. Res.* 189(1):76–92.

Shen S, Smith JC (2012) Polynomial-time algorithms for solving a class of critical node problems on trees and series-parallel graphs. *Networks* 60(2):103–119.

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

Shu K, Sliva A, Wang S, Tang J, Liu H (2017) Fake news detection on social media: A data mining perspective. *SIGKDD Explorations* 19(1):22–36.

Smith JC, Lim C, Sudargho F (2007) Survivable network design under optimal and heuristic interdiction scenarios. *J. Global Optim.* 38(2):181–199.

Wood RK (1993) Deterministic network interdiction. *Math. Comput. Model.* 17(2):1–18.

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

Zenklusen R (2010b) Network flow interdiction on planar graphs. *Discrete Appl. Math.* 158(13):1441–1455.

Zenklusen R (2014) Connectivity interdiction. *Oper. Res. Lett.* 42(6): 450–454.

Zenklusen R (2015) An o (1)-approximation for minimum spanning tree interdiction. *2015 IEEE 56th Annual Sympos. Foundations Comput. Sci.* (IEEE, New York), 709–728.

Andrea Baggio has a PhD in applied mathematics from ETH Zurich. His research interests are mathematical programming, graph theory, and algorithmic design. He is currently a quant engineer at swissQuant Group AG.

Margarida Carvalho is an assistant professor in the Department of Computer Science and Operations Research at Université de Montréal. Her research interests are algorithmic game theory, computational complexity, and mixed integer bilevel programming. She holds a FRQ–IVADO Research Chair Data Science for Combinatorial Game Theory. She was awarded the prestigious EURO Doctoral Dissertation Award in 2018.

Andrea Lodi is a full professor in the Department of Mathematical and Industrial Engineering at Polytechnique Montréal and holds the Canada Excellence Research Chair in Data Science for Real-Time Decision-Making. His research interests are mixed-integer linear and nonlinear programming and data science. He received several awards including the IBM Herman Goldstine Fellowship, the Google Faculty Research Award, and the IBM Faculty Award.

Andrea Tramontani has a PhD in operations research from the University of Bologna and is an advisory software scientist at IBM in the core team developing Cplex. His research interests are mixed-integer programming. He received the outstanding technical achievement award by IBM in 2015.