Multilevel Approaches for the Critical Node Problem

Andrea Baggio [a]

Margarida Carvalho [b]

Andrea Lodi [b]

Andrea Tramontani [c]

[a] Quant Research, swissQuant Group AG, Zurich, Switzerland

[b] Canada Excellence Research Chair in Data Science for Real-Time Decision-Making

[c] CPLEX Optimization, IBM, Bologna, Italy

baggio@swissquant.com
margarida.carvalho@polymtl.ca
andrea.lodi@polymtl.ca
andrea.tramontani@it.ibm.com

Canada Excellence Research Chair in Data Science for Real-Time Decision-Making

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. In particular, many results rely on two different viewpoints. On the one hand, network safety is investigated from a preventive perspective. In this paradigm, for a given network, the goal is to modify its structure, in order to minimize the propagation of failures. On the other hand, blocking models have been proposed for scenarios where the attack has already taken place. In this case, a harmful spreading process is assumed to propagate through the network with particular dynamics, allowing some time for an effective defensive reaction.
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, sequential game. Thus, we assume the defender not only to be able to adopt preventive strategies but also to defend the network after an attack takes place. Assuming that the attacker will act optimally, we want to chose a defensive strategy for the first stage that would minimize 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 design an exact algorithm for it based on tools developed for multilevel programming.

Keywords: Multilevel Programming; Critical Node Problem; Firefighter Problem; Mixed-Integer Programming; Defender-Attacker-Defender.

Acknowledgments: The first thanks the support by an EU grant FP7-PEOPLE-2012-ITN no. 316647, “Mixed-Integer Nonlinear Optimization”. The first author 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 the Faculty of Sciences University of Porto and INESC TEC. The second author thanks the support of Institute for data valorisation (IVADO), the Portuguese Foundation for Science and Technology (FCT) through a PhD grant number SFRH/BD/79201/2011 and the ERDF European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme within project POCI-01-0145-FEDER-006961, and National Funds through the FCT (Portuguese Foundation for Science and Technology) as part of project UID/EEA/50014/2013.

Multilevel Approaches for the Critical Node Problem

ANDREA BAGGIO *
MARGARIDA CARVALHO †
ANDREA LODI ‡
ANDREA TRAMONTANI §

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. In particular, many results rely on two different viewpoints. On the one hand, network safety is investigated from a preventive perspective. In this paradigm, for a given network, the goal is to modify its structure, in order to minimize the propagation of failures. On the other hand, blocking models have been proposed for scenarios where the attack has already taken place. In this case, a harmful spreading process is assumed to propagate through the network with particular dynamics, allowing some time for an effective defensive reaction.

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, sequential game. Thus, we assume the defender not only to be able to adopt preventive strategies but also to defend the network after an attack takes place. Assuming that the attacker will act optimally, we want to chose a defensive strategy for the first stage that would minimize 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 design an exact algorithm for it based on tools developed for multilevel programming.

Keywords: Multilevel Programming; Critical Node Problem; Firefighter Problem; Mixed-Integer Programming; Defender-Attacker-Defender

1 Introduction

Motivation. 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 propagates 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 will model as a trilevel programming optimization.

*baggio@swissquant.com. Quant Research, swissQuant Group AG, Zurich, Switzerland
†margarida.carvalho@polymtl.ca. IVADO Fellow, Canada Excellence Research Chair in Data Science for Real-Time Decision-Making, École Polytechnique de Montréal, Montreal, Canada
‡andrea.lodi@polymtl.ca. Canada Excellence Research Chair in Data Science for Real-Time Decision-Making, École Polytechnique de Montréal, Montreal, Canada
§andrea.tramontani@it.ibm.com. CPLEX Optimization, IBM, Bologna, Italy

Literature review. Since we focus on identifying critical infrastructure in a network determined by possible intentional attacks, 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 prevently in parts of the network, they become interdicted to an attack, and by attacking parts of the network, they become interdict to protection. Network interdiction problems have been vastly investigated since decades and 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 system Salmeron et al. (2009), Church et al. (2004), floods 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 knapsack problem Caprara et al. (2016), minimum spanning tree Zenklusen (2015), Frederickson and Solis-Oba (1999), connectivity of a graph Zenklusen (2014), packing Dinitz and Gupta (2013), maximum matching Dinitz and Gupta (2013), Zenklusen (2010a), independent set and vertex cover Bazgan et al. (2011), maximum s-t flow Zenklusen (2010b), Wood (1993), Phillips (1993) (often known as *network flow interdiction*), shortest path Khachiyan et al. (2008), Ball et al. (1989), 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 reason, 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). Concretely, these algorithms use as subroutines MILP models and their implementations use modern MILP solvers. The first approach for solving MIBPs has been presented in Moore and Bard (1990), where the authors applied a branch-and-bound algorithm to a relaxed formulation. Later, DeNegre and Ralphs (2009), DeNegre (2011) improved on these techniques by adding cuts based on integrality and feasibility properties. This approach has been further improved by Caramia and Mari (2015), and by Fischetti et al. (2016b), in which the authors provided significantly better performances by introducing stronger cuts. Very recently, Fischetti et al. (2017) considerably extended the latter branch-and-bound through new families of cuts and an effective pre-processing. Hemmati and Smith (2016) propose a cutting plane for integer bilevel programming problems. A lot of effort has also been dedicated to devising tailored approaches to solve particular special cases as it generally allows faster running times. For instance, Hemmati et al. (2014) considered a bilevel influence interdiction problem in networks, and DeNegre (2011), Brotcorne et al. (2013), Caprara et al. (2016) investigated interdiction versions of the knapsack problem. Fischetti et al. (2016a) build a branch-and-cut algorithm for general interdiction problems that was tested in benchmark instances, showing to be faster than any previous approach from the literature.

There have also been studies on trilevel programs. For example, a three-stage sequential game designed for modeling infrastructure defense has been presented 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.

Our contributions and paper organization. In Section 2, we define the Multilevel Critical Node problem combining two previously known paradigms from network defense into a three-stage Defender-Attacker-Defender game. The model is motivated by the real-world hypothesis that the network operator has two stages in which she can act to protect the network, namely before and after the harmful propagation starts. In Section 3, we model this problem as a trilevel Mixed-Integer Program and we design an exact algorithm for solving it. 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 IBM ILOG CPLEX Optimizer (2017) in order to solve MILPs.

In Section 4, we perform computational experiments analyzing the performances of the developed algorithm. Our test set is based on instances consisting of randomly generated graphs. Our results show that the optimal action when we consider all the three stages of the problem simultaneously has a vastly better objective function value than the value obtained by sequentially optimizing the first and the third steps separately. Such 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 Preliminaries and problem definition

Assume that we are given a directed graph $G = (V, A)$, in which V and A denote the nodes and the arcs, respectively. For $u, v \in V$, we say that (u, v) is an edge of G if the arcs (u, v) and (v, u) exist in A. 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 chose 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.

As already introduced, our main interest consists of developing a new network protection model combining in a unique three-stage sequential game two different well-studied paradigms. The first one investigates strategies that preventively act against possible viral attacks; the second one assumes the detection of an attack and aims at isolating its propagation.

In this work, we assume *perfect information* and *rationality* for the game. In other words, both agents have perfect knowledge of the graph topology, of the viral dynamics, and of the 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.

Before going into the formal definition of the Multilevel Critical Node problem, we briefly introduce separately the preventive and the blocking perspectives we consider, in Section 2.1 and 2.2, respectively. We then define the Multilevel Critical Node problem in Section 2.3 and in Section 2.4 we propose a measure to evaluate the advantage of integrating together the vaccination and protection decisions.

2.1 Preventive strategies (vaccination)

Assume an agent has the possibility to defend a network $G = (V, A)$ against a possible viral attack, but she does not know where it will take place. More precisely, she has resources to vaccinate some subset of nodes in V. Once a node get vaccinated it cannot be attacked nor infected. In other words, a vaccination for the node $v \in V$ interdicts v from being infected, and prevents v from passing the virus on the neighbouring nodes.

Since we assume the attacker to be an intelligent agent, the defender has to identify a subset of nodes whose removal transform 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, on the dynamics of propagation of the virus, and on the resources available for the attacker.
For these reasons, there might not be a clear measure how to evaluate network survivability against attacks. In the literature, many different approaches have been 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 infrastructures* so as to decrease some given graph metric, which measures the resilience of the network against attacks, which depend on 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 $\mathcal{M} : G \to \mathbb{Z}_{\geq 0}$ be a function that maps any graph to a real value, which captures the graph's resilience against viral attacks. Concretely, $\mathcal{M}(G)$ might represent the pair-wise connectivity of G Di Summa et al. (2011), Addis et al. (2013), the number of disconnected components or the size of the biggest connect component Shen and Smith (2012), Shen et al. (2012), and so on. Let $c(v) \in \mathbb{Z}_{\geq 0}$, 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, i.e., $c(D) \leq \Omega$, 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

$$\begin{array}{rrcl} \min / \max & \mathcal{M}(G[V \setminus D]) & & \\ & c(D) & \leq & \Omega \\ & D & \subseteq & V, \end{array} \tag{CNP}$$

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

2.2 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 allows for a prompt reaction which limits their spread. In this case, we assume that the attack has already taken place. Concretely, we denote by $I \subseteq V$ the subset of nodes that have been attacked. According to the viral propagation model and to 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, she 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, a s-t cut can be seen as a tool to design a protection strategy to defend a sink t from a possibly viral source s, by deleting a subset of arcs. Concretely, in Hayrapetyan et al. (2005) the authors introduced the *minimum-size bounded-capacity cut* problem, which calls for finding a cut that contains the source s, its capacity does not exceed a prescribed budget and whose size, in terms of the number of nodes of the cut, is minimized. The authors also introduced the vertex-separator version of this problem, that 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 equivalent in its general form to the minimum-size bounded-capacity vertex-cut problem. However, in order to make it compatible with our framework we need to rephrase it a bit. Moreover, for brevity, we refer to the following version of the minimum-size bounded-capacity vertex-cut problem 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 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 v to any of the nodes in I. We denote by $S(I, P) \subseteq V$ the set of saved nodes from I by P. Moreover, let $w(v) \in \mathbb{Z}_{\geq 0}$, for every $v \in V$, be the weight associated to v, which represents the reward obtained if v gets saved. The Fence problem asks, for an initial set of attacked node I, to solve the following problem:

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

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

2.3 Combining vaccination and protection

As anticipated, our goal is to take into consideration a network design problem based on the assumption that the network operator can intervene before and after an attack takes place. In particular, we would like to understand how a preventive strategy should be performed in this scenario. We formulate our protection problem as a trilevel defender-attacker-defender model motivated by the work by Brown et al. (2006) in which the authors formulated a model of defence, attack, and operation of an infrastructure system using a three-stage, sequential game.

We now combine the vaccination and protection strategies presented in Section 2.1 and 2.2. We make use of the definitions and the notation we established before. Recall that for a given set of attacked nodes I, the Fence problem asks for finding a protection strategy $P \subseteq V \setminus I$ that maximizes the weight of the saved set $w(S(I,P))$. This value strongly depends on the set of nodes I that the attacker chooses to infect. Given her full knowledge of the graph topology and of the defender's budget for protection in the third stage, the attacker will thus choose the nodes that mostly affect the network after the best blocking reaction is used. Moreover, the attack's intensity is limited by a budget constraint. Let $\Phi \in \mathbb{Z}_{\geq 0}$ be the attacker's budget and let $h(v) \in \mathbb{Z}_{\geq 0}$, for every $v \in V$, be the cost incurred for infecting v. Hence, the attacker target is modeled by the following bilevel problem:

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

Finally, as assumed by the model, the defender precedes the attacker by vaccinating a part of the network. More precisely, it is allowed to vaccinate a subset of node $D \subseteq V$, whose cost does not exceed a budget $\Omega \in N$. In the first stage, the defender vaccinates 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. Thus, the goal of the defender is to find the set $D \subseteq V$ that solves the following problem:

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

We indicate the latter problem as the *Multilevel Critical Node* (MCN) problem. Notice that, once a node is selected by some strategy, i.e., in one of the three stages, its choice becomes interdicted for all the process.

The latter trilevel problem actually extends the Critical Node problem and the Fence problem, at the same time. On the one hand, if we assume $\Lambda = 0$, set the cost for attacking nodes to be unitary, namely, $h(v) = 1 \ \forall v \in V$, and look at the problem from the defender's perspective, the derived max-min problem finds a solution for the Critical Node problem for which the metric considered is the sum of the total weights of the Φ heaviest components. On the other hand, if we assume $\Omega = 0$, for a fixed attack $I \subseteq V$, the resulting problem is exactly the Fence problem.

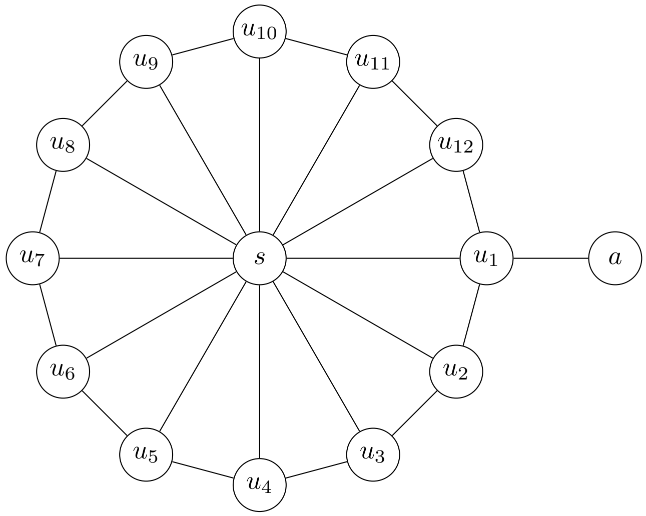
2.4 Measuring three-stage approach effectiveness

In order to check the effectiveness of our trilevel model with respect to an approach that considers preventive and blocking strategies separately, we introduced 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 no budget for protection is given, i.e., $\Lambda = 0$, 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.

Figure 1: An example for constructing an instance whose *relGain* is arbitrarily close to 1



Next, we show that *relGain* can be arbitrarily close to 1, revealing the advantage of considering the trilevel model. Let $m \in \mathbb{Z}_{\geq 5}$. We consider a directed graph $G = (V, A)$ built as follows. V contains nodes u_i, $i \in \{1, \ldots, m\}$, s, and a. All the nodes u_i, $i \in \{1, \ldots, m\}$, form a wheel whose center is s. More precisely, we first connect in both directions every node u_i to u_{i+1}, $i \in \{1, \ldots, m-1\}$, and u_1 to u_m, so as to obtain a cycle. Then, we connect in both directions every node u_i, $i \in \{1, \ldots, m\}$, to s. Finally, we connect in both direction u_1 to a. Figure 1 shows an example for $m = 12$. Furthermore, let us assume unitary cost and weight for every node, i.e., $c(v) = h(v) = w(v) = 1 \ \forall v \in V$, and $\Omega = 1$, $\Phi = 1$, and $\Lambda = 2$. On the one hand, an optimal solution for MCN is characterized by the vaccination strategy $\{s\}$ whose corresponding value OPT is m. For instance, once s is vaccinated, the most effective attack strikes the node u_1. The defender can then protect the pair $\{u_m, u_2\}$, saving, at the end of the process, m nodes by loosing only u_1 and a. Moreover, it is easy to verify that any vaccination strategy that differs from s leads to a worse scenario for the defender. On the other hand, assuming $\Lambda = 0$, MCN can be reduced to the problem that asks for finding a node whose removal minimizes the size of the biggest connected component. An optimal solution corresponds to select u_1, since it is the only option that disconnects the graph into two components. Once u_1 gets vaccinated, it is easy to verify that the best possible attacking strategy for (AP) corresponds to strike $\{s\}$. Then, any selection of two nodes among $\{u_2, \ldots, u_m\}$ is equivalent, leading to a value $VA\_AP = 4$. Finally, notice that $relGain = \frac{m-4}{m}$, which can be arbitrarily close to 1 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. 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 $v \in V$:

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

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

Note that a node $v \in V \setminus I$ is saved if either 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 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:

$$
\begin{array}{llll}
\max\limits_{z \in \{0,1\}^V} & \sum\limits_{v \in V} \alpha_v & & \\
& \sum\limits_{v \in V} z_v \leq \Omega & & \\
& \text{where } \alpha_v \text{ solves the attacker's problem} & & \\
& \min\limits_{y \in \{0,1\}^V} \quad \sum\limits_{v \in V} \alpha_v & & \\
& \qquad \sum\limits_{v \in V} y_v \leq \Phi & & \\
& \qquad \text{where } \alpha_v \text{ solves the defender's problem} & & \\
& \qquad \max\limits_{x \in \{0,1\}^V} \quad \sum\limits_{v \in V} \alpha_v & & \\
& \qquad\qquad \sum\limits_{v \in V} x_v \leq \Lambda & & \\
& \qquad\qquad \alpha_v \leq 1 + z_v - y_v & \forall v \in V & \\
& \qquad\qquad \alpha_v \leq \alpha_u + x_v + z_v & \forall\ (u,v) \in A & \\
& \qquad\qquad 0 \leq \alpha_v \leq 1 & \forall v \in V. &
\end{array}
\tag{3lvMIP}
$$

It is not difficult to see that the latter trilevel 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 was vaccinated, protected or all 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, since 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 1, i.e., saved. Furthermore, we also remark that requiring α to be binary is not necessary, since this is already enforced by the model.

Bilevel and trilevel programs are in general very difficult objects. 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 this 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 $\mathcal{U} = \left\{y \in \{0,1\}^V \mid \sum_{v \in V} y_v \leq \Phi\right\}$ be the set of all possible feasible attack strategies. With respect to the definition of x and α given above, we denote by $x(y)$ and $\alpha(y)$, respectively, the indicator variables for the protection level in response to the attack scenario $y \in \mathcal{U}$. Concretely, we define two indicator vectors $x(y), \alpha(y) \in \{0,1\}^V$ for any possible feasible attack $y \in \mathcal{U}$. Thus, (3lvMIP) is equivalent to the following MILP:

$$
\begin{array}{lrcll}
\max & \Delta & & & \\
& \Delta & \leq & \sum\limits_{v \in V} \alpha_v(y) & \forall y \in \mathcal{U} \\
& \sum\limits_{v \in V} z_v & \leq & \Omega & \\
& \sum\limits_{v \in V} x_v(y) & \leq & \Lambda & \forall y \in \mathcal{U} \\
& \alpha_v(y) & \leq & 1 + z_v - y_v & \forall v \in V,\ \forall y \in \mathcal{U} \\
& \alpha_v(y) & \leq & \alpha_u(y) + x_v(y) + z_v & \forall\ (u,v) \in A,\ \forall y \in \mathcal{U} \\
& 0 \leq \alpha_v(y) & \leq & 1 & \forall v \in V,\ \forall y \in \mathcal{U} \\
& z & \in & \{0,1\}^V & \\
& x(y) & \in & \{0,1\}^V. &
\end{array}
\tag{1lvMIP}
$$

Clearly, every element of $\mathcal{U}$ contributes to the size of the model $\Theta(|V|)$ variables and $\Theta(|V| + |A|)$ constraints. Since the size of $\mathcal{U}$ is $\Omega(|V|^{\Phi})$, the size of (1lvMIP) gets prohibitively large for efficient computations on any nontrivial graph. Thus, any reasonable approach has to take into account fewer scenarios in $\mathcal{U}$. More precisely, to follow this approach, it is necessary to find a good subset of attack strategies $\mathcal{Q} \subseteq \mathcal{U}$ for which ($1\text{lvMIP}_{\mathcal{Q}}$), i.e., (1lvMIP) restricted to $\mathcal{Q}$, still returns an optimal solution. In fact, a natural approach to arrive at an optimal solution, consists of repetitively solving ($1\text{lvMIP}_{\mathcal{Q}}$), starting from the empty set $\mathcal{Q} = \emptyset$ and adding each time a new attack strategy $y \in \mathcal{U}$ to $\mathcal{Q}$, until a guarantee of optimality is reached. The selection of any $y \in \mathcal{U}$ is done according to the impact that it potentially provides to the quality of the solution at the next step. Any new scenario added to $\mathcal{Q}$ increases the size of ($1\text{lvMIP}_{\mathcal{Q}}$) by $2|A|$ variables and $2|V| + |A| + 2$ constraints. In other words, this can be interpreted as a simultaneous column and row generation approach. Column and row generation approaches have been extensively studied in the literature, see for example Frangioni and Gendron (2009), Muter et al. (2013). This high-level approach can be formalized as follows and is guaranteed to return an optimal solution for (1lvMIP):

1. Initialize an empty set of scenarios $\mathcal{Q} = \emptyset$.
2. Solve ($1\text{lvMIP}_{\mathcal{Q}}$) (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), given the attack y and the initial vaccination z^{best}, the number of nodes surviving is strictly less than $best$, add y to $\mathcal{Q}$ and go to step 2; otherwise, return $best$ and z^{best}.

This approach is typical 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 to solve 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 where the follower's problem is linear have been already 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). We will see in the next section 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, since 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 $best$, thus it can be added to $\mathcal{Q}$, or it is optimal for (AP) and its value equals 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 the optimal strategy for MCN. For all the other strategies that are not optimal for MCN, we only require to find 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 subsection by presenting the subroutine used for the (AP) problem, $\text{AP}(V, A, \Phi, \Lambda, goal)$. We anticipate that $\text{AP}(V, A, \Phi, \Lambda, goal)$ receives as input, together with the budget variables and the value $goal$, 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 $goal$. The value $goal$ represents the quality that an attacking strategy should outmatch in order to be outputted without proof of optimality. In particural, $goal$ represents the value $best$. However, since (AP) works on the induced subgraph $G[V \setminus D]$, $goal$ has to not count towards the already saved nodes D, namely $goal = best - |D|$. In case $goal$ is strictly smaller than

MCN$(V, A, \Omega, \Phi, \Lambda)$: Solving the Multilevel Critical Node problem

$\mathcal{Q} \leftarrow \emptyset$, $best \leftarrow |V|$, $D \leftarrow \emptyset$.
While True :
  $goal \leftarrow best - |D|$.
  $(I,\ status) \leftarrow \mathrm{AP}(V_D, A_D, \Phi, \Lambda, goal)$.
  If $status =$ "opt"**:**
    Return $(D, best)$.
  If $status =$ "goal"**:**
    $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{I\}$.
    $(D = V^{z^{best}}, best) \leftarrow$ solve $(1lvMIP_{\mathcal{Q}})$.

optimal value for (AP), and hence *goal* is an underestimator of the optimal value, $\mathrm{AP}(V, A, \Phi, \Lambda, goal)$ 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. Concretely, in case the value of the strategy is smaller or equal than *goal*, the corresponding status is "goal". Otherwise, if the value of the strategy is strictly greater than *goal*, the corresponding status is "opt", since 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.

3.1 Solving the Attack-Protect problem

A bilevel formulation for the Attack-Protect problem (AP) is modeled by the second and the third level in 3lvMIP:

$$
\begin{aligned}
\min_{y \in \{0,1\}^V} \quad & \sum_{v \in V} \alpha_v \\
& \sum_{v \in V} y_v \leq \Phi \\
& \text{where } \alpha_v \text{ solves the defender's problem} \\
& \max_{x \in \{0,1\}^V} \quad \sum_{v \in V} \alpha_v \\
& \qquad \sum_{v \in V} x_v \leq \Lambda \\
& \qquad \alpha_v \leq 1 - y_v \qquad \forall v \in V \\
& \qquad \alpha_v \leq \alpha_u + x_v \qquad \forall\ (u, v) \in A.
\end{aligned}
\tag{2lvAP}
$$

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

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

with its corresponding dual problem,

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

Since, 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

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

The above mathematical program can be linearize 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 as $\gamma_v = (1-y_v)h_v$, for every $v \in V$. Since we deal with a minimization problem, it suffices to consider the two under-estimators $\gamma_v \geq 0$ and $\gamma_v \geq h_v - |V|y_v$, where we use the valid upper bound $h_v \leq |V|$. The latter upper bound follows by summing up 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 $\sum_{v\in V} h_v = |V|$, since variables $q_{(u,v)}$ cancel out for all $(u,v) \in A$. Finally, applying the latter linearization we obtain the following MILP:

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

The optimal objective value of (rlxAP) provides only an upper bound to (2lvAP). This is due to the fact that the relaxation of the third level makes the defense response more effective against any possible attack. Moreover, even in case that the solution is integral, there is no guarantee 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 solving the third level (Fence problem) once the attack strategy V^y is fixed. The optimal saved region obtained by the defensive strategy, i.e., 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 due to the fact that, 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

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

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 preceding arguments, the following procedure represents a possible approach to find 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 presented above.

As mentioned before, the algorithm we are developing for the MCN problem does not always require to find 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 $goal = best - |D|$ 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 above 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 to know whether the strategy returned by the subroutine has been outputted because of optimality or because it attains the required quality. As we already said 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$(V, A, \Phi, \Lambda, goal)$. 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) can be easily modeled as a MILP and solved by any MILP solver.

AP$(V, A, \Phi, \Lambda, goal)$: Subroutine for (AP).

```
best ← |V|,  I^best ← ∅.
While (rlxAP) is feasible:
    (I = χ^y, value) ← solve (rlxAP).
    If value ≤ goal − 1:
        Return (I, "goal").
    S ← P(V, A, I, Λ).
    If |S| ≤ goal − 1:
        Return (I, "goal").
    If |S| < best:
        best ← |S|,  I^best ← I.
    Add  ∑_{v∈S} y_v ≥ 1  to (rlxAP).
Return (I^best, "opt").
```

4 Computational Results

In this section we present some computational experiments for the algorithm we designed for the MCN problem. Algorithm MCN($V, A, \Omega, \Phi, \Lambda$) was implemented in Python 2.7.6, all MILPs have been solved with IBM ILOG CPLEX 12.7.1.0 and experiments were conducted on Intel Xeon E5-2637 processor clocked at 3.50GHz and 8 GB RAM, using a single core.
The 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 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 possible number of edges in a complete graph.

In both cases, we consider edges, i.e., whenever the arc (u, v) exists in an instance, also arc (v, u) exists. Moreover, test instances vary with respect to the number of nodes, density, and to availability of different budgets, Ω, Φ, and Λ, for the different stages of the problem.

We tested these instances with three different frameworks:

- MCN: our algorithm MCN;
- MCN^{MIX++}: MCN framework with Algorithm AP replaced by the bilevel code MIX ++[1] proposed in Fischetti et al. (2017) [2], 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 were not sufficient to attain the *goal*.

Our computational experiments investigate both the performance of our framework in terms of running time and iterations, and the relative gain defined in (1). Recall that the relative gain measures the extent to which the trilevel approach impacts the quality of an optimal solution compared to the solution obtained by solving to optimality the different levels sequentially. The relative gain, as defined in (1) is a value in the interval $[0, 1]$, but here we present it as a percentage, namely a value in the interval $[0, 100]$.

For every regimes we generated and tested 20 instances classified by the type of graph, the size, and the budget vector (Ω, Φ, Λ). We then report the average values of a certain set of attributes (see below) for these 20 instances. Every computation was aborted after a time limit of 2 hours. We reported the average values of the attributes we considered only for the instances successfully solved to optimality.
The following list summarizes the notation of the considered attributes reported 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 time limit (2 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 relative gain for instances with nonzero relative gain.
- **max**: maximum relative gain observed for the solved instances.

To complement the tables, performance profile plots for the 3 frameworks will be presented. These plots will be presented twice: one with a linear scale, which enables to see the percentage of instances that each framework solves within the time limit, and another with logarithmic scale, which allows us to determine the percentage of instances solved in less time by each framework.

Our computational analysis starts by comparing different instances with identical graph topology but with different size and vector of budgets. Figures 2 and 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 the three frameworks, instances get harder to solve as the number of nodes and the 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 the third levels, Ω and Λ. This is logic: 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 while 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 solution are only required to match the quality of some given 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 (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 due to 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 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 reveals to be slower than using Algorithm AP when only a *goal* has to be attained. For what concerns the relative gain, it seems that, for sufficiently large trees, there is a high probability the instance has a nonzero relative gain if the protection budget Λ is high.

Figures 4 and 5 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 algorithms performance declines when the attack budget Φ increases, while 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 is 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 explored 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 build on optimal (AP) solutions determined by MIX ++; on the other hand, the non necessarily optimality of the (AP) solutions used to build the cuts (cutAP) for intermediated iterations of MCN and HIB increases their total number of iterations; therefore, more time is spend in MCN and HIB intermediated iterations, reducing the impact of the last iteration computational cost over the total running time of MCN and HIB.
Figure 5 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 5b, 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 5a, it is the approach that can solve more instances within the time limit of 2 hours. As previously observed, it seems that MCN^{MIX++} benefits from the fact that it always finds attacker's optimal solutions which results in less iterations of the overall framework. Apart from small sizes, random graphs of density 5% show much higher relative gains compared to tree instances. In general, computations show many instances with nonzero relative gains and, on average, they are significant.

Finally, Figures 6 and 7 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 Figure 6, such as running time and relative gain. However, if we restrict our investigation to particular fixed budget vectors, we can point out some interesting

Figure 2: Detailed results on tree instances

Legend

- MCN
- MCN^{MIX++}
- HIB

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

size	Ω-Φ-Λ	sol%	time(s)	AP%	it	RG%	avg	max
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
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

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%.

5 Conclusions

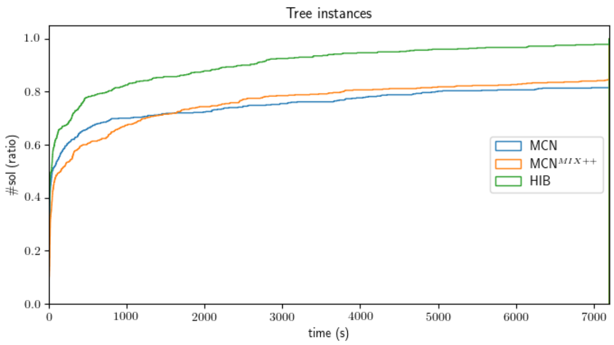
In this work we introduced a new three-stage model for protecting a network from a harmful spreading agent, based on a combination of the Critical Node and the Fence problems, which we call the Multilevel Critical Node problem. We devised an exact algorithm for this problem and tested its performance on different types of randomly generated graph structures. 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 problem sequentially.

In this line of work, devising faster algorithms for MCN is a interesting direction for future works. Moreover, replacing the Fence problem with the Firefighter problem might be a possible way to describe more sophisticated models, which better capture some specific dynamics of network defending. Additionally, simultaneous vertex and edge removal, or only edge removal, can be considered instead.

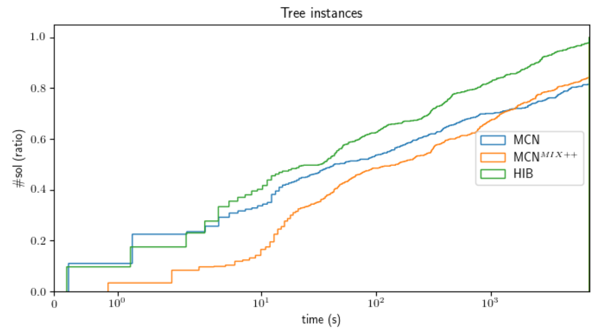
Acknowledgments

The first thanks the support by an EU grant FP7-PEOPLE-2012-ITN no. 316647, "Mixed-Integer Nonlinear Optimization". The first author contribution was performed while he was at IFOR in the Department of

Figure 3: Performance profile on tree instances.



(a) Linear scale for time.



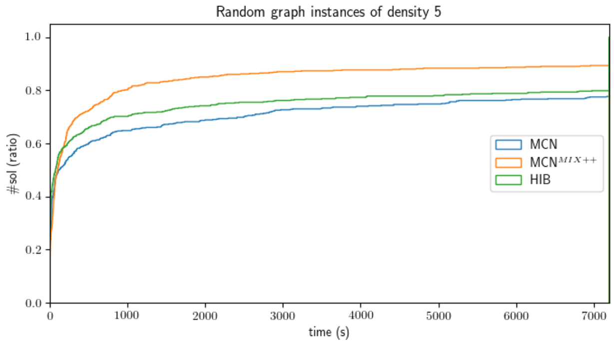
(b) Logarithmic scale for time.

Figure 4: Detailed results on random graphs of density 5%

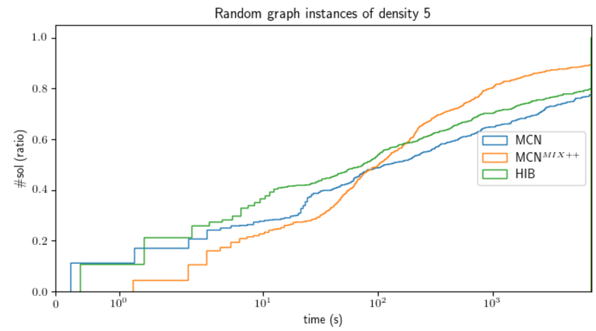
Legend
MCN
MCN^{MIX++}
HIB

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
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	-	-	-	-	-	-

Figure 5: Performance profile on random graphs of 5% density.



(a) Linear scale for time.



(b) Logarithmic scale for time.

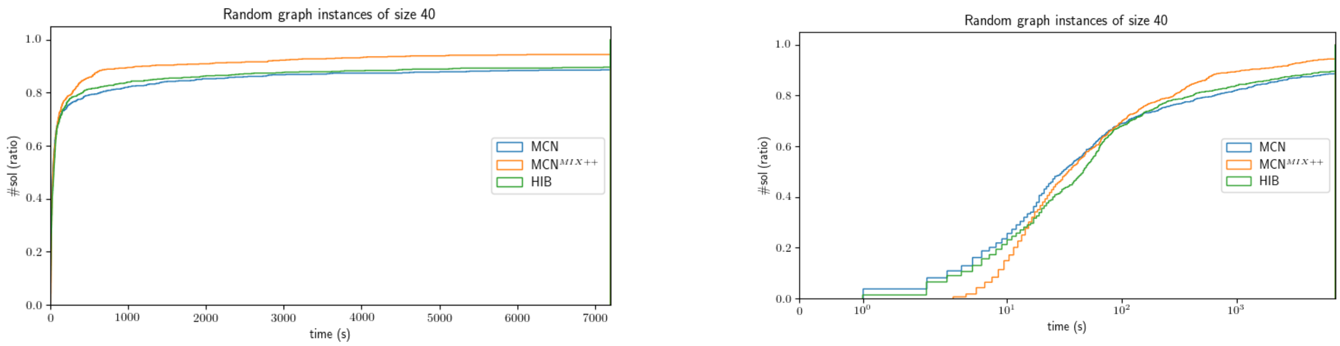
Figure 6: Detailed results on random graphs of size 40

Legend MCN MCN^{MIX++} HIB

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

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

Figure 7: Performance profile on random graphs of size 40.



(a) Linear scale for time.

(b) Logarithmic scale for time.

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 the Faculty of Sciences University of Porto and INESC TEC. The second author thanks the support of Institute for data valorisation (IVADO), the Portuguese Foundation for Science and Technology (FCT) through a PhD grant number SFRH/BD/79201/2011 and the ERDF European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme within project POCI-01-0145-FEDER-006961, and National Funds through the FCT (Portuguese Foundation for Science and Technology) as part of project UID/EEA/50014/2013.

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 Applied Mathematics* 161:2349–2360.

Alderson DL, Brown GG, Carlyle WM, Wood RK (2011) Solving defender-attacker-defender models for infrastructure defense. INFORMS, ed., *12th INFORMS Computing Society Conference*.

Assimakopoulos N (1987) A network interdiction model for hospital infection control. *Computers in Biology and Medicine* 17(6):413–422.

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

Bard JF, Moore JT (1990) A branch and bound algorithm for the bilevel programming problem. *SIAM Journal on Scientific and Statistical Computing* 11(2):281–292.

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

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

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

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

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

Church RL, Scaparra MP, Middleton RS (2004) Identifying critical infrastructure: the median and covering facility interdiction problems. *Annals of the Association of American Geographers* 94(3):491–502.

DeNegre S (2011) Interdiction and discrete bilevel linear programming .

DeNegre S, Ralphs TK (2009) A branch-and-cut algorithm for integer bilevel linear programs. *Operations Research and Cyber-Infrastructure*, 65–78 (Springer).

Di Summa M, Grosso A, Locatelli M (2011) Complexity of the critical node problem over trees. *Computers & Operations Research* 38:1766–1774.

Dinitz M, Gupta A (2013) Packing interdiction and partial covering problems. *International Conference on Integer Programming and Combinatorial Optimization*, 157–168 (Springer).

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

Fischetti M, Ljubić I, Monaci M, Sinnl M (2016a) Interdiction games and monotonicity, technical Report.

Fischetti M, Ljubić I, Monaci M, Sinnl M (2016b) Intersection cuts for bilevel optimization. *International Conference on Integer Programming and Combinatorial Optimization*, 77–88 (Springer).

Fischetti M, Ljubić I, Monaci M, Sinnl M (2017) A new general-purpose algorithm for mixed-integer bilevel linear programs. *Operations Research, Articles in Advance* 0(0):1 – 23, URL http://dx.doi.org/10.1287/opre.2017.1650.

Fomin FV, Golovach PA, Korhonen JH (2013) On the parameterized complexity of cutting a few vertices from a graph. *Mathematical Foundations of Computer Science 2013*, 421–432 (Springer).

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

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

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

Hansen P, Jaumard B, Savard G (1992) New branch-and-bound rules for linear bilevel programming. *SIAM Journal on Scientific and Statistical Computing* 13(5):1194–1217, URL http://dx.doi.org/10.1137/0913069.

Hartnell B (1995) Firefighter! an application of domination. *24th Manitoba Conference on Combinatorial Mathematics and Computing*.

Hayrapetyan A, Kempe D, Pál M, Svitkina Z (2005) Unbalanced graph cuts. *Algorithms–ESA 2005*, 191–202 (Springer).

Hemmati M, Smith JC (2016) A mixed-integer bilevel programming approach for a competitive prioritized set covering problem. *Discrete Optimization* 20:105 – 134, ISSN 1572-5286, URL http://dx.doi.org/http://dx.doi.org/10.1016/j.disopt.2016.04.001.

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

IBM ILOG CPLEX Optimizer (2017) URL http://www.cplex.com.

Israeli E, Wood RK (2002) Shortest-path network interdiction. *Networks* 40(2):97–111, ISSN 1097-0037, URL http://dx.doi.org/10.1002/net.10039.

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 of Computing Systems* 43(2):204–233.

Klein R, Levcopoulos C, Lingas A (2014) *Approximation Algorithms for the Geometric Firefighter and Budget Fence Problems*, 261–272 (Berlin, Heidelberg: Springer Berlin Heidelberg), ISBN 978-3-642-54423-1, URL http://dx.doi.org/10.1007/978-3-642-54423-1_23.

Lim C, Smith JC (2007) Algorithms for discrete and continuous multicommodity flow network interdiction problems. *IIE Transactions* 39(1):15–26, URL http://dx.doi.org/10.1080/07408170600729192.

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

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

Morton DP, Pan F, Saeger KJ (2007) Models for nuclear smuggling interdiction. *IIE Transactions* 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. *Mathematical Programming* 142(1-2):47–82.

Phillips CA (1993) The network inhibition problem. *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, 776–785 (ACM).

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

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

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 Optimization* 9:172–188.

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

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

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

Zenklusen R (2014) Connectivity interdiction. *Operations Research Letters* 42(6):450–454.

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