In this work, we analyze a sequential game played in a graph called the Multilevel Critical Node problem (MCN). A defender and an attacker are the players of this game. The defender starts by preventively interdicting vertices (vaccination) from being attacked. Then, the attacker infects a subset of non-vaccinated vertices and, finally, the defender reacts with a protection strategy. We provide the first computational complexity results associated with MCN and its subgames. Moreover, by considering unitary, weighted, undirected and directed graphs, we clarify how the theoretical tractability or intractability of those problems vary. Our findings contribute with new NP-complete, \( \Sigma_2^p \)-complete and \( \Sigma_3^p \)-complete problems.

1 Introduction

Multilevel Critical Node. Graphs are powerful mathematical structures that enable us to model real-world networks. The problem of breaking the connectivity of a graph has been extensively studied in combinatorial optimization since it can serve to measure the robustness of a network to disruptions. In this work, we will focus on the Multilevel Critical Node problem (MCN) [3]. Let \( G = (V, A) \) be graph with a set \( V \) of vertices and a set \( A \) of arcs. In MCN there are two players, designated by defender and attacker, whose individual strategies are given by a selection of subsets of \( V \). The game goes as follows: first, the defender selects a subset of vertices \( D \subseteq V \) to vaccinate subject to a budget limit \( \Omega \) and a cost \( \{\hat{c}_v\}_{v \in V} \); second, the attacker observes the vaccination strategy, and selects a subset of vertices \( I \subseteq V \setminus D \) to infect subject to a budget limit \( \Phi \) and a cost \( \{h_v\}_{v \in V} \); and third, the defender observes the infection strategy, and selects a subset of vertices \( P \subseteq V \setminus I \) to protect subject to a budget limit \( \Lambda \) and a cost \( \{c_v\}_{v \in V} \). Infected vertices propagate the infection to their neighbourhood, except to vaccinated or protected vertices. The goal of the defender is to maximize the benefit \( b_v \), of saved vertices (i.e., not infected), while the attacker aims to minimize it. We assume that all parameters of the problem are non-negative integers. The game description can be succinctly given by the following trilevel program:

\[
(MCN) \quad \max_{z \in \{0,1\}^{|V|}} \sum_{v \in V} \hat{c}_v z_v \leq \Omega \quad \min_{y \in \{0,1\}^{|V|}} \sum_{v \in V} h_v y_v \leq \Phi \quad \max_{x \in \{0,1\}^{|V|}} \sum_{v \in V} b_v \alpha_v \quad (1a)
\]

\[
\sum_{v \in V} c_v x_v \leq \Lambda \quad (1b)
\]

\[
\alpha_v \leq 1 + z_v - y_v \quad \forall v \in V \quad (1c)
\]

\[
\alpha_v \leq \alpha_u + x_v + z_v \quad \forall (u, v) \in A \quad (1d)
\]

where \( D = \{v \in V : z_v = 1\} \), \( I = \{v \in V : y_v = 1\} \) and \( P = \{v \in V : x_v = 1\} \). See [3] for further details on this mathematical programming formulation.
Contributions. To the best of our knowledge, this work is the first providing a computational complexity classification of the decision version of MCN, as well as of its subgames. Namely, we investigate the subgames (i) PROTECT, where given $D$ and $I$, the defender seeks the optimal protection strategy, (ii) ATTACK, where given $D$ and no protection budget, the attacker determines the optimal infection strategy, (iii) ATTACK-PROTECT, where given $D$, the attacker computes the optimal infection strategy, and (iv) VACCINATION-ATTACK, where given no budget for protection, the defender finds the optimal vaccination strategy. This fundamental contribution sheds light on the practical difficulties dealt in [3]. Furthermore, it contributes to the understanding of sequential combinatorial games within the polynomial hierarchy and it motivates the focus on potentially $\Omega(2^{2V})$ algorithms, heuristic methods or novel solution definitions. Table 1 summarizes our results; unitary cases assume that all costs and benefits are 1.

| Decision Versions | Undirected Graphs | Directed Graphs |
|-------------------|-------------------|-----------------|
|                    | Unitary Case      | Weighted Case   | Unitary Case      | Weighted Case   |
| PROTECT            | [1] NP-complete   | [6] NP-complete | [11] NP-complete | [16] NP-complete|
| ATTACK             | [2] Polynomial    | [7] NP-complete | [12] NP-complete | [17] NP-complete|
| ATTACK-PROTECT     | [3] NP-hard       | [8] $\Sigma_2^P$-complete | [13] NP-hard | [18] $\Sigma_2^P$-complete|
| VACCINATION-ATTACK | [4] NP-complete   | [9] $\Sigma_2^P$-complete | [14] $\Sigma_2^P$-complete | [19] $\Sigma_2^P$-complete|
| MCN                | [5] NP-hard       | [10] $\Sigma_2^P$-complete | [15] $\Sigma_2^P$-hard | [20] $\Sigma_2^P$-complete|

Table 1: Computational complexity of the decision versions of the subproblems in MCN. Entries in gray correspond to results that follow as corollaries. In increasing order, we have: [4] $\Rightarrow$ [5], [1] $\Rightarrow$ [6], [12] $\Rightarrow$ [13], [14] $\Rightarrow$ [15], and [6-10] $\Rightarrow$ [16-20].

Paper Organization. In Section 2, we revise the literature associated with MCN, allowing to position our contribution in the context of critical node problems. In Section 3, we focus on the case where graphs are undirected and each vertex benefit and cost is unitary. Section 4 adds the possibility of having non-unitary parameters, while Section 5 generalizes the game to directed graphs. Finally, Section 6 investigates structural properties of special directed graph classes that can be explored to make at least PROTECTION polynomially solvable.

2 Related literature

Assessing the vulnerability of complex infrastructures such as networks is of the utmost importance in practice. One way to measure the robustness of a given network is to study its connectivity properties, for which many metrics exist. With respect to a fixed metric, vertices often play different roles in the graph, with varying levels of importance. The most important vertices are qualified as critical. Thus, the problem of detecting subsets of critical vertices with respect to some connectivity measure is of great interest, either for defensive or for offensive purposes, and with applications in domains ranging from network immunization [2, 16] to computational biology [5, 26].

Critical Node Detection Problems (CNDP). The CNDPs have been extensively studied, with names varying with the connectivity metric to optimize and the constraints of the problem. Many of its studied versions have been shown to be NP-complete on general graphs; see Lalou et al. [20] for a recent survey. Indeed, many of these belong to the class of problems called Node-Deletion Problems [20]. They consist in deleting the smallest subset of vertices from a graph so that the induced subgraph satisfies a certain property $\pi$. Lewis and Yannakakis [21] have shown that if $\pi$ is nontrivial and hereditary, then the subsequent node deletion problem is NP-hard. In particular, MinMaxC, the problem of finding a set of vertices $D$ from a graph $G$ with a budget constraint $|D| \leq \Omega$ such that the removal of $D$ minimizes the size of the largest connected component in the remaining graph, has been shown to be NP-hard in the strong sense thanks to this argument [24]. Moreover, some CNDP problems remain NP-hard even on particular graph classes [1, 20]. For example, the original Critical Node Problem (CNP) [2] which seeks to minimize the pairwise connectivity of the graph by removing a limited number of vertices remains NP-hard on split or bipartite graphs [1].

Interdiction Games. In several CNDP, although the optimization problem is formulated with a natural single objective, the task is inherently constituted of several ones. In the CNP, minimizing the pairwise connectivity maximizes the number of connected components in the residual graph, while simultaneously minimizing the variance in the component sizes [2]. Even though in this particular case, it has been shown that the multi-objective formulation is not equivalent to the original one [27], splitting the objective in two is sometimes possible. For example, Furini et al. [13]
exhibited the hidden bilevel structure of the \textit{Capacitated Vertex Separator problem} by formulating it as a two player Stackelberg game in which a leader interdicts the network by removing some of its vertices and a follower determines the maximum connected component in the remaining graph, highlighting the link between CNDP problems and \textit{Interdiction Games}. Interdiction games on networks are a special family of two-player zero-sum Stackelberg games in which a leader interdicts parts of the network \textit{(arcs or vertices)} subject to a budget limitation in order to maximize the disruption of the follower’s objective who solves an optimization problem on the remaining graph \textit{(e.g., the maximum flow or the maximum clique)}. Whereas some interdiction games such as the \textit{network flow interdiction} are NP-complete \cite{28}, others such as the \textit{binary knapsack interdiction problem} \cite{10, 7} or the \textit{maximum clique interdiction game} \cite{14} have been shown to be $\Sigma^P_2$-complete, shading light on the intrinsic relationship between this class of problems and the second level of the polynomial hierarchy.

However, the unitary undirected version of MCN, as originally introduced by Baggio \textit{et al.} \cite{3}, is not an interdiction problem per se but contains one. Indeed, the vaccination stage of the game focuses on identifying critical infrastructures in the network to interdict them preventively to counter an intentional attack, which falls into the framework of Network Interdiction problems. Nevertheless, the game does not finish with the attack: there is a third stage where the defender tries to isolate the propagation of the infection to maximize the unharmed fraction of the network. Finding a blocking strategy to limit the diffusion of an infection is related to the \textit{Firefighter problem}, which has been shown to be NP-complete, even for trees of maximum degree three \cite{12}. Thus, the MCN problem combines two different paradigms in network protection, \textit{prevention} and \textit{blocking}, each being related to provably hard problems. The overall contraction leads to a trilevel optimization formulation for the MCN, making it fall under the \textit{Defender-Attacker-Defender (DAD)} framework introduced by Brown \textit{et al.} \cite{6} to study the defense of critical infrastructure against malicious attacks.

**Defender-Attacker-Defender.** Although the general DAD has been claimed to be NP-hard in \cite{22}, complexity results for trilevel combinatorial optimization problems are scarce. In \cite{18}, a new proof that \textit{Trilevel Linear Programming} is $\Sigma^P_2$-hard is provided, building upon the results in \cite{4, 11, 17} showing that the \textit{Multilevel Linear Programming} problem with $L + 1$ levels is $\Sigma^P_2$-hard. In fact, the decision version of MCN problem can be formulated as "given 3 integer budgets $\Omega, \Phi, \Lambda$, a graph $G$ and an integer $K$, is there a vaccination $D$ such that for all attacks $I$ there exists a protection $P$ saving at least $K$ vertices?" Thus, there seems to be a link between the MCN and the 3-alternating quantified satisfiability problem which has been shown to be $\Sigma^P_3$-complete by Meyer, Stockmeyer and Wrathall \cite{23, 29}, making one expect the MCN to be complete for this class.

### 3 Undirected graphs: the unitary case

In this section, we focus on undirected graphs $G = (V, E)$, \textit{i.e.}, for each couples of vertices $(u, v) \in V \times V$, if the arc $(u, v)$ is in $G$, then $(v, u)$ is also in the graph. We thus call $E$ the set of undirected edges. Here, we also consider unit benefits and costs, \textit{i.e.}, $\forall v \in V$, $c_v = h_v = c_o = b_o = 1$. We introduce $s$, the function that, given a graph $G$, the vaccination strategy $D$, the attack strategy $I$ and the protection strategy $P$, returns $s(G, D, I, P)$, the number of saved vertices in the end of the game. Thus, in this setting, the trilevel formulation of the problem is simply:

$$
\max_{D \subseteq V \atop |D| \leq \Omega} \min_{I \subseteq V \setminus D \atop |I| \leq \Phi} \max_{P \subseteq V \setminus (I \cup D) \atop |P| \leq \Lambda} s(G, D, I, P).
$$

To ease our analysis, guided by the relationship between \textit{Critical Node Detection Problems} and \textit{Node-Deletion Problems}, we first write the immediate \textbf{Property 3.1} stating that \textit{vaccinating or protecting} vertices has almost the same effect as \textit{removing} them from the graph with respect to $s$. Starting from $G = (V, E)$ and a subset $W \subseteq V$, we denote by $G[V \setminus W]$ the graph resulting from the deletion of the vertices in $W$ and its incident edges.

**Property 3.1.** Given $G$, $D$, $I$, $P$, we have that $s(G, D, I, P) = s(G[V \setminus (D \cup P)], \emptyset, I, \emptyset) + |D| + |P|

What \textbf{Property 3.1} actually says is that the infected vertices in $G$ are the ones in the connected components of $G[V \setminus (D \cup P)]$ where there is at least one attacked vertex in $I$.

We will start by classifying the computational complexity of \textsc{Protect}, followed by the one of \textsc{Attack-Protect}, and, finally, \textsc{Vaccination-Attack}. From the latter, we obtain the complexity of \textsc{Attack}, and the minimum complexity of MCN.

#### 3.1 The Protect problem

In \textsc{Protect}, the defender is given $D$ and $I$ and seeks to find an optimal $P$. Thus, thanks to \textbf{Property 3.1}, we can assume that the game takes place in $G_o = G[V \setminus D]$ for this last move: the defender wants to find at most $\Lambda$ vertices
and a non-negative integer \( K \). Hence, the objective of the defender being to find \( P \) minimizing the number of infected vertices \( f(P) \), we can define it as:

\[
f(P) = \sum_{i=1}^{N(P)} |C_i(P)| \times \mathbb{1}_{C_i(P) \cap I \neq \emptyset}.
\]

We will show that finding such a \( P \) is NP-complete. We argue that it is a direct consequence of the results of [1] showing that the Critical Node Problem is NP-hard on split graphs.

### 3.1.1 The Critical Node Problem on split graphs

The Critical Node Problem (CNP) [2] is a related problem to ours. The setting is very similar to Protection: we have an undirected graph \( G = (V, E) \), an integer budget \( B \), and we want to find a subset \( \bar{P} \) of vertices to remove that minimizes the pairwise connectivity of the residual subgraph \( \bar{G}[\bar{V} \setminus \bar{P}] \) under the constraint of having \( |\bar{P}| \leq B \). If we denote by \( \bar{C}_1(\bar{P}), ..., \bar{C}_{N(\bar{P})}(\bar{P}) \) the \( N(\bar{P}) \) connected components of \( \bar{G}[\bar{V} \setminus \bar{P}] \), the measure we want to minimize is:

\[
g(\bar{P}) = \sum_{i=1}^{N(\bar{P})} \left( \left| \bar{C}_i(\bar{P}) \right| \right)^2
\]

where each term in the sum is the pairwise connectivity of \( \bar{C}_i(\bar{P}) \). Here, we will focus more particularly on split graphs. A split graph is a graph \( G = (V, E) \) whose vertices \( V \) can be split in two sets \( V_1 \) and \( V_2 \), \( V_1 \) forming a clique and \( V_2 \) an independent set. Thus, the decision problem for this particular case of the CNP is:

**CNP\text{\_split}:**

**INSTANCE:** A split graph \( \bar{G} = (\bar{V}_1, \bar{V}_2; \bar{E}) \), a non-negative integer budget \( B \leq |\bar{V}| \) and a non-negative integer \( \bar{K} \).

**QUESTION:** Is there a subset \( \bar{P} \subseteq \bar{V} \) of \( \bar{P} \leq B \) such that \( g(\bar{P}) \leq \bar{K} \)?

As [1] noted, in this setting there is at most one connected component of the residual subgraph \( \bar{G}[\bar{V} \setminus \bar{P}] \) that contains more than one vertex. Moreover, it is easy to see that if this nontrivial connected component exists, it necessarily contains a subgraph of \( \bar{G}[\bar{V}_1] \). More than that, it is the only connected component of \( \bar{G}[\bar{V} \setminus \bar{P}] \) containing vertices from \( \bar{V}_1 \). Thus, we can name \( \bar{C}_1 \) the connected component containing vertices of \( \bar{V}_1 \) (in the case of \( \bar{P} \supseteq \bar{V}_1 \), then \( \bar{C}_1 \) is either a singleton from \( \bar{V}_2 \) or is empty and our reasoning still holds). Then, minimizing (4) is equivalent to minimize \( |\bar{C}_1| \). But finding the subset of vertices \( \bar{P} \) to remove to do that has been shown to be NP-hard:

**Lemma 3.2.** [1] CNP\text{\_split} is NP-hard.

### 3.1.2 Complexity result

Next, we show that the decision version of Protect is NP-complete using a reduction from MCN\text{\_split}. The decision problem is the following:

**Protect:**

**INSTANCE:** A graph \( G_a = (V_a, E_a) \), a set of attacked vertices \( I \subseteq V_a \), a non-negative integer budget \( \Lambda \leq |V_a| - |I| \) and a non-negative integer \( K \).

**QUESTION:** Is there a subset \( P \subseteq V_a \setminus I \) of \( |P| \leq \Lambda \) such that the number of infected vertices \( f(P) \leq K \)?

Note that the question can be equivalently with the inequality \( s(G_a, \emptyset, I, P) \geq |V_a| - K \).

**Theorem 3.3.** Protect is NP-complete.

**Proof.** It is easy to see that Protect is NP as determining the objective value only requires finding the connected components of \( G_a[V_a \setminus P] \) which can be done in linear time using a depth-first search (DFS).

To complete the proof, we exhibit an immediate reduction from CNP\text{\_split}. Let us take an instance of this problem, i.e. a split graph \( \bar{G} = (\bar{V}_1, \bar{V}_2; \bar{E}) \), a non-negative integer budget \( B \) and a non-negative integer \( \bar{K} \). Given that, we build a graph \( G_a \) by growing by one the size of the clique \( \bar{G}[\bar{V}_1] \) with the addition of a vertex \( u \). Thus, \( V_a = \bar{V}_1 \cup \{u\} \cup \bar{V}_2 \) and \( E_a \) is obtained by taking \( \bar{E} \) and adding an edge \((u, v_1) \) \( \forall v_1 \in \bar{V}_1 \). In fact, the new graph is still a split graph \( G_a = (\bar{V}_1 \cup \{u\}, \bar{V}_2; E_a) \). Finally, the corresponding instance of Protect is given by \( G_a, I = \{u\}, \Lambda = B \) and \( K = \left\lfloor \frac{1}{2}(3 + \sqrt{8K + 1}) \right\rfloor \) (obtained by solving \( K = \left(\frac{K - 1}{2}\right) \)). An example of such construction can be found in Figure 1. Then, as there is only one attacked vertex, minimizing (3) on this instance of Protect corresponds to chose a \( P \) that minimizes the size of the unique connected component to which \( u \) belongs in \( G_a[V_a \setminus P] \). Let’s
name $C_1$ this connected component. But as $u$ belongs to the clique part of the split graph $G_a$, $C_1$ is also the unique connected component of $G_a[V_a\setminus P]$ containing vertices from $V_1 = V_1 \cup \{u\}$. Thus, we have that $C_1 = C_1 \cup \{u\}$ and $g(P) = \left(\frac{f(P) - 1}{2}\right)$. Hence, finding $P$ that minimizes $f$ on $G_a$ is equivalent to finding $P$ that minimizes $g$ on $\bar{G}$. This finishes the proof that PROTECT is NP-hard.

\[\square\]

![Figure 1: Example of construction of $G_a$ from $\bar{G}$](image)

### 3.2 The ATTACK-PROTECT problem

We showed that solving the last level of MCN is NP-complete, now we will prove that ATTACK-PROTECT is also NP-hard. In this bilevel problem, we are taking the side of the attacker: the aim is to find the attack that will maximize the number of infected vertices after protection. The decision version of the problem is:

**ATTACK-PROTECT:**

**INSTANCE:** A graph $G_a = (V_a, E_a)$, two non-negative integer budgets $\Phi, \Lambda$ such that $\Phi + \Lambda \leq |V_a|$ and a non-negative integer $K \leq |V_a|$

**QUESTION:** Is there a subset $I \subseteq V_a$, $|I| = \Phi$ such that $\forall P \subseteq V_a \setminus I$, $|P| \leq \Lambda$, the number of infected vertices $f(P) \geq K$?

We will use a reduction from the Dominating Set problem, a known NP-complete problem [15], whose decision version is:

**DOMINATING SET:**

**INSTANCE:** A graph $\bar{G} = (\bar{V}, \bar{E})$, a positive integer $B \leq |\bar{V}|$

**QUESTION:** Is there a subset $U \subseteq V$, $|U| \leq B$, such that $\forall v \in \bar{V} \setminus U$, $\exists u \in U$ such that $(u, v) \in \bar{E}$?

**Theorem 3.4.** ATTACK-PROTECT is NP-hard.

**Proof.** Let us take a graph $\bar{G} = (\bar{V}, \bar{E})$ and a positive integer $B \leq |\bar{V}|$. The instance of ATTACK-PROTECT is simply created by taking $G_a = \bar{G}$, $\Phi = B$, $\Lambda = |V_a| - \Phi - 1$ and $K = \Phi + 1$. In this configuration, we have a protection budget $\Lambda$ which is exactly one less than the number of vertices that are not attacked. Thus, if all the protection budget is spent, there is only one vertex $u$ in the graph that is neither attacked nor protected. Therefore, if $u$ becomes infected after protection (i.e., $f(P) = K = \Phi + 1$), that means that the protection strategy did not manage to save one unit of budget while saving all the other vertices, meaning that the other vertices were all in direct contact with at least one attacked one (if it was not the case, one unit of budget could have been saved by protecting all the neighbors of the vertex that is not in direct contact with $I$). As $u$ also becomes infected, it also means that it is adjacent to one vertex in $I$. Thus, finding $I$ such that $f(P) \geq K$ means that $I$ is a dominating set of size $B$, which concludes the proof.

\[\square\]

### 3.3 The VACCINATION-ATTACK problem

In this part, we will ignore the fact that there is a protection stage at the end. This is a particular case of MCN since it is equivalent to studying it with protection budget $\Lambda = 0$. We will show that the bilevel problem VACCINATION-ATTACK is NP-complete. The decision problem is the following:

**VACCINATION-ATTACK:**

**INSTANCE:** A graph $G = (V, E)$, two non-negative integer budgets $\Omega$ and $\Phi$ such that $\Omega + \Phi \leq |V|$ and a non-negative integer $K$.

**QUESTION:** Is there a subset $D \subseteq V$, $|D| \leq \Omega$ such that $\forall I \subseteq V \setminus D$ with $|I| \leq \Phi$, the number of infected vertices $|V| - s(G, D, I, \emptyset) \leq K$?
First, we argue that in this configuration, finding the optimal attack following a given vaccination can be done in polynomial time.

**Lemma 3.5.** \textsc{Vaccination-Attack} $\in$ \textsc{NP}. Moreover, \textsc{Attack} can be solved in polynomial time.

**Proof.** Given a vaccinated set $D$, we want to verify that all the possible subsequent attacks cannot infect more than $K$ vertices. To do that, it suffices to find the best attack, i.e., solve the Attacker optimization problem, and check whether or not it complies with the inequality. But, as we highlighted it with Property 3.1, the graph on which the attack phase takes place is $G_a = G[V \setminus D]$ and the saved vertices in the end are exactly the ones in the connected components of $G_a$ that do not contain any attacked vertex. Thus, the best attack possible given $G_a$ and budget $\Phi$ is to infect one vertex in each of the $\Phi$ largest connected components of $G_a$. This can be done in linear time using a DFS. Hence, \textsc{Vaccination-Attack} $\in$ \textsc{NP}. 

In fact, this proof showed that \textsc{Vaccination-Attack} is actually equivalent to another problem: finding a subset of vertices $D$ to remove from $G$ that minimizes the sum of the sizes of the $\Phi$ largest connected components in the induced subgraph. Let’s call this problem \textsc{MinMaxC}: 

**MinMaxC:**

**INSTANCE:** A graph $G = (V, E)$, two non-negative integer budgets $\Omega$ and $\Phi$ such that $\Omega + \Phi \leq |V|$ and a non-negative integer $K$.

**QUESTION:** Is there a subset $D \subseteq V$, $|D| \leq \Omega$ such that the sum of the sizes of the $\Phi$ largest connected component in $G[V \setminus D]$ is less than $K$?

**Lemma 3.6.** \textsc{Vaccination-Attack} and \textsc{MinMaxC} are equivalent problems.

As Shen et al. [24] argued that \textsc{MinMaxC}, the problem that only seeks to minimize the size of the largest connected component in the residual graph, is \textsc{NP}-hard, we have as a direct consequence that \textsc{MinMaxC} is also \textsc{NP}-hard, which leads to the following corollaries:

**Corollary 3.7.** \textsc{Vaccination-Attack} is \textsc{NP}-complete.

**Corollary 3.8.** \textsc{MCN} is \textsc{NP}-hard.

**Proof.** Given an instance of \textsc{Vaccination-Attack}, there is a corresponding instance of \textsc{MCN} by taking the same $G, \Omega, \Phi, K$ and by setting $\Lambda = 0$. 

4 Undirected graphs: the weighted case

In this section, we study the version of \textsc{MCN} presented in problem (1) restricted to undirected graphs. We will use the subscript $w$ to denote the weighted version, \textsc{MCN}_w, as well as its subgames. In this problem, given a graph $G = (V, E)$, each vertex $v \in V$ is associated with a benefit $b_v$ and cost parameters $c_v, h_v$ and $e_v$, respectively the cost of vaccinating, attacking and protecting vertex $v$. Having introduced costs and benefits, this problem is thus intimately related to \textsc{Knapsack problems}, which we will use to demonstrate all of our complexity results in this part. First, we will highlight the direct relationship between \textsc{Attack}_w and \textsc{Knapsack}, which will get us the \textsc{NP}-completeness of this problem. Then, we will focus on the two bilevel sub-problems \textsc{Vaccination-Attack}_w and \textsc{Attack-Protection}_w and prove they are $\Sigma_2^p$-complete thanks to a \textsc{Knapsack Interdiction} problem.

Note that the \textsc{NP}-completeness of \textsc{Protect}_w is immediate from the previous section. Hence, we start with complexity results to \textsc{Attack}_w, then \textsc{Attack-Protection}_w, and \textsc{Vaccination-Attack}_w, and, to conclude, \textsc{MCN}_w. We will observe that the introduction of non-unitary parameters offers sufficient flexibility to go a level up in the polynomial hierarchy in comparison with the unitary undirected cases.

4.1 The \textsc{Attack}_w problem

In the attack phase, the vaccination already took place so we effectively work on $G_a$, which is the result of the deletion of the vaccinated vertices from the original graph. We are given a non-negative attack budget $\Phi$, and as there is no protection phase afterwards, we set $\Lambda = 0$. The goal is thus to harvest the most benefit possible by infecting vertices subject to a budget limit. The decision version of the problem is then:

\textsc{Attack}_w:

**INSTANCE:** An undirected graph $G_a = (V_a, E_a)$, a non-negative integer cost $h_v$, and value $b_v$ for each vertex $v \in V$, a non-negative integer budget $\Phi$, and a non-negative integer number $K$. 

Page 6
QUESTION: Is there a subset of vertices $I \subseteq V_a$ to attack, with cost $\sum_{v \in I} h_v \leq \Phi$ such that the sum of the benefits of the resulting infected vertices in $G_a$ is greater or equal to $K$?

To make evident the NP-completeness of the problem, we simply state the decision version of the Knapsack problem, one of the Karp’s 21 NP-complete problems [19]:

**Knapsack:**

**INSTANCE:** Finite set $U$, for each $u \in U$, a positive integer size $a_u \in \mathbb{N}$ and a positive integer profit $p_u \in \mathbb{N}$, and two positive integers $B$ and $K$.

**QUESTION:** Is there a subset $U' \subseteq U$ such that $\sum_{u \in U'} a_u \leq B$ verifying $\sum_{u \in U'} p_u \geq K$?

**Theorem 4.1.** $\text{ATTACK}_w$ is NP-complete, even on trivial graphs.

**Proof.** It is easy to see that $\text{ATTACK}_w \in \text{NP}$. Indeed, given an attack $I$, finding the subsequent infected vertices can be done in linear time thanks to a DFS. Then, it suffices to sum the values associated with each infected vertices to verify that it is greater or equal to $K$.

The reduction from Knapsack we use to show the NP-hardness is straightforward. Given an instance of Knapsack, we set $V_a = U$, $E_a = \emptyset$, $K = K$, $\Phi = B$, and $\forall v \in V_a$, $h_v = a_v$, $b_v = p_v$. In this configuration, $G_a$ having no edges, the attacked vertices are exactly the infected ones in the end, and the goal of the attacker is equivalent to filling up a knapsack with limited capacity by choosing which vertices to attack.

4.2 **The ATTACK-PROTECT$_w$ problem**

In the proof of Theorem 4.1, we highlighted how a Knapsack instance can be directly transformed into a weighted graph with no edges. In this section, as well as in the one that follows, we will use a similar transformation, but add one additional root vertex to our construction in order to build a star graph: one root vertex connected with an edge to each of the other vertices, each one representing an item of the knapsack. That way, the complexity results we devise also hold for trees.

As before, the vaccination having already been done, we start from $G_a$, the graph where the vaccinated vertices have been removed.

**ATTACK-PROTECT$_w$:**

**INSTANCE:** A graph $G_a = (V_a, E_a)$, a non-negative integer $K$, two non-negative integer budgets $\Phi$ and $\Lambda$, $\forall v \in V_a$ two non-negative integer costs $h_v$, $c_v$ and a non-negative integer benefit $b_v$.

**QUESTION:** Is there a subset $I \subseteq V_a$, with cost $\sum_{v \in I} h_v \leq \Phi$ such that $\forall P \subseteq V_a \setminus I$ with cost $\sum_{v \in P} c_v \leq \Lambda$, the sum of the benefit of the saved vertices is strictly less than $K$?

In order to show that $\text{ATTACK-PROTECT}_w$ is $\Sigma^P_2$-complete, we use the *Bilevel Interdiction Knapsack Problem* introduced by DeNegre [10] and proven to be $\Sigma^P_2$-complete in [7]. In this problem, two players, a leader and a follower, can select items in the same set of objects $\bar{O}$. First, the leader packs some items into her knapsack, then the follower chooses among the remaining ones. The aim of the leader is to interdict a subset of items, subject to a capacity constraint, in order to minimize the total profit of the follower. The objective of the follower is to maximize its profit, subject to a constraint capping the maximum profit obtainable for her. The decision problem is then:

**Bilevel Interdiction Knapsack (BIK):**

**INSTANCE:** A set of items $O$ such that each $o \in O$ has a positive integer weight $a_o$, a positive integer profit $p_o$, and $\forall o \in O$ a positive integer maximum weight capacity $A$ for the leader, a positive integer maximum profit $B$ for the follower, and a positive integer $K \leq B$.

**QUESTION:** Is there a subset $O_l \subseteq O$ of items for the leader to select, with $\sum_{o \in O_l} a_o \leq A$, such that every subset $O_f \subseteq O \setminus O_l$ with $\sum_{o \in O_f} p_o \leq B$ that the follower can create has a total profit $\sum_{o \in O_f} p_o < K$?

**Theorem 4.2.** $\text{ATTACK-PROTECT}_w$ is strongly $\Sigma^P_2$-complete, even if the graph is a tree.

**Proof.** First, $\text{ATTACK-PROTECT}_w$ is in $\Sigma^P_2$ since this decision problem is exactly of the form $\exists I \forall P \ Q(I, P)$.

Next, we prove the problem $\Sigma^P_2$-hardness. Let us begin by noting that we can restrict the instances of KIP to the ones where $K$ and $B$ are strictly inferior to $\sum_{o \in O} p_o$, otherwise, KIP is trivial to solve. This remark is used in the second part of this proof.

Starting from an instance of BIK, we construct an instance of $\text{ATTACK-PROTECT}_w$ as follows. We first build a star graph $G_a = (V_a, E_a)$ with a root vertex $r$ and a vertex $v_o$ for each $o \in O$ linked to $r$ through an edge $(r, v_o)$. We set

\[\text{Page 7}\]
Suppose first that BIK is a Yes instance. Then, there is a set of items \(O_I \subseteq O\) of total weight \(\sum_{o \in O_I} a_o \leq A\) such that for all \(O_f \subseteq O \setminus O_I\) feasible for the follower, it holds \(\sum_{o \in O_f} p_o \leq K - 1\). Consequently, in the ATTACK-PROJECT\(_w\) problem, the attacker can select the subset of vertices \(I = \{r\} \cup \{v_o : o \in O_I\}\) with a feasible attacking cost \(\sum_{v \in I} h_v = 1 + \sum_{o \in O_I} a_o \leq A + 1 = \Phi\). Now, the defender can only protect vertices in \(\{v_o : o \notin O_I\}\) and since the central vertex of the star graph is infected, the saved vertices will be the protected ones. The aim of the defender is therefore to select the subset of vertices of maximum total benefit with respect to the protection budget \(\Lambda\). This is exactly the follower’s problem in BIK. Hence, since BIK is an Yes instance, the defender (follower in BIK) cannot attain a benefit (profit in BIK) equal or greater to \(K = \bar{K}\) through a feasible action. Therefore, the ATTACK-PROJECT\(_w\) is a Yes instance.

Now suppose that ATTACK-PROTECTION\(_w\) is a Yes instance. Thus, there exists an attack strategy \(I \subseteq V_o\) such that there is no feasible subset \(P \subseteq V_o \setminus I\) of protected vertices leading to a total benefit greater or equal to \(K\) for the defender. As \(\Phi \geq 1\), it is obvious that the attacker will attack at least the central vertex \(r\), otherwise, the defender would pick it and achieve a benefit superior to \(K\) (recall that \(K = \bar{K} < \sum_{o \in O} p_o\)), contradicting ATTACK-PROTECTION\(_w\) Yes instance. Hence, the attacker is left with budget \(\Phi - h_r = A\). Once the central vertex is attacked, only the other vertices subsequently protected will not be infected. Therefore, the rest of the attack budget \(A\) is spent on a subset of vertices of \(\{v_o : v_o \in V_o : o \in O\}\) and it ensures that for any \(P = \{v_o \in V_o : v_o \in O \setminus I\}\) with \(\sum_{o \in P} c_o = \sum_{o : v_o \in P} p_o \leq \Lambda = B\), the total benefit for the defender is \(\sum_{o \in P} b_o = \sum_{o : v_o \in P} p_o \leq K - 1\). Consequently, BIK is also a Yes instance.

This completes the proof that ATTACK-PROJECT\(_w\) is \(\Sigma_2^P\)-complete. Moreover, since the BIK was shown to be NP-complete even for unary encoding, we can conclude that no pseudopolynomial-time algorithm exists to solve the ATTACK-PROJECT subgame. Since a star graph is a tree, the result stated in the theorem holds. \(\square\)

### 4.3 The Vaccination-Attack\(_w\) problem

Using a similar reduction to the one in the proof of Theorem 4.2, we show that the Vaccination-Attack\(_w\) on weighted graphs is \(\Sigma_2^P\)-complete. As in the unitary case, this is equivalent to studying MCN\(_w\) problems where we set \(\Lambda = 0\). The decision version of the problem is:

**Vaccination-Attack\(_w\):**

**INSTANCE:** A graph \(G = (V, E)\), a non-negative integer \(K\), two non-negative integer budgets \(\Omega\) and \(\Phi\), \(\forall v \in V\) two non-negative integer costs \(\tilde{c}_v\), \(h_v\) and a non-negative integer benefit \(b_v\).

**QUESTION:** Is there a subset \(D \subseteq V\), with cost \(\sum_{e \in D} c_e \leq \Omega\) such that \(\forall \text{ I } \subseteq V \setminus D\) with cost \(\sum_{e \in I} h_e \leq \Phi\), the sum of the benefit of the infected vertices is strictly less than \(K\)?

**Theorem 4.3.** Vaccination-Attack\(_w\) is strongly \(\Sigma_2^P\)-complete, even if the graph is a tree.

**Proof.** As before, Vaccination-Attack\(_w\) is in \(\Sigma_2^P\) since this decision problem is exactly of the form \(\exists D \forall I \left. Q(D, I) \right.\).

Now, we establish the problem \(\Sigma_2^P\)-hardness. We start from an instance of BIK, defined in the previous section, and we then construct an instance of Vaccination-Attack\(_w\) as follows. First, we build a star graph \(G = (V, E)\) with a central vertex \(r\) and \(|O|\) leaf vertices \(v_o\) with \(o \in O\). See Figure 3. We add an edge \((r, v_o)\) for each such leaf vertex.

The central vertex has benefit \(b_r = K\) and costs \(\tilde{c}_r = h_r = 1\). Each leaf vertex \(v_o\) with \(o \in O\) has a benefit \(b_{v_o} = p_o\), cost for the defender \(\tilde{c}_{v_o} = a_o\) and cost for the attacker \(h_{v_o} = p_o\). Finally, we fix \(\Omega = A + 1\), \(\Phi = B\) and \(K = \bar{K}\).

This is exactly the setting of BIK and one can easily complete the proof of equivalence of the two decision instances following a path very similar to the proof of Theorem 4.2.
Finally, the reduction used a star graph which is a particular case of a tree. Hence, the problem is $\Sigma_2^p$-complete even on trees.

**Corollary 4.4.** $MCN$ is $\Sigma_2^p$-hard, even if the graph is a tree.

### 4.4 The $MCN_w$ problem

In this section we show that the decision problem $MCN_w$ is $\Sigma_3^p$-complete.

**$MCN_w$:**

**INSTANCE:** A graph $G = (V, E)$, a non-negative integer $K$, three non-negative integer budgets $\Omega$, $\Phi$ and $\Lambda$, $\forall v \in V$ three non-negative integer costs $\hat{c}_v$, $h_v$ and $c_v$, and a non-negative integer benefit $b_v$.

**QUESTION:** Is there a subset $D \subseteq V$, with cost $\sum_{v \in D} h_v \leq \Phi$, that for all 0-1 assignments of $\forall v \in D$ $\sum_{v \in D} c_v \leq \Lambda$ such that the sum of the benefit of the saved vertices is greater or equal to $K$?

In order to achieve our ultimate goal, we take the 3-Alternating Quantified Satisfiability problem $(B_3 \cap 3CNF)$, known to be $\Sigma_3^p$-complete problem [25, 29], in order to prove that the generalization of BIK to a trilevel, the Trilevel Interdiction Knapsack (TIK), is $\Sigma_3^p$-complete. Then, TIK is used to demonstrate that $MCN_w$ is $\Sigma_3^p$-complete.

**3-ALTERNATING QUANTIFIED Satisfiability ($B_3 \cap 3CNF$):**

**INSTANCE:** Disjoint non-empty sets of variables $X$, $Y$ and $Z$, and a Boolean expression $E$ over $U = X \cup Y \cup Z$ in conjunctive normal form with at most 3 literal in each clause $c \in C$.

**QUESTION:** Is there a 0-1 assignment for $X$ so that for all 0-1 assignments of $Y$ there is a 0-1 assignment of $Z$ such that $E$ is satisfied?

**TRILEVEL INTERDICTIO N Knapsack (TIK):**

**INSTANCE:** A set of items $O$ such that each $o \in O$ has two a positive integer weights $a'_o$ and $a_o$ and a positive integer profit $p_o$, two positive integer maximum weight capacities $A'$ and $A$, a positive integer maximum profit $B$ and a positive integer goal $K \leq B$.

**QUESTION:** Is there a subset $O_1 \subseteq O$ of items, with $\sum_{o \in O_1} a'_o \leq A'$, such that every subset $O_2 \subseteq O \setminus O_1$, with $\sum_{o \in O_2} a_o \leq A$, there is a subset $O_3 \subseteq O \setminus O_2$, with $\sum_{o \in O_3} p_o \leq B$, such that $\sum_{o \in O_3} p_o \geq K$ holds?

**Theorem 4.5.** TIK is $\Sigma_3^p$-complete.

**Proof.** The statement of TIK is of the form $\exists O_1 \forall O_2 \exists O_3 \Phi(O_1, O_2, O_3)$, directly implying that it is in $\Sigma_3^p$.

Next, we use a reduction from the $B_3 \cap 3CNF$ which is very much in line with the reduction from 3-SAT to Subset Sum presented in [9, Theorem 34.15]:

- For each variable $u \in U$, we create two items $a_u$ and $\hat{a}_u$, one for each possible 0-1 assignment of $u$. We designate by $O_U = \{a_u : u \in U\}$ and $\hat{O}_U = \{\hat{a}_u : u \in U\}$ the two sets of items of size $|U|$.

- For each clause $c \in C$, (i) if $c$ has 1 literal, we create one item $a_{c}^1$, (ii) if $c$ has 2 literals, we create two items $a_{c}^1$ and $a_{c}^2$, and (iii) if $c$ has 3 literals, we create three items $a_{c}^1$, $a_{c}^2$, and $o_{c}^3$. We designate by $O_C$ the set of items associated with $C$.

- Weights, profits, maximum capacities, maximum profit and goal will be given by digits of size $|X| + |Y| + |Z| + |C| + 1$ in base 10. Hence, each digit position is labeled by a variable or a clause: the first $|C|$ positions (least significant numbers) are labeled by the clauses, then the next $|X|$ positions are labeled by the variables $X$, then the next $|Y|$ positions are labeled by the variables $Y$, then the next $|Z|$ positions are labeled by the variables $Z$, and, finally, the last position is labeled as forbidden.
– For each \( u \in U \), the two corresponding items \( o_u \) and \( o_\bar{u} \) have weights and profits as described next. The weights and profits of \( a'_o, a_{o_u}, p_{o_u}, a_{o_\bar{u}} \) and \( p_{o_\bar{u}} \) have digit 1 in the position labeled by the variable \( U \) and 0 in the positions labeled by other variables; the remaining digits are zero for \( a'_o, a_{o_u}, a_{o_\bar{u}} \) and \( a_{o_{\bar{u}}} \). In particular, for all \( u \in O_U \cup O_\bar{U} \), it holds \( a'_o = a_{o_u} \) and \( a_{o_{\bar{u}}} = a_{o_{\bar{u}}} \).

If the literal \( u \) appears in clause \( c \in C \), then \( p_{o_u} \) has digit 1 in the position labeled as \( c \) and 0 otherwise. Similarly, if the literal \( \bar{u} \) appears in clause \( c \in C \), \( p_{o_\bar{u}} \) has digit 1 in the position labeled by \( c \) and 0 otherwise. Finally, for all \( u \in O_U \cup O_\bar{U}, p_{o_u} \) and \( p_{o_\bar{u}} \) have digit 0 in the position labeled as forbidden.

– For each \( c \in C \), the associated items have weights and profits as follows. If \( c \) has one literal, \( a'_o, a'_{o_u}, a'_{o_\bar{u}} \) have digit 1 in the position labeled as forbidden and 0 elsewhere; \( p_{o_u} \) has digit 3 in the position labeled as \( c \) and 0 elsewhere. If \( c \) has two literals, \( a'_{o_1}, a'_{o_2}, a'_{o_1}, a'_{o_2} \) and \( a_{o_1}, a_{o_2} \) have 1 in the position labeled as forbidden and 0 elsewhere; \( p_{o_1} \) and \( p_{o_2} \) have digit 3 and 2, respectively, in the position labeled as \( c \) and 0 elsewhere. If \( c \) has three literals, \( a'_{o_1}, a'_{o_2}, a'_{o_3}, a'_{o_1}, a'_{o_2}, a'_{o_3} \) and \( a_{o_1}, a_{o_2}, a_{o_3} \) have 1 in the position labeled as forbidden and 0 elsewhere; \( p_{o_1}, p_{o_2}, p_{o_3} \) and \( p_{o_1}, p_{o_2}, p_{o_3} \) have digit 3, 2 and 1, respectively, in the position labeled as \( c \) and 0 elsewhere.

– The weight capacity \( A' \) has 1s for all digits with labels in \( X \) and 0s elsewhere. Hence, \( O_1 \) cannot contain items from \( \{ o_u, o_\bar{u} : u \in Z \cup Y \} \cup O_C \).

– The weight capacity \( A \) has 1s for all digits with labels in \( Y \), 2s for all digits with labels in \( X \) and 0s elsewhere. Hence, \( O_2 \) cannot contain items from \( \{ o_u, o_\bar{u} : u \in Z \} \cup O_C \).

– The maximum profit \( B \) has 1s for all digits with labels in \( X \cup Z \), 2s for all digits with labels in \( Y \), 4s for all digits with labels in \( C \), and 0s elsewhere. Hence, \( O_3 \) can take any item (as long as not interdicted by \( O_2 \)).

– We make \( \bar{K} \) is equal to \( B \), except for the digits with labels \( Y \), where it is 1.

See Figure 4 for an illustration of our reduction.

Let \( B_3 \cap 3CNF \) be a \( Yes \) instance. Then, take in \( O_1 \) the items \( o_u \) such that \( u \in X \) is 1 and the items \( o_\bar{u} \), otherwise. Clearly, this choice of \( O_1 \) respects the maximum weight capacity \( A' \). By construction, given this \( O_1 \), the best \( O_2 \) will take all items associated with \( X \) and not taken by \( O_1 \), as it does not interfere with the budget left for the items associated with \( Y \). Furthermore, the optimal \( O_2 \) will also take exactly one of the items \( o_u \) or \( o_\bar{u} \) for \( u \in Y \):

- The two items associated with the most significant digit whose label is in \( Y \) cannot be taken simultaneously in \( O_2 \) as it would violate the weight capacity \( A \). In fact, exactly one of these items must be taken, as otherwise \( O_3 \) would select them both, making the achievement of the profit \( \bar{K} \) only dependent on the items associated with the \( Z \); consequently, the goal would be achieved.

- The two items associated with the second most significant digit whose label is in \( Y \) cannot be taken simultaneously, since we already know that one of the items associated with the most significant digit in \( Y \) is taken which would result in a violation of the weight capacity \( A \). Hence, reasoning as before, \( O_2 \) will take exactly of the items associated with the second most significant digit in \( Y \).

- The reasoning above propagates until the least significant digit labeled in \( Y \). We conclude that the best \( O_2 \) will have exactly one of the items \( o_u \) or \( o_\bar{u} \) for \( u \in Y \).

Finally, \( O_3 \) will contain \( O_1 \) and all the items associated with \( Y \) not in \( O_2 \). This makes the rest of the items selection for \( O_3 \) completely equivalent to variable assignment in \( Z \) for \( B_3 \cap CNF \) (precisely, the standard reduction from 3-SAT to Subset Sum). Therefore, \( TIK \) is a \( Yes \) instance.

Next, suppose that \( TIK \) is a \( Yes \) instance. Certainly, an optimal \( O_1 \) must have exactly one of the items \( o_u \) and \( o_\bar{u} \) for \( u \in X \), otherwise, \( O_2 \) could interdict some \( o_u \) and \( o_\bar{u} \), making the goal \( \bar{K} \) impossible to be achieved. As argued before, an optimal reaction \( O_2 \) to \( O_1 \) will select the items associated with \( X \) not in \( O_1 \).

Assign 1 to \( u \in X \) such that \( o_u \in O_1 \), and 0 otherwise. For any valid assignment of the variables in \( Y \), the correspondence in \( TIK \) is the following: if \( u \in Y \) is 1, add \( o_u \) to \( O_2 \), otherwise add \( o_\bar{u} \). This forces \( O_3 \) to select for each \( u \in Y \), \( o_u \) if it is 1 and \( o_\bar{u} \) if it is 0; otherwise, the goal \( \bar{K} \) is not attained. Since, by hypothesis, \( TIK \) is a \( Yes \) instance, for those \( O_1 \) and \( O_2 \), there is \( O_3 \) such that the profit \( \bar{K} \) is exactly achieved which implies that there is an assignment of \( Z \) such that \( E \) is satisfied.

\[ \Box \]

**Theorem 4.6.** \( MCN_w \) is \( \Sigma_3^p \)-complete, even on trees.
| $O$ | forbidden | $Z$ | $Y$ | $X$ | $C$ |
|-----|-----------|-----|-----|-----|-----|
| $o_a$ | $a'_{o_a} = a_{o_a}$ | $d$ | $o_b$ | $c$ | $b$ | $a$ | $c_1$ | $c_2$ | $c_3$ |
| $p_{o_a}$ | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| $a'_{o_b}$ | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| $p_{o_b}$ | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| $a'_{o_c}$ | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| $p_{o_c}$ | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| $a'_{o_d}$ | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| $p_{o_d}$ | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $a'_{o_{c1}}$ | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $a_{o_{c1}}$ | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $p_{o_{c1}}$ | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 |
| $a'_{o_{c2}}$ | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $a_{o_{c2}}$ | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $p_{o_{c2}}$ | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| $a'_{o_{c3}}$ | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $a_{o_{c3}}$ | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $p_{o_{c3}}$ | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 |
| $A'$ | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| $A$ | 0 | 1 | 2 | 2 | 0 | 0 | 0 | 0 | 0 |
| $B$ | 0 | 1 | 2 | 1 | 1 | 4 | 4 | 4 | 4 |
| $K$ | 0 | 1 | 1 | 1 | 1 | 4 | 4 | 4 | 4 |

Figure 4: Example of construction of TIK from an instance $B_3 \cap 3 CNF$ with $E = (a \lor b \lor \neg c) \land (\neg a \lor \neg b \lor d) \land (a \lor c \lor b)$, where $X = \{a, b\}, Y = \{c\}, Z = \{d\}$ and the clauses are labeled from left to right.
Proof. MCN\textsubscript{w} is clearly in $\Sigma^p_3$.

Next, from an instance of TIK, we construct the following instance of MCN\textsubscript{w}:

- Let $\Omega = A'$, $\Phi = A + 1$, $\Lambda = B$ and $K = \bar{K}$.
- For each item $o \in O$ create three vertices $v_o^1$, $v_o^2$ and $v_o^3$ with
  \[- \hat{c}_{v_o^1} = \Omega + 1, \ h_{v_o^1} = \Phi + 1, \ c_{v_o^1} = p_o \text{ and } b_{v_o^1} = 0; \text{ this vertex is only available for the protection set } P;\]
  \[- \hat{c}_{v_o^2} = \Omega + 1, \ h_{v_o^2} = \Phi + 1, \ c_{v_o^2} = \Lambda + 1 \text{ and } b_{v_o^2} = p_o; \text{ this vertex cannot be vaccinated, directly infected or protected};\]
  \[- \hat{c}_{v_o^3} = a_o', \ h_{v_o^3} = a_o, \ c_{v_o^3} = \Lambda + 1 \text{ and } b_{v_o^3} = 0; \text{ this vertex is only available for the vaccination set } D \text{ and for the direct infection set } I;\]
- Create a vertex $r$ with $\hat{c}_r = \Phi + 1$, $h_r = 1$, $c_r = 1$ and $b_r = K$.
- For each item $o \in O$, add the edges $(r, v_o^1)$, $(v_o^1, v_o^2)$ and $(v_o^2, v_o^3)$.

See Figure 5 for an illustration of our reduction.

\[\begin{align*}
\hat{c}_{v_1} &= \Omega + 1, \ h_{v_1} = c_1 = 1, \ b_v = K \\
\hat{c}_{v_2} &= \Omega + 1, \ h_{v_2} = \Phi + 1, \ c_{v_2} = p_1, \ b_{v_2} = 0 \\
\hat{c}_{v_3} &= \Omega + 1, \ h_{v_3} = \Phi + 1, \ c_{v_3} = p_1, \ b_{v_3} = 0 \\
\hat{c}_{v_4} &= \Omega + 1, \ h_{v_4} = \Phi + 1, \ c_{v_4} = \Lambda + 1, \ b_{v_4} = p_1 \\
\hat{c}_{v_5} &= \Omega + 1, \ h_{v_5} = \Phi + 1, \ c_{v_5} = \Lambda + 1, \ b_{v_5} = p_1 \\
\hat{c}_{v_6} &= a_1', \ h_{v_6} = a_1, \ c_{v_6} = \Lambda + 1, \ b_{v_6} = 0 \\
\hat{c}_{v_7} &= a_1', \ h_{v_7} = a_1, \ c_{v_7} = \Lambda + 1, \ b_{v_7} = 0 \\
\end{align*}\]

Figure 5: Graph reduction from TIK to MCN\textsubscript{w} when $O = \{1, 2, \ldots, n\}$. The only vertices resulting in positive benefit are the ones in white. The vertices in gray can be vaccinated and directly attacked. The vertices in green can be protected. The vertex in black can be attacked (and protected).

The key ingredients of this reduction are the following: (i) independently of the vaccination strategy, an optimal attack will always include the vertex $r$, (ii) hence, the only way to collect a positive benefit $p_o$ is by ensuring that vertex $v_o^3$ is saved, (iii) the latter is only possible if $v_o^3$ is vaccinated and $v_o^3$ is protected or if $v_o^3$ is not attacked and $v_o^3$ is protected. These observations allow to show that TIK is a Yes instance if and only if MCN\textsubscript{w} is a Yes instance. The remainder of the proof follows a similar reasoning to the previous proofs for the weighted games.

\[\square\]

5 Directed graphs

In this section, we consider directed graphs $G = (V, A)$ and restrict costs and benefits to be unitary. We use the subscript $\text{dir}$ for these problem versions. Clearly, these problems inherit the complexity of their unitary undirected versions, as they are more general. In fact, we were able to go a level up in the polynomial hierarchy for some of its subgames in comparison with the unitary undirected cases. In this section, we first prove that the ATTACK\textsubscript{dir} is NP-complete, and then demonstrate that VACCINATION-ATTACK\textsubscript{dir} is $\Sigma^P_2$-complete. Later, in Section 6, we present special properties of PROTECT\textsubscript{dir} that allow us to easily prove NP-completeness for directed acyclic graphs and polynomial for arborescences.

It should be remarked that we do not address ATTACK-PROTECT\textsubscript{dir} and thus, it remains open whether it is $\Sigma^P_2$-complete. The difficulty on dealing with this subgame is related to the lack of $\Sigma^P_2$-hard problems involving unitary parameters or a division on the two players decision variables: in ATTACK-PROTECT\textsubscript{dir} all parameters are 1 and all vertices can be subject to infection or protection. On the other hand, as an example, non-trivial instances of KIP (presented in Section 4.2) should have weights not all 1, otherwise it becomes polynomially solvable as it can be reduced to its continuous version and, consequently, efficiently solved [8]. Another example, 2-CNF-ALTERNATING QUANTIFIED SATISFIABILITY, to be introduced in Section 5.2, and which is $\Sigma^P_2$-complete, demands each player
to control distinct sets of variables. For VACCINATION-ATTACK\textsubscript{dir}, we were able to bypass this challenge but an analogous trick does not seem easily adaptable for ATTACK-PROTECT\textsubscript{dir}.

5.1 The ATTACK\textsubscript{dir} problem

First, we study the Attack problem on directed graphs, ATTACK\textsubscript{dir}. We are given a directed graph \(G_a\) resulting from the deletion of the vaccinated vertices from the original graph, and an integer budget \(\Phi\). In this setting, there is no protection phase, i.e. \(\Lambda = 0\). The decision version of the problem is:

\textbf{ATTACK\textsubscript{dir} instance:} A directed graph \(G_a = (V_a, A_a)\), a non-negative integer budget \(\Phi \leq |V_a|\), and a non-negative integer \(K\).

\textbf{Question:} Is there a subset of vertices \(I \subseteq V_a\) with \(|I| \leq \Phi\) such that the number of infected vertices in \(G_a\) is greater or equal to \(K\)?

We saw that in the undirected case, this problem is solvable in linear time, the best strategy being to infect the \(\Phi\) largest connected components of \(G_a\). But in the directed case, the infection is only allowed to propagate itself according to the direction of the arcs, which makes the problem of choosing the right set of vertices to attack NP-complete. We will use a reduction from the 3-Satisfiability problem, which is one of the Karp’s 21 NP-complete problems [19].

\textbf{3-Satisfiability (3-SAT):}

\textbf{Instance:} Set \(U\) of variables, Boolean expression \(E\) over \(U\) in conjunctive normal form with exactly 3 literals in each clause \(c \in C\).

\textbf{Question:} Is there a 0-1 assignment for the variables in \(U\) that satisfies \(E\)?

\textbf{Theorem 5.1.} ATTACK\textsubscript{dir} is NP-complete, even on directed acyclic graphs.

\textit{Proof.} ATTACK\textsubscript{dir} \in NP as, given a set of attacked vertices \(I\), checking whether the set of infected vertices is greater than \(K\) is easily done using a DFS.

To prove that ATTACK\textsubscript{dir} is NP-hard, we take an instance of 3-SAT. We build a directed acyclic graph \(G_a\) as follows:

- For each variable \(u \in U\), we create two vertices \(v_u\) and \(v_{\overline{u}}\), one for each possible 0-1 assignment of \(u\). We call \(V_U = \{v_u; u \in U\}\) and \(V_{\overline{U}} = \{v_{\overline{u}}; u \in U\}\) the two sets of vertices of size \(|U|\). For each variable \(u\), we also create a directed path \(p_u\) of length \(|C| + |U| - 1\), with an in-going arc from both \(v_u\) and \(v_{\overline{u}}\) at the beginning of the path.

- For each clause \(c \in C\), we create a vertex \(v_c\) in \(V_C\).

- From each vertex \(v_u \in V_U\), we draw an arc \((v_u, v_c)\) to every clause in which the positive literal \(u\) appears. Similarly, we draw an arc \((v_{\overline{u}}, v_c)\) from each \(v_{\overline{u}} \in V_{\overline{U}}\) to every clause in which the negative literal \(\overline{u}\) appears.

An example of this construction can be found in Figure 6. We set \(\Phi = |U|\), \(K = |U| \times (|U| + |C|) + |C|\) and argue that answering ATTACK\textsubscript{dir} on this instance is the same as answering 3-SAT.

Indeed, suppose that 3-SAT is a \textit{Yes} instance, i.e. there is a 0-1 assignment to the variables in \(U\) such that every clause in \(E\) is true. Taking this assignment, by attacking \(v_u\) if \(u\) is set to be 1 and \(v_{\overline{u}}\) otherwise, we attack exactly \(\Phi\) vertices in \(G_a\). Moreover, each path \(p_u\) is infected, and for each pair \((v_u, v_{\overline{u}})\), there is exactly one vertex infected due to the direction of the arcs. Finally, as \(E\) is true, each clause \(c\) is true, which translates into the fact that each \(v_c\) in the graph \(G_a\) is infected. Overall, there are exactly \(|U| + |U| \times |p_u| + |C| = |U| \times (|U| + |C|) + |C|\) vertices infected in the graph.

Conversely, we prove that if ATTACK\textsubscript{dir} is a \textit{Yes} instance, i.e., there is a feasible attack \(I^*\) on \(G_a\) leading to at least \(K = |U| \times (|U| + |C|) + |C|\) vertices infected, then \(E\) is satisfiable and the corresponding 0-1 assignment can be read in \(I\). Let \(I^*\) be such an attack strategy. First, we remark that the largest possible set of infected vertices should contain all the vertices \(V_{p_u}\) of each path \(p_u\): it is possible to infect them all as \(\Phi = |U|\) and due to their size equal to \(|C| + |U| - 1\), we can prove that not infecting all of them results in a sub-optimal solution. Indeed, suppose that for one \(u'\) we do not infect any of the vertices \(V_{p_{u'}}\) of the path \(p_{u'}\). Let \(\alpha^*\) be the maximum number of vertices we can infect without infecting \(p_{u'}\). As \(p_{u'}\) is not infected, \(v_{u'}\) and \(v_{\overline{u'}}\) cannot be either. Thus, an easy upper bound \(\alpha_{u,p}\) on
\( \alpha^* \) is obtained by saying that every vertex of the graph is infected, except for the ones in \( \{v_{u'}, v_{\bar{u}'}\} \cup V_{p_u} \). Then,

\[
\alpha^* \leq \alpha_{up} = (|U| - 1) \times |p_u| + 2(|U| - 1) + |C|
= (|U| - 1) \times (|U| + |C| - 1) + 2|U| - 2 + |C|
= |U|^2 + |U| \times |C| - 2|U| - |C| + 1 + 2|U| - 2 + |C|
= |U| \times (|U| + |C|) - 1.
\]

As we assumed that the optimal attack \( I^* \) infected at least \( K = |U| \times (|U| + |C|) + |C| \) vertices, which is strictly greater than \( \alpha_{up} \), we proved that no strategy not infecting all the paths can infect \( K \) vertices.

Thus, as there is exactly \( \Phi \) different paths, we should attack exactly one element in each set of vertices \( \{v_u, v_{\bar{u}}\} \cup V_{p_u} \); if we attacked more than one, then the remaining budget would not allow to attack all the paths. As attacking \( v_u \) or \( v_{\bar{u}} \) leads to a strictly greater number of infected vertices than infecting a vertex in \( p_u \), there is no harm in assuming that no vertex inside the \( p_u \) is in \( I^* \). This implies that \( I^* \subset V_U \cup V_{\bar{D}} \). At this point, there are at least \( |p_u| \times |U| + |U| = |U| \times (|U| + |C|) \) vertices infected.

Since we supposed that we had a \( \text{Yes} \) instance to \( \text{ATTACK}_{dir} \), there must be \( K = |U| \times (|U| + |C|) + |C| \) infected vertices, which implies that all vertices in \( V_C \) are infected. Thus, \( 3\text{-SAT} \) is a \( \text{Yes} \) instance and \( I^* \) is a 0-1 assignment of \( U \) that makes \( E \) true, concluding the proof.

**Remark 5.2.** Note that the proof of Theorem 5.1 holds if \( p_u \) is replaced by a complete graph with \( |C| + |U| - 1 \) vertices (the length of the path). This observation will be useful for the reduction used in \( \text{VACCINATION-ATTACK}_{dir} \).

![Figure 6: Example of construction of \( G_a \) from the boolean expression in CNF with 3 literals in each clause \( E = (a \lor b \lor \neg c) \land (\neg a \lor b \lor c) \land (a \lor \neg b \lor c) \). We have \( U = \{a, b, c\} \) and \( |C| = 3 \). Taking \( I = \{v_u, v_{\bar{u}}, v_c\} \) is optimal.](image)

### 5.2 The \( \text{VACCINATION-ATTACK}_{dir} \) problem

Our demonstration of NP-completeness for \( \text{ATTACK}_{dir} \) inspires our proof for the \( \Sigma_2^p \)-completeness of \( \text{VACCINATION-ATTACK}_{dir} \). The formulation of this decision problem is

**\( \text{VACCINATION-ATTACK}_{dir} \):**
- **INSTANCE:** A graph \( G = (V, A) \), two non-negative integer budgets \( \Omega \) and \( \Phi \) such that \( \Omega + \Phi \leq |V| \) and a non-negative integer \( K \).
- **QUESTION:** Is there a subset \( D \subseteq V \), \( |D| \leq \Omega \) such that \( \forall I \subseteq V \backslash D \) with \( |I| \leq \Phi \), the number of infected vertices \( |V| - s(G, D, I, 0) \leq K \)?

We will use a reduction from a variant of the \( 2\text{-Alternating Quantified Satisfiability problem (B}_2 \))\(^\text{CNF} \). Historically, \( B_2 \) was the first problem shown to be \( \Sigma_2^p \)-complete [23]. If the Boolean formula studied in \( B_2 \) is in DNF with 3 literals per clause, then the problem is still \( \Sigma_2^p \)-complete [29]. Thus, if we consider expressions in CNF with 3 literals per clause, instead of seeking to satisfy the Boolean formula, we should state the question as formulated in [18]:

**2-CNF-ALTERNATING QUANTIFIED SATISFIABILITY (B}_2^{\text{CNF}} \):**
- **INSTANCE:** Disjoint non-empty sets of variables \( X \) and \( Y \), Boolean expression \( E \) over \( U = X \cup Y \) in conjunctive normal form with exactly 3 literals in each clause.
- **QUESTION:** Is there a 0-1 assignment for \( X \) so that there is no 0-1 assignment for \( Y \) such that \( E \) is satisfied?

**Theorem 5.3.** \( \text{VACCINATION-ATTACK}_{dir} \) is \( \Sigma_2^p \)-complete.
Proof. From the formulation in the form of \(\exists D\ \forall I\ Q(D,I)\), we deduce that VACCINATION-ATTACK\_dir \(\in \Sigma^P_2\). To show that it is \(\Sigma^P_2\)-hard, we take an instance of \(B_2^{CNF}\). We build \(G\) in a similar fashion to how \(G_0\) was built in the proof of the Theorem 5.1, the main difference being the use of cliques instead of paths. However, to differentiate the variables in \(X\) from the ones in \(Y\), we slightly change the construction:

- For each variable \(x \in X\), we create two vertices \(v_x\) and \(v_x\) for each possible 0-1 assignment of \(x\). We call \(V_X\) and \(V_X\) the sets of \(v_x\) and \(v_x\). We also create two cliques \(k_x\) and \(k_x\) of \(|C| + |Y| - 1\) vertices \(V_{k_x}\) and \(V_{k_x}\).
- For each variable \(y \in Y\), we create two vertices \(v_y\) and \(v_y\) for each possible 0-1 assignment of \(y\). Let \(V_Y\) and \(V_Y\) be these two sets of vertices, and \(V_U = V_X \cup V_Y\), \(V_U = V_X \cup V_Y\). We also create a clique \(y\) of size \(|C| + |Y| - 1\).
- For each variable \(y \in Y\), we create a vertex \(v_y \in V_C\).
- For each vertex \(v_u \in V_U\), we draw an arc \((v_u, v_u)\) to every clause in which the positive literal \(u\) appears. Similarly, we draw an arc \((v_u, v_u)\) to every clause in which the negative literal \(\overline{u}\) appears.
- For every \(v_x\), we draw an arc to one node in \(k_x\), and do the same thing with \(v_x\) and \(v_x\). We also draw an undirected edge between each \(v_x\) and \(v_x\).
- Finally, from each \(v_y\) and each \(v_y\), we draw an arc to one node in \(k_y\).

An example of this construction can be found in Figure 7. We set \(\Omega = |X|\), \(\Phi = |X| + |Y|\), \(K = (|X| + |Y|) \times (|X| + |C|) + |C| - 1\), and argue that answering VACCINATION-ATTACK\_dir on this instance is the same as answering \(B_2^{CNF}\).

Indeed, if we are given a solution to a Yes instance of \(B_2^{CNF}\), then by vaccinating the vertices corresponding to the opposite of the 0-1 assignment of \(X\), we oblige the attacker to infect the vertices corresponding to the truth values for \(X\). From there, by following the same reasoning as before, it is easy to see that the Yes instance of \(B_2^{CNF}\) leads to a Yes instance of VACCINATION-ATTACK\_dir, i.e., the attacker cannot infect more than \(K\) vertices.

Conversely, we show that a set \(D^*\) corresponding to a solution of a Yes instance of VACCINATION-ATTACK\_dir is a solution to a Yes instance of \(B_2^{CNF}\). The first thing to notice is that given that the vaccination budget is \(\Omega = |X|\), that the size of the cliques \(k_x\) and \(k_x\) is equal to \(|C| + |Y| - 1\) and that each clique can be disconnected from the graph by spending only one unit of vaccination budget, we necessarily have that the best vaccination strategy \(D^* \subset X \cup \{v_x, v_x\}\). Next, we show that the defender would be worse off is she decides to vaccinate both \(v_x\) and \(v_x\) for some \(x \in X\) instead of vaccinating exactly one of each member of \(\{v_x, v_x\}\). In the best case scenario, in addition to the vertices already vaccinated, deciding to vaccinate the two members of a pair will allow her to protect \(|C| - 1\) nodes in \(V_C\) (it is not possible to remove all the arcs between the \(V_U \cup V_U\) and the \(V_C\) as we suppose that \(Y \neq \emptyset\), thus at least one clause contains a variable from \(C\)). But by doing so, as \(\Omega = |X|\), the defender will also not protect at all a group of vertices \(\{v_x, v_x\} \cup \{v_x, v_x\} \cup \{v_x, v_x\}\). Thus, the attacker can then spend only one unit of her own budget to attack all of this group, a quantity of infected vertices that otherwise would have been obtained by spending two units of his budget \(\Phi\). Thus, defending the two members of \(\{v_x, v_x\}\) spared one unit of budget for the attacker, which she can then use to attack one of the disconnected cliques of size \(|C| + |Y| - 1 > |C| - 1\). Thus, making such a move for the defender is strictly worse than not doing it and \(D^*\) contains exactly one vertex from each \(\{v_x, v_x\}\).

After this stage, it is easy to see that the best move for the attacker is to attack all of the \(D^* \setminus \{v_x \cup v_x\}\), and for the variables in \(Y\), the situation reduces to the one we already discussed with ATTACK\_dir (note that it is always more interesting for the attacker to spend her budget on attacking the \(v_y\) and \(v_y\) than the disconnected cliques as it will always infect more vertices). Hence, in the end, if the attacker did not manage to infect strictly more than \(|Y| + |X|\) \times (|Y| + |C|) + |C| - 1 vertices, it means that at least one clause is false, which concludes the proof.

Corollary 5.4. MCN\_dir \(\in \Sigma^P_2\)-hard.

6 PROTECTION\_dir : tractability limits

In this section, we will concentrate on optimal protection strategies given \(I\) (directly infected vertices). Without loss of generality, in what follows, we are restricting our attention to the induced graph obtained by only considering non-saved vertices when there is no protection.
Complexity of the Multilevel Critical Node Problem

Figure 7: Example of construction of $G$ from the boolean expression in CNF with 3 literals in each clause $E = (a \lor b \lor \neg c) \land (\neg a \lor \neg b \lor d) \land (a \lor c \lor b)$. Here, $X = \{a, b\}$ and $Y = \{c, d\}$. Taking $D = \{v_a, v_b\}$, i.e. obliging both $a$ and $b$ to be False makes it impossible to satisfy $E$.

The motivation to provide a closer look to the protection problem in the directed case is based on the fact that its NP-completeness was established for split graphs in the unitary case and for trees in the weighted one. Such results do not clarify the problem complexity for directed acyclic graphs (DAGs), polytrees and arborescences. Frequently, NP-complete problems on graphs became polynomially solvable on such graph classes. In this section, we start by focusing on general results for directed acyclic graphs and then on arborescences.

6.1 Directed acyclic graphs

We will show that an optimal protection strategy can be restricted to candidate vertices for any directed acyclic graph.

**Definition 6.1.** In a directed graph $G = (V, A)$, a vertex $v \in V \setminus I$ that can be reached from a vertex of $I$ by a directed path and whose isolated protection results in a maximal set of strongly connected saved vertices, is called candidate. Denote by $C$ the set of candidate vertices.

In other words, a candidate vertex $v$ has no predecessor that implies saving $v$. See Figure 9a for an illustration: $C = \{1, 2, 3, 9\}$; e.g., vertex 5 is not a candidate, since its protection saves vertices $\{6, 7, 8\}$, but this is also guaranteed by saving vertex 2 instead, resulting in the maximal set of saved vertices $\{2, 3, 4, 5, 6, 7, 8\}$.

**Lemma 6.1.** Let $G = (V, A)$ be a directed acyclic graph. Given $I$ and $\Lambda$, there is an optimal protection strategy $P \subseteq C$.

**Proof.** Let $P \subseteq V \setminus I$ be an optimal protection strategy such that exists $v \in P \setminus C$. Then, by definition of candidate, there is a vertex $u \in C$ whose isolated protection implies saving $v$, as well as, all the vertices that $v$ alone was saving. Hence, a feasible protection strategy can be obtained by removing $v$ from $P$ and adding $u$ to $P$: note that either the used budget is maintained, if $u \notin P$, or decreased, if $u \in P$. Let this strategy be denoted by $\tilde{P} = (P - \{v\}) \cup \{u\}$.

By contradiction, suppose that $\tilde{P}$ is not optimal: there is some vertex $r$ that was saved in $P$ but not in $\tilde{P}$. In fact, we can conclude that under $P$, $r$ was saved due to $v$ being saved and possibly due to some other vertices in $P \setminus \{v\} \subseteq \tilde{P}$. However, under $\tilde{P}$, $v$ is also saved, as well, as the vertices in $P \setminus \{v\}$. Consequently, $r$ is saved in $\tilde{P}$, resulting in a contradiction.

Furthermore, we can compute the value of candidate vertices.

**Definition 6.2.** For each $v \in C$, the value of $v$ is denoted by $p_v$ and it corresponds to the number of saved vertices if $v$ is the only protected vertex.

In the example of Figure 9a, $p_1 = 1$, $p_2 = 6$, $p_3 = 1$ and $p_9 = 1$. However, note that this analysis does not make the problem trivial: in Figure 9a, if $\Lambda = 2$, the optimal protection cannot be computed in a greedy way, i.e., protecting vertices 1 and 2 is not optimal; the only optimal solution is to protect vertices 1 and 3.
Remark 6.2. Definitions 6.1 and 6.2, and Lemma 6.1 do not need the condition that $G$ is a directed acyclic graph. Hence, they extend to any directed graph and thus, undirected graphs. However, since we know that the protection problem is NP-complete for general directed graphs, for sake of simplicity, we decided to write the full section assuming DAGs.

Theorem 6.3. $\text{PROTECT}_{\text{dir}}$ is NP-complete, even for directed acyclic graphs.

Proof. The statement of $\text{PROTECT}_{\text{dir}}$ is exactly the one of $\text{PROTECT}$ in Section 3, except that the graph is directed. For sake of simplicity, we drop the subscript $a$ from $G_a$.

The problem is clearly in NP as given the protection $P$, the number of infected can be determined in polynomial time through a DFS.

Next, we reduce $\text{CNP}_{\text{split}}$ to $\text{PROTECT}_{\text{dir}}$, showing its NP-hardness. Given an instance of $\text{CNP}_{\text{split}}$, we build the following graph $G = (V, A)$:

- For each $v \in \bar{V}_1$, we create the set of vertices $T_v = \{t^1_v, t^2_v\}$ in $G$, and the arc $(t^1_v, t^2_v)$.
- For each $v \in \bar{V}_2$, we replicate it in $G$, and for each edge $(r, v) \in \bar{E}$ with $v \in \bar{V}_2$, the arc $(t^1_r, v)$ is added in $G$.
- Finally, we add the only attacked vertex $u$ to $G$ and connect it with each $t^1_v$ for $v \in \bar{V}_1$, through the arc $(u, t^1_v)$.

To complete the reduction it remains to set $\Lambda = B$ and $K = \lfloor 2 + \sqrt{8K + 1} \rfloor$ (obtained by solving $K = (\frac{K^2 - 1}{2})$). See Figure 8 for an illustration of the reduction.

![Figure 8: Example of construction of $G$ from $\bar{G}$.

First, note that $C$ of $G$ is $\{t^1_v : v \in \bar{V}_1\} \cup \bar{V}_2$, where the vertices in the first set have value at least 2, and the ones in the second have value 1. Hence, it is clear that the best protection strategy will prioritize the vertices $t^1_v$. In fact, we can argue that only those vertices can be in an optimal protection strategy. If $\Lambda = B \geq \bar{V}_1$, then the instance of $\text{CNP}_{\text{split}}$ is trivial. Therefore, we can assume $\Lambda = B < \bar{V}_1$ and thus, holds $P^* \subset \{t^1_v : v \in \bar{V}_1\}$. Consequently, choosing the optimal $P^*$ means to minimize the vertices in $T_v$, for $v \in \bar{V}_1$, and in $\bar{V}_2$ that are connected to $u$. By construction, those vertices connected with $u$ correspond to a connected component $C_1$ in $G$. Thus, $P^*$ minimizes the size of $\bigcup_{v \in C_1} \{t^2_v\} \cup \{u\} \cup C_1$. The remaining of the proof follows an analogous reasoning to the proof of Theorem 3.3. \square

6.2 Arborescence

In this section we restrict the protection problem to the case where the graph induced by $V \setminus I$ is an arborescence.

Definition 6.3. A DAG $G = (V, A)$ is an arborescence if its underlying undirected graph is a tree (forest) and there is a single vertex (root) that has a unique directed path from it to all other vertices.

In arborescence, it is direct the determination of $C$. Since all vertices in $V \setminus I$ have in-degree 1, either they are protected by their predecessor, and thus not a candidate, or they are direct successors of vertices in $I$. Therefore, $C$ is the set of all successors of vertices in $I$. For an illustration see Figure 9c. We can prove that in this case a greedy approach leads to optimality.
Lemma 6.4. Given $G = (V, A)$, $I$ and $\Lambda$, if the graph induced by $V \setminus I$ is an arborescence, then an optimal protection can be determined in polynomial time, specifically, $O(|V| \log(|V|))$. Moreover, if the induced graph is a set of arborescences, the result also holds.

Proof. We start by showing that a greedy procedure runs in time $O(|V| \log(|V|))$.

As previously observed, for arborescences, the set of candidate vertices is easy to compute: it is the set of all successors of $I$.

Next, the calculation of $p_v$ for each $v \in C$ can be performed through a depth-first-search that records the saved vertices by candidates. This requires $O(|V|)$ since the graph is an arborescence.

Finally, the $\Lambda$ candidate vertices of largest value are protected. This requires to order the vertices accordingly with $\{p_v\}_{v \in C}$. Thus, the greedy method runs in $O(|V| \log(|V|))$.

Next, we show that the described method provides an optimal protection. Let $P$ be the obtained protection through the greedy method. The key idea to prove the optimality of $P$ is essentially due to the fact that in an arborescence, $C$ is simply the set of all successors of $I$, otherwise, if we have a vertex of in-degree at least 2, we do not have an arborescence. Thus, the protection strategy $P$ cannot imply the protection of some candidate not in $P$. This shows the optimality of $P$.

\[\square\]

![Figure 9: The set $I$ is represented by black vertices and candidate vertices are dashed.](image)

Note that in trees (undirected graphs), it does not hold that $C$ is the set of successors of the vertices in $I$. Hence, Lemma 6.4 does not extend to the undirected case.

Remark 6.5. Note that in Lemmata 6.1 and 6.4, we did not use the fact that $b_v = 1$. Thus, it also holds when vertices benefits are not unitary.

Acknowledgements

The authors wish to thank the support of the Institut de valorisation des données and Fonds de Recherche du Québec through the FRQ-IVADO Research Chair in Data Science for Combinatorial Game Theory, and the Natural Sciences and Engineering Research Council of Canada through the discovery grant 2019-04557.
References

[1] Bernardetta Addis, Marco Di Summa, and Andrea Grosso. Identifying critical nodes in undirected graphs: Complexity results and polynomial algorithms for the case of bounded treewidth. *Discrete Applied Mathematics*, 161(16):2349–2360, Nov 2013.

[2] Ashwin Arulselvan, Clayton W. Commander, Lily Elefteriadou, and Panos M. Pardalos. Detecting critical nodes in sparse graphs. *Computers & Operations Research*, 36(7):2193 – 2200, 2009.

[3] Andrea Baggio, Margarida Carvalho, Andrea Lodi, and Andrea Tramontani. Multilevel approaches for the critical node problem. *Operations Research*, To appear, 2020.

[4] Charles Blair. The computational complexity of multi-level linear programs. *Annals of Operations Research*, 34(1):13–19, Dec 1992.

[5] Vladimir Boginski and Clayton W. Commander. Identifying Critical Nodes in Protein-Protein Interaction Networks, pages 153–167. Clustering Challenges in Biological Networks. World Scientific, Feb 2009.

[6] Gerald Brown, Matt Carlyle, Javier Salmerón, and R. Wood. Defending critical infrastructure. *Interfaces*, 36:530–544, 12 2006.

[7] Alberto Caprara, Margarida Carvalho, Andrea Lodi, and Gerhard J. Woeginger. A study on the computational complexity of the bilevel knapsack problem. *SIAM Journal of Optimization*, 24:823–838, 2014.

[8] Margarida Carvalho, Andrea Lodi, and Patrice Marcotte. A polynomial algorithm for a continuous bilevel knapsack problem. *Operations Research Letters*, 46(2):185 – 188, 2018.

[9] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.

[10] S. DeNegre. *Interdiction and Discrete Bilevel Linear Programming*. PhD thesis, Lehigh University, 2011.

[11] Tibor Dudás, Bettina Klinz, and Gerhard J. Woeginger. The computational complexity of multi-level bottleneck programming problems. In Athanasios Migdalas, Panos M. Pardalos, and Peter Varbrand, editors, *Multilevel Optimization: Algorithms and Application*, pages 165–179, Boston, MA, 1998. Springer US.

[12] Stephen Finbow, Andrew King, Gary MacGillivray, and Romeo Rizzi. The firefighter problem for graphs of maximum degree three. *Discrete Mathematics*, 307(16):2094–2105, Jul 2007.

[13] Fabio Furini, Ivana Ljubic, Enrico Malaguti, and Paolo Paronuzzi. Casting light on the hidden bilevel combinatorial structure of the k-vertex separator problem. In *OR-19-6, DEI, University of Bologna*, 2019.

[14] Fabio Furini, Ivana Ljubic, Sebastien Martin, and Pablo San Segundo. The maximum clique interdiction problem. *European Journal of Operational Research*, 277:112–127, 02 2019.

[15] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., USA, 1990.

[16] Jing He, Hongyu Liang, and Hao Yuan. Controlling infection by blocking nodes and links simultaneously. In Ning Chen, Edith Elkind, and Elias Koutsoupias, editors, *Internet and Network Economics*, pages 206–217, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[17] Robert G. Jeroslow. The polynomial hierarchy and a simple model for competitive analysis. *Mathematical Programming*, 32(2):146–164, Jun 1985.

[18] Berit Johannes. *New Classes of Complete Problems for the Second Level of the Polynomial Hierarchy*. PhD thesis, Technischen Universität Berlin, 2011.

[19] Richard M. Karp. *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA, 1972.

[20] Mohammed Lalou, Mohammed Amin Tahraoui, and Hamamache Kheddouci. The critical node detection problem in networks: A survey. *Computer Science Review*, 28:92 – 117, 2018.

[21] John M. Lewis and Mihalis Yannakakis. The node-deletion problem for hereditary properties is np-complete. *Journal of Computer and System Sciences*, 20(2):219–230, Apr 1980.

[22] P.A. Martin. Tri-level optimization models to defend critical infrastructure. Master’s thesis, Naval Postgraduate School, Monterey, 01 2007.

[23] A. R. Meyer and L. J. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *15th Annual Symposium on Switching and Automata Theory (swat 1972)*, pages 125–129, 1972.
[24] Siqian Shen, J. Cole Smith, and Roshan Goli. Exact interdiction models and algorithms for disconnecting networks via node deletions. *Discrete Optimization*, 9(3):172–188, Aug 2012.

[25] L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time (preliminary report). In *Proceedings of the Fifth Annual ACM Symposium on Theory of Computing*, STOC ’73, page 1–9, New York, NY, USA, 1973. Association for Computing Machinery.

[26] Vera Toomaino, Ashwin Arulselvan, Pierangelo Veltri, and Panos M. Pardalos. *Studying Connectivity Properties in Human Protein–Protein Interaction Network in Cancer Pathway*, pages 187–197. Springer US, Boston, MA, 2012.

[27] Mario Ventresca, Kyle Robert Harrison, and Beatrice M. Ombuki-Berman. The bi-objective critical node detection problem. *European Journal of Operational Research*, 265(3):895–908, Mar 2018.

[28] R. Kevin Wood. Deterministic network interdiction. *Mathematical and Computer Modelling*, 17(2):1–18, Jan 1993.

[29] Celia Wrathall. Complete sets and the polynomial-time hierarchy. *Theoretical Computer Science*, 3(1):23–33, Oct 1976.