Efficient Best Response Computation for Strategic Network Formation under Attack

Tobias Friedrich* Sven Ihde* Christoph Keßler* Pascal Lenzner*†
Stefan Neubert* David Schumann*

Abstract

Inspired by real world examples, e.g. the Internet, researchers have introduced an abundance of strategic games to study natural phenomena in networks. Unfortunately, almost all of these games have the conceptual drawback of being computationally intractable, i.e. computing a best response strategy or checking if an equilibrium is reached is NP-hard. Thus, a main challenge in the field is to find tractable realistic network formation models.

We address this challenge by investigating a very recently introduced model by Goyal et al. [WINE'16] which focuses on robust networks in the presence of a strong adversary who attacks (and kills) nodes in the network and lets this attack spread virus-like to neighboring nodes and their neighbors. Our main result is to establish that this natural model is one of the few exceptions which are both realistic and computationally tractable. In particular, we answer an open question of Goyal et al. by providing an efficient algorithm for computing a best response strategy, which implies that deciding whether the game has reached a Nash equilibrium can be done efficiently as well. Our algorithm essentially solves the problem of computing a minimal connection to a network which maximizes the reachability while hedging against severe attacks on the network infrastructure and may thus be of independent interest.

1 Introduction

Many of today's important networks, most prominently the Internet, are essentially the outcome of an unsupervised decentralized network formation process among many selfish entities [22]. In the case of the Internet these selfish entities are Autonomous Systems (AS) which interconnect via peering agreements and thereby create a connected network of networks. Each AS can be understood as a selfish player who strategically chooses a subset of other ASs to directly connect with. Each inter-AS-connection is costly and yields a benefit and a risk. The benefit is a reliable direct link towards the other AS. However, such a connection may be used by malicious software and thus harbors the risk of collateral damage if a neighboring AS is attacked.

The field of strategic network formation, started by the seminal works of Jackson & Wolinsky [15], Bala & Goyal [2] and Fabrikant et al. [11], studies the global structure and properties of networks formed by individual players making decentralized local strategic choices. In all considered models there are players trying to optimize their own benefit, while minimizing their individual cost. It is far from obvious why a collection of individual selfish strategies eventually results in useful and reliable network topologies like the Internet. Studying the properties of such models aims for revealing insights about properties of existing naturally grown networks and inspiring methods to improve them.

*Algorithm Engineering Group, Hasso Plattner Institute Potsdam, Germany
†pascal.lenzner@hpi.de

Required features of any Internet-like communication network are reachability and robustness. Such networks have to ensure that even in case of cascading edge or node failures caused by technical defects or malicious attacks, e.g. DDoS-attacks or viruses, most participating nodes can still communicate. This important focus on network robustness has long been neglected and is now a very recent endeavor in the strategic network formation community, see e.g. [6, 14, 17, 20]. We contribute to this endeavor by proving that the very recently introduced natural model by Goyal et al. [13, 14] is one of the few exceptions of a tractable network formation model. In particular, we provide an efficient algorithm for computing a utility maximizing strategy for their elegant model, which can be used to efficiently decide whether a network is in Nash equilibrium. Thus, our algorithm allows the model of Goyal et al. to be used to predict real world phenomena in large scale simulations and to analyze real world networks. Moreover, predicted structural properties of equilibria, which are obtained by a tractable model, are more plausible since it can be assumed that computationally bounded agents in the real world will eventually end up in an equilibrium state or at least close to it.

1.1 Related Work

We focus on the model for strategic network formation with attack and immunization recently proposed by Goyal et al. [13, 14]. This model essentially augments the well-known reachability model by Bala & Goyal [2] with robustness considerations. In particular, different types of adversaries are introduced which attack (and destroy) a node of the network. This attack then spreads virus-like to neighboring nodes and destroys them as well. Besides deciding which links to form, players also decide whether they want to buy immunization against eventual attacks. The model is the first model which incorporates network formation and immunization decisions at the same time.

On the one hand, the authors of [13, 14] provide beautiful structural results for their model. For example, showing that equilibrium networks are much more diverse than in the non-robust version, proving that the amount of edge overbuilding due to robustness concerns is small and establishing that equilibrium networks generally achieve very high social welfare. On the other hand, the authors raise the intriguing open problem of settling the complexity of computing a best response strategy in their model[1].

Computing a best response in network formation games can be done in polynomial time for the non-robust reachability model [2] and if the allowed strategy changes are very simple [19]. However, these examples are exceptions. The existence of an efficient best response algorithm for a network formation game is in general a rare gem. For almost all related network formation models, e.g. [4–6, 8, 10, 11, 21], where players strive for a central position in the network, it has been shown that the problem is indeed NP-hard.

To the best of our knowledge, besides the model by Goyal et al. [13, 14] there are only a few other models which combine selfish network formation with robustness considerations and all of them consider a much weaker adversary which can only destroy a single edge. The earliest are models by Bala & Goyal [3] and Kliemann [17], both essentially augment the model by Bala & Goyal [2] with single edge failures. Other related models are by Meirom et al. [20] and Chauhan et al. [6]. Both latter models consider players who try to be as central as possible in the created networks but at the same time want to protect themselves against single edge failures. In [20] heterogeneous players are considered whereas in [6] all players are homogeneous. The complexity status for computing a best response was only settled for the model by Chauhan et al. [6] where this problem was proven to be NP-hard.

Apart from network formation games, also vaccination games, e.g. [1, 7, 18, 23], are related.

[1] This question was raised in [13] for the maximum carnage adversary and is replaced in [14] with a reference to our preprint of the present paper. The preprint also appeared as a Brief Announcement [12]

There the network is fixed and the selfish nodes only have to decide if the want to immunize or not. Computing a best response in these models is trivial (there are only two strategies) but pure Nash equilibria may not exist.

1.2 Our Contribution

We establish that the natural model by Goyal et al. [13, 14] is one of the few examples of a tractable realistic model for strategic network formation and thereby answer an open question by these authors. In particular, we provide an efficient algorithm for computing a best response strategy for their main model, i.e. the "maximum carnage" adversary which tries to kill as many nodes as possible, and for the natural variant which employs the even stronger random attack adversary. Moreover, we employ our algorithm in empirical simulations which augment the extensive simulations presented in [13, 14], where players were allowed to perform only heavily restricted strategy changes. We observe fast and reliable convergence, despite the fact that the authors of [13, 14] have shown that best response dynamics can cycle, a phenomenon also observed in other network formation games [16].

1.3 Organization of the Paper

In Section 2 we introduce the model by Goyal et al. [13, 14] and some additional definitions and notation. Our main contribution follows in Section 3, where we introduce the main ideas and a detailed description of our algorithm. Several subroutines are used and we first show how these subroutines work together to solve the main problem. After providing the big picture, we then supply the subroutines with all details and correctness proofs separately.

We demonstrate the behavior of our algorithm experimentally in Section 3.7. Moreover, in Section 4, we consider the random attack adversary which is less predictable than the maximum carnage adversary. There we show how to adapt our algorithm with a few minor tweaks to cope with this more complex scenario. We conclude in Section 5 where we discuss possible future research directions.

2 Model

We work with the strategic network formation model proposed by Goyal et al. [13, 14] and mostly use their notation. In this model the n nodes of a network $G = (V, E)$ correspond to individual players $v_1, \ldots, v_n$. We will thus use the terms node, vertex and player interchangeably. The edge set E is determined by the players' strategic behavior as follows. Each player $v_i \in V$ can decide to buy undirected edges to a subset of other players, paying $\alpha > 0$ per edge, where α is some fixed parameter of the model.

If player v_i decides to buy the edge to node v_j, then we say that the edge $\{v_i, v_j\}$ is owned and paid for by player v_i.[2] Buying an undirected edge entails connectivity benefits and risks for both participating endpoints. In order to cope with these risks, each player can also decide to buy immunization against attacks at a cost of $\beta > 0$, which is also a fixed parameter of the model. We call a player *immunized* if this player decides to buy immunization, and *vulnerable* otherwise.

The strategy $s_i = (x_i, y_i)$ of player v_i consists of the set $x_i \subseteq V \setminus \{v_i\}$ of the nodes to buy an edge to, and the immunization choice $y_i \in \{0, 1\}$, where $y_i = 1$ if and only if player v_i decides to

[2] If both players v_i and v_j decide to buy the edge $\{v_i, v_j\}$, then this results in a multi-edge between v_i and v_j. However, it is easy to see that best response strategies will never contain multi-edges which is why we ignore them completely.

immunize. The strategy profile $\mathbf{s} = (s_1, \ldots, s_n)$ of all players then induces an undirected graph

$$G(\mathbf{s}) = \left(V, \bigcup_{v_i \in V} \bigcup_{v_j \in x_i} \{v_i, v_j\} \right).$$

The immunization choices $y_1, \ldots, y_n$ in $\mathbf{s}$ partition V into the set of immunized players $\mathcal{I} \subseteq V$ and vulnerable players $\mathcal{U} = V \setminus \mathcal{I}$. The components in the induced subgraph $G[\mathcal{U}]$ are called *vulnerable regions* and the set of those regions will be denoted by $\mathcal{R}_\mathcal{U}$. The vulnerable region of any vulnerable player $v_i \in \mathcal{U}$ is $\mathcal{R}_\mathcal{U}(v_i)$. *Immunized regions* $\mathcal{R}_\mathcal{I}$ are defined analogously as the components of the induced subgraph $G[\mathcal{I}]$.

After the network $G(\mathbf{s})$ is built, we assume that an adversary attacks one vulnerable player according to a strategy known to the players. We consider mostly the maximum carnage adversary [13, 14] which tries to destroy as many nodes of the network as possible. To achieve this, the adversary chooses a vulnerable region of maximum size and attacks some player in that region. If there is more than one such region with maximum size, then one of them is chosen uniformly at random. If a vulnerable player v_i is attacked, then v_i will be destroyed and the attack spreads to all vulnerable neighbors of v_i, eventually destroying all players in v_i's vulnerable region $\mathcal{R}_\mathcal{U}(v_i)$. Let $t_{max} = \max_{R \in \mathcal{R}_\mathcal{U}}\{|R|\}$ be the number of nodes in the vulnerable region of maximum size and $\mathcal{T} = \{v_i \in \mathcal{U} \mid |\mathcal{R}_\mathcal{U}(v_i)| = t_{max}\}$ is the corresponding set of nodes which may be targeted by the adversary. The set of targeted regions is $\mathcal{R}_\mathcal{T} = \{R \in \mathcal{R}_\mathcal{U} \mid |R| = t_{max}\}$, and $\mathcal{R}_\mathcal{T}(v_i)$ is the targeted region of a player $v_i \in \mathcal{T}$. Thus, if $v_i \in \mathcal{T}$ is attacked, then all players in the region $\mathcal{R}_\mathcal{T}(v_i)$ will be destroyed.

The *utility* of a player v_i in network $G(\mathbf{s})$ is defined as the expected number of nodes reachable by v_i after the adversarial attack on network $G(\mathbf{s})$ (zero in case v_i was destroyed) less v_i's expenditures for buying edges and immunization. More formally, let $CC_i(t)$ be the connected component of v_i after an attack to node $v_t \in \mathcal{T}$ and let $|CC_i(t)|$ denote its number of nodes. Then the utility (or profit) $u_i(\mathbf{s})$ of v_i in the strategy profile $\mathbf{s}$ is

$$u_i(\mathbf{s}) = \frac{1}{|\mathcal{T}|} \left(\sum_{v_t \in \mathcal{T}} |CC_i(t)| \right) - |x_i| \cdot \alpha - y_i \cdot \beta.$$

Fixing the strategies of all other players, the *best response* of a player v_i is a strategy $s_i^* = (x_i^*, y_i^*)$ which maximizes v_i's utility $u_i\big((s_1, \ldots, s_{i-1}, s_i^*, s_{i+1}, \ldots, s_n)\big)$. We will call the strategy change to s_i^* a best response for player v_i in the network $G(\mathbf{s})$, if changing from strategy $s_i \in \mathbf{s}$ to strategy s_i^* is the best possible strategy for player v_i if no other player changes her strategy.

Consider what happens if we remove node v_i from the network $G(\mathbf{s}) = (V, E)$ and we call the obtained network $G(\mathbf{s}) \setminus v_i$. In this case, $G(\mathbf{s}) \setminus v_i$ consists of connected components $C_1, \ldots, C_\ell$. The edge-set x_i^* can thus be partitioned into ℓ subsets $x_i^*(C_1), \ldots, x_i^*(C_\ell)$, where $x_i^*(C_z)$ denotes the set of nodes in C_z to which v_i buys an edge under best response strategy s_i^*. We will say that $x_i^*(C_z)$ is an *optimal partner set* for component C_z. Therefore, x_i^* is the union of optimal partner sets for all connected components in $G(\mathbf{s}) \setminus v_i$.

A best response is calculated for one arbitrary but fixed player v_a, which we call the *active player*. Furthermore let $\mathcal{C}$ be the set of connected components which exist in $G(\mathbf{s}) \setminus v_a$. Let

$$\begin{aligned} \mathcal{C}_\mathcal{U} &= \{C \in \mathcal{C} \mid C \cap \mathcal{I} = \emptyset\}, \\ \mathcal{C}_\mathcal{I} &= \mathcal{C} \setminus \mathcal{C}_\mathcal{U}, \\ \mathcal{C}_{inc} &= \{C \in \mathcal{C} \mid \exists u \in C : \{u, v\} \in E\}, \end{aligned}$$

where $\mathcal{C}_\mathcal{U}$ is the set of components in which all vertices are vulnerable, $\mathcal{C}_\mathcal{I}$ is the set of components which contain at least one immunized vertex and $\mathcal{C}_{inc}$ is the set of components to which player v_a is connected through incoming edges bought by some other player.

3 The Best Response Algorithm

A naive approach to calculate the best response for player v_a would consider all 2^n possible strategies and select one that yields the best utility. This is clearly infeasible for a larger number of players.

3.1 Key Observations

Our algorithm exploits three observations to reduce the complexity from exponential to polynomial:

Observation 1: The network $G(\mathbf{s}) \setminus v_a$ may consist of ℓ connected components that can be dealt with independently for most decisions. As long as the set of possible targets of the adversary does not change, the best response of v_a can be constructed by first choosing components to which a connection is profitable and then choosing for each of those components an optimal set of nodes within the respective component to build edges to.

Observation 2: Homogeneous components in $G(\mathbf{s}) \setminus v_a$, which consist of only vulnerable or only immunized nodes, provide the same benefit no matter whether v_a connects to them with one or with more than one edge. Thus the connection decision is a binary decision for those components.

Observation 3: Mixed components in $G(\mathbf{s}) \setminus v_a$, which contain both immunized and vulnerable nodes, consist of homogeneous regions that again have the property that at most one edge per homogeneous region can be profitable. Merging those regions into block nodes forms an auxilliary tree, called Meta Tree, which we use in an efficient dynamic programming algorithm to compute the most profitable subset of regions to connect with.

Using the above observations, we construct an algorithm with a worst-case run time complexity of $\mathcal{O}(n^4 + k^5)$ where n is the number of nodes of the network and k the number of block nodes in the largest Meta Tree.

3.2 Main Algorithm

In this section, we introduce our main algorithm, called BestResponseComputation. Its pseudo code can be found in Algorithm 1 and a schematic overview is depicted in Fig. 1.

Our algorithm solves the problem of finding a best response strategy by considering both options of buying or not buying immunization and computing for both cases the best possible set of edges to buy. Thus, the first step of BestResponseComputation is to drop the current strategy of the active player v_a and to replace it with the empty strategy $s_\emptyset = (\emptyset, 0)$ in which player v_a does not buy any edge and does not buy immunization. Then the resulting strategy profile $\mathbf{s}' = (s_1, \ldots, s_{a-1}, s_\emptyset, s_{a+1}, \ldots, s_n)$ and the set of connected components $\mathcal{C}_\mathcal{U}$ and $\mathcal{C}_\mathcal{I}$ with respect to network $G(\mathbf{s}') \setminus v_a$ is considered.

The subroutine SubsetSelect, which is described in Section 3.4.1, determines the optimal sets of components of $\mathcal{C}_\mathcal{U}$ to connect to if v_a does not immunize. This is done by solving an adjusted Knapsack problem which involes only small numbers. Two such sets of components, called $\mathcal{A}_t$ and $\mathcal{A}_v$, are computed depending on player v_a becoming targeted or not by connecting to these components. Additionally the subroutine GreedySelect, which is described in Section 3.4.2, computes a best possible subset of components of $\mathcal{C}_\mathcal{U}$ to connect with in case v_a buys immunization by greedily connecting to all components in $\mathcal{C}_\mathcal{U}$ that are profitable.

The challenging part of the problem is to cope with the connected components in $\mathcal{C}_\mathcal{I}$ which also contain immunized nodes. For such components our algorithm detects and merges equivalent nodes and thereby simplifies these components to an auxiliary tree structure, which we call the Meta Tree. This tree is then used in a dynamic programming fashion to efficiently compute the best possible set of edges to buy towards nodes within the respective component. Thus,

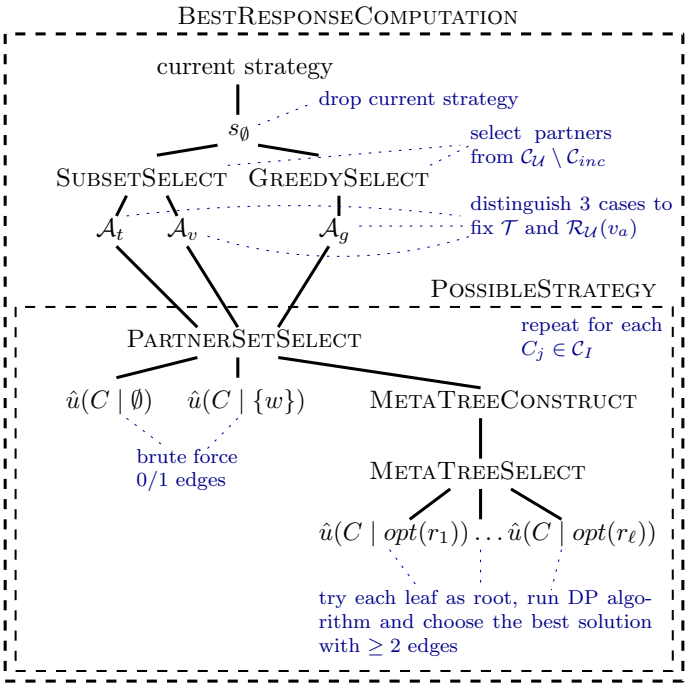

Figure 1: Schematic overview of the best response algorithm.

our approach for handling components containing immunized nodes can be understood as first performing a data-reduction similar to many approaches for kernelization in the realm of Parameterized Algorithmics [9] and then solving the reduced problem via dynamic programming.

Algorithm 1: BestResponseComputation

Input: Strategies $\mathbf{s} = (s_1, \ldots, s_n)$, Player $v_a, 1 \leq a \leq n$

Output: Best response strategy of player v_a denoted by $s_a = (x_a, y_a)$

$s_\emptyset = (\emptyset, 0)$;

Let $G(\mathbf{s}')$ be the induced game state with $\mathbf{s}' = (s_1, \ldots, s_{a-1}, s_\emptyset, s_{a+1}, \ldots, s_n)$;

Let $\mathcal{A}_t, \mathcal{A}_v$ be the solutions of SubsetSelect on $\mathcal{C}_\mathcal{U}$;

Let $\mathcal{A}_g$ be the solution of GreedySelect on $\mathcal{C}_\mathcal{U}$;

s_t = PossibleStrategy($\mathcal{A}_t$, 0);

s_v = PossibleStrategy($\mathcal{A}_v$, 0);

s_g = PossibleStrategy($\mathcal{A}_g$, 1);

$S = \{s_\emptyset, s_t, s_v, s_g\}$;

return strategy $s \in S$ which maximizes v_a's utility;

The subroutine PossibleStrategy, see Algorithm 2, obtains the best set of nodes in components in $\mathcal{C}_\mathcal{I}$. As this set depends on the number of targeted regions, it has to be determined for several cases independently. These cases are v_a not being immunized and not being targeted, v_a not being immunized but being targeted, and v_a being immunized. For each case, the subroutine PossibleStrategy first chooses an arbitrary single edge to buy into the previously selected components from $\mathcal{C}_\mathcal{U}$.

Algorithm 2: PossibleStrategy($\mathcal{A}$, y_a)

Output: Best strategy with single edges to components in $\mathcal{A}$, given immunization y_a

$M := \emptyset$;
foreach $C \in \mathcal{A}$ **do**
  $M = M \cup v_i$ for an arbitrary node $v_i \in C$;
Locally, add edges to nodes in M, update $\mathcal{R}_\mathcal{I}, \mathcal{R}_\mathcal{U}, \mathcal{R}_\mathcal{T}$ according to y_a and M;
$B \leftarrow \emptyset$;
foreach *component* $C \in \mathcal{C}_\mathcal{I}$ **do**
  $B \leftarrow B \cup$ PartnerSetSelect(C)
return $(M \cup B, y_a)$;

Then the best set of edges to buy into components in $\mathcal{C}_\mathcal{I}$ is computed independently for each component $C \in \mathcal{C}_\mathcal{I}$ via the subroutines PartnerSetSelect (see Section 3.5.1), MetaTreeConstruct (see Section 3.5.2) and MetaTreeSelect (see Section 3.5.4). The union of the obtained sets is then returned. Finally, the algorithm compares the empty strategy and the individually obtained best possible strategies for the above mentioned cases and selects the one which maximizes player v_a's utility.

We establish that all the mentioned subroutines work together as desired in Section 3.3. We specify the constructed Meta Tree in Section 3.5.2 and the MetaTreeSelect algorithm in Section 3.5.4.

The run time of our best response algorithm heavily depends on the size of the largest obtained Meta Tree and we achieve a worst-case run time of $\mathcal{O}(n^4 + k^5)$ for the maximum carnage adversary and $\mathcal{O}(n^4 + nk^5)$ for the random attack adversary, where n is the number of nodes in the network and k is the number of blocks in the largest Meta Tree (See Section 3.6 and Section 4). In the worst case, this yields a run time of $\mathcal{O}(n^5)$ and $\mathcal{O}(n^6)$, respectively. To contrast this worst-case bound, we also provide empirical results in Section 3.7 showing that k is usually much smaller than n, which emphasizes the effectiveness of our data-reduction and thereby shows that our algorithm is expected to be much faster than the worst-case upper bound.

3.3 Correctness of PossibleStrategy and BestResponseComputation

We now prove the correctness of PossibleStrategy and BestResponseComputation under the assumption that the subroutines SubsetSelect, GreedySelect and PartnerSetSelect are correct.

It is important to note, that PartnerSetSelect will not only return a best set of edges to buy into a component $C \in \mathcal{C}_\mathcal{I}$ but that all these edges connect to immunized nodes (see Lemma 5). Thus, these edges do not change the set of targeted nodes $\mathcal{T}$ or player v_a's vulnerable region $\mathcal{R}_\mathcal{U}(v_a)$.

We start with proving the correctness of subroutine PossibleStrategy. This subroutine gets a set of components from $\mathcal{C}_\mathcal{U}$ as input and then chooses an arbitrary node from each component to buy an edge to. First, we show that this is correct.

Lemma 1. *Buying at most one edge into any component $C \in \mathcal{C}_\mathcal{U}$ yields maximum profit for player v_a.*

Proof. Observe that in the case of an attack on component C it is destroyed completely, if the attack takes place elsewhere C stays connected. Hence, a single edge to any player in C provides connectivity to all surviving players in C in all cases, more edges therefore would only increase expenses but not connectivity. □

Thus, the optimal choice of edges into components in $\mathcal{C}_\mathcal{U}$ only depends on the right choice of the components by SubsetSelect and GreedySelect.

For components in $\mathcal{C}_\mathcal{I}$ we assume that PartnerSetSelect is correct and that buying edges into components in $\mathcal{C}_\mathcal{I}$ do not change $\mathcal{T}$ or $\mathcal{R}_\mathcal{U}(v_a)$. We therefore only have to prove that under this assumption it is indeed correct to handle all components in $\mathcal{C}_\mathcal{I}$ independently. For establishing this result, we first have to define player v_a's expected profit contribution of a single component $C \in \mathcal{C}_\mathcal{I}$.

3.3.1 Calculation of the Expected Profit Contribution of a Single Component

Assume that the adversary attacks vertex t. In this case, by abusing notation, let $CC_{v_a}(t) \cap C$ denote the set of nodes in component C that are still connected to player v_a after the attack on vertex t and therefore contribute to v_a's profit. In order to compute the expected profit contribution of a component $C \in \mathcal{C}_\mathcal{I}$ for player v_a, we simply have to take the expectation of $|CC_{v_a}(t) \cap C|$ over all possible choices for an attack.

Let $\hat{u}_{v_a}(C \mid \Delta)$ be the profit contribution for player v_a of a component $C \in \mathcal{C}_\mathcal{I}$ if v_a buys edges to all nodes in the set Δ. Thus,

$$\hat{u}_{v_a}(C \mid \Delta) = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} |CC_{v_a}(t) \cap C| - \alpha|\Delta|.$$

3.3.2 Conditional Independence within $\mathcal{C}_\mathcal{I}$

We proceed to proving that handling components in $\mathcal{C}_\mathcal{I}$ independently is indeed correct.

Lemma 2. *Player v_a can deal with distinct components from $\mathcal{C}_\mathcal{I}$ independently, if $\mathcal{T}$ and $\mathcal{R}_\mathcal{U}(v_a)$ do not change.*

Proof. Recall that

$$u_{v_a}(\mathbf{s}) = \frac{1}{\mathcal{T}} \sum_{t \in \mathcal{T}} |CC_{v_a}(t)| - |x_a|\alpha - y_a\beta.$$

Every summand can be split according to the components that player v can buy edges to. Let Δ_C be the edges v_a buys into C. Then

$$\begin{aligned} u_{v_a}(\mathbf{s}) = \frac{1}{\mathcal{T}} \sum_{t \in \mathcal{T}} \Big(& \sum_{C \in \mathcal{C}_\mathcal{I}} \big(|CC_{v_a}(t) \cap C| - |\Delta_C|\alpha\big) \\ & + \sum_{C \in \mathcal{C}_\mathcal{U}} \big(|CC_{v_a}(t) \cap C| - |\Delta_C|\alpha\big)\Big) - y_a\beta. \end{aligned}$$

This can be written as

$$\begin{aligned} u_{v_a}(\mathbf{s}) &= \sum_{C \in \mathcal{C}_\mathcal{I}} \left(\frac{1}{\mathcal{T}} \sum_{t \in \mathcal{T}} \big|CC_{v_a}(t) \cap C\big| - |\Delta_C|\alpha \right) \\ &\quad + \frac{1}{\mathcal{T}} \sum_{t \in \mathcal{T}} \left(\sum_{C \in \mathcal{C}_\mathcal{U}} \big(\big|CC_{v_a}(t) \cap C\big| - |\Delta_C|\alpha\big) \right) - y_a\beta \\ &= \sum_{C \in \mathcal{C}_\mathcal{I}} \hat{u}_{v_a}(C \mid \Delta_C) \\ &\quad + \underbrace{\frac{1}{\mathcal{T}} \sum_{t \in \mathcal{T}} \left(\sum_{C \in \mathcal{C}_\mathcal{U}} \big(\big|CC_{v_a}(t) \cap C\big| - |\Delta_C|\alpha\big) \right) - y_a\beta}_{(**)}, \end{aligned}$$

using linearity of expectation and the definition of $\hat{u}$. Since both $\mathcal{T}$ and $\mathcal{R}_{\mathcal{U}}(v_a)$ remain unchanged, $(**)$ is fixed.

The choice of nodes to connect to within any component C does not influence v_a's connectivity to nodes outside of C as every path from some node in C to some node outside C traverses node v_a. Thus, the choice of nodes in C is independent from the choice of nodes in other components. Hence, the first summation's terms are independent.

Because every term is non-negative, the summands may be maximized independently of each other to find the best response. □

Lemma 1 and Lemma 2 directly yield the following:

Corollary 1. *If* PartnerSetSelect *is correct and selects only edges to immunized nodes, then* PossibleStrategy *is correct.*

3.3.3 Correctness of BestResponseComputation

Now we show that the main subroutines work together as desired.

Theorem 1. *If the algorithms* SubsetSelect, GreedySelect *and* PossibleStrategy *are correct, then* BestResponseComputation *is correct.*

Proof. We assume that PossibleStrategy computes an optimal set of edges into components in $\mathcal{C}_{\mathcal{I}}$ and that all those edges connect to immunized nodes.

Firstly, observe that if v_a decides to immunize, then the player cannot change $\mathcal{T}$ or $\mathcal{R}_{\mathcal{U}}(v_a)$ by buying edges. Moreover, in this case GreedySelect computes the optimal set of components in $\mathcal{C}_{\mathcal{U}}$ to connect with via a single edge. Thus, strategy s_g (see Alg. 1) is indeed a best possible strategy for this case.

The only changes to $\mathcal{T}$ or $\mathcal{R}_{\mathcal{U}}(v_a)$ can happen in case v_a is vulnerable and buys edges to one or more vulnerable nodes. After v_a's edge purchases we distinguish the following cases:

Case 1: The new vulnerable region of v_a is strictly smaller than t_{max}: The set of targeted nodes $\mathcal{T}$ does not change and v_a remains un-targeted. By assumption SubsetSelect has chosen the optimal set of components $\mathcal{A}_v$ from $\mathcal{C}_{\mathcal{U}}$ and thus strategy s_v (see Alg. 1) is the best strategy in this case.

Case 2: The new vulnerable region of v_a has the size of exactly t_{max}: This increases the number of targeted nodes $|\mathcal{T}|$ by t_{max} as v_a becomes targeted. By assumption SubsetSelect has chosen the optimal set of components $\mathcal{A}_t$ from $\mathcal{C}_{\mathcal{U}}$ and thus strategy s_t (see Alg. 1) is the best strategy in this case.

Case 3: The new vulnerable region of v_a is larger than t_{max}: In this case v_a will be part of the single largest vulnerable region and thus killed. Clearly, in this case the only and best strategy for v_a is to not buy any edges.

Thus, BestResponseComputation covers all relevant cases and is therefore correct. □

3.4 Subroutines Dealing with Components in $\mathcal{C}_{\mathcal{U}}$

Components in $\mathcal{C}_{\mathcal{U}}$ are handled by the algorithms SubsetSelect and GreedySelect. The details of these algorithms are specified in this section.

3.4.1 SubsetSelect: Interdependent Subset Selection of $\mathcal{C}_{\mathcal{U}}$

We assume, the best response of v_a does not include immunization. By Lemma 1 we know that the problem of finding optimal edge endpoints in components in $\mathcal{C}_{\mathcal{U}}$ is equivalent to the problem of finding a subset of components in $\mathcal{C}_{\mathcal{U}}$ that is most profitable. In particular, it cannot

be beneficial for v_a to buy edges to components from $\mathcal{C}_\mathcal{U} \cap \mathcal{C}_{inc}$, as she is already connected to those. Thus, $\mathcal{C}_\mathcal{U} \setminus \mathcal{C}_{inc} = \{C_1, \ldots, C_m\}$ are the vulnerable components v_a might buy an edge to.

The algorithm SubsetSelect computes the subset of the set of components $\{C_1, \ldots, C_m\}$ which later yields the highest utility, by solving an adjusted Knapsack problem with $|C_i|$ as profit and the total number of connected nodes and the total number of newly built edges as weight limits.

In the following, let M be a 3-dimensional table, where a cell $M[x, y, z]$ of the table stores the maximum number smaller or equal to z of nodes that player v_a can connect to, using only components from $\{C_1, \ldots, C_x\}$ and building at most y edges in total. As we only buy a maximum of one edge per component, the dimensions of M are $n \times m \times n$. The dynamic program therefore uses $\mathcal{O}(n^2 m)$ space.

M is computed as follows:

$$M[0, \cdot, \cdot] = M[\cdot, 0, \cdot] = M[\cdot, \cdot, 0] = 0,$$

$$M[x, y, z] = \begin{cases} M[x-1, y, z], & |C_x| > z, \\ \max\Big(|C_x| + M\big[x-1, y-1, z-|C_x|\big],\ M\big[x-1, y, z\big]\Big), & |C_x| \leq z. \end{cases}$$

Let $r = t_{max} - |\mathcal{R}_\mathcal{U}(v)|$ be the remaining number of vulnerable nodes v_a may connect to without forming a unique targeted region. Then, the algorithm picks two solutions a_t, a_v from M defined as:

$$a_t = \max_{0 \leq j \leq m} \{M[m, j, r] - j \cdot \alpha\}$$
$$a_v = \max_{0 \leq j \leq m} \{M[m, j, r-1] - j \cdot \alpha\}$$

Let $\mathcal{A}_t, \mathcal{A}_v \subseteq \{C_1, \ldots, C_m\}$ be the corresponding subsets to the solutions a_t, a_v, which can be constructed by tracking the predecessor of a field in M.

Hence, a_t is the maximum benefit we gain by connecting to $|\mathcal{A}_t| \leq r$ additional nodes, a_v is the maximum benefit with at most $|\mathcal{A}_v| < r$ additional nodes. Thus, by using $\mathcal{A}_t$ for the solution, v_a might become part of a vulnerable region of maximum size and the best response algorithm has to decide later, whether it is beneficial to risk being attacked. By taking $\mathcal{A}_v$, the vulnerable region of v_a will remain strictly smaller than t_{max} and thus v_a won't be targeted by the adversary.

The algorithm has to store both solutions to decide later, which one is better, as the additional risk has influence on the total utility of the player.

If $a_t \leq 0$ and $a_v \leq 0$, then the best response of node v_a does not include edges to components in $\mathcal{C}_\mathcal{U}$ under the condition that v_a does not immunize.

We omit the straightforward correctness proof.

3.4.2 GreedySelect: Greedy Subset Selection of $\mathcal{C}_\mathcal{U}$

If v_a immunizes, she does not incur any risk by buying new edges into components from $\mathcal{C}_\mathcal{U}$. Thus, by Lemma 1 she can buy single edges to all components $C \in \mathcal{C}_\mathcal{U} \setminus \mathcal{C}_{inc}$ which contribute more to her expected profit than the edge cost α.

Therefore the GreedySelect algorithm computes the set $\mathcal{A}_g$ which is the set of vulnerable components player v_a buys an edge to if she immunizes:

$$\mathcal{A}_g = \{C \in \mathcal{C}_\mathcal{U} \setminus \mathcal{C}_{inc} \mid |C| \cdot p_{survive}(C) > \alpha\}$$
$$p_{survive}(c) = 1 - \frac{|C \cap \mathcal{T}|}{|\mathcal{T}|}.$$

We omit the straightforward correctness proof.

3.5 Partner Selection for Components in $\mathcal{C}_\mathcal{I}$

Let $C_1, \ldots, C_c \in \mathcal{C}_\mathcal{I}$ be the components v_a might buy edges into. By definition, each of those components contains at least one immunized node. Lemma 5 (see Section 3.5.3) ensures that we only need to consider buying edges to immunized nodes.

For computing an optimal partner set for a component $C \in \mathcal{C}_\mathcal{I}$, we consider the expected contribution of C to v_a's profit given that v_a buys edges to all nodes in a set Δ, and denote this profit by $\hat{u}_{v_a}(C \mid \Delta)$. The details of $\hat{u}$ are defined in Section 3.3.1.

3.5.1 PartnerSetSelect

For each component $C \in \mathcal{C}_\mathcal{I}$ we compute three candidate sets of players to buy edges to and finally select the candidate set that yields the highest profit contribution for the considered component C for player v_a. The three candidate sets for component C are obtained as follows:

Case 1: The player considers buying no additional edges into component C. In this case the resulting player set is empty.

Case 2: The player considers buying exactly one additional edge into component C. The resulting player set contains the immunized partner that maximizes the profit for this component.

Case 3: The player considers buying at least two edges. In this case an optimal set of at least two immunized partners is obtained using the algorithm MetaTreeSelect.

As all possible cases are covered, the most profitable set of those three candidate solutions for component C must be the optimal partner set for component C. This optimal partner set is returned. We refer to this subroutine as PartnerSetSelect.

The first two cases, buying either no or exactly one edge into component C are easily solved: if no edge is purchased by v_a, then the expected profit contribution is $\hat{u}_{v_a}(C \mid \emptyset)$. If exactly one edge is bought then the expected profit contribution is $\hat{u}_{v_a}(C \mid \{w\})$, where w is the vertex in C which maximizes v_a's expected profit for component C.

Case 3 is much more difficult to handle. It is the main point where we need to employ algorithmic techniques to avoid a combinatorial explosion. To ease the strategy selection, for each component $C \in \mathcal{C}_\mathcal{I}$ we create an auxiliary graph to identify sets of nodes which offer equivalent benefits with respect to connection. This graph is a bipartite tree which we call the Meta Tree of C. Fig. 2 shows a conversion of a graph component into its Meta Tree by merging adjacent nodes of the same type into regions and collapsing regions into blocks. So called Bridge Blocks (orange) of the Meta Tree represent targeted regions of C that would, if destroyed, decompose C into at least two components. If the adversary however chooses to attack a player in a so-called Candidate Block (blue or violet), C would remain connected. The construction procedure can be found in Section 3.5.2 and useful properties of the Meta Tree are proven in Section 3.5.3.

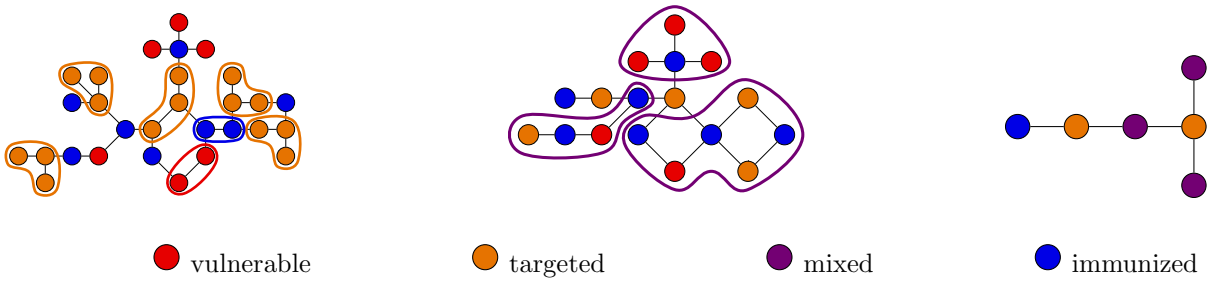

Figure 2: A graph component (left), the corresponding Meta Graph (middle), which is an intermediate step in the construction, and the obtained Meta Tree (right).

We now use the Meta Tree to identify the strategy maximizing the expected profit for v_a.

3.5.2 Meta Tree Construction

We now describe the algorithm MetaTreeConstruct for constructing the Meta Tree of component C.

To construct the Meta Tree, we first create the Meta Graph of C by merging maximally connected subcomponents of $C \cap \mathcal{U}$ and $C \cap \mathcal{I}$, respectively. Thus, the resulting undirected Meta Graph $G' = (V', E')$ is bipartite with $V' = \mathcal{R}_{\mathcal{U}}^C \cup \mathcal{R}_{\mathcal{I}}^C$, $\mathcal{R}_{\mathcal{U}}^C = \{R \in \mathcal{R}_{\mathcal{U}} \mid R \subseteq C\}$, $\mathcal{R}_{\mathcal{I}}^C = \{R \in \mathcal{R}_{\mathcal{I}} \mid R \subseteq C\}$ and $E' = \{\{R_{\mathcal{U}}, R_{\mathcal{I}}\} \in \mathcal{R}_{\mathcal{U}}^C \times \mathcal{R}_{\mathcal{I}}^C \mid \exists v_{\mathcal{U}} \in R_{\mathcal{U}}, v_{\mathcal{I}} \in R_{\mathcal{I}} : \{v_{\mathcal{U}}, v_{\mathcal{I}}\} \in E\}$.

To create the Meta Tree from the Meta Graph, we identify vertices in G' that disconnect the Meta Graph (and thus, the original graph) if the adversary attacks any player inside the region represented by these vertices. We name these vertices Bridge Blocks (BB). Regions that stay connected no matter which player the adversary targets are collapsed into so called Candidate Blocks (CB), to which the active player might buy edges. Those blocks are constructed iteratively:

(1) We create a new Candidate Block CB_x by identifying any immunized region $R_{\mathcal{I}} \in \mathcal{R}_{\mathcal{I}}^C$ which is not yet part of a Candidate Block: $CB_x = \{R_{\mathcal{I}}\}$ with $R_{\mathcal{I}} \in \mathcal{R}_{\mathcal{I}}^C$ and $R_{\mathcal{I}}$ is not yet part of a block.

(2) We add all immunized regions R that are reachable via two paths originating from any $R' \in CB_x$, which do not share a targeted region:

$$CB_x = CB_x \cup \{R \in \mathcal{R}_{\mathcal{I}}^C \mid \exists R' \in CB_x \; \exists P = R' \to R, \\ Q = R' \to R : (P \cap Q) \cap \mathcal{R}_{\mathcal{T}} = \emptyset\},$$

where $R' \to R$ denotes a simple path from vertex R' to vertex R. Note that the paths can be identical, thus two immunized regions that are connected through a vulnerable but not targeted region are collapsed as well.

(3) We add any nodes that only connect to nodes already in CB_x: $CB_x = CB_x \cup \{R_{\mathcal{U}} \in \mathcal{R}_{\mathcal{U}}^C \mid \forall \{R_{\mathcal{U}}, R\} \in E' : R \in CB_x\}$. This especially merges any non-targeted vulnerable regions into the respective adjacent Candidate Block.[3]

After we have constructed all Candidate Blocks, i.e. when all immunized nodes are part of a Candidate Block, any remaining nodes become Bridge Blocks BB_x.

Again, if an edge existed between two nodes that are now in separate blocks, these blocks are connected in the Meta Tree $M = (V^*, E^*)$ with

$$V^* = \{CB_1, ..., CB_i, BB_1, ..., BB_j\}$$

and

$$E^* = \{\{BB_x, CB_y\} \mid \exists \{R_{\mathcal{U}}, R_{\mathcal{I}}\} \in E' : R_{\mathcal{U}} \in BB_x \wedge R_{\mathcal{I}} \in CB_y\}.$$

Since the underlying Meta Graph is bipartite and no additional edges are introduced, the Meta Tree is also bipartite.

Lemma 3. *The Meta Tree M is indeed a tree.*

Proof. Our algorithm starts on a graph which is connected and only collapses nodes. This way no node can get disconnected as edges are only removed when the nodes they connected are merged. M is therefore connected.

We now use the fact that M is bipartite by construction to prove by contradiction that M contains no cycles. Assume there exists a cycle in the Meta Tree. As M is bipartite, the

[3]Note that there is always exactly one adjacent Candidate Block.

cycle must have an even length > 2, as we allow no multi-edges. Thus the cycle contains at least two Candidate Blocks $CB_1 \neq CB_2$. These two blocks contain two immunized regions $R_\mathcal{I} \in CB_1, R'_\mathcal{I} \in CB_2$. As they are arranged in a cycle, there exist two paths $P = R_\mathcal{I} \to R'_\mathcal{I}, Q = R_\mathcal{I} \to R'_\mathcal{I}$ that do not share a targeted region, thus according to the construction $R_\mathcal{I}$ and $R'_\mathcal{I}$ must be part of the same Candidate Block. This contradicts $CB_1 \neq CB_2$. □

Lemma 4. *After the construction of the Meta Tree all leaves are Candidate Blocks.*

Proof. Assume the contrary. Then there exists a Bridge Block leaf BB. Because the Meta Tree is a bipartite tree and the underlying Meta Graph contains at least one immunized region, the parent of this leaf has to be a Candidate Block $CB \neq BB$. Then BB is only connected to exactly one Candidate Block and must by construction be merged into CB. Then however BB would not be a leaf which is a contradiction, thus all leaves are Candidate Blocks. □

3.5.3 Key Properties of Using the Meta Tree

Lemma 5. *Player v_a has an optimal partner set for $C \in \mathcal{C}_\mathcal{I}$ which only buys edges to immunized players.*

Proof. Consider a best response for component $C \in \mathcal{C}_\mathcal{I}$ for player v_a which contains buying an edge to some vulnerable node $v_\mathcal{U} \in C$. Since $C \in \mathcal{C}_\mathcal{I}$, at least one immunized node $v_\mathcal{I}$ must be adjacent to $v_\mathcal{U}$'s vulnerable region $\mathcal{R}_\mathcal{U}(v_\mathcal{U})$.

On the one hand, if $\mathcal{R}_\mathcal{U}(v_\mathcal{U})$ is attacked, then player v_a gets no benefit from buying the edge to $v_\mathcal{U}$. Buying an edge to $v_\mathcal{I}$ instead, would yield at least the same benefit. On the other hand, if any other component $C_w \in \mathcal{R}_\mathcal{T} \setminus \mathcal{R}_\mathcal{U}(v_a)$ is attacked, then the vertices $v_\mathcal{U}$ and $v_\mathcal{I}$ stay connected and thus an edge to $v_\mathcal{I}$ provides the same benefit as an edge to $v_\mathcal{U}$.

Hence, each edge to some vulnerable player $v_\mathcal{U}$ can be exchanged with an edge to an immunized player $v_\mathcal{I}$ that is adjacent to $\mathcal{R}_\mathcal{U}(v_\mathcal{U})$ without decreasing the expected profit. □

We show that in case of buying at least two edges into $C \in \mathcal{C}_\mathcal{I}$, player v_a only has to consider leaves of the Meta Tree of C as endpoints. Remember that all vertices of the Meta Tree are either Candidate Blocks or Bridge Blocks and that by Lemma 4, all leaves of the Meta Tree are Candidate Blocks. Since Candidate Blocks may contain many nodes, we first have to clarify what it means to buy an edge to a Candidate Block. We show that if a player wants to buy an edge into a Candidate Block, then buying a single edge to an immunized node of the Candidate Block suffices.

Lemma 6. *It is never beneficial to buy more than one edge into a Candidate Block.*

Proof. By Lemma 5, it suffices to buy edges to immunized nodes of the Candidate Block. For any pair of immunized nodes $v_\mathcal{I}, v'_\mathcal{I}$ in the Candidate Block, one of the following statements, directly resulting from properties of the Meta Tree, will always be true:

1. There exists a direct edge or a path only containing immunized nodes between $v_\mathcal{I}$ and $v'_\mathcal{I}$.
2. For any targeted region R in the Candidate Block there exists a path connecting $v_\mathcal{I}$ and $v'_\mathcal{I}$ which does not contain R.

Thus, no matter which node will be attacked, each immunized node in the Candidate Block will always be connected to the whole Candidate Block except for the nodes that get killed by the attack. It follows, that buying at most one edge per Candidate Block suffices. □

Now we are ready to show that it is optimal to buy edges towards Candidate Blocks which are leaves of the Meta Tree.

Lemma 7. *Let M be the Meta Tree of component C. If player v_a has an optimal partner set for component C which contains buying at least two edges, then v_a also has an optimal partner set for component C which contains only leaves of M.*

Proof. By Lemma 5 w.l.o.g. v_a only buys edges to Candidate Blocks in M. Assume the best response strategy s_a^* of v_a connects to M with at least two edges with at least one of the endpoints not being a leaf of the Meta Tree. Let CB be such an inner block, root the Meta Tree at CB and inspect its subtrees.

If v_a bought edges to at least two different subtrees, one of those edges will be connected to CB no matter where the adversary attacks. Thus, by dropping the edge to CB the player does not reduce connectivity whilst decreasing her expenses, which is a contradiction to the assumption that the edge to CB is part of a best response.

If however v_a only bought edges to one of the subtrees of CB, v_a can exchange the edge to CB with an edge to a leaf CB' in another subtree Due to Lemma 4 this leaf is a Candidate Block as well. Again, independent of the attack target of the adversary, v_a is connected to CB through one of the subtrees she bought an edge to, hence the exchange does not decrease her utility.

Hence, each edge to some inner block CB can be exchanged with an edge to a leaf block CB' without decreasing the expected profit. □

3.5.4 Solving Case 3 of PartnerSetSelect

In the following let M be the Meta Tree of component C. Moreover, we assume that M has at least two Candidate Blocks, since otherwise, by Lemma 5 (see Section 3.5.3), buying at most one edge suffices.

Idea of the MetaTreeSelect Algorithm By Lemma 6 and Lemma 7 (see Section 3.5.3) we only have to consider to buy single edges into leaves of the Meta Tree which are Candidate Blocks. Thus, we only have to find the optimal combination of leaves of M to which to buy an edge.

Probing edge purchases to all possible combinations of leaves of M and comparing the expected profit contributions yields a correct solution but entails exponentially many computations. We reduce this exponential complexity by using the following observations to obtain the best possible combination of leaves. Both observations are based on the assumption that player v_a buys an edge to some leaf r of M and we consider the tree M rooted at r. Later we ensure this assumption by rooting M at each possible leaf.

Observation 1: Consider any vertex w of M. If player v_a has an edge to w, then it can be decided efficiently whether it is beneficial to buy exactly one or no edge into a subtree of w, as the influence of any additional edges into M does not propagate over w, effectively making the decisions to buy edges independent for subtrees.

Observation 2: Let w be any vertex of M and let the children of w in M be $x_1, \ldots, x_\ell$. Consider that v_a has an edge to w and it has already been decided for each subtree rooted at $x_1, \ldots, x_\ell$ whether or not to buy an edge into that subtree. If there exists at least one edge between v_a and any of those subtrees, then it cannot be beneficial to buy additional edges into the subtree rooted at w. Either w is destroyed, and the previous edge-buy decisions apply to the disconnected subtrees, or w survives, and v_a is connected to all subtrees via node w.

These observations provide us with the foundation for a dynamic programming algorithm which decides bottom-up whether it is beneficial to buy at most one edge into a given subtree by reusing the edge buy decisions of its subtrees.

Note that the algorithm never has to compare combinations of bought edges, as the only decision to make is, whether or not to buy exactly one edge into a subtree in combination with

iteratively shifting the presumed edge to the parent node of the leaves to the root r.

The MetaTreeSelect Algorithm A detailed description of the MetaTreeSelect algorithm can be found in Algorithm 3.

Algorithm 3: MetaTreeSelect(M)

Input: Meta Tree M for component C

Output: Player v_a's optimal partner set for C consisting of at least two partners

```
1 foreach leaf r of M do
2 |   M(r) ← root M at vertex r;
3 |   w ← r's only child in M(r);
4 |   opt(r) ← {some immunized node in r} ∪ RootedMetaTreeSelect(M(r), w);
5 best ← opt(r) which maximizes û_{v_a}(C | opt(r));
6 if |best| ≥ 2 then
7 |   return best;
8 else
9 |   return ∅;
```

The algorithm MetatreeSelect roots M at every leaf and assumes buying an edge towards some immunized node within the root Candidate Block. Then the subroutine RootedMetaTreeSelect (see Algorithm 4) gets the rooted Meta Tree $M(r)$ and some vertex r_T (which initially is the only child of the currently considered root leaf) as input and recursively computes the expected profit contribution of one additional edge from v_a to a block in the subtree T rooted at r_T under the assumption that v_a is already connected to the parent block $p(r_T)$ of r_T in $M(r)$.

Algorithm 4: RootedMetaTreeSelect($M(r), r_T$)

Input: rooted Meta Tree $M(r)$, vertex r_T of $M(r)$

Output: Set of nodes from T to buy an edge to

```
1  opt(r_T) ← ∅;
2  foreach child w of r_T do
3  |   opt(r_T) ← opt(r_T) ∪ RootedMetaTreeSelect(M(r), w);
4  if r_T is a Bridge Block or opt(r_T) ≠ ∅ or a player in T bought an edge to v_a
     then
5  |   return opt(r_T);
6  foreach leaf l of T do
7  |   profit(l) ← additional profit of v_a with edge to l;
8  best ← l which maximizes profit(l);
9  if profit(best) > α then
10 |   opt(r_T) ← opt(r_T) ∪ {some immunized node in best};
11 return opt(r_T);
```

Let $|T|$ denote the number of players represented by the union of all blocks in T and $opt(r_T)$ will be the set of blocks in T the algorithm decided to buy an edge to.
After processing all subtrees of r_T (Alg. 4 lines 2-3), the algorithm distinguishes three cases (Alg. 4 line 4):

Case 1: r_T is a Bridge Block. Then, as M is bipartite, $p(r_T)$ must be a Candidate Block. As the algorithm assumes the existence of an edge from v_a to $p(r_T)$, there also exists a path from v_a to r_T via $p(r_T)$ in all attack scenarios. Thus, no additional edge is needed (Alg.4 line 5).

Case 2: There exists an edge between v_a and some node x in T, either through an edge v_a buys according to the results of the recursive invocations, or through a preexisting edge bought by player x. Then, depending on the attack target, there either exists a path from v_a to r_T via x or via $p(r_T)$. Hence, no additional edge is needed (Alg.4 line 5).

Case 3: Player v_a can get disconnected from r_T by an attack on $p(r_T)$. Then the algorithm considers each leaf l of T as possible partner (Alg.4 line 6), computes the profit contribution of an edge to l (Alg.4 line 7) and selects a leaf that maximizes this profit contribution (Alg.4 line 8).

The additional profit of an edge to l is computed as follows: An edge to l only yields profit, if a Bridge Block t is attacked which either belongs to T or $t = p(r_T)$, and l is located in a subtree of t. In this case, the profit contribution equals the size of this subtree. Therefore let $\text{profit}(l \mid t)$ be the additional profit an edge to l contributes to the utility of v_a in case t is attacked and let $\mathcal{B}$ be the set of all Bridge Blocks in T. Thus

$$\text{profit}(l) = \frac{|p(r_T)|}{|\mathcal{T}|}|T| + \sum_{t \in \mathcal{B}} \frac{|t|}{|\mathcal{T}|}\text{profit}(l \mid t),$$

with

$$\text{profit}(l \mid t) = \begin{cases} 0 & l \text{ is not in any subtree of } t \\ |Y| & Y \text{ is a subtree of } t \text{ and } l \text{ is in } Y. \end{cases}$$

Finally, If the additional profit of the best possible leaf exceeds the edge costs, l is added to the set of partners of v_a (Alg.4 line 10).

3.5.5 Correctness of MetaTreeSelect and RootedMetaTreeSelect

An important detail for proving the correctness of MetaTreeSelect is the definition of $profit(l)$ from Case 3 of RootedMetaTreeSelect. There, the existence of an edge to $p(r_T)$ is assumed and the expected profit is computed by considering both the case where $p(r_T)$ is destroyed and the case where some other node is attacked. Thus, while considering an assumed edge to some parent node, it is already taken into account that this node may be attacked.

Also note that if an edge to some node of the Meta Tree is bought then this means that an edge to some immunized node of the corresponding Candidate Block is bought.

The algorithm MetaTreeSelect computes an optimal partner set for component C containing at least two nodes, if it exists. In this case, by Lemma 7, there is an optimal partner set for C containing only nodes within leaves of M. Assume that the Meta Tree M of component C is rooted at such a leaf r. Since the algorithm MetaTreeSelect compares all possibilities to root M at a leaf we will assume in the following that some node r^* in block r is contained in an optimal partner set.

Fig. 3 shows the structure of the Meta Tree in the following proofs.

Lemma 8. *Consider any subtree T with height h of M and let a be its root. If* RootedMetaTreeSelect *has processed all subtrees $T_1, T_2, \ldots$ of a under the assumption of an edge from v_a to a and at least one of those subtrees contains a node which was added to the partner set, then it cannot be profitable to buy additional edges into the complete tree T under the assumption of the existence of an edge to $p(a)$.*

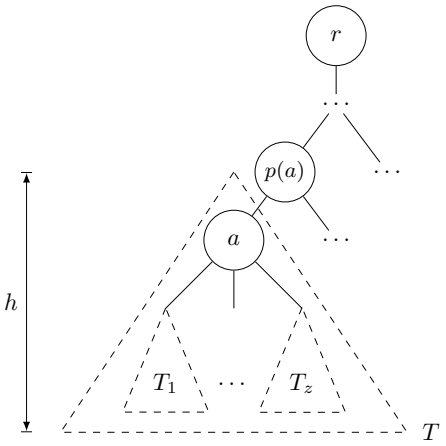

Figure 3: Inductive step of RootedMetaTreeSelect.

Proof. We prove the statement by induction over the height h. Assume there exists at least one edge between v_a and some child a' of a and assume the existence of an edge to $p(a)$, the parent of a. Thus, there exists a path from v_a to a via $p(a)$.

For $h = 1$ all children of a are leaves. The event of a being destroyed was already considered in a previous recursive call. If some other node is attacked, then all surviving nodes in T remain connected and v_a has a path via some node in a' to all of them.

Assume now, that the hypothesis holds for some arbitrary fixed height $h_0 \geq 1$. For $h = h_0+1$ we distinguish two different attack targets of the adversary.

If the attack takes place outside T, then T remains connected and analogous to above v_a is connected to all survivors via a'.

If the adversary attacks a player inside T, then v_a is connected to a via $p(a)$ which is equivalent to v_a having a direct edge to a and thus, by induction hypothesis, no additional edge should be bought into any subtree of a. □

The next statements establish that we can treat all subtrees of a node of M independently, that buying one edge into a subtree suffices if the edge-buy decision was negative for all subtrees one level below and that RootedMetaTreeSelect is correct.

Lemma 9. *Assume the existence of an edge from v_a to some node a of M and let $V_1, \ldots, V_z$ denote the node sets of a's subtrees.*

If there are optimal partner sets $O_1, \ldots, O_z$ for C which all contain r^*, then there is an optimal partner set O^* for C containing r^* where $O^* \cap V_j = O_j \cap V_j$ for all $1 \leq j \leq z$.

Proof. The choice of nodes to connect to within any subtree T of a does not influence v_a's connectivity to nodes outside of T as every path from some node in T to some node outside T traverses node a. Thus, the choice of nodes in T for all optimal partner sets for C is independent from the choice of nodes in other subtrees. □

Lemma 10. *Consider any subtree T with root a of M. If* RootedMetaTreeSelect *has correctly decided that no edge should be bought into any subtree of a, then every optimal partner set for component C which contains r^* has at most one node from within T.*

Proof. By induction over height h of T. For $h = 0$ the block a must be a leaf of the Meta Tree. By Lemma 4 and Lemma 6 the statement follows.

Assume now, that the hypothesis holds for some height $h_0 \geq 0$. For $h = h_0 + 1$ we assume the existence of an edge between v_a and $p(a)$ and distinguish two different attack targets of the adversary:

If the attack takes place outside of T, then T remains connected and thus at most one edge into T suffices.

If a node within T is attacked, then v_a is connected to a via $p(a)$ which is equivalent to v_a having a direct edge to a, which is the assumption under which RootedMetaTreeSelect has correctly decided not to buy edges into any subtree of a. □

Lemma 11. *If v_a has an edge to $p(r_T)$ and $opt(r_T)$ is returned by* RootedMetaTreeSelect$(M(r), r_T)$*, then there exists an optimal partner set for component C which contains r^* and $opt(r_T)$.*

Proof. RootedMetaTreeSelect recurses on all subtrees of r_T.

If r_T is a leaf of $M(r)$, then the algorithm computes the expected profit contribution of buying a single edge into the leaf and if this exceeds α an immunized node within this leaf is eventually added to $opt(r_T)$. Since, by Lemma 4, leaves are Candidate Blocks, buying at most one edge is correct and the algorithm compares both possible options. Otherwise, if r_T is not a leaf, then there are three cases:

Case 1: If r_T is a Bridge Block, then $p(r_T)$ must be a Candidate Block and by assumption v_a is connected to r_T via $p(r_T)$. Thus, by Lemma 9, there is an optimal partner set for C which contains r^* and the union of all the returned node sets obtained by recursing on all children of r_T.

Case 2: If there exists an edge between v_a and any node in the subtree rooted at r_T. Then, by Lemma 8 and Lemma 9, there is an optimal partner set for C which contains r^* and the union of all the returned node sets obtained by recursing on all children of r_T.

Case 3: If $opt(r_T) = \emptyset$ after all children of r_T were processed. Then, by Lemma 10, it is optimal to consider buying at most one edge into the subtree rooted at r_T. RootedMetaTreeSelect tries all possibilities and selects the most profitable one. □

Finally, we show that MetaTreeSelect is correct.

Theorem 2. *If there is an optimal partner set with at least two nodes for component C, then* MetaTreeSelect *algorithm outputs such a set.*

Proof. Assume that there exists an optimal partner set with at least two nodes for component C and assume that the Meta Tree M of component C is rooted at some leaf r. Since the algorithm compares all possibilities to root M at a leaf and by Lemma 7, at least one of those leaves must be contained in an optimal partner set. Assume that r is indeed such a leaf.

Thus, by buying r we satisfy the assumption needed for RootedMetaTreeSelect. By Lemma 11, RootedMetaTreeSelect returns a set of nodes, which together with r^* yields an optimal partner set for C. Hence, the algorithm MetaTreeSelect is correct. □

3.6 Cost Analysis

We now prove that our algorithm is efficient.

Theorem 3. BestResponseComputation *runs in time $\mathcal{O}(n^4 + k^5)$, where n is the number of nodes in the network and k is the number of blocks in the largest occurring Meta Tree. Only in terms of n, the run time is in $\mathcal{O}(n^5)$.*

of Theorem 3. We start with analyzing the five core subroutines of our algorithm: SubsetSelect, GreedySelect, PartnerSetSelect, MetaTreeConstruct and MetaTreeSelect.

SubsetSelect: This subroutine calculates a three dimensional matrix where every dimension has a maximum length of n. The calculations needed for one matrix entry can be done in constant time. This results in a time complexity of $\mathcal{O}(n^3)$.

GreedySelect: Every single component of $\mathcal{C}_\mathcal{U}$ is tested individually if an edge into it is beneficial. For the calculation of the expected profit contribution only the size of the component is needed. Thus, this task can be done for all components in $\mathcal{O}(n)$.

PartnerSetSelect: This subroutine is executed for every component $C \in \mathcal{C}_\mathcal{I}$. Let p denote the number of nodes of any processed component $C \in \mathcal{C}_\mathcal{I}$ and let q denote its number of edges. The algorithm covers three cases:

Case 1: No edge is bought into C: Here an attack is simulated on every targeted node y in component C and the remaining profit is calculated. Calculating the profit after an attack on node y can be done in linear time $\mathcal{O}(p+q)$ using breadth first search (BFS). Since there are $\mathcal{O}(p)$ many targeted nodes in C the overall cost is in $\mathcal{O}(p(p+q))$. In the worst case this is in $\mathcal{O}(p^3)$.

Case 2: One edge is bought into C: The profit of every combination of buying an edge to node x in C and having node y in C attacked is calculated. This amounts to a total cost of $\mathcal{O}(p^2(p+q))$, which is in $\mathcal{O}(p^4)$.

Case 3: Two or more edges are bought into C: In this case MetaTreeConstruct and MetaTreeSelect are executed.

- MetaTreeConstruct: To construct the Meta Tree we first create the Meta Graph of C. To do so we use BFS from an uncollapsed node to find all adjacent nodes of the same types. All of these nodes are then merged and disregarded in further iterations of BFS. There may be $\mathcal{O}(p)$ calls to BFS and (assuming a representation as adjacency list) merging neighboring nodes of the same type can be done in $\mathcal{O}(p+q)$. Thus the construction of the Meta Graph happens in $\mathcal{O}(p(p+q))$ which is in $\mathcal{O}(p^3)$.

  The construction of the Meta Tree consists of two steps: Finding cycles to collapse and collapsing the nodes on the cycle. The first part can be done using BFS and in every iteration of BFS at least one node is collapsed. Thus, the overall cost for all BFS computations is in $\mathcal{O}(p(p+q))$. The second part can be done by checking the edges of all neighboring nodes. This can be done in $\mathcal{O}(p+q)$ time per merge. Hence we have a total cost of $\mathcal{O}(p(p+q))$ which is in $\mathcal{O}(p^3)$ for the construction of the Meta Tree.

- MetaTreeSelect: Let k denote the number of nodes of the created Meta Tree. The MetaTreeSelect algorithm calculates the expected profit contribution for connecting with the currently considered component via two or more edges. It roots the Meta Tree at every leaf, and then traverses it bottom up. During this traversal the following calculation is made at every node: The number of reachable nodes from a node x given an attack on node y is evaluated for every possible node pair x, y of the Meta Tree. As seen above this cannot exceed a run time of $\mathcal{O}(k^3)$. As this is done for every node for every possible root, the total cost is in $\mathcal{O}(k^5)$.

Thus Case 3 has total cost in $\mathcal{O}(p^3 + k^5)$. It follows that the total cost of PartnerSetSelect can be upper bounded by $\mathcal{O}(p^4 + k^5)$.

PossibleStrategy: First, an arbitrary node for every previously selected vulnerable component is chosen. This can be done in constant time per component. Adding the corresponding edges and updating immunized, vulnerable and targeted regions can be done via at most n BFS computations and thus in $\mathcal{O}(n^3)$. Next, PartnerSetSelect is called for each of the c components in $\mathcal{C}_\mathcal{I}$. Let p_j and k_j denote the number of nodes and the size of the Meta Tree for component $C_j \in CI$ for $1 \leq j \leq c$. Therefore, the cost of PossibleStrategy can be upper bounded by $\mathcal{O}(n^3 + \sum_{1 \leq j \leq c}(p_j^4 + k_j^5))$ which is in $\mathcal{O}(n^4 + k^5)$.

BestResponseComputation: The total cost for BestResponseComputation thus is in $\mathcal{O}(n^3 + n + n^3 + n^4 + k^5) = \mathcal{O}(n^4 + k^5)$. Since $k < n$ this yields cost in $\mathcal{O}(n^5)$. □

3.7 Empirical Results

To evaluate our proposed best response algorithm, we apply it to randomly generated networks. To allow a direct comparison with the extensive experiments by Goyal et al. [13, 14], we chose the same setup, i.e. the initial networks were generated via the Erdős-Renyi model with average degree 5 and $\alpha = \beta = 2$.

In Fig. 4 (left) we show the number of rounds required until the best response dynamic arrives at a Nash equilibrium averaged over 100 experiments per configuration. Here a round

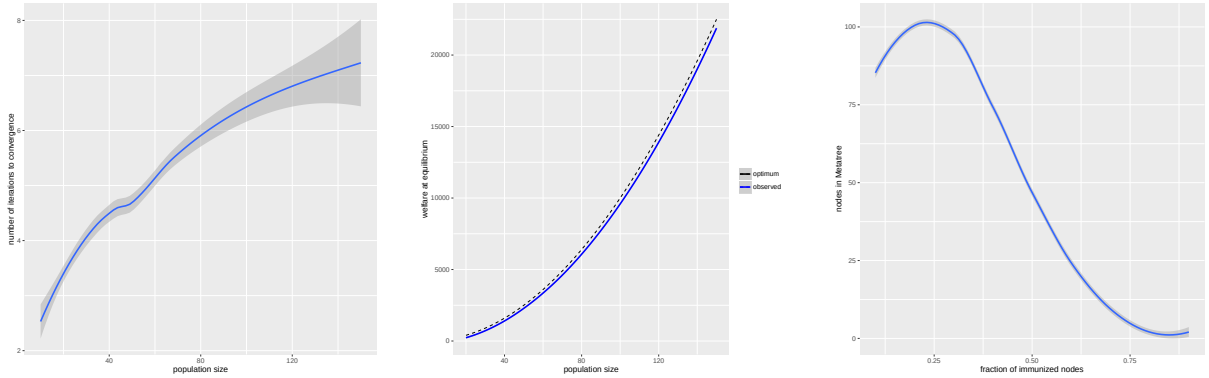

Figure 4: Number of iterations until convergence (top); average welfare at non-trivial equilibria (middle) $\alpha = \beta = 2$. Bottom: Meta Tree size with respect to immunization density.

consists of a best response strategy update by every player in some fixed order. In contrast with the results in [13, 14], where much weaker strategy updates called swapstable best response have been used, our experiments indicate a speed-up of 50% by updating to best possible strategies.

Fig. 4 (middle) plots the welfare of networks at (non-trivial) equilibria over the population size achieved by performing best response dynamics with the initial setting as described above. For every configuration a random sample from the 100 independent experiments was chosen. We observe that the achieved global welfare is quite close to the optimal value of $n(n-\alpha)$ which indicates that best response dynamics yield favorable outcomes whenever they converge to a Nash equilibrium[4].

The run time for computing the partial best response strategy for attaching to connected components from $\mathcal{C}_{\mathcal{I}}$, as described in Section 3.5, heavily depends on the number of Candidate Blocks in the resulting Meta Tree for the component. To illustrate the benefits of using the Meta Tree as a preprocessing step, we compared the number of nodes of a random connected network and the number of Candidate Blocks in the corresponding Meta Tree, varying the fraction of immunized players in the initial network. For each experiment we used connected $G_{n,m}$ random networks with $n = 1000$ and $m = 2n$ edges.

The results are depicted in Fig. 4 (right). There the number of Candidate Blocks of the Meta Tree over the fraction of immunized players averaged over 100 runs per parameter combination is shown. As expected, the number of Candidate Blocks in the Meta Tree shrinks rapidly as the fraction of immunized vertices in the initial network increases. We also note that the maximum number of Candidate Blocks in the Meta Tree is roughly 10% of the number of nodes in the initial network.

We illustrate the behavior of best response dynamics with Fig. 5, where snapshots from a sample run are shown. The initial network is sparsely connected with $n/2 = 25$ edges and contains no immunized players. During the first round, a player with a sufficient number of

[4]Note that this property is not guaranteed, since Goyal et al. [14] present a best response cycle, that is, a configuration where a sequence of best response strategy updates returns to the initial network and may thus never converge.

incident edges bought by other players decides to obtain immunization. Then all following players choose to connect to this newly immunized hub. The subsequent rounds distribute players away from the newly formed targeted regions until an equilibrium is achieved after four rounds.

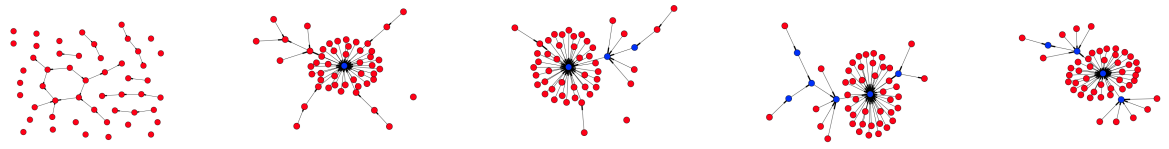

Figure 5: A sample run of the best response dynamic ($n = 50, \alpha = \beta = 2$). Initial state. After round 1. After round 2. After round 3. After round 4.

4 Random Attack Adversary

In the following we will consider the random attack adversary [14] which is a less predictable adversary compared to the maximum carnage adversary.

The Modified Model and its Implications The random attack adversary attacks one vulnerable node uniformly at random, i.e. with probability $\frac{1}{|\mathcal{U}|}$ per player in $\mathcal{U}$. Facing this adversary, the probability for a vulnerable player $v_{\mathcal{U}}$ to be destroyed increases linearly with the size of player $v_{\mathcal{U}}$'s vulnerable region $\mathcal{R}_{\mathcal{U}}(v_{\mathcal{U}})$. Moreover, every vulnerable player belongs to the set of targeted vertices, that is, $\mathcal{T} = \mathcal{U}$.

Clearly, the behavior of the chosen adversary influences the utility of a player's strategy heavily. Thus, it is not at all clear that our algorithm for computing a utility maximizing strategy for the maximum carnage adversary can be adapted to the random attack adversary. We now show that this can actually be done with only some minor adjustments. Thereby we increase the run time slightly, but our algorithm is still efficient.

Adaptation of our Algorithm With the deterministic maximum carnage adversary, a player was either risk-free or potentially targeted with probability $\frac{1}{|\mathcal{T}|}$. Introducing the random attack adversary results in up to n different probabilities for the player v_a to be destroyed due to different possible sizes of targeted regions.

Note that none of the subroutines GreedySelect, MetaTreeConstruct and MetaTreeSelect depend on t_{max}, but are only formulated in terms of $\mathcal{T}$ and $\mathcal{R}_{\mathcal{T}}$. Even though these sets change in comparison to the maximum carnage adversary, correctness still holds for those parts of the algorithm. In particular, the Meta Tree still fulfills the same properties as before, even though the number of Bridge Blocks increases for many input graphs, as all vulnerable regions are targeted regions now (see Fig. 6).

Thus, only the subroutine SubsetSelect has to be adjusted.

SubsetSelect for the Random Attack Adversary With the random attack adversary the maximum expected profit generated by all possible sizes of the vulnerable region of v_a need to be considered. A larger region might lower the overall profit as it increases the attack probability on the vulnerable region of v_a. Instead of two sets of selected components $\mathcal{A}_v, \mathcal{A}_t$, we now need to consider up to n sets that maximize the profit for each possible size of $\mathcal{R}_{\mathcal{U}}$.

As before, the interdependent subset selection on $\mathcal{C}_{\mathcal{U}} \setminus \mathcal{C}_{inc}$ with $|\mathcal{C}_{\mathcal{U}} \setminus \mathcal{C}_{inc}| = m$ fills the matrix M. In the end, the plane $M[m, \cdot, \cdot]$ contains all possible numbers of players from these components that v_a can buy edges to. These values constitute all possible sizes of $\mathcal{R}_{\mathcal{U}}(v)$. There

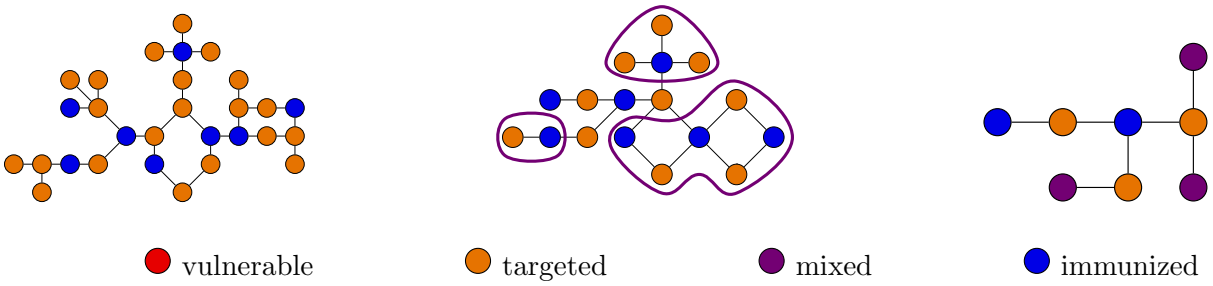

Figure 6: A graph component (left) and its Meta Graph (middle) and Meta Tree (right) for the random attack adversary.

might be different subsets of $\mathcal{C}_\mathcal{U} \setminus \mathcal{C}_{inc}$ that result in the same size of the active player's vulnerable region. Observe that the maximum utility is always achieved with the subset that uses the least amount of edges.

So for every row $M[m, \cdot, z]$ the algorithm selects the minimum y so that $M[m, y, z] = z$. This can be done in $\mathcal{O}(n^2)$ and yields a maximum of n values. We call this modified version UniformSubsetSelect.

BestResponseComputation for the Random Attack Adversary The modified main algorithm can be found in Algorithm 5.

The only change compared to the maximum carnage adversary is that for each of the $\mathcal{O}(n)$ solutions of UniformSubsetSelect the subroutine PossibleStrategy is executed. This yields an overall run time of $\mathcal{O}(n^3 + n(n^4 + k^5)) = \mathcal{O}(n^5 + nk^5)$, which is in $\mathcal{O}(n^6)$.

Observe that we calculate best responses for all possible sizes of $\mathcal{R}_\mathcal{U}(v_a)$, and $\mathcal{T}$ remains unchanged across all cases.

Algorithm 5: BestResponseComputation for the random attack adversary.

Input: Strategies $\mathbf{s} = (s_1, \ldots, s_n)$, active player $v_a, 1 \leq a \leq n$

Output: Best response strategy of player v_a denoted by $s_a = (x_a, y_a)$

$s_\emptyset = (\emptyset, 0)$;

Let $G(\mathbf{s}')$ be the induced game state with $\mathbf{s}' = (s_1, \ldots, s_{a-1}, s_\emptyset, s_{a+1}, \ldots, s_n)$;

Let $\mathfrak{A}$ be the solutions of UniformSubsetSelect on $\mathcal{C}_\mathcal{U}$;

Let $\mathcal{A}_g$ be the solution of GreedySelect on $\mathcal{C}_\mathcal{U}$;

$S_v = \{\text{PossibleStrategy}(\mathcal{A}, 0) \mid \mathcal{A} \in \mathfrak{A}\}$;

$s_g = \text{PossibleStrategy}(\mathcal{A}_g, 1)$;

$S = \{s_\emptyset, s_g\} \cup S_v$;

return strategy $s \in S$ which maximizes v_a's utility;

5 Conclusion

For most models of strategic network formation computing a utility maximizing strategy is known to be NP-hard. In this paper, we have proven that the model by Goyal et al. [13, 14] is a notable exception to this rule. The presented efficient algorithm for computing a best response for a player circumvents a combinatorial explosion essentially by simplifying the given network and thereby making it amenable to a dynamic programming approach. An efficient best response computation is the key ingredient for using the model in large scale simulations and

for analyzing real world networks. Moreover, our algorithm can be adapted to a significantly stronger adversary and we are confident that further modifications for coping with other variants of the model are possible.

Future Work Settling the complexity of computing a best response strategy with respect to the maximum disruption adversary is left as an open problem. Besides this, it seems worthwhile to consider a variant with directed edges, originally introduced by Bala & Goyal [2]. Directed edges would more accurately model the differences in risk and benefit which depend on the flow direction. Using the analogy of the WWW, a user who downloads information benefits from it, but also risks getting infected. In contrast, the user providing the information is exposed to little or no risk. Moreover, a constant cost for immunization seems unrealistic. In reality a highly connected node would have to invest much more into security measures than any node with only a few connections. Thus, it would be interesting to consider a version where immunization costs scale with the degree of a node. We believe that this variant of the model yields more diverse optimal networks and also a greater variety of equilibria.

References

[1] J. Aspnes, K. Chang, and A. Yampolskiy. Inoculation strategies for victims of viruses and the sum-of-squares partition problem. *Journal of Computer and System Sciences*, 72(6):1077–1093, 2006.

[2] V. Bala and S. Goyal. A Noncooperative Model of Network Formation. *Econometrica*, 68(5):1181–1229, 2000.

[3] V. Bala and S. Goyal. A strategic analysis of network reliability. In *Networks and Groups*, pages 313–336. Springer, 2003.

[4] D. Bilò, L. Gualà, S. Leucci, and G. Proietti. Locality-based network creation games. In *SPAA'14*, pages 277–286, 2014.

[5] D. Bilò, L. Gualà, and G. Proietti. Bounded-distance network creation games. *ACM TEAC*, 3(3):16:1–16:20, 2015.

[6] A. Chauhan, P. Lenzner, A. Melnichenko, and M. Münn. On selfish creation of robust networks. In *SAGT'16*, pages 141–152, 2016.

[7] P.-A. Chen, M. David, and D. Kempe. Better vaccination strategies for better people. In *EC'10*, pages 179–188. ACM, 2010.

[8] A. Cord-Landwehr and P. Lenzner. Network creation games: Think global - act local. In *MFCS'15*, pages 248–260, 2015.

[9] R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

[10] S. Ehsani, S. S. Fadaee, M. Fazli, A. Mehrabian, S. S. Sadeghabad, M. A. Safari, and M. Saghafian. A bounded budget network creation game. *ACM Transactions on Algorithms*, 11(4):34, 2015.

[11] A. Fabrikant, A. Luthra, E. N. Maneva, C. H. Papadimitriou, and S. Shenker. On a network creation game. In *PODC'03*, pages 347–351.

[12] T. Friedrich, S. Ihde, C. Keßler, P. Lenzner, S. Neubert, and D. Schumann. Brief announcement: Efficient best response computation for strategic network formation under attack. In *SPAA'17*, to appear.

[13] S. Goyal, S. Jabbari, M. Kearns, S. Khanna, and J. Morgenstern. Strategic Network Formation with Attack and Immunization. *arXiv preprint arXiv:1511.05196*, 2015.

[14] S. Goyal, S. Jabbari, M. Kearns, S. Khanna, and J. Morgenstern. Strategic network formation with attack and immunization. In *WINE'16*, pages 429–443, 2016.

[15] M. O. Jackson and A. Wolinsky. A strategic model of social and economic networks. *Journal of economic theory*, 71(1):44–74, 1996.

[16] B. Kawald and P. Lenzner. On dynamics in selfish network creation. In *SPAA'13*, pages 83–92. ACM, 2013.

[17] L. Kliemann. The price of anarchy for network formation in an adversary model. *Games*, 2(3):302–332, 2011.

[18] V. A. Kumar, R. Rajaraman, Z. Sun, and R. Sundaram. Existence theorems and approximation algorithms for generalized network security games. In *ICDCS'10*, pages 348–357. IEEE, 2010.

[19] P. Lenzner. Greedy selfish network creation. In *WINE'12*, pages 142–155.

[20] E. A. Meirom, S. Mannor, and A. Orda. Formation games of reliable networks. In *INFOCOM'15*, pages 1760–1768, 2015.

[21] M. Mihalák and J. C. Schlegel. The price of anarchy in network creation games is (mostly) constant. In *SAGT'10*, pages 276–287, 2010.

[22] C. H. Papadimitriou. Algorithms, games, and the internet. In *STOC'01*, pages 749–753, 2001.

[23] S. Saha, A. Adiga, and A. K. S. Vullikanti. Equilibria in epidemic containment games. In *AAAI*, pages 777–783, 2014.