Efficient Best Response Computation for Strategic Network Formation Under Attack

Tobias Friedrich, Sven Ihde, Christoph Keßler, Pascal Lenzner(✉), Stefan Neubert, and David Schumann

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

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. [14] 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 through the network via neighboring nodes.

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.

V. Bilò and M. Flammini (Eds.): SAGT 2017, LNCS 10504, pp. 199–211, 2017.
DOI: 10.1007/978-3-319-66700-3_16

The field of strategic network formation, started by the seminal works of Jackson and Wolinsky [15], Bala and 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.

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.

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 and 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.

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, that the amount of edge overbuilding due to robustness concerns is small and that equilibrium networks generally achieve very high social welfare. Besides this, 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 [16,19]. 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,

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

it has been shown that the problem is indeed NP-hard. The model by Goyal et al. [13,14] seems on the first glance computationally easier than the above mentioned centrality models since players only strive for reaching all other players. However, the presence of a strong adversary and the possibility of immunization renders finding a best possible strategy a non-trivial problem.

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 and Goyal [3] and Kliemann [17], both essentially augment the model by Bala and 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 of computing a best response was only settled for the model by Chauhan et al. [6] where it was proven to be NP-hard.

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

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.

Due to space constraints, we refer to [12] for all omitted details.

2 Model

We consider the 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.

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. 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 immunize. The strategy profile $\mathbf{s} = (s_1, \ldots, s_n)$ of all players then induces an undirected graph $G(\mathbf{s}) = \big(V, \bigcup_{v_i \in V} \bigcup_{v_j \in x_i} \{v_i, v_j\}\big)$. 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 is $\mathcal{R}_\mathcal{U}$. The vulnerable region of any 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 player $v_i \in \mathcal{U}$ is attacked, then v_i will be destroyed and the attack spreads to all vulnerable neighbors of v_i, eventually destroying all players in $\mathcal{R}_\mathcal{U}(v_i)$. Let $t_{max} = \max_{R \in \mathcal{R}_\mathcal{U}} \{|R|\}$ be the number of nodes in the vulnerable region of maximum size and $\mathcal{T} = \{v_i \in \mathcal{U} \mid |\mathcal{R}_\mathcal{U}(v_i)| = t_{max}\}$ be the corresponding set of nodes which may be targeted. 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 $\mathcal{R}_\mathcal{T}(v_i)$ 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 $\mathcal{C}_\mathcal{U} = \{C \in \mathcal{C} \mid C \cap \mathcal{I} = \emptyset\}$, $\mathcal{C}_\mathcal{I} = \mathcal{C} \setminus \mathcal{C}_\mathcal{U}$ and $\mathcal{C}_{inc} = \{C \in \mathcal{C} \mid \exists u \in C : \{u, v\} \in E\}$, 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.

3.2 Main Algorithm

Our algorithm, called BestResponseComputation, is described in Algorithm 1 and a schematic overview can be found in Fig. 1.

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 $s_a = (x_a, y_a)$

1 $s_\emptyset = (\emptyset, 0)$;
2 Let $G(\mathbf{s}')$ be the induced game state with $\mathbf{s}' = (s_1, \ldots, s_{a-1}, s_\emptyset, s_{a+1}, \ldots, s_n)$;
3 Let $\mathcal{A}_t, \mathcal{A}_v$ be the solutions of SubsetSelect on $\mathcal{C}_\mathcal{U}$;
4 Let $\mathcal{A}_g$ be the solution of GreedySelect on $\mathcal{C}_\mathcal{U}$;
5 s_t = PossibleStrategy($\mathcal{A}_t$, 0);
6 s_v = PossibleStrategy($\mathcal{A}_v$, 0);
7 s_g = PossibleStrategy($\mathcal{A}_g$, 1);
8 $S = \{s_\emptyset, s_t, s_v, s_g\}$;
9 **return** strategy $s \in S$ which maximizes v_a's utility;

Algorithm 2: PossibleStrategy

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

1 $M := \emptyset$;
2 **foreach** $C \in \mathcal{A}$ **do**
3 $\quad M = M \cup v_i$ for an arbitrary node $v_i \in C$;
4 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;
5 $B \leftarrow \emptyset$;
6 **foreach** *component* $C \in \mathcal{C}_\mathcal{I}$ **do**
7 $\quad B \leftarrow B \cup$ PartnerSetSelect(C)
8 **return** $(M \cup B, y_a)$;

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.

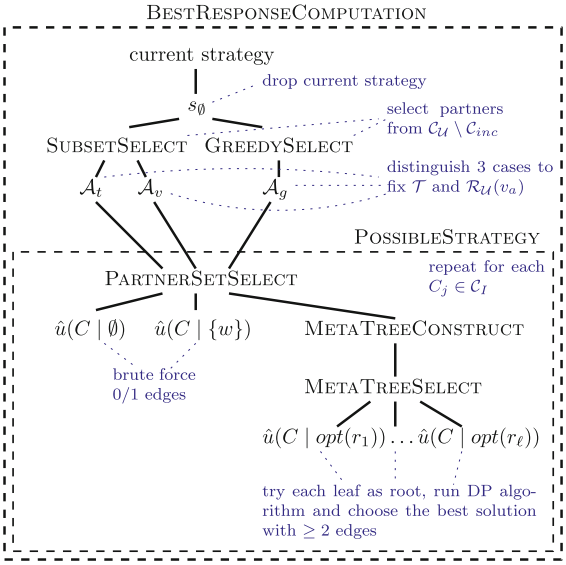

Fig. 1. Schematic overview of the best response algorithm.

The subroutine SubsetSelect 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 greedily computes a best possible subset of components of $\mathcal{C}_\mathcal{U}$ to connect with in case v_a buys immunization.

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, 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.

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. The correctness of this is guaranteed by the following lemma.

Lemma 1. *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.*

For each case, PossibleStrategy first chooses an arbitrary single edge to buy into the previously selected components from $\mathcal{C}_\mathcal{U}$. This is correct since we have:

Lemma 2. *Buying at most one edge into any component $C \in \mathcal{C}_\mathcal{U}$ yields maximum profit for player v_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, MetaTreeConstruct and MetaTreeSelect. 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. All in all we get:

Theorem 1. *The algorithm* BestResponseComputation *is correct and runs in polynomial time.*

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. 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 in [12] empirical results 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 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. The next statement ensures that we only need to consider buying edges to such nodes.

Lemma 3. *Player v_a has an optimal partner set for $C \in \mathcal{C}_\mathcal{I}$ which only buys edges to immunized players.*

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 $\hat{u}_{v_a}(C \mid \Delta)$.

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 C. In this case the resulting player set is empty.

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

Case 3: The player considers buying at least two edges. An optimal set of at least two immunized partners is obtained via the algorithm MetaTreeSelect.

As all possible cases are covered, the most profitable set of those three candidate solutions 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. Figure 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. Details of the Meta Tree are discussed in [12]. An important property is guaranteed by the following lemma.

Lemma 4. *All leaves of the Meta Tree are Candidate Blocks.*

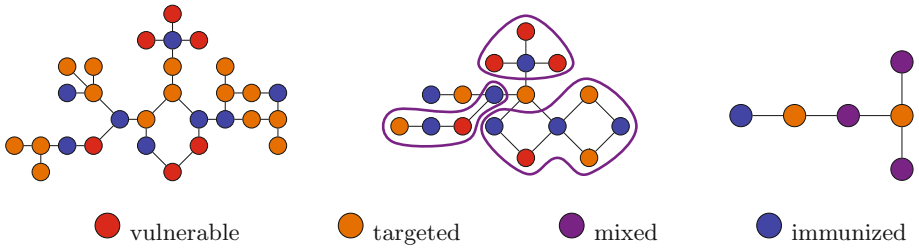

Fig. 2. A graph component (left), the corresponding Meta Graph (middle), which is an intermediate step in the construction, and the obtained Meta Tree (right). (Color figure online)

We now use the Meta Tree for maximizing the expected component profit for v_a.

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 3 buying at most one edge suffices.

Idea of the MetaTreeSelect *Algorithm.* The following two lemmas imply that 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.

Lemma 5. *Buying more than 1 edge to a Candidate Block is never beneficial.*

Lemma 6. *Let M be the Meta Tree of component C. If player v_a has an optimal partner set for C which contains buying at least two edges, then v_a also has an optimal partner set for C which contains only leaves of M.*

Probing all possible combinations of leaves of M yields exponential runtime. We use the following two observations to compute the best possible combination of leaves efficiently. 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. Let w be any vertex of M.

Observation 1: 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. Hence decisions are independent for subtrees.

Observation 2: 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*: The MetaTreeSelect algorithm can be found in Algorithm 3. It 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 $M(r)$. 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.

Algorithm 3: MetaTreeSelect

Input: Meta Tree M for component C
Output: Player v_a's optimal partner set for C consisting of at least two partners

foreach *leaf* r *of* M **do**
  $M(r) \leftarrow$ root M at vertex r;
  $w \leftarrow r$'s only child in $M(r)$;
  $opt(r) \leftarrow$ {some immunized node in r} $\cup$ RootedMetaTreeSelect$(M(r), w)$;
$best \leftarrow opt(r)$ which maximizes $\hat{u}_{v_a}(C \mid opt(r))$;
if $|best| \geq 2$ **then**
  return $best$;
else
  return $\emptyset$;

Algorithm 4: RootedMetaTreeSelect

Input: rooted Meta Tree $M(r)$, vertex r_T of $M(r)$
Output: Set of nodes from T to buy an edge to

$opt(r_T) \leftarrow \emptyset$;
foreach *child* w *of* r_T **do**
  $opt(r_T) \leftarrow opt(r_T) \cup$ RootedMetaTreeSelect$(M(r), w)$;
if r_T *is a Bridge Block* ***or*** $opt(r_T) \neq \emptyset$ ***or*** *a player in* T *bought an edge to* v_a **then**
  return $opt(r_T)$;
foreach *leaf* l *of* T **do**
  profit$(l) \leftarrow$ additional profit of v_a with edge to l;
$best \leftarrow l$ which maximizes profit(l);
if profit$(best) > \alpha$ **then**
  $opt(r_T) \leftarrow opt(r_T) \cup$ {some immunized node in $best$};
return $opt(r_T)$;

After processing all subtrees of r_T (Algorithm 4 lines 2–3), the algorithm distinguishes three cases (Algorithm 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 (Algorithm 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 (Algorithm 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 (Algorithm 4 line 6), computes the profit contribution of an edge to l (Algorithm 4 line 7) and selects a leaf that maximizes this profit contribution (Algorithm 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 $\text{profit}(l) = \frac{|p(r_T)|}{|T|}|T| + \sum_{t \in \mathcal{B}} \frac{|t|}{|T|}\text{profit}(l \mid t)$, with

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

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 (Algorithm 4 line 10).

The correctness of MetaTreeSelect is based in the following statement:

Lemma 7. *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 $opt(r_T)$.*

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 6, 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 7, RootedMetaTreeSelect returns a set of nodes, which together with r^* yields an optimal partner set for C. Hence, the algorithm MetaTreeSelect is correct. □

4 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 and 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.

References

1. Aspnes, J., Chang, K., Yampolskiy, A.: Inoculation strategies for victims of viruses and the sum-of-squares partition problem. J. Comput. Syst. Sci. **72**(6), 1077–1093 (2006)
2. Bala, V., Goyal, S.: A noncooperative model of network formation. Econometrica **68**(5), 1181–1229 (2000)
3. Bala, V., Goyal, S.: A strategic analysis of network reliability. Rev. Econ. Des. **5**(3), 205–228 (2000). doi:10.1007/s100580000019. ISSN 1434-4750
4. Bilò, D., Gualà, L., Leucci, S., Proietti, G.: Locality-based network creation games. In: SPAA 2014, pp. 277–286 (2014)
5. Bilò, D., Gualà, L., Proietti, G.: Bounded-distance network creation games. ACM TEAC **3**(3), 16:1–16:20 (2015)
6. Chauhan, A., Lenzner, P., Melnichenko, A., Münn, M.: On selfish creation of robust networks. In: Gairing, M., Savani, R. (eds.) SAGT 2016. LNCS, vol. 9928, pp. 141–152. Springer, Heidelberg (2016). doi:10.1007/978-3-662-53354-3_12
7. Chen, P.-A., David, M., Kempe, D.: Better vaccination strategies for better people. In: EC 2010, pp. 179–188. ACM (2010)
8. Cord-Landwehr, A., Lenzner, P.: Network creation games: think global - act local. In: MFCS 2015, pp. 248–260 (2015)
9. Downey, R.G., Fellows, M.R.: Fundamentals of Parameterized Complexity. Texts in Computer Science. Springer, Heidelberg (2013)
10. Ehsani, S., Fadaee, S.S., Fazli, M., Mehrabian, A., Sadeghabad, S.S., Safari, M.A., Saghafian, M.: A bounded budget network creation game. ACM Trans. Algorithms **11**(4), 34 (2015)

11. Fabrikant, A., Luthra, A., Maneva, E.N., Papadimitriou, C.H., Shenker, S.: On a network creation game. In: PODC 2003, pp. 347–351 (2003)
12. Friedrich, T., Ihde, S., Keßler, C., Lenzner, P., Neubert, S., Schumann, D.: Efficient best-response computation for strategic network formation under attack. CoRR, abs/1610.01861 (2016)
13. Goyal, S., Jabbari, S., Kearns, M., Khanna, S., Morgenstern, J.: Strategic Network Formation with Attack and Immunization. arXiv preprint arXiv:1511.05196 (2015)
14. Goyal, S., Jabbari, S., Kearns, M., Khanna, S., Morgenstern, J.: Strategic network formation with attack and immunization. In: Cai, Y., Vetta, A. (eds.) WINE 2016. LNCS, vol. 10123, pp. 429–443. Springer, Heidelberg (2016). doi:10.1007/978-3-662-54110-4_30
15. Jackson, M.O., Wolinsky, A.: A strategic model of social and economic networks. J. Econ. Theory **71**(1), 44–74 (1996)
16. Kawald, B., Lenzner, P.: On dynamics in selfish network creation. In: SPAA 2013, pp. 83–92. ACM (2013)
17. Kliemann, L.: The price of anarchy for network formation in an adversary model. Games **2**(3), 302–332 (2011)
18. Kumar, V.A., Rajaraman, R., Sun, Z., Sundaram, R.: Existence theorems and approximation algorithms for generalized network security games. In: ICDCS 2010, pp. 348–357. IEEE (2010)
19. Lenzner, P.: Greedy selfish network creation. In: WINE 2012, pp. 142–155 (2012)
20. Meirom, E.A., Mannor, S., Orda, A.: Formation games of reliable networks. In: INFOCOM 2015, pp. 1760–1768 (2015)
21. Mihalák, M., Schlegel, J.C.: The price of anarchy in network creation games is (mostly) constant. In: Kontogiannis, S., Koutsoupias, E., Spirakis, P.G. (eds.) SAGT 2010. LNCS, vol. 6386, pp. 276–287. Springer, Heidelberg (2010). doi:10.1007/978-3-642-16170-4_24
22. Papadimitriou, C.H.: Algorithms, games, and the internet. In: STOC 2001, pp. 749–753 (2001)
23. Saha, S., Adiga, A., Vullikanti, A.K.S.: Equilibria in epidemic containment games. In: AAAI, pp. 777–783 (2014)