Algorithms for Subgame Abstraction with Applications to Cyber Defense

Anjon Basak[1], Marcus Gutierrez[2], and Christopher Kiekintveld[3]

[1] University of Texas at El Paso, El Paso, TX, 79968, USA,
abasak@miners.utep.edu
[2] University of Texas at El Paso, El Paso, TX, 79968, USA,
mgutierrez22@miners.utep.edu
[3] University of Texas at El Paso, El Paso, TX, 79968, USA,
cdkiekintveld@utep.edu

Abstract. Normal Form Games (NFGs) are one of the most familiar representations for modeling interactions among multiple agents. However, modeling realistic interactions between agents often result in massive games, causing the computation of standard solutions like Nash equilibrium to be intractable. We propose an approach for solving large games that have a particular structure such that they can be (approximately) decomposed into strategically isolated subgames. We propose an abstraction approach that can exploit such structure in arbitrary NFGs and also present a cyber defense scenario in which this structure arises naturally. We show that our algorithms are much more scalable than standard methods, and achieve high-quality approximations even when the full game has only an approximate subgame structure.

Keywords: Game theory, abstraction, Nash Equilibrium, solution quality, cybersecurity

1 Introduction

A Normal Form Game (NFG) is one of the most common representations for modeling interactions between multiple decision makers (i.e., players). While an NFG is very general, it is often problematic to represent and solve an NFG for real-world scenarios because enumerating all possible strategies results in tremendous game model. Solving an NFG is known to be a computationally hard problem [6], and most existing algorithms (e.g., implemented in the Gambit software package [18]) do not scale well in practice.

To analyze games that are beyond the limits of standard solution algorithms, an increasingly common approach is to apply some form of automated abstraction to simplify the game. The simplified game is then analyzed using an available solver, and the solution is mapped back into the original game. If the reduced game can retain the vital strategic features of the original game, then in principle the solution of the simpler game may be a reasonable approximation of the solution to the original game. This general approach has been successful in developing computer poker agents, and most of the successful players in the annual competition in computer poker over the past years have used some variation of abstraction (e.g., [9, 10, 12, 13, 23]).

In current literature, most of the abstraction techniques are on extensive form games, a decision tree based structure where players move sequentially. And in most of the literature Poker is used to do the evaluations. Some abstraction techniques are specially focused on Poker [9] [10] [12] [13]. There is lossless abstraction technique [11] and also lossy abstraction technique [20]. There is also decision theoretic clustering [2] where the Bard et al. used utility based clustering technique for an NFG. Sandholm gave error bounds on the abstraction techniques [20] [15] because as we know, abstraction will discard information if it is lossy. Furthermore, an error in solution quality will arise after the reverse mapping of the strategies to the original game. Most recently Brown et al. used CFR [23] and imperfect recall abstraction with earth mover's distance [8] for a hierarchical abstraction [4] technique. Another widespread approach to handle massive games is Double Oracle (DO) Algorithm [3] [19] which relies on the concept of column/constraint generation techniques. In this paper we will focus on a recursive abstraction technique [5] by Conitzer et al. which is the motivation for our proposed algorithm.

Two works very closely related to an NFG reduction are one by Conitzer et al. [5] and another one by Bard et al. [2]. In the former paper, the authors gave an abstraction technique which can be used in a class of NFGs called *Any Lower Action Gives Identical Utility* (ALAGIU). The authors show that their technique can be applied recursively in ALAGIU games to abstract the game and find approximate Nash equilibrium. Motivated by the approach, we introduce an abstraction technique we call Iterative Subgame Abstraction and Solution Concept (ISASC). To evaluate this technique, we use a class of NFGs where some actions give identical utility that we call *Approximately Identical Outside Subgames* (AIOS). We also introduce a Pure Strategy Nash Equilibrium solution concept called *Minimum Epsilon Bound* (MEB). Finally, we evaluated ISASC and MEB by comparing against some other solution concepts by using NFGs of class AIOS.

Next, we present a motivating domain based on a cyberdefense scenario that naturally leads to games with this structure, and we also consider clustering methods that can be used to find this type of structure in arbitrary games. Based on this structure we present methods for abstracting an NFG, and algorithms for solving the abstracted game and mapping the reduced solution back to a solution in the original game.

Our main contributions are as follows: (1) we introduce the AIOS structure and present a realistic class of games with this structure, (2) we present fast methods for building and solving abstracted games based on clustering strategies, (3) we offer more sophisticated iterative algorithms for solving games using abstraction with both exact and approximate AIOS structure, (4) we present experimental evaluation of our algorithms on both generic games and games based on the cyber defense scenario, showing that our algorithms substantially improve scalability over baseline equilibrium solution algorithms.

2 Games with AIOS Structure

A *Normal Form Game* (NFG) is a standard representation in Game Theory in which the outcomes of all possible combinations of strategies are represented using a payoff

matrix. The tuple (N, A, u) represents a finite N-player NFG [21]. N is a finite set of players, indexed by i. The set of actions (pure strategies) is given by $A = A_1 \times ... \times A_n$, where A_i is the set of actions for player i. Each vector $a = (a_1, ..., a_n) \in A$ is an action profile. $a_{i,k}$ is the kth action for player i. We extend to mixed strategies $s_i \in S_i$, and use the notation $\pi^i(a_i)$ to refer to the probability of playing action a_i for player i. Each player has a real-valued utility (payoff) function $u = (u_1, ..., u_n)$ where $u_i : A \mapsto R$, extended to mixed strategies as usual by using expected utility.

We will consider abstracted games represented as (simpler) NFGs. For these games, we use the same notation but with a hat to denote that it is an abstracted game, $(\hat{N}, \hat{A}, \hat{u})$. We also use $A_i(O)$ to refer to the set of available actions for player i in NFG O.

We now introduce a game structure based on the idea of forming subgames with strong interactions within a subgame, but weak interactions outside of the subgame. This structure is similar to ALAGIU games [5], but extended to multiple subgames. We call this *Approximately Identical Outside Subgames* (AIOS), as shown in Figure 1. The fundamental idea is to create clusters of strategies for both players that form subgames. Within a subgame, the strategies and payoffs can vary arbitrarily. However, outside of the subgame, the strategies for each player should have payoffs as similar as possible for playing against any opponent strategy, not in the subgame. Games with exact AIOS have identical payoffs outside the subgame, while games with approximate AIOS weaken this to allow some variation in the payoffs outside the subgames.

Suppose the row player is player 1 and column player is player 2. If player 1 decides to play any strategy from $\{1-10\} \in c_{1,1}$, he needs to worry only about the probabilities assigned by player 2 to strategies $\{1-10\} \in c_{2,1}$. Intuitively this is because if player 2 plays from strategies outside of $c_{2,1}$ the payoff is the same for the row player no matter which action he chooses among $\{1-10\} \in c_{1,1}$. The subgame G_1 is formed by considering only actions in $c_{1,1}$ and $c_{2,1}$.

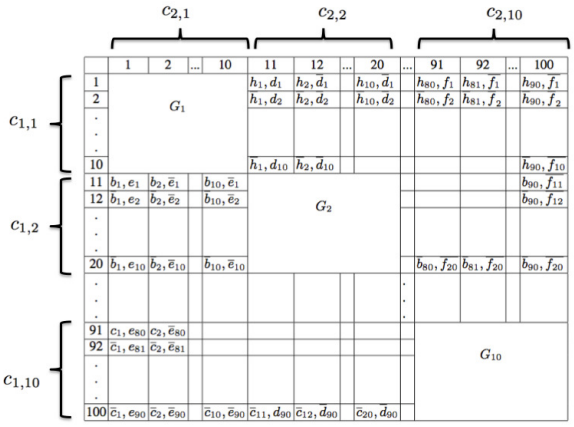


Fig. 1: AIOS Structure in an NFG

3 A Cyber Defense Game with AIOS

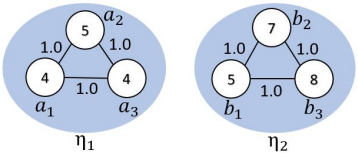


Fig. 2: Example network with $t_{i,j} = 1$ and $T(\eta_k, \eta_l) = 0$

We now present a cybersecurity scenario for a *Botnet* attack where the AIOS (Section 2) structure arises naturally. Most real-world networks are divided into subnets to increase performance and security, but there are limited resources to inspect/harden devices against attacks. Automated intrusion detection systems (IDS) [22] [17] are an important defense, but it may not be possible to use a costly IDS on every network host [1]. We present a game-theoretic model for stopping the spread of an attacker (e.g., a botnet) through a network that has a subnet architecture. Botnets often spread easily within a subnet using worms that exploit open ports and unpatched vulnerabilities. However, spreading between subnets requires moving through more secure and highly monitored routers that limit connectivity. This locality leads a game model with AIOS structure.

Figure 2 shows an example with 2 subnets containing 3 nodes each. A network is a collection of nodes that belong to exactly 1 subnet η_k. Every host has a value v_i. $t_{i,j}$ represents the *intra-transmission probability* for the botnet to propagate from node i to j within the same subnetwork η_k. $T(\eta_k, \eta_l)$ represents the *inter-transmission probabilities* for the botnet propagating from subnet η_k to η_l. We model a one-shot game where the Defender selects a node i to *defend* (e.g., closing ports, patching vulnerabilities, increasing monitoring). The defend action reduces the transmission probabilities for all edges connected to i and stops any attack that spreads to node i. The attacker selects an initial node to attack, which spreads according to the transmission probabilities (which can be estimated using simulation).

If the botnet spreads to a defended node and is detected, the Defender pays a cost equal to the total value of the infected nodes (to clean up the attack), but the attacker receives a payoff of zero. If the attack does not interact with a defended node, the attacker receives the sum of the values of all the infected nodes. We estimate the payoff matrix for a particular game using Monte Carlo simulation to estimate the spread of the infection for each pair of strategies.

Figure 3(a) shows the NFG representation for the example in Figure 2 assuming $t_{i,j} = 1$ and no edges exist between subnets. In this case, the game has an exact AIOS structure. When the two players play on the same subnet, there is a strategically interesting game. However, when the two players play outside of the same subnets, there is no interaction. Intuitively, this is because the Defender will never be able to detect the Attacker's botnet because no connection exists between subnets. Figure 3(b) shows

Attacker

Defender	a_1	a_2	a_3	b_1	b_2	b_3
a_1	-4.0, 0.0	-7.01, 2.27	-6.47, 2.24	-20.0, 20.0	-20.0, 20.0	-20.0, 20.0
a_2	-5.97, 1.97	-5.0, 0.0	-8.0, 2.02	-20.0, 20.0	-20.0, 20.0	-20.0, 20.0
a_3	-9.0, 2.19	-9.0, 2.29	-4.0, 0.0	-20.0, 20.0	-20.0, 20.0	-20.0, 20.0
b_1	-13.0, 13.0	-13.0, 13.0	-13.0, 13.0	-5.0, 0.0	-10.99, 3.75	-11.46, 3.69
b_2	-13.0, 13.0	-13.0, 13.0	-13.0, 13.0	-8.97, 3.18	-7.0, 0.0	-13.0, 3.2
b_3	-13.0, 13.0	-13.0, 13.0	-13.0, 13.0	-12.0, 2.99	-12.0, 3.08	-8.0, 0.0

(a)

Attacker

Defender	a_1	a_2	a_3	b_1	b_2	b_3
a_1	-4.0, 0.0	-7.62, 3.09	-7.12, 3.04	-20.7, 18.58	-20.69, 18.53	-20.7, 18.44
a_2	-6.83, 2.97	-5.0, 0.0	-8.82, 2.93	-20.88, 18.48	-20.84, 18.45	-20.89, 18.44
a_3	-8.6, 2.38	-8.66, 2.33	-4.0, 0.0	-19.89, 18.66	-19.93, 18.77	-19.91, 18.79
b_1	-12.97, 12.17	-12.96, 12.15	-12.96, 12.18	-5.0, 0.0	-10.64, 3.96	-11.28, 4.01
b_2	-14.01, 12.36	-14.02, 12.34	-14.03, 12.38	-9.39, 4.09	-7.0, 0.0	-13.14, 4.06
b_3	-14.41, 12.33	-14.45, 12.26	-14.44, 12.3	-12.39, 3.86	-12.41, 4.08	-8.0, 0.0

(b)

Fig. 3: (a) Game for Figure 2 with $t_{i,j} = 1$ and $T(\eta_k, \eta_l) = 0$. (b) Game for Figure 2 with $t_{i,j} = [0.85, 1.0]$ and $T(\eta_k, \eta_l) = 0.10$

the network seen in Figure 2 with a low inter-transmission probability between subnets where $T(\eta_k, \eta_l) = 0.10$ and transmission probabilities within the subnets in the range $t_{i,j} = [0.85, 1.0]$. When we add these weak interactions between subnets (i.e., relatively low transmission probabilities) we have a game with an approximate AIOS structure where actions in different subnets have only limited effects on the payoffs.

4 Fast Clustering Abstraction

We first describe a fast abstraction method based on clustering. The first step in the abstraction is to identify clusters of similar actions and group these actions to define smaller games.

The basic idea of a clustering abstraction is to find groups of similar actions and to merge them to generate a smaller game. There are many different ways to perform clustering, including different algorithms and different ways to define similarity. We use one of the most common clustering methods, k-means [16, 7] clustering with a Euclidean distance metric on the payoff vectors for each strategy. Other clustering techniques may work as well (e.g., density-based clustering). However, we consider that for future work.

The set of clusters for player i is denoted using $c_i = \{c_{i,1}, ..., c_{i,m}\}$, where $c_{i,m}$ is the m^{th} cluster for player *i*, and $c_{i,m} = \{a_{i,1}, ..., a_{i,k}\}$. Every action belongs to exactly one cluster, so $c_{i,1} \cap c_{i,2} \cap ... \cap c_{i,k} = \emptyset$. For example, if we apply k-means clustering to the game shown in Figure 4(a) we might find the following clusters for each player: $c_{1,1} = \{2, 4\}, c_{1,2} = \{1, 3\}$ and $c_{2,1} = \{3, 4\}, c_{2,2} = \{1, 2\}$.

After clustering each strategy, each cluster maps to a single pure strategy in the abstracted game. The outcomes in the abstracted game correspond to each player choosing to play one of the clusters of strategies in the original game. For this initial solution method, we do not attempt to solve the individual subgames; we will introduce methods in the next section that do use solutions to the subgames. Here we compute a quick estimate of the payoffs in the subgames by averaging the payoffs, effectively assuming that both players play a uniform random strategy in each subgame. More formally:

$$\hat{u}_i(\hat{a}_i, \hat{a}_j) = \frac{\sum_{\substack{\forall a_{i,k} \in g(c_{i,m}) \\ \forall a_{j,l} \in g(c_{j,n})}} u_i(a_{i,k}, a_{j,l})}{|c_{i,m}| \times |c_{j,n}|} \tag{1}$$

In Equation 1, $\hat{a}_i = c_{i,m}, \hat{a}_i \in \hat{A}_i(R)$ and $\hat{a}_j = c_{j,n}, \hat{a_j} \in \hat{A}_j(R)$. g is the *reverse mapping function* that we describe next. Figure 4(b) provides an example of an abstracted game calculated from the NFG in Figure 4(a) using this method.

		p_2		
p_1	9, 2	2, 7	2, 4	8, 5
	2, 8	7, 3	9, 9	9, 9
	3, 2	2, 7	9, 10	3, 3
	3, 5	7, 8	9, 5	5, 3

(a)

		p_2
p_1	8, 6.5	4.75, 5.5
	5.5, 6	4, 4.5

(b)

Fig. 4: (a) A NFG, (b) Abstracted NFG

Since the strategy space of the abstracted game is not the same as the original we need the *reverse mapping function* $g(\hat{s}_i) \rightarrow s_i$ to map every strategy in the abstracted game into a strategy in the original game. We also use a version of this function for clusters: $g(c_{i,m}) = \{a_{i,1}, \ldots, a_{i,k}\}$. For a clustering abstraction, the original equilibrium actions can be obtained using the equation $\pi^i(a_{i,k} \in g(c_{i,m})) = \frac{\pi^i(c_{i,m})}{|c_{i,m}|}$. The NE for the abstract game in Figure 4(b) is $\{(1, 0), (1, 0)\}$. The reverse mapping gives the strategy for the original game in Figure 4(a) as $\{(0, 0.5, 0, 0.5), (0, 0, 0.5, 0.5)\}$. In general, the solution to the abstracted game (e.g., an NE) may not be an exact solution in the original game since we do not directly analyze the subgames using this method, but they can serve as a quick approximation method.

4.1 Solving Games

We consider several solution methods for solving games, including the abstracted games described above. We consider both Pure and Mixed-Strategy Nash Equilibrium, as well as a different concept that directly minimizes the bound on the approximation quality in the original game.

Approximate Pure Strategy Nash Equilibrium In a Pure-Strategy Nash Equilibrium (PSNE) all players play pure strategies that are mutual best-responses. However, PSNE is not guaranteed to exist. Therefore, we instead look for the pure-strategy outcome that is the best approximate equilibrium. We first calculate the values of deviations for each action a_i and then select the action profile that minimizes the maximum benefit to deviating.

Mixed Strategy Nash Equilibrium We also calculate a version of mixed-strategy Nash equilibrium using the software package Gambit [18]. There are several different solvers for finding Nash equilibria in this toolkit. We used one based on Quantal

response equilibrium (QRE) [14]. QRE produces different versions of mixed strategy equilibrium depending on a parameter $\lambda = [0, \infty]$. When $\lambda = 0$ the players play uniform randomly and when $\lambda = \infty$ players play very close to a rational player which produces an approximate Nash equilibrium. We choose $\lambda = \infty$ for our work.

Minimum Epsilon Bounded Equilibrium When solving an abstracted game, the best analysis may not be finding a Nash Equilibrium, since this may not be an equilibrium of the original game. As an alternative, we introduce *Minimum Epsilon (ε) Bounded* equilibrium (MEB). This alternative is an improved version of PSNE that considers additional information about the abstraction in the analysis. Instead of considering deviations to clusters of actions (and the average payoff of the cluster), we use the maximum expected payoff for any of the actions in the original game. This allows for a better estimate of how close the outcome will be to an equilibrium in the original game. The difference in comparison with PSNE is in the calculation of $\varepsilon(a_i^*)$. Equation 2 is used to compute the ε for MEB.

$$\varepsilon(\hat{a}_i^*) = max_{\forall \hat{a}_i \in \hat{A}_i, \hat{a}_j \in \hat{A}_j} [\overline{u}_i(\hat{a}_i, \hat{a}_j) - \hat{u}_i(\hat{a}_i^*, \hat{a}_j)] \tag{2}$$

In the above equation $\overline{u}_i(\hat{a}_i, \hat{a}_j)$ returns a payoff from an upper bound game $\overline{R}$. Payoffs for the upper-bounded game $\overline{R}$ are computed using Equation 3. Equation 3 calculates the maximum expected payoff for an abstracted action by reverse mapping to the original actions and calculating the expected payoff for every original action, selecting the maximum one. Next, where $\forall \hat{a_i} \in \hat{A}_i(R), \forall \hat{a_j} \in \hat{A}_j(R), (i,j) \in N, i \neq j$, the equation iterates over all the actions for every player and calculates the payoffs for the upper-bounded game $\overline{R}$. Equation 2 cannot be used in the original game because we need an upper-bounded game where we use reverse mapping. Unless we have an abstracted game, it is not possible to compute an upper-bounded game.

$$\overline{u_i}(\hat{a_i}, \hat{a_j}) = max_{\forall a_{i,k} \in g(\hat{a}_i)} \frac{\sum_{\forall a_{j,l} \in g(\hat{a}_j)} u_i(a_{i,k}, a_{j,l})}{|g(\hat{a}_j)|} \tag{3}$$

Double Oracle Algorithm (DO) The Double Oracle is not a solution concept. It is a technique used to handle massive games. Double Oracle Algorithms [3] [19] utilize the method of column/constraint generation. The idea is to restrict the strategies of all the players and solve the restricted game exactly using the LP [21] for solving an NFG. Next, for each player, the best response is computed given the strategy of the opponent. The best response can be any strategy in the original game. Next, we add the best responses to the strategy of the restricted game. If it's not possible to add anymore best responses, then the game is solved. In our case, we do not have a zero-sum game to use the LP. So, we used the QRE [14] [18] to solve the restricted general-sum NFG. The QRE gives an approximate Nash Equilibrium. Thus, in our case the DO we used to compare against our proposed algorithm provides an approximate Nash Equilibrium.

5 Hierarchical Solution Method

We now describe a more refined solution approach that constructs subgames based on strategy clusters and uses the solutions to these subgames to create a more accurate abstracted game. When games have exact AIOS structure, this will result in finding an exact solution to the original game by composing the results of the subgame solutions. In cases where games have approximate AIOS structure, we propose an iterative solution method that improves solution quality by taking into account error from outside of the subgames.

5.1 Subgames

Consider the AIOS example shown in Figure 1. Ten subgames correspond to ten pairs of clusters of actions for the players. For example, G_1 is played using clusters $c_{1,1}$ and $c_{2,1}$. Now we consider building an abstracted game by first solving each of the subgames G_1 to G_{10} utilizing any solution concept to get a mixed strategy for each player in each game.

As before, the abstracted game will have one action for each player corresponding to each cluster (10 in the example). To fill in the payoffs for each pair of clusters (a 10x10 matrix), we compute the expected payoffs using the mixed strategies for the corresponding clusters (for the subgames, this is the expected payoff from the solution to the game). Figure 5 shows the resulting abstracted game R.

	$c_{2,1}$	...	$c_{2,10}$
$c_{1,1}$	$G_{1,1}, G_{1,2}$	...	
.	.		.
.	.	...	.
.	.		.
$c_{1,10}$	...	...	$G_{10,1}, G_{10,2}$

Fig. 5: Abstracted (hierarchical) Game R

Next, we solve R using any solution concepts mentioned in section 4.1. To get the reverse mapping here we must distribute the probabilities of $c_{1,1}, c_{1,2}, ..., c_{1,10}$ over all the actions in $c_{1,1}, c_{1,2}, ..., c_{1,10}$ for player 1 to get the strategy for the original game (resp. for player 2). Equation 4 gives this reverse mapping, where $i \in N, \forall a_{i,k} \in g(c_{i,m})$, which is different from the uniform mapping described previously. In equation 4 the probabilities $\pi^i(a_{i,k})$ on the right-hand side are the mixed strategies for the subgames.

$$\pi^i(a_{i,k}) = \pi^i(c_{i,m}) \times \pi^i(a_{i,k}) \tag{4}$$

5.2 Approximate AIOS Games

The AIOS structure is strict if we require exactly identical payoffs outside of the subgames. However, it is much more plausible to find approximate forms of this structure. For example, in Section 3 we saw how low transmission probabilities between subnets lead to an approximate AIOS game. For a noisy version of AIOS, we define the *delta* (δ) parameter to specify how much variation in the payoffs is allowed outside of the subgames. Let $\delta_{i,k}$ be the maximum payoff difference for any pair of actions in cluster k for player i for any strategy of the of the opponent that is not in the same subgame.

δ_i is the maximum of $\delta_{i,k}$ for player i, where k can be from 1 to a number of clusters. Equation 5 picks the maximum δ considering all of the clusters and players. Equation 6, where $(i,j) \in N, i \neq j$, calculates δ for one cluster $c_{i,k}$ for a player i.

$$\delta = max_{i \in N}(max(\delta_{i,k})), \qquad k = 1, ..., |\hat{A}_i(R)| \tag{5}$$

$$\delta_{i,k} = max(u_i(a_{i,m}, a_{j,t}) - u_i(a_{i,n}, a_{j,t})) \tag{6}$$

5.3 Iterative Solution Algorithm

For games with approximate AIOS structure, simply composing (as above) the solutions of the subgames may not be an equilibrium of the original game. The solution may occasionally play in quadrants of the game that are *not* one of the subgames solved explicitly, which results in an error when the payoffs do not match identically. We now introduce an iterative solution technique that (partially) accounts for this error. After solving the subgames and abstracted game as previously, we now calculate the expected payoff for each strategy outside its subgame. Then, we modify the subgames using this error term added to the payoffs in the subgame and solve them again, and then recalculate the abstracted game and solve it again. This process results in a sequence of modified solutions that account for the differences in payoffs outside of the subgames from the previous iteration. We call this algorithm the Iterative Subgame Abstraction and Solution Concept (ISASC).

Consider the subgame G_1 in Figure 1. We want to internalize the noise outside of the subgame into the payoffs of the subgame. So, before solving G_1, we update the payoffs for both player 1 and player 2. For action $\{1-10\} \in c_{1,1}$, we calculate the expected utility when player 2 does not play the actions in the subgame. That means that when player 1 plays $\{1-10\}$, we calculate the expected utility of $\{1-10\}$, denoted Ω_i, by considering the probabilities of player 2 playing $\{11-100\}$ from the strategy on the previous iteration. Then we update the payoffs of G_1 for player 1 for action $\{1-10\} \in c_{1,1}$ for every $\{1-10\} \in c_{2,1}$ by adding the Ω_i. This process repeats for all strategies in the game.

Pseudocode for updating the subgames is shown in Algorithm 1. $[u_i(a_i, a_j) = u_i(a_i, a_j) + \Omega_i(a_i)]$, where $\forall a_i \in A_i(G), \forall a_j \in A_j(G), \forall (i,j) \in N, i \neq j$, line 4-12, is used to update the subgames. Lines 5-7 are used to compute the expected payoff Ω_i, for an action of player i, when player j plays outside of the subgame G. $A_i(G)$ gives us the action set for player i in game G. $\pi_{T-1}(a_j)$ is the probability of action a_j from the mixed strategy for iteration $T-1$.

To improve the speed of the algorithm, we do not need to solve all of the subgames in each iteration. If a subgame does not occur with positive probability in the solution to the abstracted game, it may not need to be solved again in the next iteration. We can also create variations of ISASC by using different solution algorithms to solve either the subgames or the abstracted game. For example, we can first use the PSNE solution concept to get an NE. Then we calculate the ϵ for that equilibrium. If the ϵ is greater than 0, we can use QRE to get another equilibrium which we can use instead of the PSNE, drastically reducing computation time. In our experiments, we test several different variations of the algorithm.

Algorithm 1 Update Subgame Algorithm

Input: Subgame `G'`, original game `G`, player i

```
Subgame actions o' = Actions(G', i)
Opponent actions in G', p = Actions(G, G', i_op)
Opponent actions outside G', p' = Actions(G, G', i_op)
for j ← 1, o' do                                   ▷ for every action of player i in G'
    for k ← 1, p' do                               ▷ for every action of opponent ∉ G'
        ω = ω + PayOff(j,k,G) ×π(k)
    end for
    for l ← 1, p do                                ▷ for every action of opponent in G'
        Outcome o = [ j, l ]
        G'(o,i) = PayOff(G, o) + ω                 ▷ update the payoff in G'
    end for
end for
```

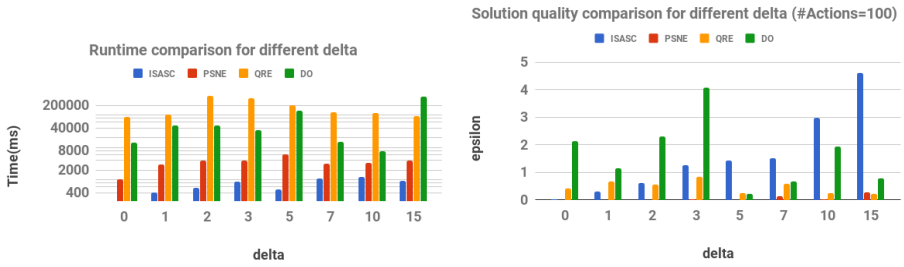
6 Experiments

In the experiment section, we used two criteria to measure the performance of our proposed algorithm: (a) runtime (b) epsilon (ϵ), which is different from ε. We know that the idea of using abstraction is to make the solution concepts scalable. However, this usage also suffers from loss of solution quality. So, We think that runtime combined with solution quality measurement with epsilon will provide a better performance measurement.

ϵ measures whether for a player there is an incentive to switch to another pure strategy from the current Nash Equilibrium strategy (which can be either a mixed strategy Nash Equilibrium using QRE or a pure strategy Nash Equilibrium using PSNE, MEB). To compute ϵ of an approximate Nash Equilibrium strategy for player i first we calculate the expected payoff of player i given the approximate Nash Equilibrium strategy of the players. Next, we check whether there is an incentive for player i to switch to a pure strategy from the current approximate Nash Equilibrium strategy (which can be either pure or a mixed strategy Nash Equilibrium). Finally, we take the maximum of all the players' ϵ which gives us the ϵ for an approximate Nash Equilibrium. In a Nash Equilibrium, there is no incentive to switch to a pure strategy for all the participating players ($\epsilon = 0$).

For our first experiment we generated 100 2-player games for different values of $\delta = \{0, 1, 2, 3, 5, 10, 15\}$ with 100 actions for each player in the original game. The strategies for each player are partitioned into 10 clusters with 10 actions each: $|c_1| = |c_2| = 10$, $|c_{1,m}| = 10$, $m = 1, 2, ..., 10$. The subgames are completely random games with payoffs generated uniformly between 0 and 100. The payoffs outside of the subgames are generated randomly with the constraint that in every cluster the maximum payoff difference between the payoffs for the actions is δ for all actions of the opponent that are not part of the subgame. Meaning, for every action outside the subgames we add noise.

We begin by showing whether there is any benefit of using the ISASC for different levels of AIOS structure. To do that we compare the runtime performance and solution



(a) Runtime comparison (b) Solution quality comparison

Fig. 6: Measuring performance against PSNE, QRE and DO which are applied in the original game without any use of abstraction

quality of ISASC against different solution methods: PSNE, QRE and DO, when these different methods are applied to the original game without the use of any abstraction as shown in Figure 6a. The results clearly show that ISASC provides solutions using considerably less time compared to QRE and DO. Even though PSNE provides an approximate pure strategy Nash Equilibrium very quickly, ISASC produces a mixed strategy approximate Nash Equilibrium much quicker than the PSNE. Since ISASC uses abstraction, the solution quality gets worse as we add more noise to the AIOS structure by increasing δ as shown in Figure 6b. Providing a theoretical bound on the solution quality is an interesting future work direction.

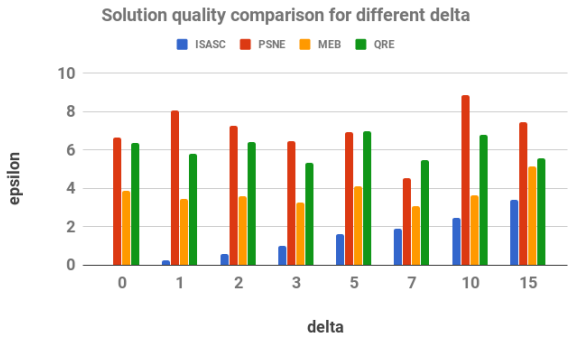


Fig. 7: Comparison of solution quality when PSNE, MEB and QRE are applied using fast clustering abstraction

The next experiment compares the solution quality of ISASC with the fast clustering abstraction methods. We use basic k-means clustering and the three different solution methods to solve the abstracted games: PSNE, MEB, and QRE. We want to show that

with the use of abstraction, ISASC is superior to other abstraction methods which use the fast clustering abstraction methods.

Figure 7 shows the results for a different set of games (generated using same parameters used for the first experiment) with varying levels of δ. ISASC does very well in cases with low δ, as expected. However, it continues to perform better when the values of δ are much more significant. We also note that MEB provides better solution quality than PSNE and QRE. For this experiment, we did not show the runtime comparison because currently, we do not have an adequate clustering or community detection method for finding the subgames of an NFG. It's a hard problem since the cluster depends not only the player whose strategy we want to cluster but also on every opponents' actions and payoffs. So, for ISASC we assume that the subgames are known, and for the other algorithms, we used k-means clustering. We understand that this assumption is not fair. However, later (in the cyber defense scenario) we show that even when PSNE, MEB and QRE have the exact clusterings (because we know the subnets), our proposed algorithm ISASC still outperforms all the other solution concepts.

6.1 Cyber Defense Games

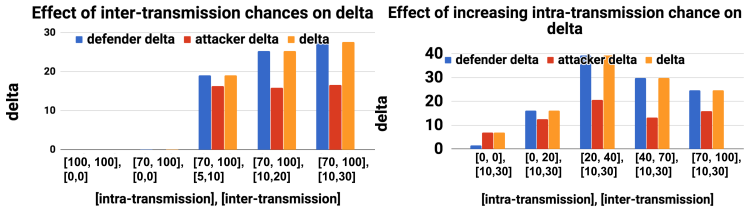
We now consider the more realistic cyber defense games described in Section 3. We compared our ISASC algorithm against the PSNE, MEB and QRE solution concepts with fast clustering abstraction. We generated 20 games using the parameter settings shown in Figure 8. Each parameter is drawn uniformly from the given range. The number of edges in subnet η_l is $|e_{\eta_l}|$ in the range $|e_{min}, e_{max}|$ where e_{min} and e_{max} are the minimum and maximum number of edges respectively. All networks are connected, and worm propagation is controlled by $T(\eta_k, \eta_l)$ and $t_{i,j}$, where $t_{i,j} >> T(\eta_k, \eta_l)$. We use Monte Carlo simulation for $10,000$ iterations to estimate the payoffs based on the propagation of the attack. Each subnet forms a cluster of actions for our solution methods. Since we already know the subnets for this cyber defense scenario, and thus the subgames, we assumed that the clusters are identified. PSNE, MEB and QRE use the known clusters to make the abstracted game to find an approximate NE in the abstracted game and then use reverse mapping to find the final approximate NE strategy for the original game.

Parameter	Value range
$t_{i,j} = t_{j,i}$	$[70, 100]$
$T(\eta_i, \eta_j)$	$[10, 30]$
v_i	$[6, 10]$
$c^d(v_i)$	$[1, 3]$
$c^a(v_i)$	$[1, 3]$
$\lvert e_{\eta_k,\eta_l} \rvert$	1
$\lvert e_{\eta_k} \rvert$	$\lvert e_{min}, e_{max} \rvert$

Fig. 8: Network settings

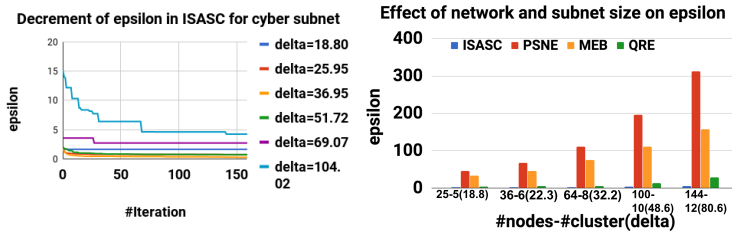
Our first experiment shows how *delta* varies as we vary the *inter* and *intra-transmission probability* in Figure 9a,9b. We used games with 50 nodes and 5 subnets with the same number of nodes. We can see in Figure 9a that when $t_{i,j} = [100, 100]$ and $T(\eta_k, \eta_l) = 0$ so the subnets are totally disconnected from each other *delta* $= 0$. In this case we can find an exact equilibrium by composing subgame solutions. However, when $T(\eta_k, \eta_l)$ starts to increase *delta* increases. In both Figure 9a and Figure 9b, we see that δ reaches a maximum when $T(\eta_k, \eta_l) \approx t_{i,j}$ as the spreading of botnet becomes random across the entire network, losing the AIOS structure.

Our next experiment shows that the ISASC algorithm improves solution quality when there is *delta* > 0 in the cyber defense games. We increased the size of the



(a) Impact: $T(\eta_k, \eta_l)$ on δ (b) Impact: $t_{i,j}$ on δ

Fig. 9: Effect of transmission parameters on δ

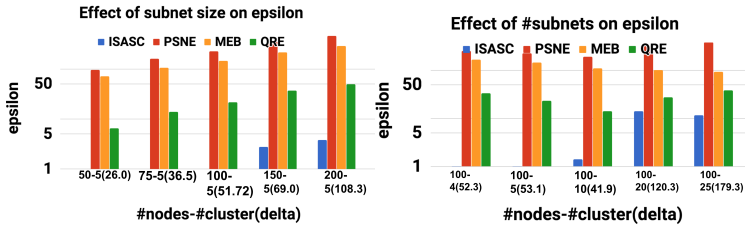


(a) Iterative decrements of ϵ (b) #node and #subnet vs ϵ

Fig. 10: Performance of ISASC

subnets to control *delta* for this experiment. As we can see in Figure 10a, when *delta* is large, ISASC improves solution quality significantly through the iterative procedure.

Next, we show how *delta* and ϵ change when we vary both network size and subnet size. In Figure 10b we can see that ISASC performs favorably compared to the other solution algorithms. Next, we increase subnet size but keep the number of subnets fixed. Figure 11a shows that as the subnet size increases *delta* increases. However, the ISASC algorithm continues to provide better solution quality with very low ϵ for higher δ.



(a) Subnet size vs ϵ (b) #subnet vs ϵ

Fig. 11: Solver comparisons

Our last experiment shows how the number of subnets affects ϵ. We kept the number of nodes fixed to 100. Figure 11b shows the result. ISASC dominates the results with

low ϵ. In all of the experiments, ISASC gives very high solution quality compared to other algorithms.

7 Conclusion

Solving large NFG is a fundamental problem in computational game theory, and abstraction is a promising direction for scaling up to solve the largest and most challenging games. We propose a new class of abstraction methods for NFG based on the AIOS structure, motivated by earlier game reduction techniques proposed by Conitzer et al. [5]. Identifying subgames in a random NFG and providing a bound on the solution quality of ISASC are possible future directions. We show that realistic cybersecurity scenarios can lead to this type of structure. We also show that there exist several abstraction-based solution methods that can take advantage of this structure to quickly find solutions to huge games by decomposing them into subgames. For games with only approximate AIOS structure, we show that iterative solution methods can give us very high-quality approximations to the solution of the original game.

8 Acknowledgments

This research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-13-2-0045 (ARL Cyber Security CRA). The views and conclusions contained in this document are those of the authors andshould not be interpreted as representing the official policies,either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes not with standing any copyright notation here on.

References

1. Alpcan, T., Basar, T.: An intrusion detection game with limited observations. In: 12th Int. Symp. on Dynamic Games and Applications, Sophia Antipolis, France, vol. 26 (2006)
2. Bard, N., Nicholas, D., Szepesvári, C., Bowling, M.: Decision-theoretic clustering of strategies. In: AAMAS (2015)
3. Bosansky, B., Kiekintveld, C., Lisy, V., Pechoucek, M.: An exact double-oracle algorithm for zero-sum extensive-form games with imperfect information. Journal of Artificial Intelligence Research **51**, 829–866 (2014)
4. Brown, N., Ganzfried, S., Sandholm, T.: Hierarchical abstraction, distributed equilibrium computation, and post-processing, with application to a champion no-limit Texas hold'em agent. Tech. rep. (2014)
5. Conitzer, V., Sandholm, T.: A technique for reducing normal-form games to compute a nash equilibrium. In: AAMAS, pp. 537–544 (2006)
6. Fabrikant, A., Papadimitriou, C., Talwar, K.: The complexity of pure nash equilibria. In: Proceedings of the thirty-sixth annual ACM Symposium on the Theory of Computing, pp. 604–612 (2004)
7. Forgy, E.W.: Cluster analysis of multivariate data: efficiency versus interpretability of classifications. Biometrics **21**, 768–769 (1965)

8. Ganzfried, S., Sandholm, T.: Potential-aware imperfect-recall abstraction with earth mover's distance in imperfect-information games. In: Conference on Artificial Intelligence (AAAI) (2014)
9. Gilpin, A., Sandholm, T.: A competitive texas hold'em poker player via automated abstraction and real-time equilibrium computation. In: Proceedings of the National Conference on Artificial Intelligence (AAAI), vol. 21, p. 1007 (2006)
10. Gilpin, A., Sandholm, T.: Better automated abstraction techniques for imperfect information games, with application to texas hold'em poker. In: AAMAS, p. 192 (2007)
11. Gilpin, A., Sandholm, T.: Lossless abstraction of imperfect information games. Journal of the ACM (JACM) **54**(5), 25 (2007)
12. Gilpin, A., Sandholm, T., Sørensen, T.B.: Potential-aware automated abstraction of sequential games, and holistic equilibrium analysis of texas hold'em poker. In: Proceedings of the Conference on Artificial Intelligence (AAAI), vol. 22, p. 50 (2007)
13. Gilpin, A., Sandholm, T., Sørensen, T.B.: A heads-up no-limit texas hold'em poker player: discretized betting models and automatically generated equilibrium-finding programs. In: AAMAS, pp. 911–918 (2008)
14. Goeree, J.K., Holt, C.A., Palfrey, T.R.: Quantal response equilibrium. The New Palgrave Dictionary of Economics. Palgrave Macmillan, Basingstoke (2008)
15. Kroer, C., Sandholm, T.: Extensive-form game abstraction with bounds. In: Proceedings of the fifteenth ACM conference on Economics and computation, pp. 621–638 (2014)
16. Lloyd, S.: Least squares quantization in pcm. Information Theory, IEEE Transactions on **28**(2), 129–137 (1982)
17. Matta, V., Di Mauro, M., Longo, M.: Ddos attacks with randomized traffic innovation: botnet identification challenges and strategies. IEEE Transactions on Information Forensics and Security **12**(8), 1844–1859 (2017)
18. McKelvey, R.D., McLennan, A.M., Turocy, T.L.: Gambit: Software tools for game theory (2006)
19. McMahan, H.B., Gordon, G.J., Blum, A.: Planning in the presence of cost functions controlled by an adversary. In: Proceedings of the 20th International Conference on Machine Learning (ICML-03), pp. 536–543 (2003)
20. Sandholm, T., Singh, S.: Lossy stochastic game abstraction with bounds. In: Proceedings of the 13th ACM Conference on Electronic Commerce, pp. 880–897 (2012)
21. Shoham, Y., Leyton-Brown, K.: Multiagent systems: Algorithmic, game-theoretic, and logical foundations. Cambridge University Press (2008)
22. Venkatesan, S., Albanese, M., Shah, A., Ganesan, R., Jajodia, S.: Detecting stealthy botnets in a resource-constrained environment using reinforcement learning. In: Proceedings of the 2017 Workshop on Moving Target Defense, pp. 75–85. ACM (2017)
23. Zinkevich, M., Johanson, M., Bowling, M., Piccione, C.: Regret minimization in games with incomplete information. In: Advances in neural information processing systems, pp. 1729–1736 (2007)