DEEP BLOC: A GAME THEORETIC APPROACH TO ORCHESTRATE CPS AGAINST CYBER ATTACKS

by

Alireza Tahsini, B.S.

A thesis submitted to the Graduate Council of
Texas State University in partial fulfillment
of the requirements for the degree of
Master of Science
with a Major in Computer Science
August 2019

DEDICATION

In dedication to my wonderful Parents, whose unending love and support has allowed me grow more than I could ever imagine.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my deepest gratitude to my advisor Professor Guirguis for his encouragement, patience and plentiful guidance. It would be impossible to count all the ways that you have helped me to grow step by step during this three years. I do appreciate the honor that you granted me to work under your supervision.

Next, I would like to praise Professor Eftekhari, who supported me through my undergraduate studies both technically and spiritually. Without your hints I never would have embarked on this wonderful journey in the first place. Lastly, I want to thank you for directing my research path deliberately, which provided me further opportunities and achievements.

A very special thanks to my coauthor, colleague and friend Noah Dunstatter. We had a tense but fun adventure together. DNQN works after all of hard work, discussions, disappoints, long nights and hours spent on complicated equations. than you! Everyday of working with you was a blast for me and I hope some day I work with my inspiring, kind and intelligent friend again.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

One important aspect in protecting CPS is ensuring that the proper control and measurement signals are propagated within the control loop. The CPS research community has been developing a large set of check blocks that can be integrated within the control loop to check signals against various types of attacks (e.g., false data injection attacks). Unfortunately, it is not possible to integrate all these "checks" within the control loop as the overhead in checking signals may violate the delay constraints dedicated by the control loop. Moreover, these blocks do not completely operate in isolation of each other, but dependencies exist among them in terms of their effectiveness against detecting a subset of attacks. Thus, it becomes a challenging and a complex problem to assign the proper checks, specially with the presence of a rational adversary who can observe the check blocks assigned and optimizes her own attack strategies accordingly. This paper tackles the inherent state-action space explosion that arise in securing CPS through developing a unifying framework in which Deep Reinforcement Learning algorithms are utilized to provide optimal/sub-optimal assignments of check blocks to signals. The framework models stochastic games between the adversary and the CPS defender and derives mixed strategies for assigning check blocks to ensure the integrity of the propagated signals, while abiding to the real-time constraints dictated by the control loop. Furthermore, the strategies obtained reflect various factors such as the aggressiveness and risk associated to the players in taking defense/attack actions. Our results show that our framework can obtain assignment strategies that outperform other strategies and heuristics.

I. INTRODUCTION

Cyber Physical Systems (CPS) will be an essence to our daily activities, as we can already see applications of CPS in a variety of domains such as transportation, power, healthcare, and manufacturing systems. These systems are mounted on critical foundations, which directly touch human lives and governments' establishments.
Compromising the operation of such strategic infrastructures is fascinating for any adversaries. This adversaries can be arbitrary ones who use classic attack strategies (e.g., DoS and ransomeware) to gain some immediate pay offs. Conversely, they can be sophisticated ones capable of running attacks that can mitigate the normal operation of the underlying infrastructure, in order to slowly drive the system into an insecure stage and, still, stay hidden. We have already witnessed incidents stemmed from such forms of attacks like stuxnet worm to the recent attacks on the Ukrainian power grid[1]. Soon we will encounter more instances of these attacks that cause vehicles to veer off the road, manipulate devices responsible for power generation and consumption and exploit robotics/drone systems for malicious and terrorism-related activities.
This is while, the vast majority of the CPS built are validated through *ad hoc* trial-and-error approaches. Photographs of CPS engineers in hard hats with laptops debugging their CPS attached to buildings and bridges dominate the general opinion of CPS. A recent empirical study of verification and validation techniques used by CPS experts [2] has demonstrated a pervasive frustration with the lack of rigorous tools; confirming that *in situ* trial-and-error debugging is the most commonly used approach.
Despite the recent (and rich) research efforts in identifying attacks on CPS and developing resilient defense mechanisms against them, a coherent approach in which these mechanisms can be "put" together is lacking. Blindly throwing more checks and defenses at the CPS may not just be unnecessary, but could

negatively impact the stability margins of the system allowing more attacks to be mounted.

Problem Statement

Ensuring the correctness, timing and integrity of the control and measurement signals requires orchestrating the control loop with various "check blocks" – such as threshold checks, model predictors, learning modules, assertions and action blocks. Due to their different timing, overhead and effectiveness characteristics, it becomes challenging to choose the right one(s), specially against a rational adversary who is aware of the blocks present and seeks to inflict the maximum damage. Moreover, they must abide by the capacity constraints dictated by the process controlled. In this work, we present a game-theoretic framework – which we coin BLOC – that dynamically equips the control loop with the appropriate blocks at the right location(s) to protect it against a rational adversary. Through an intelligent orchestration of blocks that are the results of mixed strategies, the system can be defended against an adversary that can choose the best attack method against the system. BLOC considers the range of possible attacks, the importance of each measurement and control signal, the nature of each block (e.g., stateless or stateful) as well as the effectiveness and timing of each block.

Contributions

We summarize our contributions below:

1. We present the BLOC framework, which orchestrates the CPS with various check blocks in a coherent manner. These blocks are enabled at four strategic locations with access to the intended control and measurement signals.

2. We introduce Deep Nash Q-Network(DNQN), to derive the near optimal policy on Markov games. This algorithm is capable of handling

environments with exponentially large and discrete action spaces in presence of asymmetric agents.

3. We develop a Markov game model to capture the CPS dynamics, agents' behavior and security measurements. On top of that, we apply DQNQ to find the a policy for the defender, in order to assign/remove check blocks during the horizon to properly protect the system. At the same time, DQNQ approximates the attacker's optimal policy to choose the best attack method and targets to attack.

II. RELATED WORK

There has been a lot of research efforts in securing CPS and networked control systems against spoofing (e.g., [3, 4, 5, 6, 7, 8]) and jamming/delaying attacks (e.g., [9, 10, 11]). These studies differ in their assumptions about the attacker's knowledge and the types of signals that can be attacked. In [4], the authors show false data injection attacks on state estimators in power grids. The idea is to craft an attack vector – in which each element corresponds to an injected measurement from a meter – that when combined with the state estimators, would still pass detection. The authors assume the attacker knows the configuration of the power system but may not have access to all the meters. The work in [5] generalizes false data injection attacks on control systems with specific controllers (e.g., Kalman) and shows that these attacks can cause the system to become unstable. In [7], the authors illustrate a scheme in which the control signals can be spoofed in tandem with the measurement signals to hide the effect of the attack. This scheme in effect hijacks the operation of the CPS. In terms of jamming attacks, the authors in [12] study the performance of a linear control system subject to various Denial of Service (DoS) attack models on the measurements and control signals (e.g., random, Bernoulli, constrained and general). Wireless jamming has been shown to cause severe effects that may cripple the entire system (e.g., [13, 14]).

To defend against these attacks, in [6], the authors construct a safety envelope from the measurements obtained under the normal operation of the system (without attacks). Attack detectors are then constructed that compare the measurements received during the operation of the system to the ones maintained by the safety envelope. The authors in [8] assume knowledge of the state of the system and derive correlation graphs without attacks to study how that information impacts decisions. In [3] the authors prevent an adversary from finding attack vectors through identifying two sets: a set of sensors to protect

and a set of state variables that can be independently verified. While the authors in [15] investigate the combination and configuration of various defense mechanisms using stateless and stateful detection schemes for CPS, they do not consider a game-theoretic approach. In our work, we present a framework in which these defenses – in addition to other blocks – can be orchestrated through rigorous game-theoretic approaches. The use of game theory has been instrumental in advancing the state-of-the-art in security games and their wide range of applications (e.g., as it operates under the worst-case scenario).
In [16], authors represent a human-CPS interaction model in a smart city, where the CPS elements are considered to form a graph. In that graph, the cyber and physical elements are interdependent nodes; consequently, each node is a battlefield in the proposed Colonel Blotto game, where the attacker and defender assign recourses to win a battle. Although that formulation is a very general one, it does not explicitly take into account the specific interaction of resources in each battlefield. The authors in [17] investigate the process of transmitting data from a sensor to a controller in a CPS, while the attacker tries to strike this communication with a jamming attack. By considering power constraints for both players, a static and a dynamic game is formulated for finding the optimal transmission and attack strategies for the players, given a time horizon.
Another effort in [18] presents a model for assuring the security in a CPS by solving a parametric game matrix. Those specified parameters express the cost, loss and benefit of players after taking an action. They have defined a small state space to capture the dynamic behavior of the system in normal situation and under attacks.
The work in [19] presents a receding-horizon methodology to approximate an infinite-horizon dynamic Stackelberg game to securely control a CPS. The model assumes two attackers: one can attack the measurement signals and the other one can attack the control signals. The model is a two-level Stackelberg game, where the measurement jammer makes a decision before the operator and the

control jammer. In the first-level, the measurement jammer is the leader and the operator (controller) and the control jammer are the followers. In the second one, the operator is the leader and the control jammer is the follower. The attackers try to manipulate the signals to destabilize the system, while minimizing the cost of the attack by solving a quadratic program. On the other hand, the operator tries to find a control law that maintains the stability of the system. Our work is different from the above works as it considers a different formulation in which the defender is seeking an assignment of concrete check blocks – that have varying effectiveness to protect against various attack methods while abiding to the delay constraints imposed by the control loop as well as accounting for the uncertainty in the number of signals received.

In [20], the authors propose a hybrid game-theoretic approach for resilient system control that also considers the system security and robustness. The scenario is stated as a cross-layer design in which two games are intertwined – a zero-sum differential game is used for robust control and a stochastic one is used for the design of the defense mechanisms. Unlike this work, our framework captures details about the exact blocks used, their effectiveness and impact on the signals propagated in the network.

Aside from CPS, security games have been instrumental in improving security of other cyber domains and non-cyber domains. The work in [21] framed the airport passenger screening problem as a Stackelberg game and solved for optimal allocation strategies that were then implemented at LAX international airport. Similarly, [22] used a Stackelberg game formulation to optimize fare inspection points in Los Angeles' metro system in an attempt to deter fare evasion. The authors in [23] provide an overview of game-theory's various successes in the physical domain and end with a discussion of possible applications and challenges in the cyber domain. Recent works applied game-theoretic screening strategies to the cyber-alert assignment problem faced by network security teams with [24, 25] solving for static screening strategies and [26] employing a dynamic screening strategy that considers a temporal element of play.

Incidentally, if we wish to derive time-dependent policies for defense we could use self-play as discussed in [27, 28, 29]. In these, methods agents are trained one at a time, in the way a player approximates the best response on a Markov Decision Process (MDP), which is made by the other players' strategy profile.

Interchangeably, frame the problem in the context of multi-agent reinforcement learning (MARL) on a Markov game as suggested in [30]. It has been shown in [31] that such systems can be proven to converge to Nash values under certain settings and provides us with a reliable way to derive game-theoretic and time-dependent policies. In fact, this is exactly what was done in [26] where dynamic programming was used to perform value-iteration and arrive at optimal defense strategies. However, such solution methods do not scale well to larger more realistic state spaces.

III. BLOC FRAMEWORK

In this section, we present a general model for CPS, the adversary model, the check blocks, and the BLOC framework.

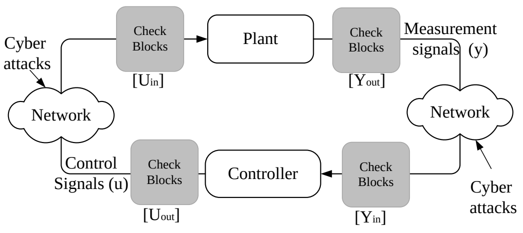

Figure III.1: A general block diagram for a CPS.

The CPS model

Figure III.1 shows a generic block diagram of a CPS composed of a plant and a controller. Let x_k denote the state of the system at time k. A vector of measurement signals y_k is generated from the CPS and is fed to decision making entities (e.g., controllers that are centralized or distributed systems). The y_k signals intend to *capture* the state x_k in as representative way as possible, for instance by sensing the x_k values directly or by inferring the desired components from other dependent ones. The decision making entities process the y_k (measurement signals) and take control decisions u_k that change the state of the CPS. The dynamics of the plant and the controller can be captured through various models such as linear time invariant, linear time variant, non-linear or hybrid models. We assume that measurement signals and/or control signals traverse network components that are subject to cyber attacks.

The Adversary Model

We consider an adversary who is choosing different attack vectors on the measurement and control signals. If we let $\Gamma_u(u)$ and $\Gamma_y(y)$ denote the attack function on the control and measurement signals, respectively, then $\bar{u} = \Gamma_u(u)$

and $\bar{y} = \Gamma_y(y)$ are the bad control and measurement signals received and acted upon by the CPS components. In case of a jamming attack, the attacked signal is simply omitted from the vector. Such “attacks” could just as easily be non-malicious but just as dangerous errors or unknowns, whether in the communication medium, in sensing, in actuation, or in some combination. We adopt a rational adversary model who is effectively selecting the $\Gamma_u()$ and/or $\Gamma_y()$ to bypass the check mechanisms in place to achieve the maximum damage.

The Check Blocks

Check blocks are the components that check signals and take actions (e.g., watermark, encrypt, alert, etc). In general, depending on their operation, these blocks would need to maintain and utilize a state of previous values. For example, a cusum (cumulative sum for change detection) check block is an example of a stateful one that is required to keep a history and can have different effectiveness based on the threshold chosen; whereas a simple threshold check would be considered stateless. Our models treat these checks as parameterized components in which their operations and parameters impact their effectiveness. Check blocks include differentiators, aggregators, model predictors, state estimators, etc. as well as more sophisticated elements that utilize machine learning methods. We present examples of them within our BLOC framework.

The BLOC Framework

BLOC orchestrates the CPS control loop with various “check blocks" within four major locations as depicted in Fig. III.1. The four major locations are: $[Y_{in}]$, $[Y_{out}]$, $[U_{in}]$ and $[U_{out}]$. Due to their unique locations, each one processes different types of signals—some are guaranteed to be valid (e.g., because of a tight coupling to a physical component), while some may be spoofed or distorted by noise.

- **The** $[Y_{in}]$ **location:** This location has access to all the previous measurement signals received $\bar{y}_1$, $\bar{y}_2$, ... $\bar{y}_k$ (which could have been spoofed or otherwise exhibit errors) and uses these historical values and the most recent measurements to calculate a current best estimate of the actual physical state. A common check that fits in this block is to measure the standardized residual using the χ^2 statistic which ensures that the received measurements do not deviate from the estimated state based on a threshold typically chosen based on a hypothesis testing criterion [32]. As illustrated in [4], such a check would not be enough to protect against spoofing attacks.

  Another component that fits in this location is a safety envelope test in which the new measurement signals received, $\bar{y}_k$, are compared against a measurement data model obtained under no attack using machine learning algorithms [6].

- **The** $[U_{in}]$ **location:** This location receives all the new control signals, $\bar{u}_1$, $\bar{u}_2$, ..., $\bar{u}_k$ (which could have been spoofed) and also has access to the current state of the physical system. It is also reasonable to assume that this component knows the initial state of the system, x_0, and thus can track the evolution of the state based on the control signals received (subject to the computational capabilities available). One of the primary goals of this block is to assert that the control signals received will not cause immediate violations of any of the system's target invariant properties if the actions embedded in the $\bar{u}_k$ are applied to the system (i.e., it simulates the result of applying the control signal). Because it has access to the historical state, this component can also verify properties relating to control signals changing too fast or too slow.

- **The** $[Y_{out}]$ **Location:** This location has access to all the previously generated measurements, y_1, y_2, ..., y_k and it can compare the new measurements to previous ones to detect immediate inconsistencies—in essence ensuring that the applied control did not violate any of the system's constraints. This location can also infer across elements within y_k over time to determine whether the vector of measurement signals is internally consistent. Finally, and perhaps the most useful against spoofing attacks, a check block can watermark y_k deliberately, so that the Y_{in} block can detect any tampering [33].

- **The** $[U_{out}]$ **Location:** This location can run/utilize a predictor (e.g., Model Predictive Control [34, 35]), given the current state estimate and the new control signal generated. Doing so, this block predicts the *expected state* of the physical elements of the system and verifies that this expected state does not violate any of the system properties. It assumes that a component in this location has access to the state space parameters, which is reasonable. Much of what is possible in this block is analogous to that in the $[U_{in}]$ block, but blocks in this location operate *before* the signals have traversed the network, where errors and attacks can happen.

  As in the $[Y_{out}]$ location, a block in the $[U_{out}]$ location can watermark the control signals and have a block in the $[U_{in}]$ location verify their presence to protect against spoofing attacks.

IV. MARKOV MODEL

We consider a game-theoretic formulation of a 2-player Markov game between the defender and the adversary. The defender seeks to protect the CPS against various attacks through enabling a set of check blocks. The adversary seeks to attack the CPS through selecting a set of target signals (e.g., a measurement or a control signal) to attack and a particular attack method to employ (e.g., spoof, jam, delay). Given a certain performance budget that the CPS can tolerate in the presence of these blocks, the defender seeks to assign blocks within the $[U]$ and $[Y]$ locations to maximize their utility subject to the adversary choosing her best response. In this problem we have:

- **Signal Targets** T: The adversary can choose to attack different target signals – measurements or control. We let U_t denote the utility the defender loses after an attack on target t, when t is unprotected.
- **Attacks Methods** M: The adversary can choose one attack method m from a set of available attacks M to target the CPS.
- **Check Blocks** B: The defender can choose to protect signals from attacks by enabling check blocks. We let E_b^m denote the effectiveness (i.e., the probability) of block b in protecting a target against attack m. We assume that each block imposes some delay on the system when checking signals, therefore we let C_t denote the number of check blocks target t can tolerate before violating the delay constraints on the control loop. To address the statefulness property of check blocks, we define the maximum warm-up period W_b to be the number of time steps check block b needs to observe the signals before being fully functional.

We model this problem as a two-player zero-sum Markov game, in which a sub-game is played in every time-step during an infinite horizon. This Markov game is represented by the tuple $\langle \mathcal{S}, \mathcal{A}_a, \mathcal{A}_d, \mathcal{T}, \mathcal{R}, \beta \rangle$ where:

- $\mathcal{S}$ is the finite set of system states, where each $s \in \mathcal{S}$ describes the assignment of blocks to targets and the attacker budget.
  - **Block assignment** N represents assignment status of blocks on targets. Each entry, $n_{b,t}$, demonstrates number of times steps needed by b to be fully effective on t.
  - **Predictability** p is a measure to capture the versatility of the defender's policy. Every state's Block Assignment is compared with its predecessor and p will be the number of blocks that have remained fully active.
  - **Attacker budget** ψ is an integer bounded by Ψ, which conceptually represents the attacker capabilities for striking the system, e.g. amount of attacker resources, her knowledge about the aimed CPS and the risk level she is willing to take.
- $\mathcal{A}_a$ is a finite set of actions the attacker can choose from in order to attack the system. Each action contains an attack method m and a vector v specifying for every target whether it is attacked (1) or not (0).
- Defender's actions $\mathcal{A}_d$ is, also, a finite set. Actions include adding or removing one block, or making no changes to the block assignment.
- $\mathcal{T} : \mathcal{S} \times \mathcal{A}_a \times \mathcal{A}_d \times \Rightarrow \Pi(\mathcal{S})$ is the state evolution function based on the agents action pair, where Π is a discrete probability distribution over $\mathcal{S}$. Accordingly, we let $\mathcal{T}(s, a, d, s')$ denote the probability of transitioning into state s' from state s when the attacker and the defender take actions $a \in \mathcal{A}_a$ and $d \in \mathcal{A}_d$, respectively.

  Based on the adversary's action a, her budget ψ is either decremented based on the number of attacked targets, or incremented by a unit value ψ_0.

  Block assignment matrix will be altered, as well; for partially active blocks the warm up time gets decremented by one. In a case, the defender decides to deploy a block on a target, the corresponding element in N will be set to W_b. Additionally, if he decides to remove a block, the corresponding element will be deactivated. Beside that, N is exposed to modification due to the environment uncertainty $\Pi(\mathcal{S})$, which is set to carry the uncertainty in the physical part of the CPS. In every transition, with probability λ, system will be affected by a shock, where some blocks get removed or assigned in a random scenario.

- $\mathcal{R} : \mathcal{S} \times \mathcal{A}_a \times \mathcal{A}_d \times \mathcal{S} \Rightarrow \mathrm{I\!R}$ is the reward obtained by the defender. We let $\mathcal{R}(s, a, d, s')$ denote the reward received from transitioning from state s to s' under actions $a \in \mathcal{A}_a$ and $d \in \mathcal{A}_d$. In this environment due to the zero-sum nature of the game, one reward function suffices for both agents. Equation IV.1 captures the protection of the targets (first term) and predictability of block assignment (second term), where γ is a constant to punish the defender for a stagnant assignment.

$$\mathcal{R}(s, a, d, s') = -\sum_{t \in v} x_{m,t} U_t - \gamma p \tag{IV.1}$$

$$x_{m,t} = \prod_{b \in B} (1 - E_b^m \times (1 - \frac{n_{b,t}}{W_b})) \tag{IV.2}$$

$$p = \sum_{b \in B} \sum_{t \in T} 1_{n_{b,t} \; = \; 0 \; in \; s \; and \; s'} \tag{IV.3}$$

In equation IV.2, $x_{a,t}$ is the overall effectiveness of the check block assignment against attack action m on target t. As shown in the equation, partially effective blocks have an effectiveness proportional to the number of remaining time steps to become fully effective. Equation IV.3 captures the stagnancy of the assignment, where 1_Z is an indicator function associated with event Z. In essence, it counts the number of check blocks which remain fully active between two successive states.

- β : is the discount factor such that $0 < \beta < 1$.

V. METHODOLOGY

In this section we propose, Deep Nash Q-Network (DNQN), a new approach for inferring suboptimal policies for multi-agent environments. We extended Deep Q-Networks (DQN) in a similar manner the authors in [36, 31] introduced Nash Q-learning as a version of Q-learning [37].

As shown in V.1, Q-learning uses Bellman's equation to seek a policy with the maximum expected discounted reward:

$$Q(s_t a_t) = (1-\alpha)Q(s_t, a_t) + \alpha(r + \beta \max_{a_{t+1}} Q(s_{t+1}, a_{t+1})), \tag{V.1}$$

where α is the learning factor, r is the immediate reward, β is the discount factor and $Q(s_t, a_t)$ is the quality of action a_t in state s_t. The equation iteratively updates $Q(s_t, a_t)$ with regards to the *best* action available in the next state, s_t. When it comes to multi-agent environments, the *best* action has a more complicated meaning, as all agents should agree upon the quality of the chosen action. Therefore, depending on the nature of the environment, *max* operator would need to be replaced with more complicated ones, as in Nash Q-learning, $Nash$ is the substitute for the *max* operator when agents are non-cooperative:

$$Q(s_t, a_t, d_t) = (1-\alpha)Q(s_t, a_t, d_t) + \alpha(r + \beta Nash(s_{t+1}))$$

$Nash$ calculates a one stage Nash equilibrium strategy profile and the corresponding utility of the sub-game generated based on players' actions, where each entry of the game matrix is the Q-value of the corresponding action pair.

DQN, introduced in [38, 39], is a Q-learning method, with the capability of operating in environments with exponentially large state and/or actions spaces (e.g., Atari games). Utilizing a deep neural network, DQN can generalize Q-values over a smaller number of observations; we let $Q(s_t, a_t; \theta)$ denote the

predication of the neural network with parameters θ about the quality of a_t in s_t. To ensure the convergence of DQNs, a second neural network – called the target network – is typically defined that takes advantage of experience replay. Target network estimates the target Q-values, $Q(s_t, a_t; \theta^-)$, that will be used to compute the Bellman error and will only be updated periodically with the training network's parameters. Furthermore, experience replay buffer is used to keep the historic interaction of the agent with the environment. The agent randomly samples this buffer to learn from diverse past experiences, instead of focusing on the recent events. Random sampling breaks the data correlation in the mini-batch, as well, which is a highly preferred event by the neural network.
As represented in Algorithm 1, with same concept as Q-learning, DQN uses the *max* operator to choose the action with the highest Q-value to minimize the Bellman error. To take an advantage of DQN in multi-agent environments, such as our proposed BLOC framework, we replace the *max* operator with $Nash$, as shown in Algorithm 2.
The training starts, with initializing the target and training networks, as well as the Experience Replay with random transitions. In every state, a game matrix will be generated based on the available actions to the defender and the attacker, where the value of each entry is estimated by the training network $Q(s_t, a_t, d_t; \theta)$. Then, agents transit to the next state regarding the strategy profile derived from $Nash(s)$ or a random policy with probability ϵ. After adding the transition to the replay buffer, a mini-batch of transitions is randomly sampled from the buffer. At this point, for every sample in the mini-batch a game matrix gets generated for the next state s_{t+1} using the target network and $Nash$ computes the games' Nash value. Afterwards, the training network gets updated using the inferred Bellman error.

Algorithm 1 Deep Q-learning with Experience Replay

```
Initialize $i = 0$
Initialize learning network with random weights $\theta_i$
Initialize target network weights $\theta^- = \theta_i$
Initialize $\tau$ to desired update cycle
Initialize replay memory $\mathcal{Z}$ to capacity $N$
for episode= $1, E$ do
    Randomize starting state $s_1 \in S$
    for t = $1, T$ do
        With probability $\epsilon$ select random action $a_t$
            otherwise select $a_t = \text{argmax}_a Q(s_t, a_t; \theta)$
        Execute $a_t$ and observe reward $r_t$ and next state $s_{t+1}$
        Store transition $(s_t, a_t, r_t, s_{t+1})$ in $\mathcal{Z}$
        Sample minibatch of transitions $(s_j, a_j, r_j, s_{j+1}) \in \mathcal{D}$
        Set $y_j = r_j + \beta \max_{a'} Q(s_j, a'; \theta^-)$
        Perform gradient descent step on $(y_j - Q(s_j, a_j; \theta))^2$
        Set $i = i + 1$
        if $i$ mod $\tau = 0$, then
            $\theta^- = \theta_i$
        end if
    end for
end for
```

Fictitious Play

Theoretically, DNQN is capable of finding Nash Equilibrium on Markov games, however, in practice implementation of $Nash$ determines if it is feasible to use DNQN in realistic environments. Deterministic solutions (e.g. linear program) are not able to solve the game matrix in a timely manner, especially in the case of environments with large actions spaces. Therefore, we need to approximate the Nash Equilibrium in environments similar to BLOC.

Introduced in [40] Fictitious Play is an iterative approach for learning in normal form games. In this algorithm, players play the game repeatedly tracking the other players' behavior, then they react with the best response strategy regarding the historic information, Algorithm 3.

The fact that Fictitious Play is not precise does not essentially imply a disadvantage. In line 10 of Algorithm 2, Fictitious Play outputs the Nash strategy profile, this is while in line 11 and 13 agents get a chance to explore the

Algorithm 2 Deep Nash Q-Network with Experience Replay

```
1: Initialize $i = 0$
2: Initialize learning network with random weights $\theta_i$
3: Initialize target network weights $\theta^- = \theta_i$
4: Initialize $\tau$ to desired update cycle
5: Initialize replay memory $\mathcal{Z}$ to capacity $N$
6: for episode $= 1, E$ do
7:     Randomize starting state $s_1 \in S$
8:     for $t = 1, T$ do
9:         $\mathcal{Q}_s$ = GetQMatrix($s$, $\theta_i$)
10:        $\langle \pi_A,\ \pi_D \rangle$ = Nash($\mathcal{Q}_s$)
11:        With probability $\epsilon$ select a random action $a_t$
12:            otherwise sample $a_t \sim \pi_A$
13:        With probability $\epsilon$ select a random action $d_t$
14:            otherwise sample $d_t \sim \pi_D$
15:        Execute actions $\langle a_t, d_t \rangle$ in and observe reward $r_t$ and next state $s_{t+1}$
16:        Store transition $(s_t, a_t, d_t, r_t, s_{t+1})$ in $\mathcal{Z}$
17:        Sample random mini-batch of transitions $(s_j, a_j, d_j, r_j, s_{j+1})$ from $\mathcal{Z}$
18:        $\mathcal{Q}_{s_{j+1}}$ = GetQMatrix($s_{j+1}$, $\theta^-$)
19:        $\langle v_{s_{j+1}} \rangle$ = Nash($\mathcal{Q}_{s_{j+1}}$)
20:        Set $y_j = r_j + \beta v_{s_{j+1}}$
21:        Take gradient descent step on $\big(y_j - Q(s_j, a_j, d_j; \theta_i)\big)^2$
22:        Set $i = i + 1$
23:        if $i$ mod $\tau = 0$, then
24:            $\theta^- = \theta_i$
25:        end if
26:    end for
27: end for
```

environment. To justify, we can argue the small inaccuracy in strategy profile would be seen as a part of the exploration process.

With that being said DNQN is capable of estimating Nash Equilibrium in 2 player zeros-sum games for symmetric or asymmetric agents. This process is depicted in V.1.

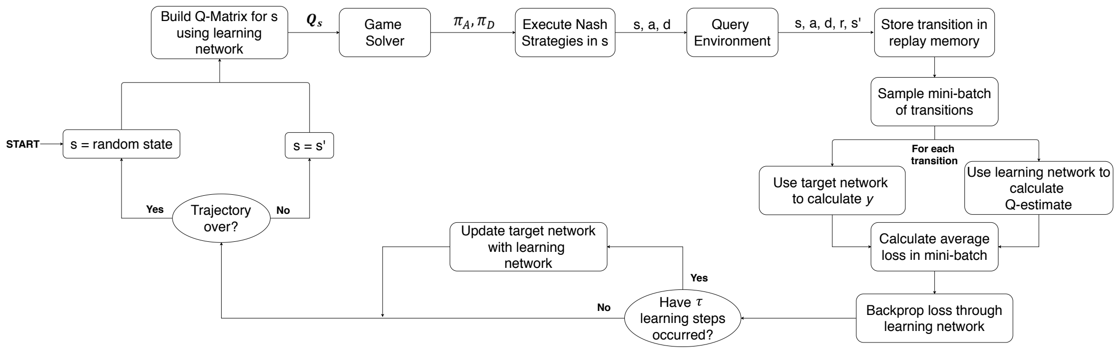
START
s = random state
Build Q-Matrix for s using learning network
Q_s
Game Solver
π_A, π_D
Execute Nash Strategies in s
s, a, d
Query Environment
s, a, d, r, s'
Store transition in replay memory
Sample mini-batch of transitions
For each transition
Use target network to calculate y
Use learning network to calculate Q-estimate
Calculate average loss in mini-batch
Backprop loss through learning network
Have τ learning steps occurred?
Yes
Update target network with learning network
No
Trajectory over?
Yes
No
s = s'

Figure V.1: Diagram of the DNQN learning process

Algorithm 3 Iterative Fictitious Play

```
$R$ is an $m \times n$ matrix of rewards
$rowReward$ and $rowCnt$ are $m$-length arrays of zeros
$colReward$ and $colCnt$ are $n$-length arrays of zeros
Initialize $bestResponse$ to any random row action
for $iterations$ do
    $colReward \ = \ colReward \ + \ R[bestResponse, \ \cdot \ ]$
    $bestResponse \ = \ \text{argmin}(colReward)$
    $colCnt \ = \ colCnt \ + \ 1$
    $rowReward \ = \ rowReward \ + \ R[\ \cdot \ , \ bestResponse]$
    $bestResponse \ = \ \text{argmax}(rowReward)$
    $rowCnt \ = \ rowCnt \ + \ 1$
end for
$gameValue \ = \ \Big(\big(\max(rowReward) \ + \ \min(colReward)\big) \ / \ 2\Big) \ / \ iterations$
$rowMixedStrat \ = \ rowCnt \ / \ iterations$
$colMixedStrat \ = \ colCnt \ / \ iterations$
```

VI. PERFORMANCE EVALUATION

In this section, we evaluate the performance of our intelligent block assignment method. We show that BLOC framework will provide the highest protection level comparing to other common heuristics.

Environments

We assessed the performance of BLOC on four different instances of the environment, Table VI.1.

Table VI.1: Sample Environment Settings

Name	ENV-SMALL	ENV-MID	ENV-MID-E	ENV-LARGE
$\|\mathbf{T}\|$	3	5	5	8
$\|\mathbf{B}\|$	3	6	6	10
$\|\mathbf{M}\|$	3	6	6	10
$\mathbf{C}$	$[2, 2, 1]$	$[4, 3, 6, 2, 5]$	$[4, 3, 6, 2, 5]$	$[4, 8, 6, 4, 8, 5, 9, 7]$
$\mathbf{U}$	$[25, 15, 40]$	$[35, 25, 40, 30, 32]$	$[35, 25, 40, 30, 32]$	$[36, 16, 47, 10, 20, 30, 15, 8]$
$\mathbf{W}$	$[1, 2, 4]$	$[1, 12, 6, 2, 10, 5]$	$[1, 12, 6, 2, 10, 5]$	$[2, 12, 1, 12, 1, 6, 14, 13, 10, 4]$
$\mathbf{E}$	$Random(0, 1)$	$Random(0, 0.5)$	Hand Crafted	$Random(0, 0.4)$
Ψ	9	15	15	48
$\|\mathbf{S}\|$	$\sim 2E + 5$	$\sim 7E + 18$	$\sim 7E + 18$	$\sim 3E + 89$
λ	0	0.1	0.1	0.1

ENV-SMALL is moderately a simple environment, on which we could calculate the optimal policy. This environment is helpful as we could show the policies derived from DNQN are very close to the optimal polices inferred from Dynamic Programming (DP). Next, we approached two mid-size environments, ENV-MID and ENV-MID-E, since we could assess the impact of blocks' effectiveness on the behavior of our agents. Lastly, we trained our agents on ENV-LARGE to show the scalability of our approach on notably large environments.

Hyper Parameters

Neural Network

One of the main challenges that we faced was identifying a *consistent* neural network architecture and hyper parameters that would capture different environment sizes. Due to the complexity of DNQN, a pure "trial and error" process in identifying quality features and proper hyper parameters is not feasible. Therefore, we designed a supervised learning problem, where we fitted a neural network to the reward function on the data gathered from the random interaction of agents with the environment. This process is almost similar to what happens in DNQN before the first update of the target network. When combined with a trial and error process, we decided on a fully connected neural network with 6 layers with the following number of neurons in each layer: $[30, 30, 20, 20, 60, 1]$. We chose the activation function, loss function, optimizer, batch size and learning factor to be RELU, Mean Square Error, Adam, 32 and 0.005, respectively. It is worth mentioning that this architecture have yielded strong results across all environments and experiments.

DNQN

To find proper values for the hyper parameters in DNQN, we run many instances of this expensive algorithm. Eventually, we set the size of the replay buffer to 3200, ψ_0 to 1 and the discount factor β to 0.99. We also found that fixing the number of iterations in the Fictitious Play algorithm to 500 gave us a good balance between accuracy and running time. After generating and solving 10,000 random games, Fictitious Play was found to be approximately 20 times faster and has the accuracy of almost 94%.

Features

We set $\phi(s, a, d)$ to be a function that maps the state s and the action pair a, d to a set of features that is recognizable by the neural network. For the state, N is represented in a way that each entry shows the extent to which a block is effective (i.e., $1 - \frac{n_{t,b}}{W_b}$). The Predictability and Attacker Budget are passed directly as features without any changes. The defender's action is formulated with an integer that indicates the type of action (assignment, removal or no change) together with the utility of the affected target and a vector showing the effectiveness of chosen block b against all attack methods, E_b. Finally, the attacker's action is shown with a one hot vector with the chosen attack method set to 1 in conjunction with a vector resulted from element wise product of v and U.

Training and Stability

After finalizing the features and finding the proper parameters we trained our agents on the described environments. On ENV-SMALL, we trained two pair of agents; first we used DP, similar to [41], to find the optimal policy for the attacker and defender, which took about 10 hours, running in parallel on a machine with 18 $2.2GHz$ CPU cores. Figure VI.1 shows the convergence of average Q-values across all states during the training process. Then, we trained the agents using DNQN; this process took about 32 hours on an IBM PowerPC equipped with Tesla $p100 - sxm2$ GPUs and Intel(R) Xeon(R) $E5 - 2698v4@2.20GHz$ CPUs. Figure VI.2 demonstrates the obtained loss value through the training, which is computed in line 21 of Algorithm 2.
Also, figure VI.3 depicts the obtained loss values during the training process of ENV-MID-E, which lasted for 7 days on the specified IBM machine. It worth mentioning, due to the size of state space it was not feasible to obtain the optimal strategies with dynamic programming in this environment.

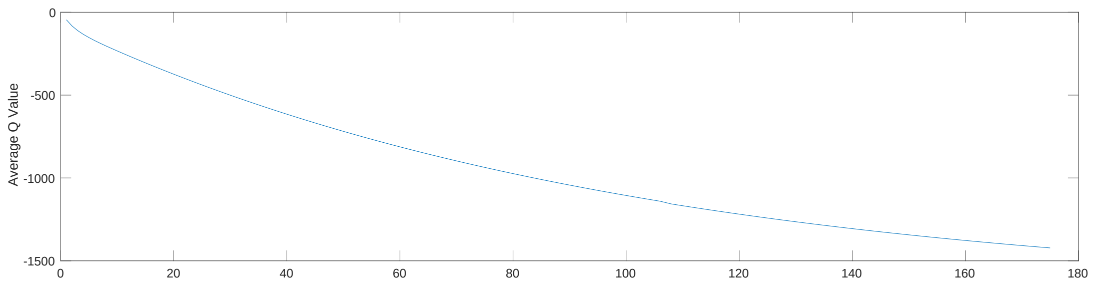

Figure VI.1: Convergence of average Q value in Dynamic Programming method

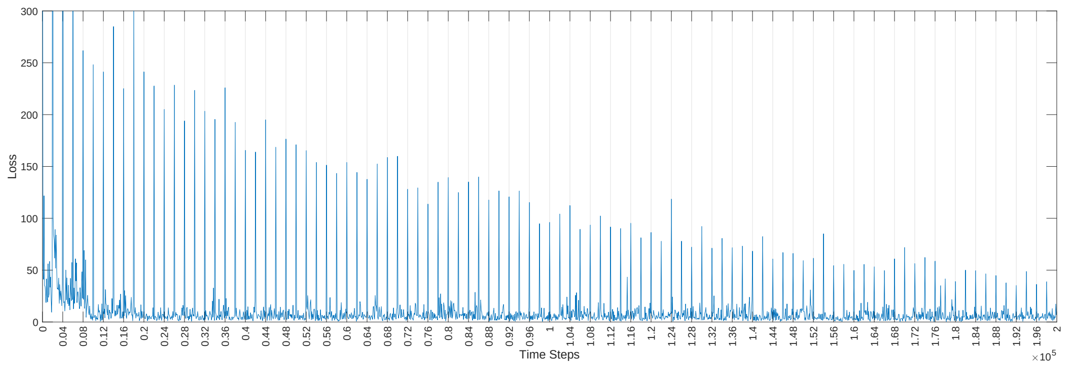

Figure VI.2: Loss value obtained from DNQN training - ENV-SMALL

Main Evaluation

Once the agents were trained, we simulated the system and assessed the behavior of the learned strategies against random and greedy strategies. Random players pick actions at random. The greedy defender picks an action that assigns the block with highest average effectiveness relative to its max warm-up time, $mean(E_b)/W_b$, on the target with the highest utility. The greedy attacker chooses the action with the highest immediate reward.

To compare between policies, we ran 300 simulations with 400 time steps for each policy pair and averaged the achieved cumulative discounted reward, Utility, in every time step. Figures VI.4 and VI.5 show the simulation results on BLOC-SMALL. As we can see in VI.4 DP defender can achieve the highest utility against Greedy, DNQN and DP attackers. This implies that BLOC framework protects the system the most compare to other heuristics. Yet, DP defender is not the best policy against the random attacker. We argue although

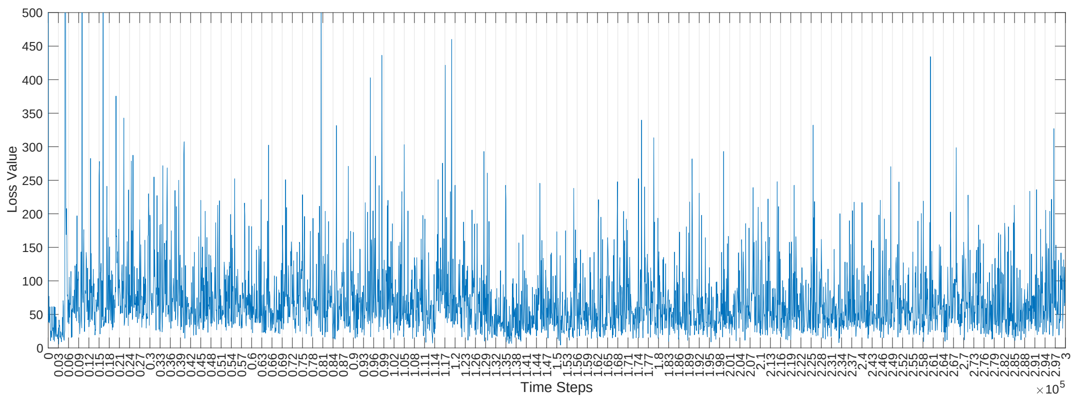

Figure VI.3: Loss value obtained from DNQN training - ENV-MID-E

DP defender can be exploited by some policies, it still achieves the highest worst-case utility among other polices. In VI.4 we compared performance of different attack methods. ENV-SMALL is a simple environment and the attacker task is straightforward, as such the greedy attacker can achieve a comparable utility to the DP attacker. Besides that, it can be observed DNQN agents have a very similar behavior as DP agents, so we can determine that DNQN is a legitimate approximator for DP.

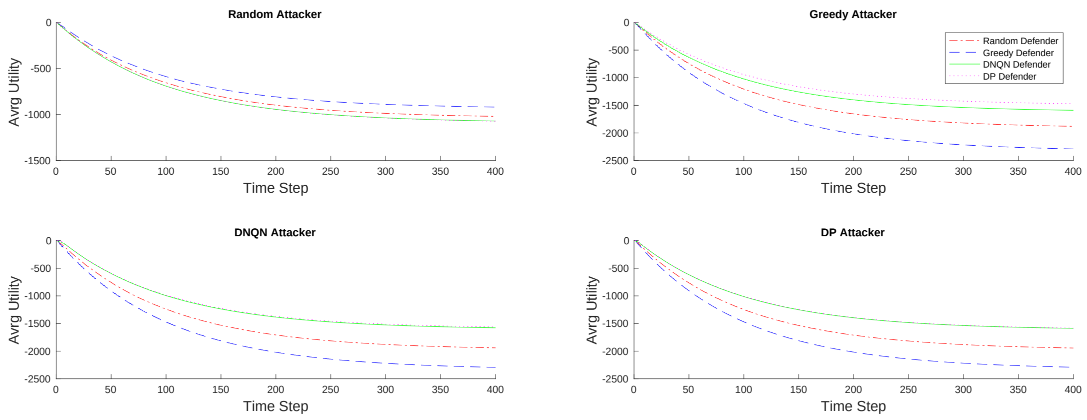

Figure VI.4: Cumulative utilities obtained by each attacker policy against defender policies ENV-SMALL

Likewise, we evaluated the performance of trained agents using DNQN on ENV-MID and ENV-MID-E. As shown in figure VI.6 and VI.7, the attacker trained by DNQN have a considerably better performance in all cases, however

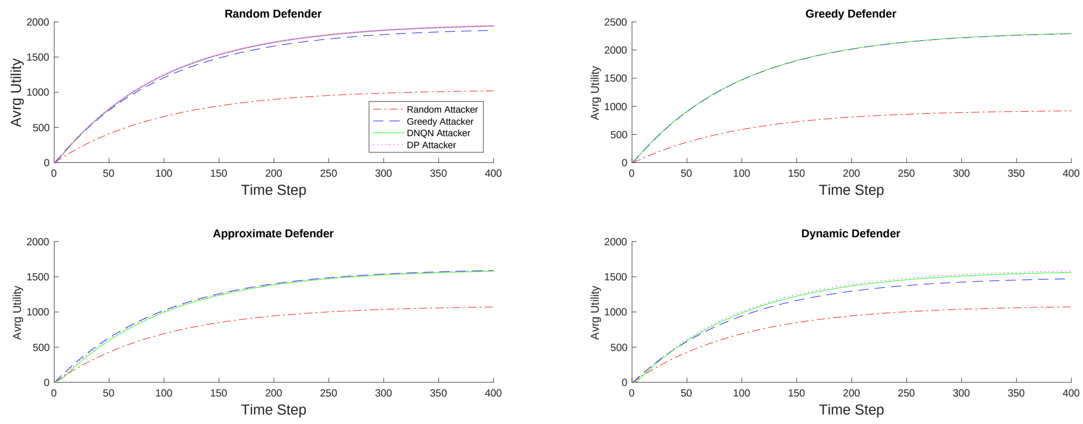

Figure VI.5: Cumulative utilities obtained by each defender policy against attacker policies on ENV-SMALL

DNQN's defender is marginally better. In ENV-MID-E, we handcrafted the effectiveness matrix in order to show how our approach can capitalize on certain patterns in contrast to other strategies. This is because a random policy does not consider E at all; the greedy policy does not incorporate the elements in E and decisions are just based on the average effectiveness of blocks. As shown in figures VI.6 and VI.7 both DNQN agents in ENV-MID-E have a significantly better performance than other heuristics. Unlike BLOC-SMALL, ENV-MID and ENV-MID-E have the uncertainty component, which leads to a particular behavior. Through out the simulations we observed that the DNQN defender does not max out the capacity constraints, as opposed to the greedy defender. This helps DNQN to recover easier from the impact of uncertainty, since when the capacity is reached, block removal is more probable than an assignment. On the other hand, the DNQN attacker usually accumulates her budget and strikes the system right after the occurrence uncertainty in consecutive time steps.
To demonstrate the scalability of our proposed BLOC framework in large environments, we conducted our last experiment on ENV-LARGE. This environment is fairly challenging to operate on, especially for the defender as he needs to consider the uncertainty while keeping his behavior unpredictable on a very large state space. Figures VI.10 and VI.11 show that DNQN achieves the highest utility in all cases. Furthermore, the potential of our methodology in

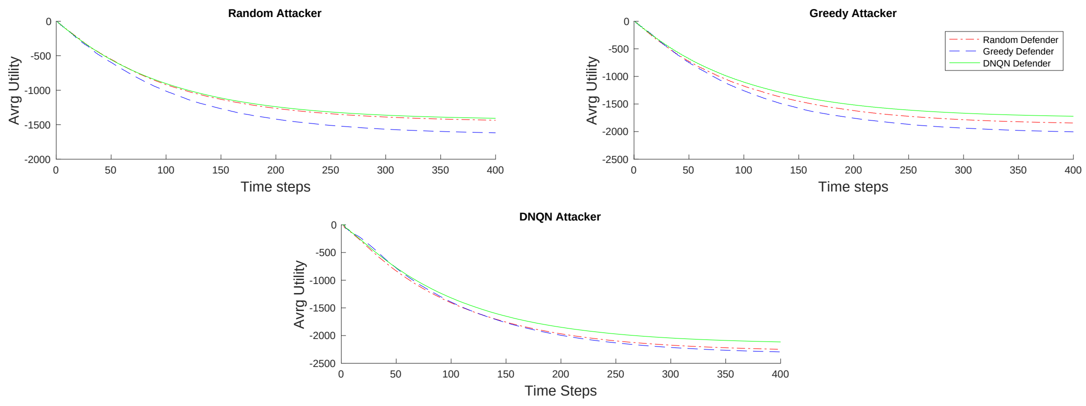

Figure VI.6: Cumulative utilities obtained by each attacker policy against defender policies on ENV-MID

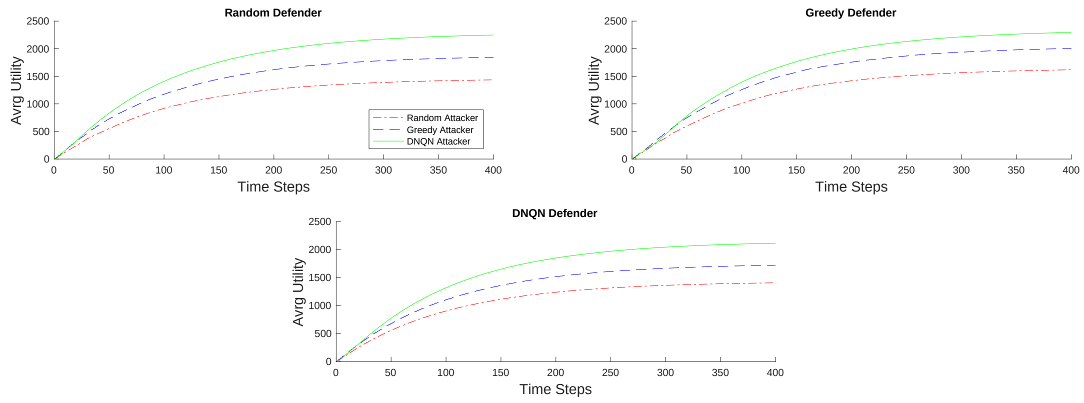

Figure VI.7: Cumulative utilities obtained by each defender policy against attacker policies on ENV-MID

obtaining higher average utility in comparison to other heuristics increases the size of the state space increases.

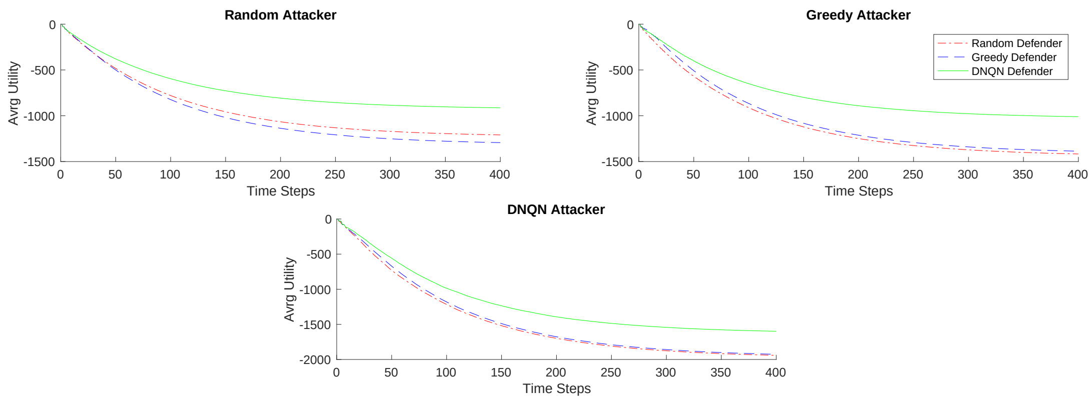

Figure VI.8: Cumulative utilities obtained by each attacker policy against defender policies on ENV-MID-E

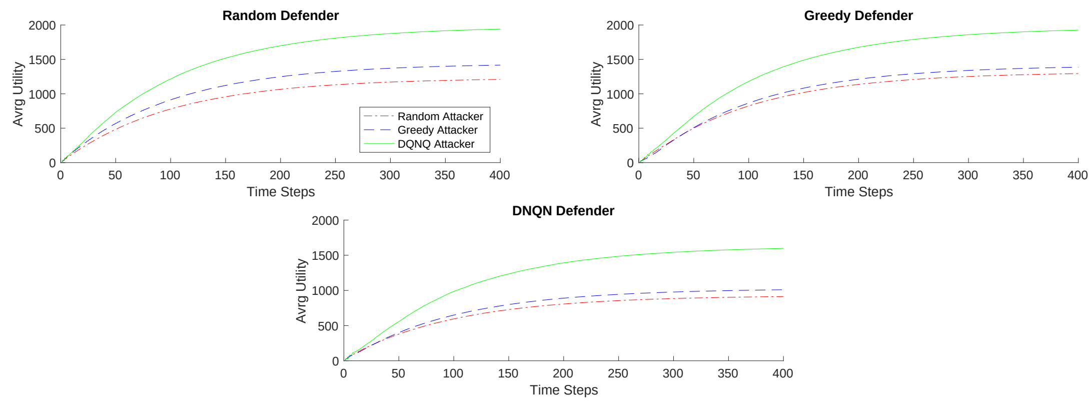

Figure VI.9: Cumulative utilities obtained by each defender policy against attacker policies on ENV-MID-E

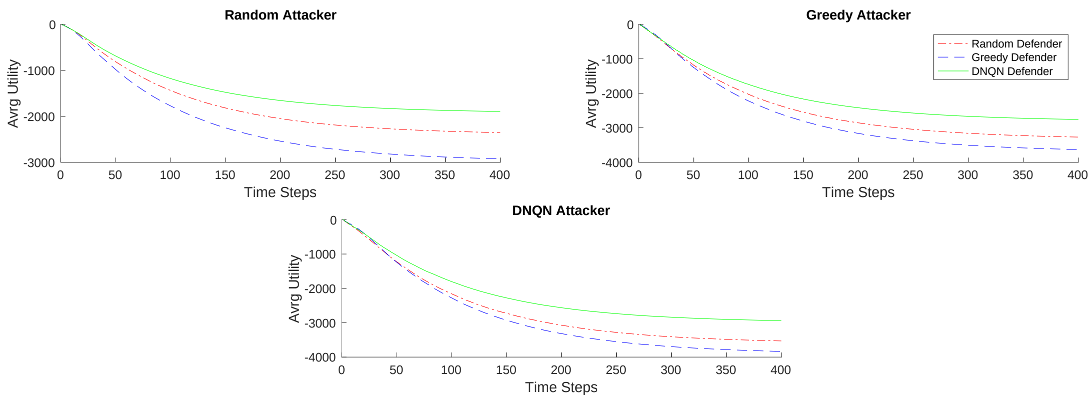

Figure VI.10: Cumulative utilities obtained by each attacker policy against defender policies on ENV-LARGE

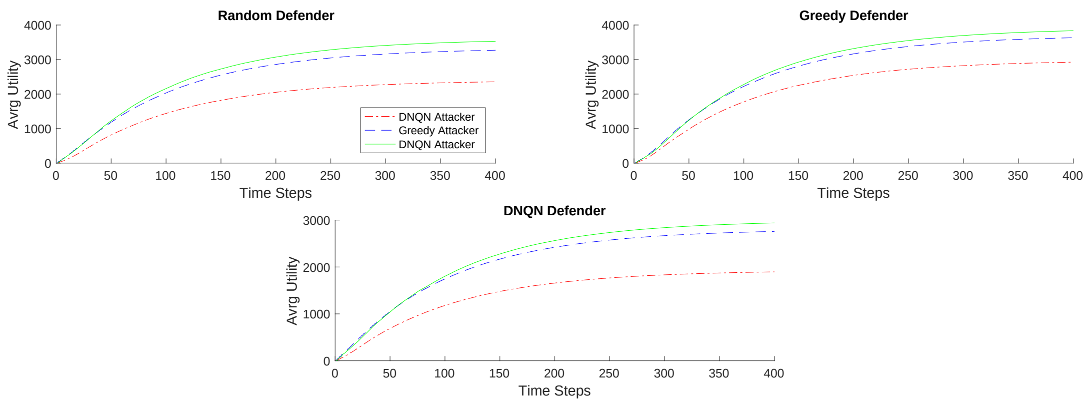

Figure VI.11: Cumulative utilities obtained by each defender policy against attacker policies on ENV-LARGE

VII. CONCLUSION

Cyber attacks on CPS are becoming more sophisticated and the defense mechanisms to check against have not just become largely specific – but increasingly more computationally expensive (e.g., using deep learning). This work presents a coherent framework – BLOC – that orchestrates the CPS at runtime with the proper "check blocks" that are assigned through game-theoretic approaches. Additionally, they are randomized through the mixed-strategy of the defender and operate within the delay bound dictated by the CPS control loop. We developed a Markov game model that captures agent's behavior and the environment characteristics. Later, we derived sub-optimal policies despite the combinatorial nature of the problem, using DNQN. We introduced DNQN as a novel Deep Reinforcement Learning algorithm to operate in adversarial multi-agent environments, where a single reward function suffices all agents. We showed the defender policy obtained from DNQN could protect the system $\%15$ and $\%25$ more than random and greedy approaches respectively, in a case of environment with more than $3E+89$ states.

REFERENCES

[1] J. Finkle, "U.S. official sees more cyber attacks on industrial control systems," 2016.

[2] X. Zheng, C. Julien, M. Kim, and S. Khurshid, "Perceptions on the state of the art in verification and validation in cyber-physical systems," *IEEE Systems Journal*, vol. 11, no. 4, pp. 2614–2627, 2017.

[3] R. Bobba, K. Rogers, Q. Wang, H. Khurana, K. Nahrstedt, and T. Overbye, "Detecting False Data Injection Attacks on DC State Estimation," in *The First Workshop on Secure Control Systems, CPS Week*, 2010.

[4] Y. Liu, P. Ning, and M. Reiter, "False Data Injection Attacks against State Estimation in Electric Power Grids," in *the 18th ACM Conference on Computer and Communications Security*, (Chicago, IL), November 2009.

[5] Y. Mo and B. Sinopoli, "False Data Injection Attacks in Control Systems," in *Proceedings of the 1st workshop on Secure Control Systems*, pp. 1–6, 2010.

[6] A. Tiwari, B. Dutertre, D. Jovanović, T. de Candia, P. Lincoln, J. Rushby, D. Sadigh, and S. Seshia, "Safety Envelope for Security," in *Proceedings of the 3rd international conference on High confidence networked systems*, pp. 85–94, ACM, 2014.

[7] N. Trcka, M. Moulin, S. Bopardikar, and A. Speranzon, "A Formal Verification Approach to Revealing Stealth Attacks on Networked Control Systems," in *Proceedings of the 3rd International Conference on High Confidence Networked Systems*, (Chicago, IL), April 2014.

[8] Y. Wang, Z. Xu, J. Zhang, L. Xu, H. Wang, and G. Gu, "SRID: State Relation Based Intrusion Detection for False Data Injection Attacks in SCADA," in *Computer Security-ESORICS*, Springer, 2014.

[9] J. Hespanha, P. Naghshtabrizi, and Y. Xu, "A Survey of Recent Results in Networked Control Systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 138–162, 2007.

[10] L. Schenato, B. Sinopoli, M. Franceschetti, K. Poolla, and S. Sastry, "Foundations of Control and Estimation Over Lossy Networks," *IEEE*, vol. 95, no. 1, p. 163, 2007.

[11] T. Yang, "Networked Control System: A Brief Survey," *IEE Proceedings Control Theory and Applications*, vol. 153, no. 4, pp. 403–412, 2006.

[12] S. Amin, A. Cárdenas, and S. Sastry, "Safe and Secure Networked Control Systems under Denial-of-Service Attacks," *Hybrid Systems: Computation and Control*, pp. 31–45, 2009.

[13] K. Pelechrinis, M. Iliofotou, and V. Krishnamurthy, "Denial of Service Attacks in Wireless Networks: The Case of Jammers," *IEEE Communications Surveys & Tutorials*, vol. 13, no. 2, 2011.

[14] M. Wilhelm, I. Martinovic, J. Schmitt, and V. Lenders, "Short Paper: Reactive Jamming in Wireless Networks: How Realistic is the Threat?," in *ACM conference on Wireless network security*, 2011.

[15] D. I. Urbina, J. A. Giraldo, A. A. Cardenas, N. O. Tippenhauer, J. Valente, M. Faisal, J. Ruths, R. Candell, and H. Sandberg, "Limiting the impact of stealthy attacks on industrial control systems," in *ACM CCS*, 2016.

[16] A. Ferdowsi, W. Saad, B. Maham, and N. B. Mandayam, "A colonel blotto game for interdependence-aware cyber-physical systems security in smart cities," in *Proceedings of the 2nd International Workshop on Science of Smart City Operations and Platforms Engineering*, pp. 7–12, ACM, 2017.

[17] Y. Li, L. Shi, P. Cheng, J. Chen, and D. E. Quevedo, "Jamming attacks on remote state estimation in cyber-physical systems: A game-theoretic approach," *IEEE Transactions on Automatic Control*, vol. 60, no. 10, pp. 2831–2836, 2015.

[18] H. Orojloo and M. A. Azgomi, "A game-theoretic approach to model and quantify the security of cyber-physical systems," *Computers in Industry*, vol. 88, pp. 44–57, 2017.

[19] M. Zhu and S. Martinez, "Stackelberg-game analysis of correlated attacks in cyber-physical systems," in *American Control Conference (ACC), 2011*, pp. 4063–4068, IEEE, 2011.

[20] Q. Zhu and T. Basar, "Game-theoretic methods for robustness, security, and resilience of cyberphysical control systems: Games-in-games principle for optimal cross-layer resilient control systems," *IEEE control systems*, vol. 35, no. 1, pp. 46–65, 2015.

[21] A. S. M. T. M. Brown, A. Sinha, "One size does not fit all: A game-theoretic approach for dynamically and effectively screening for threats," *AAAI conference*, 2016.

[22] Z. Yin, A. Jiang, M. Tambe, C. Kiekintveld, K. Leyton-Brown, T. Sandholm, and J. Sullivan, "Trusts: Scheduling randomized patrols for fare inspection in transit systems using game theory," in *Proceedings of the 24th IAAI*, (Palo Alto, CA), 2012.

[23] D. K. M. B. M. T. A. Sinha, T. Nguyen and A. Jiang, "From physical security to cybersecurity," *Journal of Cybersecurity*, 2015.

[24] A. Schlenker, H. Xu, M. Guirguis, M. Tambe, A. Sinha, C. Kiekintveld, S. Sonya, N. Dunstatter, and D. Balderas, "Towards a game-theoretic framework for intelligent cyber-security alert allocation," in *Proceedings of the 3rd IJCAI workshop on Algorithmic Game Theory, Melbourne, Australia*, 2017.

[25] A. Schlenker, H. Xu, M. Guirguis, M. Tambe, A. Sinha, C. Kiekintveld, S. Sonya, N. Dunstatter, and D. Balderas, "Don't bury your head in warnings: A game-theoretic approach for intelligent allocation of

cyber-security alerts," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pp. 381–387, 2017.

[26] N. Dunstatter, M. Guirguis, and A. Tahsini, "Allocating security analysts to cyber alerts using markov games," 2018.

[27] J. Heinrich, M. Lanctot, and D. Silver, "Fictitious self-play in extensive-form games," in *International Conference on Machine Learning*, pp. 805–813, 2015.

[28] J. Heinrich and D. Silver, "Deep reinforcement learning from self-play in imperfect-information games," *arXiv preprint arXiv:1603.01121*, 2016.

[29] M. Lanctot, V. Zambaldi, A. Gruslys, A. Lazaridou, K. Tuyls, J. Pérolat, D. Silver, and T. Graepel, "A unified game-theoretic approach to multiagent reinforcement learning," in *Advances in Neural Information Processing Systems*, pp. 4190–4203, 2017.

[30] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Machine Learning Proceedings 1994*, pp. 157–163, Elsevier, 1994.

[31] M. L. Littman, "Value-function reinforcement learning in markov games," *Cognitive Systems Research*, vol. 2, no. 1, pp. 55–66, 2001.

[32] A. Wood and B. Wollenberg, *Power Generation, Operation, and Control.* John Wiley & Sons, 2012.

[33] R. Chabukswar, "Secure Detection in Cyberphysical Control Systems," *Ph.D. Thesis – CMU*, 2014.

[34] E. Camacho and C. Alba, *Model Predictive Control.* Springer Science & Business Media, 2013.

[35] D. Mayne, J. Rawlings, C. Rao, and P. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, 2000.

[36] J. Hu and M. P. Wellman, "Nash q-learning for general-sum stochastic games," *Journal of machine learning research*, vol. 4, no. Nov, pp. 1039–1069, 2003.

[37] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[38] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[39] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[40] G. W. Brown, “Iterative solution of games by fictitious play,” in *Activity Analysis of Production and Allocation* (T. C. Koopmans, ed.), New York: Wiley, 1951.

[41] M. Guirguis, A. Tahsini, K. Siddique, C. Novoa, J. Moore, C. Julien, and N. Dunstatter, “Bloc: A game-theoretic approach to orchestrate cps against cyber attacks,” in *2018 IEEE Conference on Communications and Network Security (CNS)*, pp. 1–9, 2018.