A Deep Learning Game Theoretic Model for Defending Against Large Scale Smart Grid Attacks

James D. Cunningham, Alexander Aved, *Senior Member, IEEE*, David Ferris, Philip Morrone, and Conrad S. Tucker

***Abstract*—Power grids that are interdependent with communication networks create more possible modes of failure (e.g., cyberattacks) as well as more complex propagation of failure through the coupled networks. To ensure robust defense of smart grids, it is important to model both attacker and defender as intelligent, a scenario that the framework of game theory provides methods to analyze. However, prior works in applying game theoretic models to smart grid security limit the problem space to a small number targets under threat due to the inability of state-of-the-art methods to scale to large networks. Our method scales to large networks by combining neural networks that use featurized action representations with an approximation of large combinatorial actions to generalize knowledge about the best targets to attack/defend across graphs of various topologies and sizes. Our model's invariance to the size of the input graph allows us to transfer knowledge from games played on small graphs during training to large graphs during evaluation. Our experiments show that our method can learn Nash equilibrium strategies on small networks, and demonstrate low exploitability when generalized to large networks, especially compared to the common heuristics currently used to simulate attacks on large graphs.**

***Index Terms*—Deep learning, game theory, cybersecurity, smart grid security.**

I. Introduction

THE MODERNIZATION of electric grids are bringing increased efficiency and new functionalities to consumers via two-way communication between various components of the network [1]. However, by adding more sophisticated communication to the grid, numerous cybersecurity vulnerabilities are introduced [2], [3], [4]. Moreover, the propagation of failure through the coupled grids becomes more complex, as failing communication nodes have an impact on surrounding power grid nodes and vice versa [5]. These vulnerabilities can create cascading failures through a grid, and were exploited in the 2015 cyber attack on the Ukrainian grid that left a quarter million without power [6]. This demonstrates a need for methods that can analyze the complex topologies of these systems and determine an effective strategy to distribute limited defense resources across many possible adversarial targets.

Many works in modeling cascading failures study the effects of, and mitigation strategies for, a set of initiating element failures such as power lines or substations [5], [7], [8], [9]. When studying cascading failures, these works consider a maximum initiating failure size of anywhere from 10% to 35% of grid elements out of hundreds or thousands. These targets are selected either randomly or via a deterministic heuristic [5], [7]. However, an adversary will usually not behave randomly, but instead seek to cause maximum damage given their limited resources [10]. On the other hand, the deterministic heuristics assume that the attacker will only ever attack the targets that have the largest impact in terms of cascading failure, making these strategies very predictable. However, if an attack strategy is very predictable, an intelligent defender would take extra measures to defend this target, typically in the form of human resources such as physical security or active monitoring [10], [11], [12]. In this case, the attacker must balance the importance of the targets chosen with the unpredictability of their strategy.

The game theoretic framework of a security game accounts for both the value and predictability aspects of the scenario by assuming that an attacker and defender simultaneously make decisions about targets to protect or attack. The attacker maximizes its payoff by choosing the highest impact target that the defender leaves unprotected. The defender has the opposite incentive, to protect the target the attacker will choose to attack, especially high impact targets. Consider the example of a security game in which there are n targets that an adversary may choose to attack and that a defender may choose to protect, and the payoffs for the attacker and defender are determined by the target selected by each player. Also assume that each player may play a mixed strategy that defines a probability distribution over the n possible targets. This is a

Manuscript received 5 December 2021; revised 29 March 2022 and 27 June 2022; accepted 10 August 2022. Date of publication 17 August 2022; date of current version 20 February 2023. This work was supported in part by the Griffith Institute and Air Force Research Lab under Contract SA1003202012030. Paper no. TSG-01928-2021. *(Corresponding author: Conrad S. Tucker.)*

James D. Cunningham is with the Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: jamescun@andrew.cmu.edu).

Alexander Aved, David Ferris, and Philip Morrone are with the Information Exploitation Division, Air Force Research Laboratory, Rome, NY 13441 USA (e-mail: alexander.aved@us.af.mil; david.ferris.3@us.af.mil; phili.morrone.6@us.af.mil).

Conrad S. Tucker is with the Department of Mechanical Engineering and the Machine Learning, Robotics, and Biomedical Engineering Departments, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: conradt@andrew.cmu.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TSG.2022.3199187.

Digital Object Identifier 10.1109/TSG.2022.3199187

normal form game with a finite number of possible actions, thus there must exist a Nash equilibrium of mixed strategies for each player such that neither player has an incentive to deviate from their strategy [13]. In this sense, these strategies belonging to a Nash equilibrium can be considered the optimal strategies for each player. Calculating this Nash equilibrium directly is possible given that the payoff matrix is known for all combinations of attack and defense actions.

There exist prior works that have modeled smart grid defense scenarios as a two player security game, and either directly solve for the Nash equilibrium or apply tabular Reinforcement Learning (RL) to learn approximate optimal strategies [10], [11], [12], [14], [15]. However, these methods are only applied at the scale of selecting a single target at a time from a set of no more than 50. This is due to the fact that the methods used in these works do not scale to large networks. Consider for example a security game with a 100-node network. Let us assume that an adversary can attack 5 nodes on this graph simultaneously, and that a defender can likewise protect 5 nodes on the graph. Once the node selections have been made, a cascading failure simulator is used to determine the payoffs for the attacker and defender. In this case, the attacker and defender each have $\binom{100}{5} \approx 7.5 \times 10^7$ possible combinations of 5 nodes they could choose, each resulting in a potentially different payoff for each player. The corresponding payoff matrix would be of size $(7.5 \times 10^7)^2 \approx 5.7 \times 10^{15}$, making tabular approaches to finding optimal strategies intractable. In this work, we generate security game strategies for the attack and defense of 50 nodes in a coupled power and communication grid model based off of the topology of the Polish power grid with 2383 nodes.

Towards learning to play a security game on a large-scale graph, this work makes the following contributions:

- We propose a neural network model that is invariant to the size of the input graph to predict the payoff for a set of attacker and defender target selections.
- We also introduce the Reduced Combinatorial Rollout (RCR) sampling trick for RL to approximate a large combinatorial action as a set of smaller combinatorial actions.
- We demonstrate through experiments that by combining the first two contributions, effective security game strategies can be generated for large-scale coupled power and communication network test cases, including a topology based on the Polish grid.

The rest of this paper is organized as follows: Section II reviews the most relevant literature to this problem and Section III presents the details of the two-player game scenario and method for learning it. Section IV details the simulation model used and the experimental results. Finally, Section V gives concluding remarks and possible directions of future work.

II. Literature Review

A. Coupled Power and Communication Network Models

As communication and power networks have trended towards interdependence, there have been several works that have investigated modeling failure propagation between them. Sturaro *et al.* [7] introduce a heterogeneous model of cascading failure in interconnected communication and power grids that captures some of the complex and realistic interdependencies between the two network domains on networks as large as 1022 nodes. They show that under both random attacks and targeted attacks (based on a node degree heuristic) of up to 15% of nodes that simpler models tend to underestimate the cascading effect of failures in the coupled network.

Cai *et al.* [8] study the effects of different coupled network topologies on cascading failure, and find that a "double-star" structure is the most resilient. Korkali *et al.* [5] study the effects that the degree of coupling between the two network domains has on the propagation of failure through the coupled network. They find that for a topology based on the Polish power grid with 2383 nodes under a set of random initial node failures of up to 35%, a higher degree of coupling improves resilience to cascading failure on a realistic network topology based on the Polish power grid nodes. Cordova-Garcia *et al.* [9] propose a coupled network model that accounts for communication delay between nodes, which affect both the collection of information and the timing of control signals sent to generators. They also provide a control algorithm that accounts for this delay to perform optimal load shedding to mitigate the cascading failure after a random 10% of power lines initially fail in a network with a maximum of 186 lines.

While these works provide quality models for the coupled power and communication networks at realistic scale, they all lack a model of an intelligent attacker with respect to a defender who can take preventative measures, and instead test against random node failures or a deterministic heuristic such as attacking nodes with the highest degree. In this work, we introduce a method to apply game theoretic models to a coupled power and communication network simulation of realistic scale.

B. Security Games Applied to Smart Grids

The modeling of intelligent adversarial agents is the domain of game theory, and the scenario of smart grid attacks are well modeled as a security game. Yan *et al.* [14] use tabular Q-learning to learn a sequential topology attack on a power grid. By using this approach, critical sequences to create cascading failures were identified. However, this work only models the adversary, and not the defender side of the two player game.

Wei *et al.* [11] model both the attacker and defender of a power grid in a two-player game and solve for a Nash equilibrium. However, on a test case with 177 targets in which each player can select 2 (a natural action space of $\approx 15,000$), the authors assume only 40 of these actions need to be considered based on the structure of the network. Each of these 40 attack actions are simulated in a cascading failure simulator with a built in load shedding controller, and only the 9 actions that give positive utility to the attacker are used in the final security game model. In these ways, this work circumvents the large combinatorial action space problem by manually pruning actions until the action space is small enough for RL.

TABLE I
COMPARISON OF THE FEATURES OF THIS WORK TO THOSE OF THE MOST CLOSELY RELATED LITERATURE

Features	Wei 2018 [20]	Ni 2019 [21]	Paul 2020 [10]	Guo 2021 [12]	Korkali 2017 [5]	This Work
Realistic Scale Network				✓	✓	✓
Game Theoretic Adversary	✓	✓	✓	✓		✓
Generalizes Across Topologies					✓	✓
Sequential Decision Making		✓	✓	✓		

Paul *et al.* [10] and Guo *et al.* [12] both implement a Minimax Q learning approach to find the Nash equilibrium strategies for a graph with a total of 46 targets of which each player can select a single target at a time. Both of these works also manually prune the action space from 46 for each player down to 10 and 6 respectively. Minimax Q learning is a well-known method for finding Nash equilibriums in two-player zero sum games that are too large to directly solve for the Nash equilibrium. Nonetheless, it still suffers from an inability to scale to large action spaces and, as these works demonstrate, requires manual pruning of the action space for large problems.

Each of the aforementioned methods use tabular RL, in which the payoff associated with each combination of state and player actions is stored in a table and updated from experiences of interaction with the environment. A drawback of this approach is that it not possible to generalize to arbitrary graph topologies. This is because it is impossible to create a table large enough to store payoffs for every possible graph topology for even small scale problems. For example, a 10 node graph with homogeneous undirected edges and no loops (edges from a node to itself) has 2^{45} possible discrete topologies. Assuming each value in the Q-table holds a 4 byte floating point value, the table would require at least 140 TB of memory. Even worse, this memory requirement scales exponentially with the size of the network. As this example demonstrates, tabular methods are severely limited by state space complexity due to their reliance on discrete states.

However, outside of the smart grid domain, Deep RL (DRL) methods that leverage neural networks have demonstrated success in domains of multi-player games with very large state spaces, including the board games Go, Chess, and Shogie [16], [17], as well as competitive video games such as DOTA2 [18] and StarCraft2 [19]. These approaches use featurized state representations and neural networks to automatically learn to generalize knowledge about good action selections from states encountered during training to unseen states. While DRL has proven effective in domains with large state spaces, large action spaces still pose a challenge. This is due to the fact that an agent cannot infer information about the performance of an action it hasn't seen from actions it has seen the same way it does with states, and instead needs to execute all of its actions many times in different states to learn an effective strategy. For reference, the game of Go has at most 361 discrete actions available on a given turn.

We bridge this gap in the setting of combinatorial action spaces over graphs by featurizing both the state space and action space using continuous node embeddings in such a manner that they are invariant to the size of the network. We train this neural network model on small graphs with an action space of tractable size, and then evaluate the trained model on large graphs. The key assumption of this approach is that the features of valuable targets in small networks are similar to the features of valuable targets in large networks.

One aspect that is present in most of the referenced works that we choose not to model is sequential decision making. Depending on the scenario of interest, it can be reasonable to model the security game as playing out over multiple time steps or a single time step. Of the four works reviewed in Section II-A, only Cai *et al.* [8] chose to model the outages as occurring sequentially, with the rest choosing to model the outages as occurring simultaneously. It should be noted that one could be motivated to model the decisions over multiple time steps precisely because of the technical limitations of modeling large combinatorial action spaces. For example, having an action space of size N with a time horizon of 10 could be implemented with standard RL, whereas an action space size of $\binom{N}{10}$ at a single time step cannot. Because this action space size limitation relates to the number of targets that can be selected by an agent simultaneously, we choose focus on a one-step game in this work to address that limitation in the simplest setting, and leave extensions to sequential decision domains to future work.

Table I summarizes the advantages of this work relative to other works reviewed in this section.

III. METHOD

We consider the game theoretic setting of a finite normal form two-player zero-sum game, where player 1 has action space $\mathbf{A}_1$, player 2 has action space $\mathbf{A}_2$, and the players share the state space $\mathbf{S}$. For a normal form game, we can define the utility function $Q_1(\mathbf{s}, a_1, a_2), \forall \mathbf{s} \in \mathbf{S}, a_1 \in \mathbf{A}_1, a_2 \in \mathbf{A}_2$ as the payoff for player 1 given any state and pair of player actions. For any finite normal form game, at least one Nash equilibrium must exist [13]. Because this is a zero-sum game, the payoff for player 2 is $-Q_1(\mathbf{s}, a_1, a_2)$. Thus, we will use Q as shorthand for both player's utility functions.

Each agent $i \in \{1, 2\}$ follows some mixed strategy $\pi_i(\mathbf{s}) \in \nabla(\mathbf{A}_i)$, where $\nabla(\mathbf{A}_i)$ is the space of probability distributions over the action space $\mathbf{A}_i$, and $\pi_i(\mathbf{s}, a_i)$ denotes the probability of choosing action a_i under strategy π_i. The Minimax Q-learning algorithm derives a mixed strategy π_i via the following equation:

$$\pi_i(\mathbf{s}) = \max_{\pi_i'(\mathbf{s}) \in \nabla(\mathbf{A}_i)} \min_{a_{-i} \in \mathbf{A}_{-i}} \sum_{a_i \in \mathbf{A}_i} \pi_i'(\mathbf{s}, a_i) Q_i(\mathbf{s}, a_i, a_{-i}) \quad (1)$$

If both players in a two-player zero-sum game know Q and play their mixed strategies as defined by Equation (1), they must be at a Nash Equilibrium [22]. Furthermore, Littman and Szepesvári [23] proved that Minimax Q-learning is guaranteed to converge to the true utility function Q given sufficient

exploration. Equation (1) can be solved via the linear program:

$$\begin{aligned} &\max_{\pi_i'(\mathbf{s}),\omega} \ \omega \\ &\omega \leq \sum_{a_i \in A_i} \pi_i'(\mathbf{s}, a_i) Q_i(\mathbf{s}, a_i, a_{-i}), \forall a_{-i} \in \mathbf{A}_{-i} \\ &\sum_{a_i \in \mathbf{A}_i} \pi_i'(\mathbf{s}, a_i) = 1 \\ &\pi_i'(\mathbf{s}, a_i) \geq 0, \forall a_i \in \mathbf{A}_i \end{aligned} \tag{2}$$

where ω is a dummy variable that allows the minimization term in Equation 1 to be encoded as an inequality constraint in the linear program [13].

As discussed in Section II, when the state and action spaces are small and discrete, tabular RL methods can be used to learn Q via interactions with the game environment. However, for very large and/or continuous state or action spaces, tabular methods fail. To overcome these limitations, we propose learning Q using a neural network with special featurization of the graph-based representation of the state and action space motivated by the smart-grid defense application. This featurization is invariant to the size of the graph, and thus can be used to generalize from small graphs to large graphs, reducing the computational complexity of training. We also propose a sampling trick that allows us to approximate a large combinatorial action with a sequence of smaller combinatorial actions.

A. Neural Network Q Model

Inspired by Dai *et al.*'s [24] method of using deep RL to learn combinatorial optimization problems over graphs, we propose a neural network model for Q in our two-player game setting that uses a similar architecture, but adapted for the multi-agent setting. This method is specific to graph-based environments, in which the task can be formulated as choosing the best node(s) in the graph based on the the overall graph topology.

For a graph G with nodes V and edges E, let us define a generic node embedding function

$$\mu_v = f(v), \forall v \in V \tag{3}$$

where $\mu_v \in \mathbb{R}^p$, and p is the number of node features. We can then summarize the state of the entire graph by taking the mean over all embedded nodes $\frac{1}{|V|}\sum_{u\in V}\mu_u$. For example, if we take a simple case where the only node feature considered is the degree (number of edges) of the node, then we would have $p = 1$, μ_v would be the degree of v, and the state of the graph would be the mean of the degrees of all nodes in the graph. Note that the dimensionality of this state representation is $p = 1$ as well, regardless of the size of the graph. For example, consider the case where the average node degree in the entire graph is $\mathbf{s} = 2.5$, the attacker plans on attacking a node with a degree of $\mathbf{a_1} = 2$, and the defender plans on defending a node with a degree of $\mathbf{a_2} = 3$. In this example, the neural network model can be thought of as receiving the following query: "For a graph with an average node degree of 2.5, an attacked node with degree 2, and a defended node with degree 3, what percentage of nodes in the grid will fail after cascading?" In our experiments, we use multiple node features beyond just node degree.

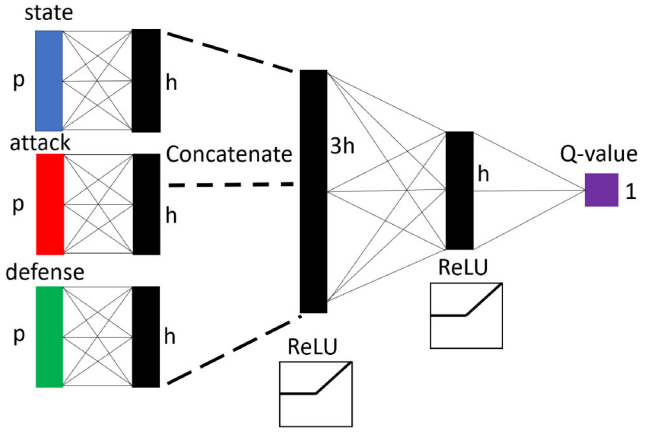

Fig. 1. The neural network model used to represent Q in our proposed method. Takes the featurized graph state, and the node embeddings of the attacker's and defender's node choices as input and returns the Q-value as output. Dimensions of each component of the architecture are shown on the figure, where p is the number of node features and h is the hidden layer size.

Considering a scenario where an attacker attacks node v, and defender defends node w, we can parameterize Q as follows:

$$Q_\theta(s, v, w) = \theta_1^T \text{ReLU}\left(\theta_2^T \text{ReLU}\left(\left[\theta_3^T \sum_{u\in V}\mu_u, \theta_4^T \mu_v, \theta_5^T \mu_w\right]\right)\right) \tag{4}$$

where

- $\theta_1 \in \mathbb{R}^h$
- $\theta_2 \in \mathbb{R}^{3h\times h}$
- $\theta_3, \theta_4, \theta_5 \in \mathbb{R}^{p\times h}$
- $[\cdot,\cdot]$ represents concatenation.

Equation (4) can be represented via the neural network architecture shown in Figure 1. The first hidden layer of this architecture learns features relevant to how the state and each agent's actions relate individually to the Q-value, while the second hidden layer learns how the relationship between the state and agent actions impacts the Q-value. Because we are considering one-step normal form games, we treat learning the parameters θ as an online supervised learning problem with respect to the learning experience tuples $[\mathbf{s}, a_1, a_2, r]$ consisting of the state and actions of each player, and reward generated by the environment. We use the Huber ℓ-1 loss function defined as:

$$L(Q_\theta(\mathbf{s}, a_1, a_2), r) = \begin{cases} 1/2(Q_\theta - r)^2, & |Q_\theta - r| < 1 \\ |Q_\theta - r|, & \text{otherwise} \end{cases} \tag{5}$$

to update the parameters θ via gradient ascent.

Note that the size of the neural network model only depends on of the number of node features p and the hidden layer size h, which are design choices, and not the size of the graph. However, in order to get the optimal policy π from the linear program (2), we still must evaluate Q_θ with every combination of attack and defense actions. In a large combinatorial action space, this can still be prohibitive. To overcome this limitation we propose an approximation of the combination of actions as a set of smaller combinations of actions that share credit for the result.

TABLE II
ILLUSTRATIVE EXAMPLE OF THE DIFFERENCE BETWEEN RCR WITH $m = 1$ THE FULL ACTION SPACE $m = 2$

$m = 1$	Node 2	Node 5	Node 7
Q_0	0	0	0
Q_1	0.5	0.5	0
Q_2	0.35	0.5	0.2

$m = 2$	Nodes 2&5	Nodes 2&7
Q_0	0	0
Q_1	0.5	0
Q_2	0.5	0.2

B. Reduced Combinatorial Rollout

In the standard RL paradigm, learning experiences are collected by sampling an action from the strategy π and passing the chosen action to the environment, which then generates the reward. The learning experience consisting of the state, action selected, and reward are stored in an experience buffer. Periodically the neural network training algorithm samples experiences from this buffer to perform an optimization step.

We propose the Reduced Combinatorial Rollout (RCR) method for RL that allows for the selection of k nodes from a graph with $|V|$ nodes with an action space that is smaller than $\binom{|V|}{k}$. Let $\binom{|V|}{m}$ be the size of the reduced action space such that $m < k$. RCR changes the paradigm of sampling only a single action from the strategy on a given time step to sampling k/m times without replacement. This generates k/m experiences, as opposed to a single experience, that are all stored in the experience buffer with the same reward. In this way, all actions share equal credit for their collective performance, and over a diverse set of node selections the best performing actions become more represented in experiences with higher rewards.

When $m = 1$, each node is considered a separate action and thus can only be evaluated individually. This means that in the best case scenario, the agent successfully learns to choose the top n nodes that each individually contribute the most its utility. However, by increasing m to 2, the agent's individual actions correspond to selecting pairs of nodes until $k/2$ unique pairs have been chosen. This allows the agent to learn higher order effects that occur from two nodes that may be dependent on each other to cause cascading, but when paired with other nodes are less impactful.

As an illustrative example, consider a graph with 10 nodes where the learning agent is tasked with selecting two nodes on the graph to attack/defend. Suppose that in the first episode, the agent selects nodes 2 and 5, and receives a reward of 0.5. Second, it selects nodes 2 and 7 and receives a reward of 0.2. Table II shows how a tabular representation of Q would update its values with RCR with $m = 1$ compared to the full action space ($m = 2$). While action space with $m = 1$ is reduced from $\binom{10}{2}$ to 10, we can see that the Q-value of node 2 is ultimately the average of its performance with node 5 and node 7. Whereas with $m = 2$ it can represent the performance of node 2 paired with node 5 separately from its performance with node 7 for the trade-off of a larger action space of $\binom{10}{2}$.

While this example was specifically constructed to show the disparity that can arise from not considering higher order effects between combinatorial actions, if node 2 performed similarity regardless of what other node it was paired with, then the cost of reducing the action space size is small. Thus the choice of m should match the degree to which the reward (as dictated by the simulation environment) is nonlinear with respect to combinations of actions.

This idea can be extended to higher values of m, however the larger m is the more the original problem curse of dimensionality returns. A systematic method for identifying the optimal choice of m is left for future work. In this work we limit to considering $m = 1$ and $m = 2$.

The main trade-off of RCR comes from the approximation error of representing an $\binom{n}{k}$ problem as a set of smaller $\binom{n}{m}$ problems. While unlike manual pruning of possible targets, RCR allows for any of combination of nodes in the network to be selected with some probability, there are some mixed strategies over combinations of nodes in the larger combinatorial action space that cannot be represented as sequences of the smaller. However, we expect that this approximation error will contribute a small enough portion of the overall error to the Nash equilibrium strategy for large problems that it is worth the savings in computational complexity. We evaluate this claim in Section IV.

Another drawback of this approach comes from the need to store k/m experiences per time step as opposed to only one. This trade-off creates the need for either a larger buffer size (which may be constrained by available computer memory), or results in an experience buffer that fills more quickly with more temporally correlated experiences. Minh *et al.* [25] demonstrated that these correlations can decrease training stability. However, the memory requirements scale only linearly with k, whereas the action space complexity in the original paradigm scales exponentially with k, making this a favorable trade-off for problems with large k. If computer memory is a hard constraint, a technique such as reservoir sampling [26] could be used to reduce correlations in the experience buffer.

IV. APPLICATION

This section details the experiments that were conducted to evaluate the method we propose. First, we describe the coupled power and communication network environment that determines the payoffs for the security game. Then, we describe the specific experiments conducted and analyze the results.

A. Coupled Networks Environment

The coupled network simulation model used in this work is adapted from the open-source model by Korkali *et al.* [5]. Key features and modifications of this model will be presented in this section, however the reader is referred to that work for more details on the coupled network model.

Consider a coupled network model represented as a graph $G(V, E)$ with nodes V and edges E. Let us define subgraphs $G_c(V_c, E_c)$ and $G_p(V_p, E_p)$ which are the nodes and edges of G that exist entirely within the communication and power domains respectively. All nodes belong to either V_c or V_p, however there exist edges that connect nodes in V_c to nodes in V_p. While Korkali *et al.* investigate a few different models of

failure propagation between network domains, we focus on the "intermediate" model of dependency that was concluded to be the most realistic. This model is referred to as "intermediate" because it strikes a balance between assuming that a failure in a power node will always cause a failure in a connected communication node and assuming that the same failure will never cause a communication failure (due to for example a backup battery in the communication node). Instead, the intermediate model assumes there is some probability that a power failure causes a communication failure (that the battery backup will fail). We also assume, as Korkali *et al.* did, that the adversary will only target power nodes.

Within the communication domain itself, it is assumed that communication is possible between two nodes as long as there exists a connected path between them in the graph G_c. In G_p, power failures propagate according to DC power-flow equations with generator and consumer nodes that place various generations and loads on the network and transmission lines that have a fixed capacity. While DC power flow models do not capture all aspects of cascading failure (e.g., dynamic instability, voltage collapse, and distance relays [27]), they are computationally efficient and numerically stable. Moreover, even highly complex AC models must make some simplifying assumptions, and there is not currently an established AC power flow model that is publicly available [27]. For these reasons DC power flow models have been chosen for several cascading failure studies [5], [27], [28], [29]. Furthermore, because our RL method in particular represents the environment as a black box, it can be extended straightforwardly to any cascading failure model that is not prohibitively computationally expensive to train with. The DC power flow model used in this work can be summarized as follows:

$$\mathbf{G} - \mathbf{D} = \mathbf{B}\theta \tag{6}$$

$$F_{ij} = \frac{1}{x_{ij}(\theta_i - \theta_j)} \tag{7}$$

where $\mathbf{G}$ and $\mathbf{D}$ are vectors of power generation and load, $\mathbf{B}$ is a weighted Laplacian matrix that encodes the network's topology, θ is a vector of phase voltage angles, F_{ij} is the power flow from node i to node j, and x_{ij} is the normalized inductance of the transmission line. When a component fails, flows are recomputed according to this model. If the updated flows exceed the flow capacity of a power line, then it will disconnect in an amount of time that is proportional to the overload. This triggers another recomputation of the power flow. If the networks separates into islands and there is not a feasible solution to Equation (6) due to an imbalance between supply and demand, a combination of generator adjustments and load reductions are used to arrive at a new feasible solution to Equation (6).

The simulation model also utilizes an automated load shedding algorithm that reactively reduces the cascading of failures through the network based on node failures caused by the attacker (details provided in [5]). However, this controller relies on the communication network to measure load injections at various points in the graph, therefore, if communication nodes fail due to the adversary's attack, it will reduce the effectiveness of the load shedding algorithm.

We create G_v using both random graphs generation and a graph modeled after the Polish power grid [30]. Random graphs are constructed with the same average number of edges per node as the Polish grid, regardless of the total number of nodes in the graph. The graph G_c is then constructed using the following procedure: First, a copy of G_p is made such that $V_c = V_p$. Then 10% of the edges in G_c are rewired, excluding those rewiring that would result in self-loops or duplicate edges. Finally, based on a coupling parameter $q \in [0, 1]$, a node in V_c is connected to its corresponding node in V_p with probability q.

The main modification of the coupled network simulation environment for this work was to make it compatible with the agent-environment interface that is standard in RL. In this paradigm, the environment is a black box from the agents' perspective. The environment receives actions from the agents, and generates a reward (and a new environment state in multi-step environments). In this work, the reward is represented as the proportion of nodes in the graph that fail after cascading. For example, if 50% of nodes in the graph go down after cascading completes, then the attacker agent receives a reward of 0.5 and the defender receives a reward of -0.5.

The modified environment, as well the learning agent implementations, can be found on the project's GitHub repository.[1]

B. Experimental Results

There are many possible choices of node embedding functions to use in Equation (3). The heuristics of node degree, betweenness centrality, and harmonic centrality were used as features for this work. The addition of other heuristic node features beyond these three had little impact on performance. The node index is also included as a feature, so that the agent can tell if two nodes are the same node or different nodes with identical features, for a total of $p = 4$ features. As for h, ablations from 16 to 128 had no significant impact on performance. The results presented in this section use $h = 64$.

Our method relies on training on small-scale graphs, and then generalizing to larger graphs. Thus, we conduct our training on 10-node graphs. We start with the simplest case of attacking a single node. Because of the small scale, we are able to calculate the Nash equilibrium directly. We can then compare the divergence of the learned policy from the true Nash equilibrium policy as a metric of its closeness to optimality. We can then also compare their performance to the benchmarks used in the literature.

We evaluate our method under two conditions: 1. training on a fixed graph topology and then evaluating on that same topology; 2. training on random topologies and then evaluating on a testing set of fixed topologies. Condition 1 is similar to the condition under which tabular RL methods are successful, such as those used in [10], [11], [12], and thus we benchmark against a tabular Minimax Q Learning implementation. Condition 2 involves generalizing from random topologies during training to unseen topologies at test-time. As discussed in Section II, this is significantly more challenging than condition

[1] https://github.com/jdc5549/coupled-networks

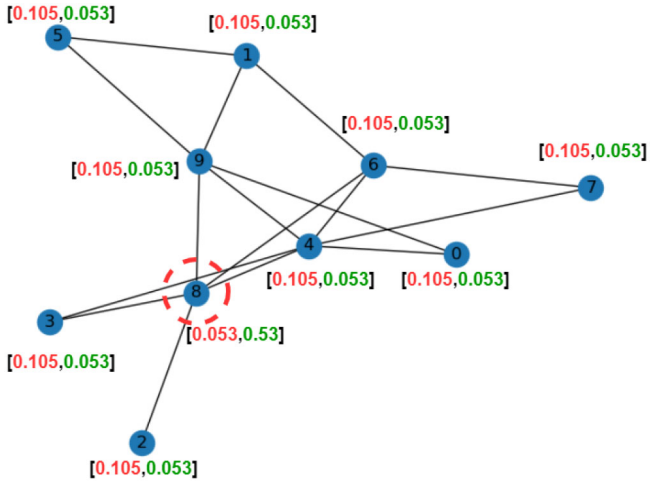

Fig. 2. Illustration of Nash equilibrium mixed strategies for the attacker and defender with respect to a fixed 10 node graph. Node 8 is highlighted as a critical node that the defender defends most often.

1 and computationally prohibitive to implement in tabular form even for this 10-node example.

Figure 2 shows one of the fixed networks that we use to evaluate our method under each of the specified conditions, as well as the directly calculated Nash equilibrium strategy for this network. It should be noted that each of the fixed networks used for training has a unique Nash equilibrium. We can observe that for this fixed network in Figure 2, node 8 is clearly the most valuable target in this network, as the defender defends it with the highest probability, and all others uniform randomly. Thus, the attacker's best strategy is to attack this node with only a small probability and the rest of the time attack any other node uniform randomly. Figure 3 shows the Kullback–Leibler (KL) divergence between the agent strategies and nash equilibrium strategies during training. From this figure, we can observe that under the first training condition, the divergence converges to near 0 for both our method and the tabular RL benchmark, and that the tabular RL method reaches a lower error. This is to be expected at this small scale given that it uses an exact representation of Q whereas the neural network approaches uses the approximation Q_θ. Under the second training condition, the KL divergence in Figure 3 converges to a value of less than 1. This larger deviation from the NashEQ strategy is also expected, as the model is being evaluated on the much harder problem of generalizing to a testing set of 10 topologies it was not exposed to during training, which as discussed in Section II is not scalable even on this 10 node network using tabular RL, and thus we are not able to benchmark against these methods.

For strategies that do not achieve a near 0 KL-divergence to the NashEQ strategies, we can also evaluate them by calculating their approximate *exploitability*. The exploitability of a strategy is defined as the expected average payoff of the best response to that strategy. An exploitability of 2δ yields at least a δ-Nash equilibrium [31], which means that the Nash equilibrium has an exploitability of 0 by definition. We can approximate this best response strategy by training new agents specifically against the fixed "ego" agent models we wish to

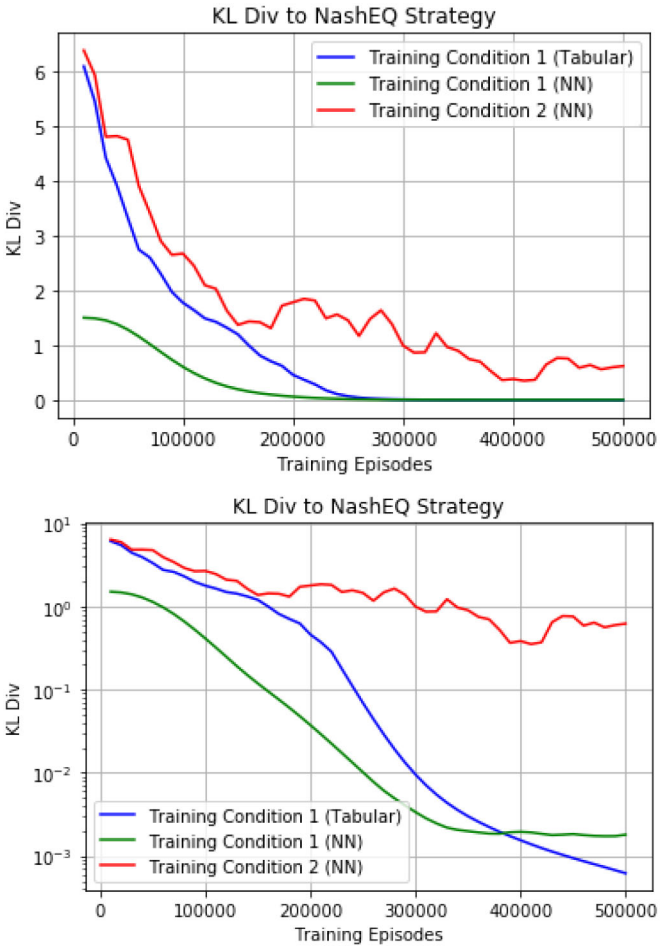

Fig. 3. KL divergence of agent strategies and Nash equilibrium strategies for 10 choose 1 test case. For the condition 1 curves, divergences are calculated with respect to the single fixed network in Figure 2. For the condition 2 curves, divergence is calculated with respect to a testing set of 10 random topologies including the one in Figure 2.

evaluate. Because these "exploiter" agents always play against a fixed ego agent strategy, we can use more computationally efficient single-agent RL techniques to train them. We use the Advantage Actor-Critic (A2C) [32] algorithm to train exploiter agents with the ego agent strategy treated as part of the environment.

More precisely, if we have trained ego agents π_{Atk} and π_{Def} that have average payoffs playing against each other of δ_g and $-\delta_g$ respectively, we would train exploiter π_{XA} against π_{Atk} and π_{XD} against π_{Def} using A2C, and they would achieve average payoffs of δ_{XA} and δ_{XD} respectively. Our calculated exploitability would then be

$$\delta = \left(\delta_{XD} - \delta_g\right) + \left(\delta_{XA} + \delta_g\right) \tag{8}$$

After training, the learned ego and exploiter models were evaluated for 1000 episodes on the set of 10 testing graphs. The heuristic benchmarks from the literature: random selection and deterministically selecting the node with highest degree are also evaluated.

Table III shows the average KL divergence and exploitability of each of the evaluated methods. All of the learned models demonstrate exploitability that is either near 0 or negative,

TABLE III
AVERAGE EXPLOITABILITY AND NASH KL DIVERGENCE OF EACH EVALUATED STRATEGY ON THE TESTING SET OF 10 GRID TOPOLOGIES FOR THE 10 CHOOSE 1 TEST CASE

Ego Strategy	δ	KL Div
Training Condition 1 (Tabular)	-0.015	1.08×10^{-9}
Training Condition 1	-0.014	3.04×10^{-3}
Training Condition 2	1.00×10^{-5}	0.459
Random (Sturaro 2016, Korkali 2017)	0.0658	0.237
Node Degree (Sturaro 2016, Korkali 2017)	0.412	2.92

while both of the benchmarks have a positive exploitability. While the random benchmark has a smaller divergence from the NashEQ than the model under condition 2, it is still more easily exploited on average. It should be noted that this is true despite the fact that for 3 of 10 test scenarios, the NashEQ strategy is coincidentally uniform random for both players.

For a concrete example of the strategies learned for the fixed graph, Table IV shows the average strategies over the 1000 episodes for both trained models compared against the Nash equilibrium strategy. We can see that the condition 1 tabular model learns the exact Nash equilibrium with respect to the precision of the figures in the table, and our model is very close to exact. Under training condition 2, we can see that the influence of the node features is apparent, especially node degree. The highest degree nodes are 4, 9, and 8, while the lowest degree nodes are 0, 2, and 3. The high degree nodes are all among the most often defended, and the lowest degree nodes are the least often defended. However, the model correctly defends node 8 most often, despite it only being the second highest degree node, indicating that the model has learned to predict high impact nodes with respect to cascading failure. These results illustrate the key aspect that makes training condition 2 more challenging. Instead of memorizing via trial and error the best solution for a particular network, the agent must learn which node features correspond to high value nodes and prioritize their targets accordingly.

Next, we examine a slighty more complex test case of each player choosing 2 nodes out of 5. We compare modeling the full action space versus using RCR with $m = 1$. Figure 4 shows the learning curves for 2 million steps of training for each of these cases. The assumption referenced in Section III that for larger problems the overall error to the Nash equilibrium will be larger than the error induced by RCR seems to hold here, as RCR achieves a KL divergence of less than 1, while in the full action space little progress is made.

With this in mind, we examine the large-scale $\binom{100}{5}$ test case. Unlike the small-scale experiments, we are unable to directly calculate Nash equilibrium.

For the $\binom{100}{5}$ test case, we train A2C exploiters using RCR with $m = 1$ for $100,000$ time steps against the RCR agent model for the $\binom{10}{2}$ test case applied to the $\binom{100}{5}$ scenario as the ego agents. We also benchmark against both random node selections and the largest node degree heuristic. We generate a test set of 10 randomly generated 100-node graphs to train the exploiter agents, and then evaluate the trained agent for 1000 the average and standard deviation of exploitability across the 10 test networks for our neural network agent and

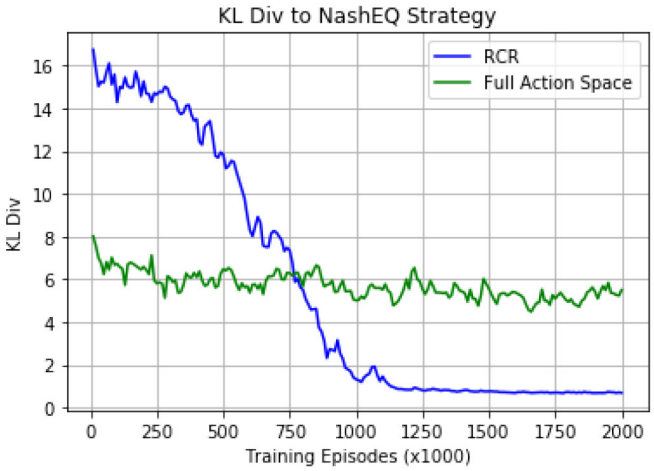

Fig. 4. Comparison of RCR versus full combinatorial action space for the "5 choose 2" graph test case.

the benchmark strategies. Table V shows the exploitability of our proposed method compared to the exploitability of the random and node degree heuristics that are used in practice for large-scale graphs. We can see that our proposed method is less exploitable than these heuristics by a factor of 19 and 24 respectively. This means that an intelligent agent is able to inflict a much larger cost against an agent that is using these standard heuristics as opposed to our proposed game theoretic strategy of preemptive defense.

The final test case we consider is the topology based on the Polish grid. This topology constitutes a $\binom{2383}{50}$ test case. Table VI shows the exploitability metrics for this network. We find an exploitability of -0.00151 for the Polish grid, which indicates that the exploiter agents performed slightly worse than the ego agents. However, the exploiter agents still had success against the two benchmark heuristics.

These results certainly do not indicate that our method's strategies are close to the Nash equilibrium despite the low exploitability, due to the imperfection of the best response strategies. However, the fact that the A2C exploiters were largely unable to exploit the strategies learned by the ego agents, especially compared to common heuristic strategies employed on large-scale graphs, indicates that the ego agents learned intelligent strategies for attack and defense on large networks with many simultaneous target selections. This is in contrast to state-of-the-art game theoretic methods that avoid modeling many simultaneous target selections entirely due to their inability to scale.

V. CONCLUSION

We introduced a neural network model that uses node embeddings and minimax Q-learning to predict the utility of attacking or defending nodes in a graph. This model does not depend on the size of the graph, which enables us to train on small graphs and leverage those same learned features on graphs of realistic scale. We also introduced a sampling trick we call RCR, that allows for the approximation of a large combination of actions as a series of smaller combinations of actions. Combining both of these contributions, we were able

TABLE IV
COMPARISON OF STRATEGIES FOR THE ATTACKER AND DEFENDER ON THE 10 NODE TEST CASE SHOWN IN FIGURE 2. THE NASH EQUILIBRIUM STRATEGY IS COMPARED WITH THE STRATEGIES LEARNED BY THE TABULAR AGENT UNDER TRAINING CONDITION 1 AND THE NEURAL NETWORK (NN) AGENT UNDER TRAINING CONDITIONS 1 AND 2

Attack Node	0	1	2	3	4	5	6	7	8	9
Nash EQ	0.105	0.105	0.105	0.105	0.105	0.105	0.105	0.105	0.053	0.105
Tabular C1	0.105	0.105	0.105	0.105	0.105	0.105	0.105	0.105	0.053	0.105
NN C1	0.109	0.109	0.105	0.105	0.107	0.105	0.105	0.102	0.052	0.102
NN C2	0.142	0.0460	0.104	0.0970	0.0792	0.0905	0.109	0.149	0.120	0.0644

Defense Node	0	1	2	3	4	5	6	7	8	9
Nash EQ	0.053	0.053	0.053	0.053	0.053	0.053	0.053	0.053	0.526	0.053
Tabular C1	0.053	0.053	0.053	0.053	0.053	0.053	0.053	0.053	0.526	0.053
NN C1	0.051	0.052	0.051	0.052	0.056	0.051	0.054	0.051	0.528	0.055
NN C2	0.0308	0.0537	0.0461	0.0521	0.137	0.0677	0.00	0.0415	0.305	0.267

TABLE V
EXPLOITABILITY OF EGO AGENT STRATEGIES WITH RESPECT TO A2C EXPLOITERS

Ego Strategy	δ
Our Method	**0.00140**
Random (Sturaro 2016, Korkali 2017)	0.0267
Node Degree (Sturaro 2016, Korkali 2017)	0.0337

TABLE VI
EXPLOITABILITY OF EGO AGENT STRATEGIES WITH RESPECT TO A2C EXPLOITERS FOR THE POLISH GRID TOPOLOGY

Ego Strategy	δ
Our Method	**-0.00151**
Random (Sturaro 2016, Korkali 2017)	0.00267
Node Degree (Sturaro 2016, Korkali 2017)	0.00130

to learn both an attack and defense strategy on a large-scale coupled power and communication network that was far less exploitable with respect to a dedicated RL exploiter agent than common heuristic strategies for node failures.

While this work is an important first step towards scaling security games to large-scale graphs, there are several possible directions to expand upon this work. Firstly, very simple node features were used for the node embeddings, and this likely limited the ability of the network identify optimal strategies from node features alone. To overcome reliance on heuristics, graph convolutional neural networks such as "struc2vec" could be used to automatically learn embeddings of the nodes that are useful for Q-value prediction.

Another direction for future work is expanding to more complex security game models, for example multi-step environments where agents must apply long-term credit assignment to their actions, as well as more complex models of how attackers and defenders can distribute resources over possible targets. Additionally, relaxing the assumption of zero-sum payoff would require more complex adversarial learning methods than minimax Q-learning, but would allow for modeling of scenarios where the attacker and defender have different priorities and incentives. Moreover, relaxing the zero-sum assumption would effectively increase the training time required, as in this work the fact both sides of the game can be represented with the same Q-function (with inverted sign) was exploited to reduce training time. This motivates methods of improving the computational efficiency of the proposed method as an additional avenue for future work.

ACKNOWLEDGMENT

Thanks to Alex Aved for concept development. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Research Laboratory or the U.S. Government. The authors would also like to thank John Salerno for his contribution to this project.

REFERENCES

[1] T. D. Le, A. Anwar, R. Beuran, and S. W. Loke, "Smart grid co-simulation tools: Review and cybersecurity case study," in *Proc. 7th Int. Conf. Smart Grid (icSmartGrid)*, 2019, pp. 39–45.
[2] C.-C. Sun, A. Hahn, and C.-C. Liu, "Cyber security of a power grid: State-of-the-art," *Int. J. Elect. Power Energy Syst.*, vol. 99, pp. 45–56, Jul. 2018.
[3] Z. El Mrabet, N. Kaabouch, H. El Ghazi, and H. El Ghazi, "Cyber-security in smart grid: Survey and challenges," *Comput. Elect. Eng.*, vol. 67, pp. 469–482, Apr. 2018.
[4] Gö. N. Ericsson, "Cyber security and power system communication—Essential parts of a smart grid infrastructure," *IEEE Trans. Power Del.*, vol. 25, no. 3, pp. 1501–1507, Jul. 2010.
[5] M. Korkali, J. G. Veneman, B. F. Tivnan, J. P. Bagrow, and P. D. H. Hines, "Reducing cascading failure risk by increasing infrastructure network interdependence," *Sci. Rep.*, vol. 7, pp. 1–13, Mar. 2017.
[6] "Analysis of the cyber attack on the Ukrainian power grid: Defense use case," Elect. Inf. Sharing Anal. Center, Washington, DC, USA, E-ISAC–388, 2016.
[7] A. Sturaro, S. Silvestri, M. Conti, and S. K. Das, "Towards a realistic model for failure propagation in interdependent networks," in *Proc. Int. Conf. Comput. Netw. Commun. (ICNC)*, 2016, pp. 1–7.
[8] Y. Cai, Y. Cao, Y. Li, T. Huang, and B. Zhou, "Cascading failure analysis considering interaction between power grids and communication networks," *IEEE Trans. Smart Grid*, vol. 7, no. 1, pp. 530–538, Jan. 2016.
[9] J. Cordova-Garcia, X. Wang, D. Xie, Y. Zhao, and L. Zuo, "Control of communications-dependent cascading failures in power grids," *IEEE Trans. Smart Grid*, vol. 10, no. 5, pp. 5021–5031, Sep. 2019.
[10] S. Paul, Z. Ni, and C. Mu, "A learning-based solution for an adversarial repeated game in cyber-physical power systems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 11, pp. 4512–4523, Nov. 2020.
[11] L. Wei, A. I. Sarwat, W. Saad, and S. Biswas, "Stochastic games for power grid protection against coordinated cyber-physical attacks," *IEEE Trans. Smart Grid*, vol. 9, no. 2, pp. 684–694, Mar. 2018.

[12] Y. Guo, L. Wang, Z. Liu, and Y. Shen, "Reinforcement-learning-based dynamic defense strategy of multistage game against dynamic load altering attack," *Int. J. Elect. Power Energy Syst.*, vol. 131, Oct. 2021, Art. no. 107113.

[13] Y. Guo, L. Wang, Z. Liu, and Y. Shen, "Reinforcement-learning-based dynamic defense strategy of multistage game against dynamic load altering attack," *Int. J. Elect. Power Energy Syst.*, vol. 131, Oct. 2021, Art. no. 107113.

[14] J. Yan, H. He, X. Zhong, and Y. Tang, "Q-learning-based vulnerability analysis of smart grid against sequential topology attacks," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 1, pp. 200–210, Jan. 2017.

[15] I. Ahmad, A. Clark, A. Sabol, D. Ferris, and A. Aved, "Maximizing resilience under defender attacker model in heterogeneous multi-networks," in *Proc. 3rd Int. Conf. Data Intell. Secur. (ICDIS)*, 2020, pp. 117–126.

[16] D. Silver *et al*. "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[17] D. Silver *et al*. "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," 2017, *arXiv:1712.01815*.

[18] C. Berner *et al*. "Dota 2 with large scale deep reinforcement learning," 2019, *arXiv:1912.06680*.

[19] O. Vinyals *et al*. "Grandmaster level in StarCraft II using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[20] F. Wei, Z. Wan, and H. He, "Cyber-attack recovery strategy for smart grid based on deep reinforcement learning," *IEEE Trans. Smart Grid*, vol. 11, no. 3, pp. 2476–2486, May 2020.

[21] Z. Ni and S. Paul, "A multistage game in smart grid security: A reinforcement learning solution," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 9, pp. 2684–2695, Sep. 2019.

[22] M. L. Littman, "Value-function reinforcement learning in Markov games," *Cogn. Syst. Res.*, vol. 2, no. 1, pp. 55–66, 2001.

[23] M. L. Littman and C. Szepesvári, "A generalized reinforcement-learning model: Convergence and applications," in *Proc. ICML*, vol. 96, 1996, pp. 310–318.

[24] H. Dai, E. B. Khalil, Y. Zhang, B. Dilkina, and L. Song, "Learning combinatorial optimization algorithms over graphs," 2017, *arXiv:1704.01665*.

[25] V. Mnih *et al*. "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[26] M. Osborne, A. Lall, and B. Van Durme, "Exponential reservoir sampling for streaming language models," in *Proc. 52nd Annu. Meeting Assoc. Comput. Linguist.*, 2014, pp. 687–692.

[27] M. J. Eppstein and P. D. H. Hines, "A 'random chemistry' algorithm for identifying collections of multiple contingencies that initiate cascading failure," *IEEE Trans. Power Syst.*, vol. 27, no. 3, pp. 1698–1705, Aug. 2012.

[28] R. Pfitzner, K. Turitsyn, and M. Chertkov, "Statistical classification of cascading failures in power grids," in *Proc. IEEE Power Energy Soc, General Meeting*, 2011, pp. 1–8.

[29] A. Bernstein, D. Bienstock, D. Hay, M. Uzunoglu, and G. Zussman, "Power grid vulnerability to geographically correlated failures—Analysis and control implications," in *Proc. IEEE INFOCOM IEEE Conf. Comput. Commun.*, 2014, pp. 2634–2642.

[30] R. D. Zimmerman, C. E. Murillo-Sánchez, and R. J. Thomas, "MATPOWER: Steady-state operations, planning, and analysis tools for power systems research and education," *IEEE Trans. Power Syst.*, vol. 26, no. 1, pp. 12–19, Feb. 2011.

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

[32] V. Mnih *et al.*, "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1928–1937.