IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 21, 2020, accepted October 28, 2020, date of publication November 3, 2020, date of current version November 13, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3035362

Achieving Correlated Equilibrium by Studying Opponent's Behavior Through Policy-Based Deep Reinforcement Learning

KUO CHUN TSAI[1], (Senior Member, IEEE), AND ZHU HAN[1,2], (Fellow, IEEE)
[1]Department of Electrical and Computer Engineering, University of Houston, Houston, TX 77204, USA
[2]Department of Computer Science and Engineering, Kyung Hee University, Seoul 446-701, South Korea

Corresponding author: Kuo Chun Tsai (kevintsai159@gmail.com)

This work was supported in part by U.S. Multidisciplinary University Research Initiative under Grant 18RT0073, Grant NSF EARS-1839818, Grant CNS1717454, Grant CNS-1731424, and Grant CNS-1702850.

ABSTRACT Game theory is a very profound study on distributed decision-making behavior and has been extensively developed by many scholars. However, many existing works rely on certain strict assumptions such as knowing the opponent's private behaviors, which might not be practical. In this work, we focused on two Nobel winning concepts, the Nash equilibrium, and the correlated equilibrium. We proposed a policy-based deep reinforcement learning model which instead of just learning the regions for corresponding strategies and actions, it learns why and how the rational opponent plays. With our proposed policy-based deep reinforcement learning model, we successfully reached the correlated equilibrium which maximizes the utility for each player. Depending on the scenario, the equilibrium can reach outside of the Nash equilibrium convex hull to achieve higher utility for the players, while the traditional non-regret algorithms cannot. In addition, we also proposed a mathematical model to inverse the calculation of the correlated equilibrium probability to estimate the rational opponent player's payoff. Through simulations, with limited interaction among the players, we showed that our proposed method can achieve the optimal correlated equilibrium where each player gains an equal or higher utility than the Nash equilibrium.

INDEX TERMS Correlated equilibrium, deep learning, game theory, joint distribution, machine learning, neural network, reinforcement learning.

I. INTRODUCTION

Game theory is the study of mathematical models regarding the rationality of decision making strategic interaction. Originally, game theory addressed the two-player zero-sum non-cooperative games where each participant's gains or losses are exactly balanced by those of the other participants [1]. Today, game theory applies to a wide range of behavioral relations such as cooperative game [2], contract theory, auction theory, matching game, dynamic game and more [3], [4].

The main idea for game theory is for each rational player to maximize his or her utility. Take the Nobel winning Nash equilibrium solution concept for example. The Nash equilibrium solution concept, named after the mathematician John Forbes Nash Jr., is a solution of a non-cooperative game involving two or more players. A Nash equilibrium in the game is where no player has utility increment by changing only their strategy [5]. However, the solution under the Nash equilibrium might be far from the centralized solution. Hence, in 1974, there was another mathematician named Robert Aumann discussed another Nobel winning solution concept called correlated equilibrium [6]. The idea in the correlated equilibrium concept is that all players will choose an action according to a public signal. If all players are satisfied by the recommended strategy, the distribution is called a correlated equilibrium distribution [7]. With this concept, players can achieve equal or higher utility in the game since they consider the joint distribution instead of the marginal distribution as the Nash equilibrium. However, there does not exist a distributed strategy on how to reach correlated equilibrium outside the Nash equilibrium convex hull since there does not exist a clear way on how to design the public signal for the players to obtain. Some existing works using non-regret learning [8], [9] can only achieve the Nash equilibrium convex hull but cannot learn the better correlated equilibrium.

The reason that the players in a non-cooperative game are unable to achieve the correlated equilibrium is due to the

The associate editor coordinating the review of this manuscript and approving it for publication was Kathiravan Srinivasan.

fact that they are unable to mine some of the information out from the public signal provided to them. Fortunately, a technique called machine learning was developed for studying the underlying factor of the data. Many works, such as [10]–[13] and more, showed the robustness of the machine learning technique for data mining tasks. However, even today, the link of machine learning with the correlated equilibrium solution concept in game theory is seldom studied. Although few articles tried to apply machine learning with correlated equilibrium solution concept in some applications, there are concerns within their research when applying to real-life scenarios. For example, some research showed that the Q-Learning method [14], [15] could achieve the correlated equilibrium within the system [16]–[18]. However, the issues with the Q-Learning method are high space and power consumption. Moreover, in order for them to achieve the correlated equilibrium, they need a centralized system to perform the equilibrium calculation which is not practical for a large system in real-life scenarios.

Motivated by the above facts, we proposed a policy-based deep reinforcement learning model to determine the strategy to reach equilibrium under limited information given to the player. We then estimate the payoffs of other players via our proposed mathematical model. Once we have all the payoffs from the other players, we can determine the correlated equilibrium between other players without them interacting with each other. Thus, the contribution we made in this work can be summarized as follows:

- We first define the public signal in the system for the player to obtain that contains limited information of the players.
- With limited information, players will start to interact with the environment. By applying our proposed deep reinforcement learning model, the player not only can understand the structure of the environment but also learn the joint distribution among all of the players when exploring the environment. In the end, the players can reach a correlated equilibrium where no one wants to deviate.
- With the correlated equilibrium probability distribution that all players satisfied with, we proposed a mathematical model that combines the concept of the correlated equilibrium and the force of tension to estimate the payoff vectors of other players.
- By knowing the payoff vector of other players in the system, we could compute the correlated equilibrium. In other words, with those payoff vectors, deep reinforcement learning learns why and how the rational opponent plays, instead of just learning the regions for corresponding strategies and actions.
- This paper combines game theory with machine learning in the sense that the proposed machine learning learns what is the game player's payoff, instead of just categorizing the strategies of actions according to the current situation.

The rest of the paper is organized as follows. In Section II, we discuss the basic concept in the Nash equilibrium and correlated equilibrium concepts along with our system model and problem formulation. Next, in Section III, we show how the player interacts with the environment with our proposed policy-based deep reinforcement learning model and learn the joint distribution among the players. In the same section, we also proposed a mathematical model to estimate the information of the opponent players. Next, we show the numerical results for our proposed methods in Section IV. Finally, we conclude our work in Section VI.

II. SYSTEM MODEL AND BASIC EQUILIBRIUM CONCEPT

In this section, we will first go through the basic concept of the Nash equilibrium and the correlated equilibrium in Section II-A. In Section II-B, we will study the relationship between the environment and the players in our system model. Finally, in Section II-C, we will discuss the problem formulation.

A. NASH EQUILIBRIUM AND CORRELATED EQUILIBRIUM BASICS

The Nash equilibrium is a solution concept for a non-cooperative game for two or more players [5]. The equilibrium outcome of a non-cooperative game is one where no player wants to deviate from his or her chosen strategy after considering the opponent's decision. In other words, an individual cannot increment his or her utility from unilaterally changing his or her strategies, assuming the other players remain their strategies. There might be none or multiple Nash equilibrium in a non-cooperative game depending on the setup of the game and the strategies used by each player. A formal definition of Nash equilibrium is as follows.

Definition 1: An I-player game is characterized by an action set Φ_i. Let $B_i(\varsigma_{-i}) \subset \Phi_i$ be the set of player p_i's best response strategy against $\varsigma_{-i} \in \Phi_{-1}$. $\varsigma^* = (\varsigma_1^*, \ldots, \varsigma_I^*) \in \Phi$ is a Nash equilibrium if $\varsigma^* \in B_i(\varsigma_{-i}^*)$ for every $i \in I$.

On the other hand, the correlated equilibrium is also a solution concept that is more general than the Nash equilibrium [19]. The idea of the correlated equilibrium solution concept is that each player chooses their decision according to their observation of a public signal [20]. A strategy assigns an action to every possible decision set D_h a player can choose. If no player wants to deviate from the recommended strategy, the distribution is called a correlated equilibrium. A formal definition is as follows.

Definition 2: An I-player strategic game (I, Φ_i, u_i) is characterized by an action set Φ_i and utility function u_i for each player i. when player i chooses strategy $\varsigma_i \in \Phi_i$ and the remaining players choose a strategy profile ς_{-i} described by the $I-1$ tuple. Then the player i's utility is $u_i(\varsigma_i, \varsigma_{-i})$. A strategy modification for player i is a function $\phi_i : \Phi_i \to \Phi_i$. That is ϕ_i tells player i to modify his or her behavior by playing action $\phi_i(\varsigma_i)$ when instructed by play ς_i. Let (Ω, Ψ) be a countable probability space. For each player i, let F_i be his or her information partition, q_i be i's posterior and let

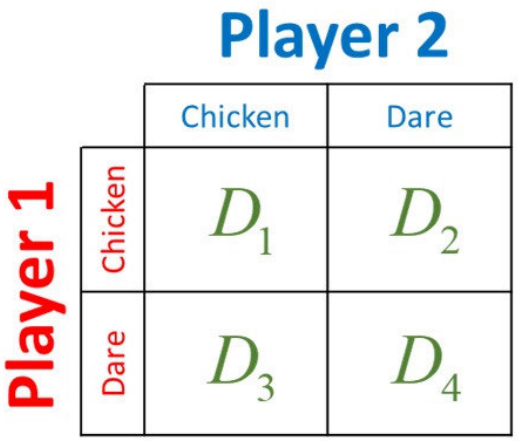


FIGURE 1. Game of chicken decision set layout.

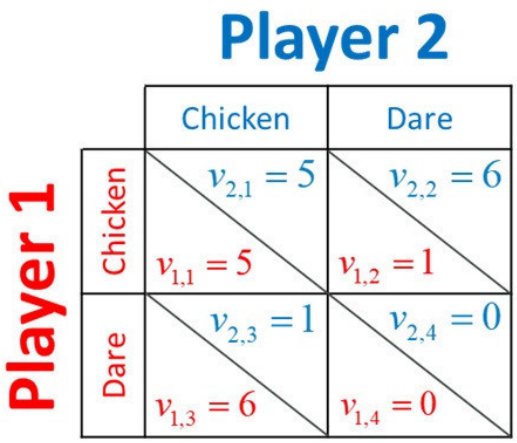


FIGURE 2. Game of chicken variables layout.

$s_i : \Omega \rightarrow \Phi_i$, assigning the same value to states in the same cell of i's information partition. Then $((\Omega, \Psi), F_i, s_i)$ is the correlated equilibrium of the strategic game (I, Φ_i, u_i) if it satisfies the condition

$$\sum_{\omega \in \Omega} q_i(\omega) u_i(s_i(\omega), s_{-i}(\omega)) \geq \sum_{\omega \in \Omega} q_i(\omega) u_i(\phi_i(s_i(\omega)), s_{-i}(\omega)) \quad (1)$$

for every player i and for every strategy modification ϕ_i.

B. SYSTEM MODEL

There are two major components in our system model – the players and the environment. We can consider the environment is a set of states $\mathbb{S}$ where each state $s_k = (\rho_1, \rho_2, \ldots, \rho_H)$ is a unique tuple which contains the probabilities for each decision set $D_h \in \mathbb{D}$ for $h = 1, 2, \ldots, H$, where $\mathbb{D}$ is a set of permutation of all players' decision. In other words, state s_k contains a probability distribution of the decisions in $\mathbb{D}$. The order of the players' decision in the decision set D_h will always start from player p_1 to player p_I. Take the game of chicken (Fig. 1) for example. The set $\mathbb{D}$ contains total four elements where D_1, D_2, D_3, and D_4 are set to (Chicken, Chicken), (Chicken, Dare), (Dare, Chicken), and (Dare, Dare), respectively.

On the other hand, there is a set of players P in our system model where P contains a total of I players where $2 \leq I \in \mathbb{Z}$. Each player p_i has their own payoff vector V_i, policy π_i, and a set of actions $\mathbb{A}$. The payoff vector V_i contains rewards $v_{i,h}$ that player p_i will receive when agreeing on performing the decision set D_h in the game. The sum of all the elements in the payoff vector V_i has to be equal to one. This allows us to determine the ratio between each element when we estimate the payoff vector for the other players in Section III-E. Next, the policy π_i is the behavior on how player p_i will interact with the environment. Policy π_i states that player p_i will follow a certain probability distribution to choose an action $a_j \in \mathbb{A}$ to perform based on the current state s_k that the player is currently in and the previous state s_{k-1} that the player came from. The action set $\mathbb{A}$ is the permutation of increase, decrease, or no change on the probability of each of the decision set D_h for $1 \leq h \leq H - 1$. The amount of increasing or decreasing the probability is set to be $\vartheta \in (0, 1]$. The reason why the action does not contain the change of decision D_H is due to the fact that the probability distribution of all decisions in $\mathbb{D}$ needs to sum up to one. Hence, we can get the probability of D_H by subtracting the summation of all other decisions' probabilities from one. In addition, even though the probability adjustment amount ϑ is a constant, if the probability is out of the range of [0, 1] after increasing or decreasing the amount of ϑ, depending on the action, the probability will only increase or decrease a certain amount so the probability will still within the range of [0, 1]. Next, depending on the action the player chooses, the player will be moving to another state. The detail on how the player will interact with the environment will be discussed in Section III-B.

Moreover, according to Robert J. Aumann, the correlated equilibrium is a general form of strategy randomization than mixing. This means that the solution of the Nash equilibrium is within the convex hull of the correlated equilibrium. Hence, we set a restriction where there must be at least two Nash equilibrium sets in the system. Moreover, the Nash equilibrium sets have to be found using the mixed strategy. This restriction future expresses as the rewards in V_i for player p_i will be unique values when the decisions are fixed for other players. Otherwise, there will be no solution to the mixed strategy. Take Fig. 2 as an example. The rewards constraint is that $v_{1,1} \neq v_{1,3}$ and $v_{1,2} \neq v_{1,4}$ for player 1, and $v_{2,1} \neq v_{2,2}$ and $v_{2,3} \neq v_{2,4}$ for player 2.

C. PROBLEM FORMULATION

Although players could achieve correlated equilibrium to obtain a higher reward by using the correlated strategy instead

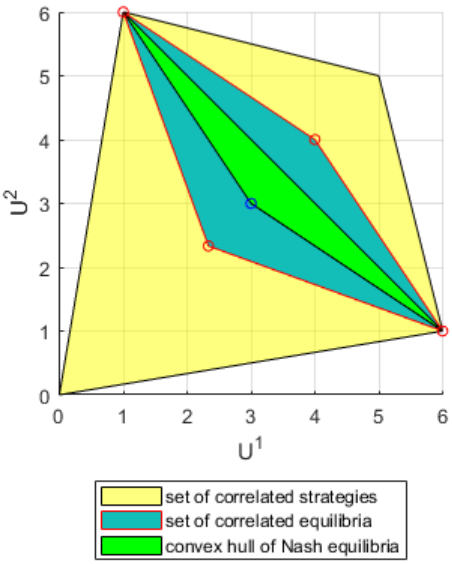


FIGURE 3. Nash equilibrium vs. correlated equilibrium.

of the mixed strategy in a non-cooperative game, there exist few challenges within the correlated equilibrium solution concept itself. First, as mentioned in Section II-A, the correlated equilibrium concept is a more general form of strategy randomization than the mixing Nash equilibrium strategy. We can see the relation between the correlated equilibrium convex hull and the Nash equilibrium convex hull from the game of chicken shows in Fig. 3. Although it is true that the reward obtained by the player might be higher when applying the correlated strategy rather than the Nash equilibrium strategy. However, at the same time, this also means there is a probability that the player can obtain a lower reward. Second, the correlated equilibrium concept is built on the idea of having a public signal for players to observe. Based on this public signal, the player will choose the suggested decision. However, this public signal must fulfill constraint (1). This means that the public signal must be generated based on the payoff vectors from all of the players. This further implies that the payoff vectors from all of the players are public information which violates the purpose of a game. Otherwise, this will become a centralized system with centralized solutions. Hence, without knowing the payoff vectors from other players, there is no guarantee the game can reach a correlated equilibrium. Thus, the above challenges within the correlated equilibrium solution concept motivate us to design a public signal that contains limited information about each of the players and a strategy for players that allow them to achieve the correlated equilibrium. Moreover, the correlated equilibrium they achieved will also maximize each players' reward.

III. PROPOSED METHOD

Our proposed method will be split into two major parts – correlated equilibrium distribution determination and payoff vectors estimation. The first part is to determine the correlated equilibrium distribution between players and maximize the summation of each of the player's reward under this distribution. We first provide an overview of the proposed process in Section III-A. Then we discuss how players interact with the environment and collect data during the interaction in Section III-B. Next, we then discuss our proposed a policy-based deep reinforcement learning model and the model training process that leads players to the correlated equilibrium in Section III-C and Section III-D, respectively. The second part is to estimate the payoff vectors of others via our proposed mathematical model that involves the idea of the force of tension in Section III-E. In Section III-A, we summarize the entire process along with the use of our proposed models.

A. PROCESS OVERVIEW

In our system, we start with the player p_{main}'s point of view. This means that at this point, we only know the information of player p_{main} such as player p_{main}'s payoff vector V_{main} and policy π_{main}. The process flowchart is shown in Fig. 4 and outlined as followed.

We first find the set of players $\mathcal{P}_{against} \in P$ who obtain more than one Nash equilibrium when interacting with player p_{main}. We considered the player p_t in set $\mathcal{P}_{against}$ to be against the main player p_{main} and the rest of the players are cooperating with player p_{main}. The reason we say the players in $\mathcal{P}_{against}$ are against to player p_{main} is due to the payoff vectors of those players are monotonously increasing in the opposite direction as main player p_{main}. The detail on the monotonously increasing property will be discussed in Section III-E.

Once we have the set $\mathcal{P}_{against}$, we will let players p_{main} and $p_t \in \mathcal{P}_{against}$ to interact with the system. During the interaction, both players will have to try their best to cooperate but also not to reveal too much information to others. This means each player not only has to discover the environment but also analyze the observed public information during the interaction. The public signals in our system model are the players' state after each action they performed. This information only tells a player what the other player's preference for each of the decision set D_h but does not reveal the actual payoff vector of others. Some may argue that revealing the player's state gives too much information to the opponents. However, this is not true. Take the mixed strategy for example, although no player will know what the probabilities have been assigned to each pure strategy by the opponents, after certain rounds of game play, the player can base on the statistics to determine those probabilities. Therefore, revealing the players' states as the public signals is valid. Along with player's own information, each player will learn from their experience through their own deep neural networks which will be discussed in Section III-C. Our proposed policy-based deep reinforcement learning model will learn the joint distribution between two players and lead them to the state that both players agreed on while still obtaining a certain amount of reward.

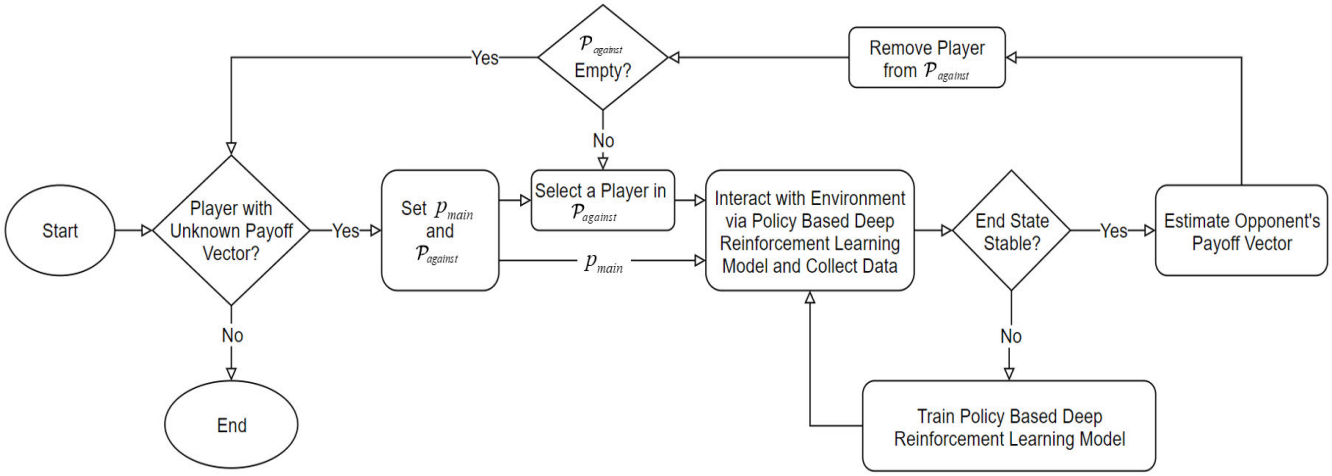


FIGURE 4. Proposed method process flowchart.

As for the termination of the interaction, since no player knows the payoff vector of the other player, we cannot terminate the interaction process based on a certain expect rewards of a player nor can we set a distribution goal to indicate the players have reached the correlated equilibrium distribution with an expect reward from both of the players. Hence, the only termination condition we can set is the number of actions the player can perform during the interaction. The number of actions has a restriction which will be discussed in Section III-B along with the interaction process and data collection process. After a certain amount of interactions, we can see that no matter how long the players interact with the environment, they will always be ending up in the same state. This means that the learning has been completed and they have reached state s_{ce} where the probability distribution of decision set $\mathbb{D}$ satisfied the correlated equilibrium distribution definition and also maximize the joint rewards of both of the players based on this distribution.

Once we get the estimated correlated equilibrium distribution state s_{ce}, we can now go to the second part of our proposed method to estimate payoff vector $\hat{V}_t$ of player p_t. In order to estimate the payoff vectors, we proposed a method that combined the idea of the rationality defined in the theorem of correlated equilibrium in game theory and the idea of the force of tension. First, due to the fact that correlated equilibrium distribution is calculated with the rationality conditions regarding the payoff vectors of both of the players, when we estimating the payoff vector $\hat{V}_t$ for player p_t, the constraints still need to fulfill the rationality conditions. Next, we can treat the probabilities in the probability distribution as a type of preference for the decisions for each of the players. Based on the preference, we can determine the tension on a decision D_h between each player. With these constraints, we can estimate the payoff vector V_t by solving a linear equation to maximize the reward of the player p_t.

Once we estimate the payoff vector of player p_t, we will repeat this process until we go over each player in set $\mathcal{P}_{against}$ and estimated the payoff vector for all players in set $\mathcal{P}_{against}$. As mentioned before, the player who is not in the set of $\mathcal{P}_{against}$ is considered as the player whose cooperating with the main player p_{mian}. This means these players who cooperate with player p_{main} will most likely be against the player $p_t \in \mathcal{P}_{against}$. Hence, if we can set either one of the players in $\mathcal{P}_{against}$ as main player p_{main}, we can find the players who are against the new main player p_{main} and repeat the process until we get the payoff vector for all players. Keep in mind that the termination condition can be triggered before each player interacts with each other. Hence, this means some players will not interact with some other players at all. However, we can still compute the correlated equilibrium among those players simply by the correlated equilibrium theorem that we discussed in Section II-A.

B. DATA COLLECTION

We let player $p_l \in \{p_{main}, p_t\}$ interact with the environment for M rounds independently. At the start of each round, player p_l will always start at the same state $s_{def} \in \mathbb{S}$. From this state, player p_l will choose an action $a_j \in \mathbb{A}$ to perform according to player p_l's policy π_l. Once the action has been completed, player p_l will enter another state $s_k \in \mathbb{S}$. Player p_l will then repeat the process of performing the next action and entering into another state until player p_l obtained N states for each round m for a total of M rounds. The number of states, N, a player needs to obtain is an integer greater or equal to $\lceil \vartheta^{-1} \rceil$. The reason being that since the player does not know the exact state which is satisfied by the other player, each player must have the ability to discover the entire environment in order to have the chance to reach the correlated equilibrium state. Also, since each of the probability in a state s_k is within the range of $[0, 1]$, a minimum of $\lceil \vartheta^{-1} \rceil - 1$

actions will guarantee that the player has the ability to reach any state in the environment.

Back to the data collection process, player p_l will record the action number of a_k, i.e., k, that performed at the n-th action in round m in choice $c_{l,m,n}$ and the corresponding state in $s_{l,m,n}$ along the process. Here, the choice $c_{l,m,n}$ is an one-hot encode data. For example, if there are five actions in the action set $\mathbb{A}$ and the player chose action a_2 at the n-th action in round m, than the choice $c_{l,m,n} = 01000$. Once M rounds of interactions have been completed, player p_l will have a set of actions' number C_l, i.e.,

$$C_l = \begin{bmatrix} c_{l,1,1} & \cdots & c_{l,1,N-1} \\ \vdots & \ddots & \vdots \\ c_{l,M,1} & \cdots & c_{l,M,N-1} \end{bmatrix} \tag{2}$$

and a set of corresponding states S_l, i.e.,

$$S_l = \begin{bmatrix} s_{l,1,1} & \cdots & s_{l,1,N} \\ \vdots & \ddots & \vdots \\ s_{l,M,1} & \cdots & s_{l,M,N} \end{bmatrix}. \tag{3}$$

where $s_{l,m,1} = s_{def}$.

Once player p_l has finished interacting with the environment for M rounds, we will need to calculate the rewards that player p_l gained in each state in S_l. However, we cannot just calculate the reward for player p_l based only on S_l since for each action that player p_l performed was depending not only on player p_l's policy π_l but also depends on the action of the other player. Hence, when we calculate the reward gained in state $s_{l,m,n}$, we need to consider the n's states in round m from the other player where $1 \leq m \leq M$ and $1 \leq n \leq N$. The relationship between both of the n's states in round m from each of the players is that those states are equally important. Hence, the final state $\hat{s}_{m,n}$ for both players will be at the middle point of both of the n's states in round m from both players. Thus, in order to do determine the state $\hat{s}_{m,n}$, we need to gather both of the state sets from each player and averaging them element-wise to form the average state set S_{avg}, i.e.,

$$S_{avg} = \begin{bmatrix} \hat{s}_{1,1} & \cdots & \hat{s}_{1,N} \\ \vdots & \ddots & \vdots \\ \hat{s}_{M,1} & \cdots & \hat{s}_{M,N} \end{bmatrix} \tag{4}$$

where

$$\hat{s}_{m,n} = \frac{1}{2} \sum_{l=1}^{2} s_{l,m,n}. \tag{5}$$

With the calculated average state set, we can then calculate the reward $r_{l,m,n}$ that player p_l gained in the each of the state in the average state set S_{avg} by computing the dot product of the average state $\hat{s}_{m,n}$ and the transport of the player's payoffs vector V_l, i.e.,

$$r_{l,m,n} = \hat{s}_{m,n} \cdot V_l^T. \tag{6}$$

C. POLICY-BASED DEEP REINFORCEMENT LEARNING NEURAL NETWORK

The structure of our proposed policy-based deep reinforcement learning neural network (Fig. 5) has two input layers where the input data for the first input layer will always be a set of states $s_{l,m,n}$ and the input data for the second input layer will be a set of states $s_{l,m,n-1}$ for $2 \leq n \in N$. With these two input layers, the neural network can determine which direction the player is heading to. However, before determining the relationship between states $s_{l,m,n}$ and $s_{l,m,n-1}$, the network has to understand and recognize the information delivered by the probability distributions in each of the states. Therefore, right after each of the input layers, there is a hidden layer to analyze the input data. After the input analyzation layers, we concatenate the output of those two hidden layers with respect to the second axis of the data.

Followed by the concatenation layer are three sequential fully connected hidden layers. These three hidden layers are for the neural network to determine the relationship between the two input data. Since the difference between the two input data might result in some negative values, the activation function for these three hidden layers should allow negative values to pass through. Moreover, as mentioned at the beginning of Section III, the neural network works similarly as a classifier. This means the output values should only be positive numbers. Hence, the activation function for these three hidden layers are set to the leaky rectified linear unit (Leaky ReLU) [21], i.e.,

$$f(x) = \begin{cases} 0.2x, & \text{for } x < 0, \\ x, & \text{for } x \geq 0. \end{cases} \tag{7}$$

This allowed certain negative values to pass through and also allowed the neural network to focus more on the positive values to match the output values.

What follows next is also another three sequential fully connected hidden layers. However, the difference is that the number of units in each of these hidden layers is twice the size of the layer in the previous three sequential hidden layers. Also, the activation functions in these three hidden layers are rectified linear unit (ReLU). The main purpose of these three hidden layers is to learn the joint distribution among the players in order for the deep neural network to determine the optimal action to perform.

Finally, the output layer is a fully connected layer with J units which is the same as the number of actions in the action set $\mathbb{A}$. The activation function in the output layer is the softmax activation function, i.e.,

$$f(\mathbf{x})_i = \frac{e^{x_i}}{\sum_{j=1}^{J} e^{x_j}} \tag{8}$$

for $i = 1, \ldots, J$ where x_i is the element of the input vector $\mathbf{x}$. The softmax activation function ensures the elements in the output vector to be within the range $(0, 1)$ and also ensures the L^1-norm of the output vector is equal to 1. In other words, the output vector represents the probability distribution of the possible actions based on the input states.

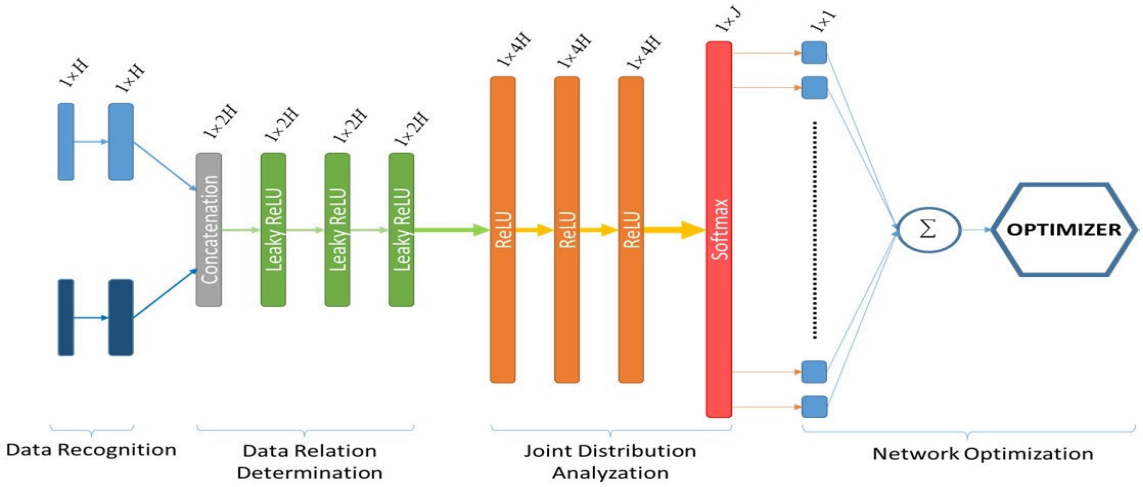


FIGURE 5. Policy-based deep reinforcement learning neural network structure.

D. MODEL TRAINING

We have discussed the structure of our proposed policy-based deep reinforcement learning neural network in the previous subsection where the output of the network is the probability distribution of the actions. Now we are going to discuss how we update the weights in the neural network. First of all, the output of the deep learning neural network works similarly as most of the multi-class classification but not exactly the same. In most of the multi-class classification, the loss function will be fed in a batch of output data along with the corresponding one-hot encoded labels. After each training via an optimizer, the probability for the correct predict label will be increased. There is nothing wrong with this process. However, most of the time, even if the input data only contains one category, some probabilities for the incorrectly predicted labels will also be slightly increased. As mentioned before in Section II-B, the way we choose action a_k is not based on the arguments of the maxima among the output vector but instead the probability distribution of the actions given by the output vector. Hence, when we increase the probability of one of the labels, we need to make sure the other probabilities do not increase. Therefore, the way we achieved this property is to calculate the loss for each of the probabilities. Here, we use the logarithmic loss for our loss function and scale the value by a weight. The absolute value of the weight indicates the change rate of the probability and the sign of the weight indicates whether the probability should move tower or away from the target for the positive and negative sign, respectively. The best and valid value to present as the weight for the loss function in our system is the reward $r_{l,m,n}$ that we have mentioned in Section III-B. However, we will need to perform post-processing on the reward data in order for our model to efficiently learn the joint distribution between the players.

In the post-processing procedure, we first multiply the reward $r_{l,m,n}$ by a discount factor γ to the power of $N - n$ and get $\bar{r}_{l,m,n}$, i.e.,

$$\bar{r}_{l,m,n} = \gamma^{(N-n)} r_{l,m,n} \tag{9}$$

where $\gamma \in [0, 1]$. This is due to the different importance of each action that the player performed during the interaction. Action $a_{l,m,N-1}$ will have more influence on which state player p_l will be ended up on than the action $a_{l,m,N-2}$, and so on and so forth. In other words, we can consider actions $a_{l,m,n}$ for $n = 1, 2, \ldots, N$ is a time series data set since action $a_{l,m,n}$ will always be performed after action $a_{l,m,n-1}$. In other words, $a_{l,m,N-1}$ will be the most recent action that player p_l performed in round m. Based on the property of time series data, longer time horizons have much more variance as they include more irrelevant information, while short time horizons are biased towards only short-term gains. Hence, we need a discount factor to reduce the variance in the data set. With the calculated rewards $\bar{r}_{l,m,n}$, we get another set $\bar{R}$.

Now, the discount factor γ has taken care of the variance caused by the long-term gains in the data set. There is another concern from the player's policy π_l itself. As mentioned before in Section II-B, player p_l will sample an action according to the probability distribution given by their policy π_i for the state that the player is currently in. This means that there is a chance that some actions will never or rarely be chosen based on the probability distribution. The problem with using the rewards from $\bar{R}$ to update the weights of the deep learning neural network is that the deep learning neural network will only be increasing the probabilities for sampled actions since reward $\bar{r}_{l,m,n}$ is always positive. Hence, the probabilities for those actions that have not been sampled or rarely sampled will be decreased throughout the training process. In order to overcome this problem, we will subtract the reward by a bias. Here, the bias for the reward is designed to be the mean $\mu_{l,n}$ of the rewards of the n-th observed state in the M rounds of interaction. We also divided the reward by

the standard deviation $\sigma_{l,n}$ of the rewards of the n-th observed state in the M rounds of interaction. This will normalize the data for stability purposes during the training process. The post-processed reward $\hat{r}_{l,m,n} \in \hat{R}$ is expression as the following,

$$\hat{r}_{l,m,n} = \frac{\bar{r}_{l,m,n} - \mu_{l,m}}{\sigma_{l,m}} \tag{10}$$

where

$$\sigma_{l,n} = \sqrt{\frac{1}{M}\sum_{m=1}^{M}\left(\bar{r}_{l,m,n} - \mu_{l,n}\right)^2} \tag{11}$$

and

$$\mu_{l,n} = \frac{1}{M}\sum_{m=1}^{M}\bar{r}_{l,m,n}. \tag{12}$$

After calculating the loss for each of the probabilities, we will sum up the losses and feed into the optimizer. In our proposed deep learning neural network, we applied Adam optimization with a learning rate of 0.001 to update the weights. We will keep repeating the process of data collection and model training until the player's last state in each round becomes stable before going to the next process of estimating the payoff vector of the other players.

E. OPPONENT PAYOFF ESTIMATION

Once we have determined the correlated equilibrium state s_{ce}, the next step will be estimating the payoff vectors for the other player by using the properties from both the decision choosing rationality in game theory and the force of tension. In addition, starting from this section, each player will do the calculation on their own. This means there will be no more information sharing among the players. Keep in mind that the payoff vectors estimation process works for every player in the system. However, as mentioned before, we will be discussing the process in player p_{main}'s point of view. Therefore, we only know the information about payoff vector V_{main} and the probability distribution that is the correlated equilibrium state s_{ce} at this point.

First, the correlated equilibrium distribution is based on the rational decision chosen by both of the players. The optimal probability distribution $\mathbb{P}$ for both of the players is the probability distribution that will maximize the joint reward of both of the players instead maximize their own reward. Therefore, under the circumstance where we know the exact payoff vectors for both of the players, the objective function $O_{old}(P)$ for finding the optimal probability distribution $\mathbb{P}$ is expressed as follows,

$$\max_{P} \quad O_{old}(P) = P \cdot (V_{main})^T + P \cdot (V_t)^T \tag{13}$$

$$\text{subject to } \rho_h \in P \geq 0,$$

$$\sum_{h=1}^{H}\rho_h = 1, \tag{14}$$

and constraint (1).

Now we want to reverse the calculation to estimate the payoff vectors V_t when given the optimal probability distribution $\mathbb{iP}$, the rationality conditions in (1) should still be fulfilled and the objective function is still be maximizing the reward summation of all players but based on the payoff vectors, i.e.,

$$\max_{V_t} \quad O_{new}(V_t) = \mathbb{P} \cdot (V_t)^T. \tag{15}$$

Here, we can eliminate the reward of player p_{main} since it is just a constant.

However, with (1) being the only constraint, the objective function will only be maximizing the payoff vectors as if each player's decision is independent with each other. As mentioned before, the constraint in (1) only gives a convex hull boundary on the probability distributions over the pure strategies. This means for each player, there will be at least one probability distribution that will benefit himself or herself but not necessary for others. Hence, even though the solution $\bar{V}_t$ will maximize the objective function $O_{new}(V_t)$, the solution $\bar{V}_t$ will never equal to the actual payoff vectors with (1) being the only constraint. Moreover, if we try to solve the objective function $O_{old}(P)$ with $\bar{V}_t$, the optimal solution will never be equal to $\mathbb{P}$. Hence, there need to be some other constraints that indicate the relationship between the players' payoff vectors. This is where the idea of the force of tension steps in.

As mentioned before in Section II-A, a distribution is a correlated equilibrium distribution if and only if it lies within the convex hull based on (1). In order to get the optimal probability distribution $\mathbb{P}$, the player needs to maximize the objective function $O_{old}(P)$. In the force of tension point of view, we can consider optimal probability distribution $\mathbb{P}$ as an equilibrium point where the tensions for each decision set D_h from both of the players have reached an equilibrium. Moreover, since the values in the optimal solution $\mathbb{P}$ are probabilities, we can treat those probabilities as the preferences of each decision set D_h for the players. Based on this idea, we can list out the constraints regarding the relationship between the rewards in the payoff vector. There are a few types of constraints that we will be discussing in this section. For easier understanding, starting from here, we will use the game setup shown in Fig. 6 as the scenario where player p_1 and player p_2 are "Player 1" and "Player 2" in Fig. 6, respectively. Keep in mind that we do not know the payoff vector V_2 at this time.

We first need to reorder the elements in the payoff vectors by sorting both of the payoff vectors and the probability distribution $\mathbb{P}$ based on the payoff vector V_1 in the ascending order. The payoff vector V_2 has to future rearrange the elements where the first element will be move to the end of the vector. After the reordering process, we get three new vectors $V_1 \rightarrow \bar{V}_1$, $V_2 \rightarrow \bar{V}_2$, $\mathbb{P} \rightarrow \bar{\mathbb{P}}$, and $\mathbb{D} \rightarrow \bar{\mathbb{D}}$, i.e.,

$$\bar{V}_1 = \left\{\bar{v}_{1,1}, \bar{v}_{1,2}, \bar{v}_{1,3}, \bar{v}_{1,4}\right\} = \left\{v_{1,4}, v_{1,2}, v_{1,1}, v_{1,3}\right\}, \tag{16}$$

$$\bar{V}_2 = \left\{\bar{v}_{2,1}, \bar{v}_{2,2}, \bar{v}_{2,3}, \bar{v}_{2,4}\right\} = \left\{v_{2,4}, v_{2,2}, v_{2,1}, v_{2,3}\right\}, \tag{17}$$

$$\bar{\mathbb{P}} = \{\bar{\rho}_1, \bar{\rho}_2, \bar{\rho}_3, \bar{\rho}_4\} = \{\rho_4, \rho_2, \rho_1, \rho_3\}, \tag{18}$$

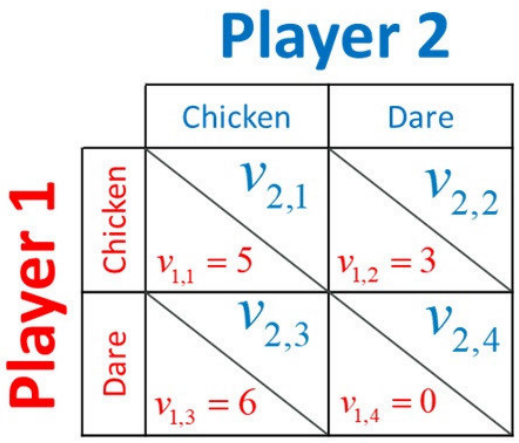


FIGURE 6. Game setup for payoff vector estimation.

$$\begin{aligned}
&\qquad \bar{D}_1 \quad \bar{D}_2 \quad \bar{D}_3 \quad \bar{D}_4 \quad \bar{D}_1 \\
&\bar{\mathbb{P}} = \quad \bar{\rho}_1 \quad \bar{\rho}_2 \quad \bar{\rho}_3 \quad \bar{\rho}_4 \quad \bar{\rho}_1 \\
&\bar{V}_1 = \quad \bar{v}_{1,1} \rightarrow \bar{v}_{1,2} \rightarrow \bar{v}_{1,3} \rightarrow \bar{v}_{1,4} \\
&\qquad\qquad \bar{v}_{2,2} \leftarrow \bar{v}_{2,3} \leftarrow \bar{v}_{2,4} \leftarrow \bar{v}_{2,1} \quad = \bar{V}_2
\end{aligned}$$

FIGURE 7. Payoff vector reorder.

and

$$\bar{\mathbb{D}} = \left\{\bar{D}_1, \bar{D}_2, \bar{D}_3, \bar{D}_4\right\} = \{D_4, D_2, D_1, D_3\}. \tag{19}$$

The idea here is that players will try to move from the decision set with the lowest reward to the decision set with the highest reward where the direction for player p_1 is from the left to the right and the right to the left for player p_2. This setup is due to the monotonously increasing property for the convex set and function [22], [23]. The illustration is shown in Fig. 7 where we also showed the associated decision set for each reward. The arrow in the figure indicates the direction of the force that the player puts on the decision state. With this setup, we can start to discuss the types of conditions.

The first constraint type is regarding one outgoing force for each decision set from each player. We look at the first decision set D_h where the corresponding rewards for both player p_1 and player p_2 have an outgoing arrow. We then subtract the force from player p_1 by the force from player p_2 to get the net force $\vec{f}_h$, i.e.,

$$\vec{f}_h = \bar{\rho}_{h+1}\left(\bar{v}_{1,h+1} - \bar{v}_{1,h}\right) - \bar{\rho}_{h-1}\left(\bar{v}_{2,h-1} - \bar{v}_{2,h}\right) \tag{20}$$

We can than list a constraint based on $\delta_{\vec{f}_h}$ where

$$\delta_{\vec{f}_h} = \begin{cases} \vec{f}_h \geq 0, & \text{if } \bar{\rho}_{h-1} < \bar{\rho}_h \leq \bar{\rho}_{h+1}, \\ \vec{f}_h = 0, & \text{if } \bar{\rho}_{h-1} \leq \bar{\rho}_h \text{ and } \bar{\rho}_{h+1} \leq \bar{\rho}_h, \\ \vec{f}_h \leq 0, & \text{if } \bar{\rho}_{h-1} > \bar{\rho}_h \geq \bar{\rho}_{h+1}. \end{cases} \tag{21}$$

The reason for different signs of outgoing net force $\vec{f}_h$ is due to the preference according to probability $\bar{\rho}_h$. Reward $\bar{v}_h$ with higher probability $\bar{\rho}_h$ means it has a higher preference to the player which also means that the player will be willing to move to from a reward with lower preference. Take the reward $\bar{v}_{1,3}$ for example. If probability $\bar{\rho}_4$ is greater than probabilities $\bar{\rho}_3$ and $\bar{\rho}_2$, this means p_1 will be trying to move from $\bar{v}_{1,3}$ to $\bar{v}_{1,4}$. Since no player will be against to himself or herself, the only force that that will be against to player p_1 will came from player p_2 which is the force from $\bar{v}_{2,3}$ to $\bar{v}_{2,2}$. However, since the preference on reward $\bar{v}_{2,4}$ for player p_2 is also higher than reward $\bar{v}_{2,3}$ and $\bar{v}_{2,2}$, player p_2 will also be moving to reward $\bar{v}_{2,4}$ as well. Therefore, the force from reward $\bar{v}_{1,3}$ to reward $\bar{v}_{1,4}$ should be greater or equal to the force from player p_2 from reward $\bar{v}_{2,3}$ to reward $\bar{v}_{2,4}$. The same idea applied on other two signs in (21).

The second constraint type is regarding one incoming force for each decision set from each player. We look at the first decision set D_h where the corresponding rewards for both player p_1 and player p_2 has an incoming arrow. We then subtract the force from player p_1 by the force from player p_2 to get the difference $\overleftarrow{f}_h$, i.e.,

$$\overleftarrow{f}_h = \bar{\rho}_h\left(\bar{v}_{1,h} - \bar{v}_{1,h-1}\right) - \bar{\rho}_h\left(\bar{v}_{2,h} - \bar{v}_{2,h-1}\right) \tag{22}$$

for

$$\bar{v}_{2,h-1} = \begin{cases} \bar{v}_{2,1}, & \text{if } h = H, \\ \bar{v}_{2,h-1}, & \text{otherwise.} \end{cases} \tag{23}$$

We also can than list a constraint based on $\overleftarrow{f}_h$ where

$$\delta_{\overleftarrow{f}_h} = \begin{cases} \overleftarrow{f}_h \geq 0, & \text{if } \bar{\rho}_{h-1} > \bar{\rho}_h \geq \bar{\rho}_{h+1}, \\ \overleftarrow{f}_h = 0, & \text{if } \bar{\rho}_{h-1} \leq \bar{\rho}_h \text{ and } \bar{\rho}_{h+1} \leq \bar{\rho}_h, \\ \overleftarrow{f}_h \leq 0, & \text{if } \bar{\rho}_{h-1} < \bar{\rho}_h \leq \bar{\rho}_{h+1}. \end{cases} \tag{24}$$

for

$$\bar{\rho}_{h+1} = \begin{cases} \bar{\rho}_1, & \text{if } h = H, \\ \bar{\rho}_{h-1}, & \text{otherwise.} \end{cases} \tag{25}$$

The different signs of the incoming net force $\overleftarrow{f}_h$ is the same idea as we mentioned in constraint (21).

We will repeat these two types of constraints with L continuous outgoing force for the first type constraint and L continuous incoming force for the second type constraint for $L = \left\{2, 3, \ldots, \left\lfloor \frac{H}{2} \right\rfloor\right\}$. Finally, we will add few more constraints as follows,

$$v_{i,h} \geq 0, \tag{26}$$

$$\sum_{h=1}^{H} v_{i,h} = 1, \tag{27}$$

and the equality in the Nash equilibrium with the mixed strategy has to be established.

Algorithm 1 Overall Proposed Method

Definition:
$D_{i,j}$ is a set of all possible decision combination $d_{i,j,k}$ of all players in P except player p_i and p_j where $i \neq j$;
$NE_{i,j,k}$ be the set of the Nash equilibria with the mixed strategy between player p_i and player p_j when other players decision is fixed to $d_{i,j,k}$;
Statement $f\ (p_i, \pi_i)$ indicates player p_i will interact with the environment with policy π_i based on p_i's DNN to obtain N states for M rounds and return C_i and S_i;
Function $\Re\left(\hat{S}, V_i\right)$ calculates the post-processed reward set $\hat{R}_i$ for player p_i;
Initialization:
for each $p_i \in P$ **do**
  for each $p_j \in P$ **AND** $i \neq j$ **do**
    for each $d_{i,j,k} \in D_{i,j}$ **do**
      if $\{p_j, p_i, d_{j,i,k}\} \notin \Im$ **then**
        $\Im$ append $\{p_i, p_j, d_{i,j,k}\}$;
      end if
    end for
  end for
end for
while $\Im \neq \emptyset$ **do**
  for $\{p_i, p_j, d_{i,j,k}\}$ in $\Im$ **do**
    Other players' decision set to $d_{i,j,k}$;
    if V_i **OR** V_j is known **then**
      if size$(NE_{main,i,d_{i,j,k}}) \geq 2$ **then**
        if V_i is known **then**
          $p_m = p_i$ and $p_a = p_j$;
        else
          $p_m = p_j$ and $p_a = p_i$;
        end if
        $\tilde{\mathbb{P}} = \varnothing$
        while $\tilde{\mathbb{P}}$ not stable **do**
          $C_m, S_m = f\ (p_m, \pi_m)$;
          $C_a, S_a = f\ (p_a, \pi_a)$;
          $\hat{S} = 0.5\ (S_m + S_a)$;
          $\hat{R}_m = \Re\left(\hat{S}, V_m\right)$;
          $\hat{R}_a = \Re\left(\hat{S}, V_a\right)$;
          Update π_m with C_m, S_m, and $\hat{R}_m$;
          Update π_i with C_a, S_a, and $\hat{R}_a$;
          $\tilde{\mathbb{P}} = \hat{S}\ [-1]$;
        end while
        Set proposed constraints;
        Maximize $O\left(\tilde{V}_a\right)$ with respect to $\tilde{V}_a$;
      end if
      Remove $\{p_i, p_j, d_{i,j,k}\}$ from $\Im$;
    end if
  end for
end while
Compute correlated equilibrium among players with $\mathbb{V}$;

F. METHOD SUMMARY

Now we have gone through the details of each part of our proposed method. We can see that from our proposed method, all players can learn the joint distribution between each other with the policy-based deep reinforcement learning model and reach the correlated equilibrium. The correlated equilibrium probability distribution they reached allowed the player to obtain the maximum reward under a rational decision making in the game. With the correlated equilibrium probability distribution, the player can calculate the opponent's payoff vector based on our proposed mathematical model which involves the idea of the force of tension. The overall process is summarized in Algorithm 1 where we can see that not all players will interact with all other players. However, we still can compute the correlated equilibrium among those players who do not have interaction at all since we already have the payoff vectors of those players. This reduced the computation in respect of the entire system.

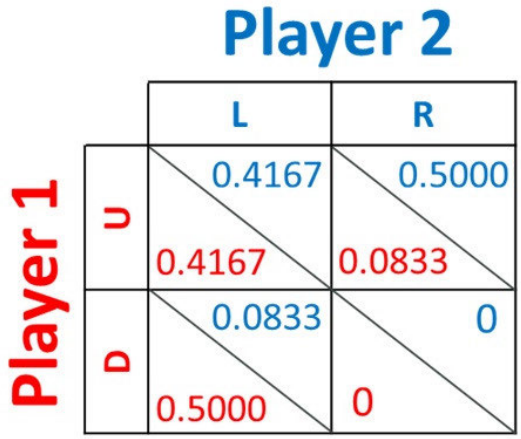


FIGURE 8. Simulation setup.

IV. PERFORMANCE EVALUATION

A. SIMULATION SETUP

In our simulation, the game environment is set up based on a two-person game where the players are player p_1 and player p_2. The decisions available to player p_1 are "U" and "D" where the decisions available to player p_2 are "L" and "R". Therefore, the decision sets D_h for $h = \{1, 2, 3, 4\}$ are (U,L), (U,R), (D,L), and (D,R), respectively. The payoff vectors $V_1 = \{0.3571, 0.4286, 0.2143, 0\}$ and $V_2 = \{0.3571, 0.2143, 0.4286, 0\}$. The illustration of the setup is shown in Fig. 8. The step size ϑ is set to 0.005. Therefore, the action set A is the permutation set among the elements in each of the three sets $\{-0.005, 0, 0.005\}$, $\{-0.005, 0, 0.005\}$, and $\{-0.005, 0, 0.005\}$ which has total of 27 actions. The player will interact with the environment for $M = 40$ rounds and will perform $N = 200$ actions in each round.

B. NUMERICAL RESULTS

With the setup of our simulation, we let the player p_1 be the main player. This means, at this point, we only know the payoff vector V_1 but not the payoff vector V_2. However,

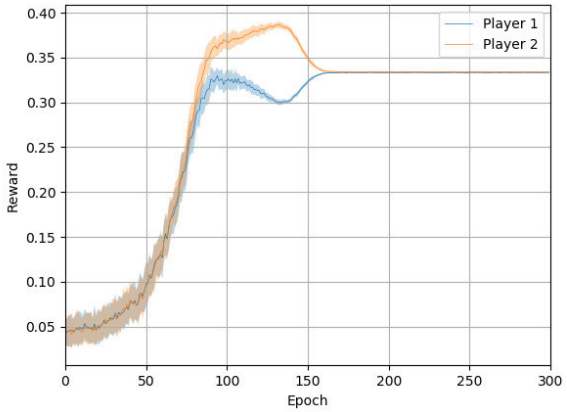


FIGURE 9. Average reward gained in each epoch.

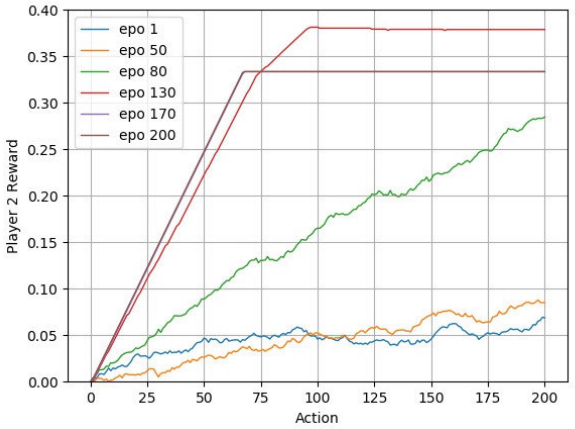


FIGURE 11. Player 2 reward increasing after learning.

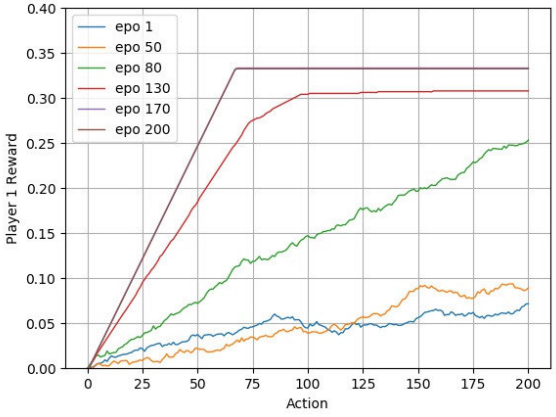


FIGURE 10. Player 1 reward increasing after learning.

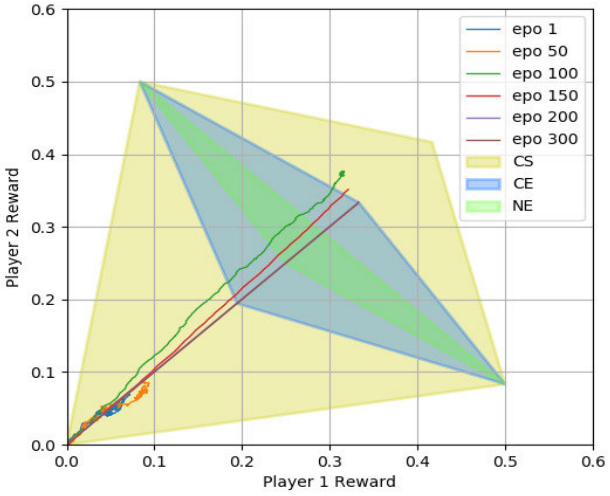


FIGURE 12. Track of both players after learning.

in order to illustrate the relation between both players, we will also show the data from player p_2 in the later figures.

We first let two players interact with the environment for more than 300 epochs where the player interacts 40 rounds in each epoch. As we can see in Fig. 9, both players are gaining small and unstable rewards at the beginning. As they interact with the environment with their own policy-based deep reinforcement learning neural network in each epoch, they learn from the environment and the opponent and receive higher rewards as the number of epochs increases. In addition, the more they learn, the faster they reach the higher reward. The results were shown in Fig. 10 and Fig. 11 for player p_1 and player p_2, respectively. However, around 80 epochs where they reach mixed Nash equilibrium, they started to diverge from each other. More specifically, player p_2 starts to pull player p_1 tower decision set $D_2 = (\mathrm{U,R})$. Around epoch 130, player p_1 realizes that he or she must act to stop the reduction on his or her reward. Hence, player p_1 starts to pull back player p_2. Around epoch 170, both players reach an equilibrium point where the state at the end of each round of interaction was stabled at $\left\{\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, 0\right\}$. In Fig. 12, we can see the track of the interaction from both players in different epochs where the areas of "CS", "CE", and "NE" represent the set of correlated strategies, the convex hull of the Nash equilibrium, and the convex hull of correlated equilibrium, respectively. We can see that as they observe the environment and learn the opponent's behavior, they achieve outside of the Nash equilibrium convex hull and become stable at the correlated equilibrium where maximizes both players' reward. Even though we simulate more than 500 epochs, the final state remains stable after epoch 170. Hence, we only showed the result of up to 300 epochs in each figure.

Once the both players have reached a stable state after well trained, we use this state as the estimate correlated equilibrium probability distribution where $\tilde{\mathbb{P}} = \left\{\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, 0\right\}$ to determine the estimate payoff table for player p_2. Moreover, if we calculate the correlated equilibrium with the objective function (13), we can see that the solution $\mathbb{P} = \left\{\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, 0\right\}$ is exactly the same as the probability distribution in state s.

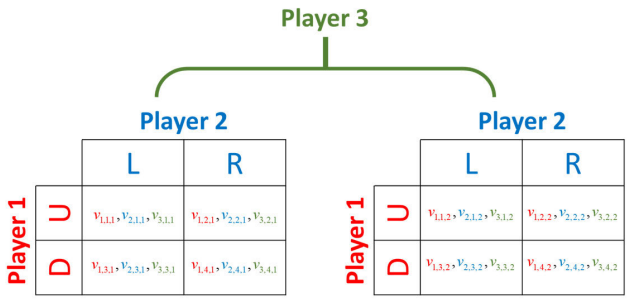


FIGURE 13. Three players game.

With the estimated correlated equilibrium probability distribution $\tilde{\mathbb{P}}$, we are able to list out the objective function and the constraints based on the steps in Section III-E. By solving this constrained linear multivariable function, we get the estimate payoff vector $\tilde{V}_2 = \{0.4167, 0.5000, 0.0833, 0\}$ of player p_2. Compared with the actual payoff vector $V_2 = \{0.4167, 0.5000, 0.0833, 0\}$, we can see that there is no error between these two vectors. Hence, we have successfully computed the payoff vector for the other player.

In the beginning, the players can only apply the mixed strategy to reach the Nash equilibrium and obtain the reward 0.25 and 0.25 for player p_1 and player p_2, respectively. Now, with our proposed deep reinforcement learning model, both players can obtain a higher reward of 0.3333 and 0.3333 for player p_1 and player p_2, respectively, by reaching the estimated correlated equilibrium.

Now, in the case that we have three players in the game as shown in Fig. 13 where player three p_3 can choose the left or right matrix they will be playing. With player p_3 choosing the left matrix, we will do the same thing as before where we let player p_1 interact with player p_2 and get the payoff vector V_2. However, we will also let either player p_1 interact with player p_3. By doing so, we can get the payoff vector V_3. By having the payoff vectors V_2 and V_3, we can compute the correlated equilibrium between player p_2 and player p_3 without them interacting with each other. The same thing for the case where the player p_3 chooses the right matrix. This way, we reduced the computation of the interaction between player p_2 and player p_3 in the entire process.

V. DISCUSSION OF APPLICATION SCENARIOS

The correlated equilibrium solution concept has been adopted in many different areas. However, most of them still relied on either a centralized system or information sharing between parties. In this section, we will be discussing the applications that involve the correlated equilibrium in wireless communication, smart grid, and resource allocation, where our proposed scheme can reduce signaling and improve the performance.

We first look at the applications in wireless communication where the most common wireless technologies use radio waves. With radio waves, the transmission distances can be as short as a few centimeters such as NFC and as far as millions of miles for deep-space radio communications. However, there are some challenges in wireless communication such as signal interference, data throughput, and more need to be solved in order to have a stable communication between devices. The author in [24] proposed a distributed cooperation policy selection scheme for interference to perform subcarrier assignment for uplink multi-cell OFDMA systems by adopting the correlated equilibrium solution concept that achieves better performance by allowing each user to consider the joint distribution among users' actions to minimize the interference. Also, in [16], the author showed by using a game-theoretic learning algorithm which is based on correlation equilibrium in the problem of multi-user multichannel access in distributed high-frequency diversity communication networks can completely avoid interference and get optimal throughput. Moreover, it also guarantees fairness among all user equipment. Besides the concerns on the interference in wireless communication, the data throughput is also a main concern as well. In the work [25], the author showed a game-theoretic approach based on correlated equilibrium and regret-matching learning can provide significant gains in terms of average cell throughput in the Monte Carlo simulations of Long Term Evolution - Advanced like system.

Next, the smart grid is an electrical grid that includes a variety of operation and energy measures including smart meters, smart appliances, renewable energy resources, and energy-efficient resources [26]. Electronic power conditioning and control of the production and distribution of electricity are important aspects of the smart grid [27]. In energy-aware ad hoc networks, energy efficiency is a crucial requirement. The authors in [28] present a cooperative behavior control scheme based on the correlated equilibrium to reduce and balance energy consumption.

Finally, in the resource allocation problem which arises in many application domains ranging from the social sciences to engineering [29], [30], the objective is to allocate resources to different areas under some concerns such as energy consumption, fairness, and more. In [31], the authors proposed an energy-efficient resource allocation scheme by using the correlated equilibrium. Furthermore, the authors present a linear programming method and a distributed algorithm based on the regret matching procedure to implement the CE. With their proposed method, they can determine the desired resource allocation in an uplink orthogonal frequency division multiple access (OFDMA) system.

From the above applications that we discussed, we can see that the correlated equilibrium solution concept can highly improve the performance in many areas. However, as mentioned before, most of the applications that adopt the correlated equilibrium solution concept still rely on a centralized system. The issue has been that the system will need to allow all the nodes or users within the system to be able to communicate with each other or through a master node. This means that the data transmission and computation power are two of the main factors that limited the system's bottleneck.

Moreover, information sharing among the system will also lead to some privacy issues. With our proposed method, these issues could be overcome since each node or user does not have to directly communicate with each other to share their private information but to learn other's behavior and reach the correlated equilibrium within the system which could highly improve the system performance.

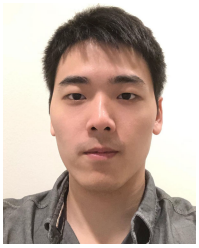
VI. CONCLUSION

In this work, we have successfully overcome the issue regarding the public signal in the correlated equilibrium solution concept through our proposed policy-based deep reinforcement learning model. We can see from the numerical results that the model learned the joint distribution of all the players and reached the state of the correlated equilibrium probability distribution. Moreover, with the information from the player himself or herself and the correlated equilibrium probability distribution, achieved from the deep reinforcement learning model, we propose a mathematical model to estimate the payoff vector of the other player, which combines the concept of the rationality in game theory and the force of tension. Once we have the estimated payoff vectors of other players, we can compute the correlated equilibrium among the player and the other players who have not interacted with each other. This paper combines game theory with machine learning in the sense that the proposed machine learning learns what is the game player's payoff, instead of just categorizing the strategies of actions according to the current situation. Moreover, unlike most of the systems out there that adopted the correlated equilibrium solution concept, our proposed system does not require a centralized synchronized system. However, due to the limited information sharing in our system, it might lead to performance loss for a small system that could ignore the data transmission latency or privacy concern. In our future work, we will be looking into these concerns and improve our system model to overcome it.

REFERENCES

[1] S. Bowles, *Microeconomics: Behavior, Institutions, and Evolution*. Princeton, NJ, USA: Princeton Univ. Press, 2009.
[2] R. Branzei, D. Dimitrov, and S. Tijs, *Models in Cooperative Game Theory*, vol. 556. Springer, 2008. [Online]. Available: https://www.springer.com/gp/book/9783540779537
[3] Z. Han, D. Niyato, W. Saad, T. Başar, and A. Hjørungnes, *Game Theory in Wireless and Communication Networks: Theory, Models, and Applications*. Cambridge, U.K.: Cambridge Univ. Press, 2011. [Online]. Available: https://books.google.com/books?id=mvaUAwAAQBAJ
[4] Z. Han, D. Niyato, W. Saad, and T. Başar, *Game Theory for Next Generation Wireless and Communication Networks: Modeling, Analysis, and Design*. Cambridge, U.K.: Cambridge Univ. Press, 2019.
[5] M. J. Osborne and A. Rubinstein, *A Course in Game Theory*. Cambridge, MA, USA: MIT Press, 1994.
[6] R. J. Aumann, "Subjectivity and correlation in randomized strategies," *J. Math. Econ.*, vol. 1, no. 1, pp. 67–96, Mar. 1974. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0304406874900378
[7] D. Fudenberg and J. Tirole, *Game Theory*. Cambridge, MA, USA: Massachusetts Institute Technology, vol. 393, no. 12, 1991, p. 80.
[8] Z. Han, C. Pandana, and K. J. R. Liu, "Distributive opportunistic spectrum access for cognitive radio using correlated equilibrium and no-regret learning," in *Proc. IEEE Wireless Commun. Netw. Conf.*, Hong Kong, Mar. 2007, pp. 11–15.
[9] M. Löschenbrand, "Finding multiple Nash equilibria via machine learning-supported Gröbner bases," *Eur. J. Oper. Res.*, vol. 284, no. 3, pp. 1178–1189, Aug. 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377221720300783
[10] T. G. Dietterich, "Machine learning for sequential data: A review," in *Structural, Syntactic, and Statistical Pattern Recognition*, T. Caelli, A. Amin, R. P. W. Duin, D. de Ridder, and M. Kamel, Eds. Berlin, Germany: Springer, 2002, pp. 15–30.
[11] K. C. Tsai, L. Wang, and Z. Han, "Caching for mobile social networks with deep learning: Twitter analysis for 2016 U.S. election," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 1, pp. 193–204, Jan. 2020.
[12] K. C. Tsai, W. Hu, X. Wu, J. Chen, and Z. Han, "Automatic first arrival picking via deep learning with human interactive learning," *IEEE Trans. Geosci. Remote Sens.*, vol. 58, no. 2, pp. 1380–1391, Feb. 2020.
[13] L. Deng and X. Li, "Machine learning paradigms for speech recognition: An overview," *IEEE Trans. Audio, Speech, Language Process.*, vol. 21, no. 5, pp. 1060–1089, May 2013.
[14] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 279–292, 1992.
[15] R. Dearden, N. Friedman, and S. Russell, "Bayesian Q-learning," in *Proc. Aaai/iaai*, 1998, pp. 761–768.
[16] W. Li, Y. Xu, Y. Cheng, Y. Yang, X. Chen, M. Wang, and D. Liu, "Distributed multichannel access in high-frequency diversity networks: A multi-agent learning approach with correlated equilibrium," *IEEE Access*, vol. 7, pp. 85581–85593, 2019.
[17] M. Tan, C. Han, X. Zhang, L. Guo, and T. Yu, "Hierarchically correlated equilibrium Q-learning for multi-area decentralized collaborative reactive power optimization," *CSEE J. Power Energy Syst.*, vol. 2, no. 3, pp. 65–72, 2016.
[18] T. Yu, H. Z. Wang, B. Zhou, K. W. Chan, and J. Tang, "Multi-agent correlated equilibrium Q(λ) learning for coordinated smart generation control of interconnected power grids," *IEEE Trans. Power Syst.*, vol. 30, no. 4, pp. 1669–1679, Jul. 2015.
[19] A. Brandenburger and E. Dekel, "Rationalizability and correlated equilibria," *Econometrica*, vol. 55, no. 6, pp. 1391–1402, 1987. [Online]. Available: http://www.jstor.org/stable/1913562
[20] R. J. Aumann, "Correlated equilibrium as an expression of Bayesian rationality," *Econometrica*, vol. 55, no. 1, pp. 1–18, 1987. [Online]. Available: http://www.jstor.org/stable/1911154
[21] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. ICML*, Atlanta, GA, USA, 2013, vol. 30, no. 1, p. 3.
[22] G. Szekeres, "On a property of monotone and convex functions," *Proc. Amer. Math. Soc.*, vol. 7, no. 3, pp. 351–353, 1956. [Online]. Available: http://www.jstor.org/stable/2032739
[23] S. Boyd, S. Boyd, and L. Vandenberghe, *Convex Optimization* (Berichte Über Verteilte Messysteme). Cambridge, U.K.: Cambridge Univ. Press, 2004. [Online]. Available: https://books.google.com.tw/books?id=mYm0bLd3fcoC
[24] J. Zheng, Y. Cai, and D. Wu, "A correlated-equilibrium-based subcarrier allocation scheme for interference minimization in multi-cell OFDMA systems," in *Proc. Int. Conf. Wireless Commun. Signal Process. (WCSP)*, Nanjing, China, Nov. 2011, pp. 1–6.
[25] P. Sroka and A. Kliks, "Distributed interference mitigation in two-tier wireless networks using correlated equilibrium and regret-matching learning," in *Proc. Eur. Conf. Netw. Commun. (EuCNC)*, Bologna, Italy, Jun. 2014, pp. 1–5.
[26] M. S. Saleh, A. Althaibani, Y. Esa, Y. Mhandi, and A. A. Mohamed, "Impact of clustering microgrids on their stability and resilience during blackouts," in *Proc. Int. Conf. Smart Grid Clean Energy Technol. (ICSGCE)*, Offenburg, Germany, Oct. 2015, pp. 195–200.
[27] M. Lee, O. Aslam, B. Foster, D. Kathan, J. Kwok, L. Medearis, R. Palmer, P. Sporborg, and M. Tita, "Assessment of demand response and advanced metering," Federal Energy Regulatory Commission, Washington, DC, USA, Tech. Rep., 2013.
[28] D. Wu, Y. Cai, L. Zhou, Z. Zheng, and B. Zheng, "Cooperative strategies for energy-aware ad hoc networks: A correlated-equilibrium game-theoretical approach," *IEEE Trans. Veh. Technol.*, vol. 62, no. 5, pp. 2303–2314, Jun. 2013.
[29] B. Kauffmann, F. Baccelli, A. Chaintreau, V. Mhatre, K. Papagiannaki, and C. Diot, "Measurement-based self organization of interfering 802.11 wireless access networks," in *Proc. IEEE INFOCOM-26th IEEE Int. Conf. Comput. Commun.*, Anchorage, AK, USA, May 2007, pp. 1451–1459.

[30] J. R. Marden and T. Roughgarden, "Generalized efficiency bounds in distributed resource allocation," *IEEE Trans. Autom. Control*, vol. 59, no. 3, pp. 571–584, Mar. 2014.
[31] D. Wu, L. Zhou, and Y. Cai, "Energy-efficient resource allocation for uplink orthogonal frequency division multiple access systems using correlated equilibrium," *IET Commun.*, vol. 6, no. 6, pp. 659–667, Apr. 2012.



KUO CHUN TSAI (Senior Member, IEEE) received the B.S. degree from The University of Texas at Austin, Austin, TX, USA, in June 2016, and the Ph.D. degree from the Department of Electronics and Computer Engineering, University of Houston, Houston, TX, USA, in May 2020. His research interests include machine learning, and computing and data processing on hardware and software.

ZHU HAN (Fellow, IEEE) received the B.S. degree in electronics engineering from Tsinghua University in 1997, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Maryland, College Park, in 1999 and 2003, respectively.

From 2000 to 2002, he was a Research and Development Engineer of JDSU, Germantown, MD, USA. From 2003 to 2006, he was a Research Associate at the University of Maryland. From 2006 to 2008, he was an Assistant Professor at Boise State University, ID, USA. He is currently a John and Rebecca Moores Professor with the Department of Electrical and Computer Engineering as well as the Department of Computer Science, University of Houston, TX, USA. His research interests include wireless resource allocation and management, wireless communications and networking, game theory, big data analysis, security, and smart grid. He received the NSF Career Award in 2010, the Fred W. Ellersick Prize of the IEEE Communication Society in 2011, the EURASIP Best Paper Award for the *Journal on Advances in Signal Processing* in 2015, the IEEE Leonard G. Abraham Prize in the field of Communications Systems (Best Paper Award for IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS) in 2016, and several best paper awards from IEEE conferences. He was an IEEE Communications Society's Distinguished Lecturer from 2015 to 2018, and has been an AAAS Fellow since 2019 and an ACM Distinguished Member since 2019. He has been a 1% highly cited researcher since 2017 according to the Web of Science. He is also the winner of 2021 IEEE Kiyo Tomiyasu Award, for outstanding early to mid-career contributions to technologies holding the promise of innovative applications, with the following citation: "for contributions to game theory and distributed management of autonomous communication networks."