Empirical Game-Theoretic Methods to Minimize Regret Against Specific Opponents

Moinul Morshed Porag Chowdhury[a], Jose Perez[b], Christopher Kiekintveld[b], Tran Cao Son[a], William Yeoh[c], and Enrico Pontelli[a]

[a]New Mexico State University
[b]The University of Texas at El Paso
[c]Washington University in St. Louis

ABSTRACT

In retail markets, a broker makes sequential decisions simultaneously with other brokers to trade products through publishing tariffs. A tariff is a contract between a broker and a customer where a broker's main objective is to maximize revenue by using a retail tariff strategy. Our work includes developing an isolated generalized retail market simulator to control the dynamic and stochastic variables of the vibrant, complex retail market, and formulating the retail trading and pricing problems as a game using game-theoretic concepts. We use *Deep Q-Networks* (DQN) to learn the *Best Response* (BR) strategy for a specific retail strategy played in this simulator. We propose *Clustered Double Oracle Empirical Game-Theoretic Analysis* (CDO-EGTA), a novel method for finding new strategies to minimize regret in retail trading. CDO-EGTA clusters the existing pool of candidate retail strategies into several groups and learns a new class of BR strategies. Empirical results show that our method outperforms the current state-of-the-art methods in terms of regret.

Keywords: Empirical Game Theory, Deep Reinforcement Learning, Retail Trading, Nash Equilibrium

1. INTRODUCTION

In a multi-agent repeated game, if there is an opportunity to sense the opponent agent strategy from previous rounds, an agent can exploit its opponent in payoffs by playing a specific Best Response (BR) strategy. In many real-world multi-domain applications, we often know something in advance about our opponent's strategy (e.g., from watching previous interactions or historical data), and know that their strategies are often too complex to change rapidly. It allows us to learn more ahead of time about our opponent strategy. The optimal way to exploit the opponents in such games is to learn specific BR strategies for each candidate's strategies and play corresponding BR strategies against them. However, in a large strategy space, learning BR policies for all the retail strategies is infeasible. We need to find a reasonable point to learn some BR strategies to exploit the opponent strategies, which motivates us to propose a novel method for finding better agent strategies while facing a specific set of opponents in a repeated game. Our method is called *Clustered Double Oracle Empirical Game-Theoretic Analysis* (CDO-EGTA), which builds upon the classic *Double Oracle* (DO) framework based on *Deep Q-Network* (DQN) and clustering methods.

Our work includes formulating a multi-agent stochastic simultaneous, sequential decision-making problem as a game using game-theoretic concepts and developing an isolated generalized game simulator to control the game's dynamic, stochastic variables. In this work, we choose the general retail trading market as our domain where the agents play the retail trading game to maximize their revenue. A broker (i.e., an autonomous trading agent) makes sequential decisions simultaneously with other brokers to trade products through publishing tariffs where a tariff is a contract between a broker and a customer. We focus on two broker retail games, where we use Deep Q-Networks (DQN) to learn the BR strategy for a specific retail strategy. Our main contribution is a

Further author information: (Send correspondence to M.M.P.C)
M.M.P.C.: E-mail: porag@nmsu.edu, T.C.S.: E-mail: stran@nmsu.com, E.P.: E-mail: epontell@nmsu.com, C.K.: E-mail: cdkiekintveld@utep.com, J.P.: E-mail: jperez50@miners.utep.edu, W.Y.: E-mail:wyeoh@wustl.edu

domain-independent novel method called Clustered Double Oracle Empirical Game-Theoretic Analysis (CDO-EGTA), which explores new strategies for the agent to minimize regret in payoffs. The CDO-EGTA clusters the existing pool of opponents' strategies into some groups, learns a BR strategy for each group and maps group to BR strategy. While playing the game, the CDO-EGTA agent finds the rival's group from the map and plays the corresponding BR strategy to exploit the opponent. Empirical results show that our method outperforms the current state-of-the-art methods in terms of regret.

2. BACKGROUND AND RELATED WORK

An abstracted game is a reduced model of a complex interaction which is used to analyze complex games (e.g., poker).[1,2] Empirical game theory estimates the payoff of a strategy by simulating a complex game (e.g., stock market games, cybersecurity games, and trading agent competitions), instead of analyzing it to the atomic level.[3] An oracle is a function that takes the current profile and returns the best-response pure strategy for a player. Using specialized algorithms as oracles,[4] introduced the double-oracle algorithm and applied it to a robot path-planning game. *Empirical Game-Theoretic Analysis* (EGTA)[5,6] refers to the body of work that solves game models and searches for Nash equilibria from simulations. In the *Double Oracle Empirical Game-Theoretic Analysis* (DO-EGTA) method, we use an oracle for each player to derive the *Best Response* (BR) strategy against the current equilibrium opponent.[7] bridged RL with EGTA by applying tabular Q-learning with tile coding to learn better strategies in a continuous double auction trading game.

Deep learning has been successfully used to learn superhuman strategies for extensive-form games like go, chess, and shogi.[8,9] It uses a neural network as a function approximator to generalize the experience, which is more robust than tabular Q-learning methods and can handle high-dimensional state spaces. *Deep Reinforcement Learning* (DRL) is a framework that uses a *Deep Neural Network* (DNN) as a function approximator instead of storing the Q-values to learn policies. Deep Q-Networks[10] have been demonstrated to learn strong policies for playing Atari games using only pixel input, based on the Q-learning method of.[11] DRL has also been successfully applied to other problems with imperfect information, multiple players with asymmetric roles, and non-zero-sum payoffs from cooperative-competitive games[12] to video games like Super Smash Bros.,[13] DOTA 2,[14] and Doom.[15] Recently,[16] presented work on applying DQN as the oracle for a zero-sum security game. *History Aware Double Oracle EGTA* (HADO-EGTA)[3] is the latest variant of DO-EGTA, considered to be the state-of-the-art, where the algorithm keeps track of previous equilibrium strategies and learn a new DQN strategy over the mix strategies of current and prior equilibria.

3. PROPOSED METHODOLOGY MOTIVATION

We now describe a variant of the Rock Paper Scissor (RPS) game and some initial results to provide intuition for our proposed method. In this two-player RPS game, we have ten different sizes (weighted from 0 to 9) of rocks (R0–R9), papers (P0–P9), and scissors (S0–S9). Like a general RPS game, a rock beats a scissor, a scissor beats a paper, and a paper beats a rock. If both players pick the same type of object, the heavy item wins the round. If both players choose the same type and weight object, we considered it a draw. A player gets 1 point for a win, 0 for a draw, and -1 for a loss.

Suppose we have an opportunity to estimate a player's policy while playing in this type of game. In that case, by detecting the opponent's policy nature, we can have some advantages, which is our proposed methodology's central intuition. In this RPS game, we can treat picking an item as a pure strategy; thus, we have 30 different pure strategies. By simulating this game 50000 times using a random sampling of pure strategies for both players, we find that the giant rock (R9), paper (P9), and scissor (S9) get the highest payoffs. So, this game's mixed strategy Nash-equilibria (MSNE) is to pick the R9, P9, and S9 objects with 1/3 probability.

In this game, playing the MSNE is not an optimal strategy because when we uniform randomly select an object from R9, P9, and S9, it is not guaranteed that we can always win. For example, if we choose S9 as an outcome object and the opponent selects any rock, we will lose the round. The best way to play the opponent is by picking the best competitive item against it. Clustering objects can help us to achieve this goal. Our experiment in figure 1 follows a Zero Intelligence (ZI) policy, which uniform randomly picks an object from two different ranges i.e. (R0–R9, P0–P4) & (P5–P9, S0–S9) among the 30 objects and simulates against every objects

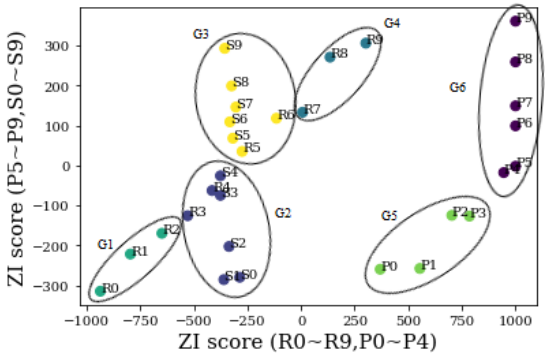

Figure 1. Clustering Objects in a Rock Paper Scissor Variant

for 1000 times. The X-axis is the score when an object is played against ZI(R0–R9, P0–P4), and the Y-axis is when played against ZI (P5–P9, S0–S9). By applying the k-means clustering algorithm on this data, we can find six groups. If we learn a specific best response object for each group, it will give us a clear advantage while playing the game. For example, if we can estimate that the opponent object belongs to group G3, we can easily play any best response objects from the range (R7–R9) and win the round. Thus, learning best responses using clustering has clear advantages over the other state-of-the-art approaches such as machine learning SelfPlay, DO-EGTA, and HADO-EGTA. SelfPlay never converges to the MSNE because it tries to find a single object which wins over all other objects and eventually results in an exploitable strategy. DO-EGTA and HADO-EGTA will eventually converge to this MSNE as their output strategy, which will have regrets. Moreover, clustering helps to speed the convergence process compared to approaches without clustering.

A conceptual diagram of DO-EGTA (see Figure 2) shows three types of contours, where the solid contour is payoff w.r.t. the previous-previous opponent, the dashed contour is w.r.t. the previous opponent and the dotted contour is w.r.t. the current opponent. It shows how the new BR strategy moves forward from learning against different equilibria and converging to the ultimate BR strategy (red circle). Blue pentagons are the points where DO-ETGA converges to a specific set of strategies. Both DO, and HADO-EGTA methods output a single BR strategy that can get better or equal payoffs over all the strategies in the candidate strategy set. However, it is not guaranteed that it can effectively exploit them payoff-wise, which is one of these methods' main limitations. Moreover, we may need many iterations to converge to the ultimate BR strategy for a large strategy space, which can be computationally expensive.

Our proposed CDO-EGTA method tries to learn a portfolio of different BR strategies that will do well against different types of opponents since a one-size-fits-all strategy is likely insufficient for this domain. In a duopoly retail environment, when retailers compete against each other, they can have some action preferences against specific types of opponents. As the strategies are some probability distributions of actions, there might be some inner correlation between strategies and opponent types. We try to find the inner correlations of candidate strategies and cluster them into groups. Then, by applying the DO-EGTA method on each group, we try to find a mapping of BR strategies to groups of opponents.

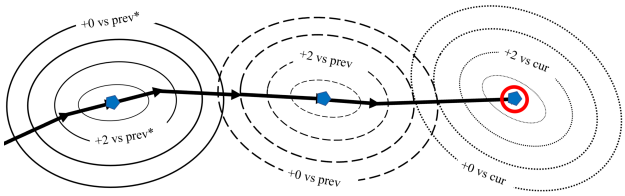

Figure 2. Conceptual Diagram of DO-EGTA

4. RETAIL PRICING GAME

A (finite, n-person) Normal-form Game (NFG) is a tuple (N, A, u), where:

1. N is a finite set of n players, indexed by i;
2. $A = (A_1, \ldots, A_n)$, where A_i is a finite set of actions available to player i. Each vector $a = (a_1, ..., a_n)$, where $a_i \in A_i$, is called an action profile;
3. $u = (u_1, \ldots, u_n)$, where $u_i : A \to R$ is a real-valued utility (or payoff) function for player i.

The retail pricing game is a complex game where retailers can play and transition between games like *Iterated Prisoner's Dilemma* (IPD), *Repeated Coordination Games*, *The Tragedy of the Common*, *The Boxed Pigs*, etc.[17]

4.1 Formal Definition of 3SM Game

We define the retail trading problem as a *Stochastic Simultaneous Sequential Market* (3SM) game as follows:

- A finite set of players $N = \{1, 2, \ldots, n\}$.
- A state space M, where a state $m \in M$ is defined by a set of features $\{f_1, f_2, \ldots, f_x\}$ using domain knowledge. The features can be time, players' public and private information, etc.
- Each player $i \in N$ has an action set A^i, where an action $a \in A^i$ is a real number $-PP_{max} \leq a \leq PP_{max}$ in a continuous range, where PP_{max} is the maximum (positive) energy price in the market.
- Players can select a strategy s, which can be pure or mixed, using their own action set at anytime. A pure strategy is playing only one action with probability 1, while a mixed strategy is playing several actions with a probability distribution. At time t, each player $i \in N$ chooses a strategy s_t^i from its strategy set S^i.
- At time t, each player $i \in N$ plays only one action in the game from their selected strategy, which we call the outcome action $o_t^i \in o_t$.
- Action profiles, $O = A^1 \times A^2 \times \ldots \times A^n$, consist of all the outcome actions chosen by individual players.
- The payoffs of players are defined by a *payoff function* $g : M \times O \to \mathbb{R}^N$, which maps each state and action profile to a real number. The i^{th} element of g, g^i, is the payoff to player i as a function of the state $m \in M$ and action profile $o \in O$.
- Players have preferences over the payoffs at time t, denoted by $\preceq_i$. For player k and any action profiles o_i and o_j, $o_i \preceq_k o_j$ iff $g^k(m_t, o_i) \leq g^k(m_t, o_j)$.
- At time t, players fully observe the action profile o_{t-1} and partially observe the state m_{t-1}. A player's cost, profit, market shares, etc. are private information that is not shared with others, which makes it an imperfect-information game.
- At time t, players simultaneously choose a strategy $s_t^i \in S^i$, play moves $o_t^i \in o_t$, and customers (nature) selects the next state m_{t+1} based on the current state m_t and action profile o_t according to the probability $P(m_{t+1}|m_t, o_t)$.
- As players have perfect recall, each player keeps a history H of action profiles and visible features $\hat{m}$ of a state m, where $H_{t+1} = \{o_1, \hat{m_1}, o_2, \hat{m_2}, \ldots, o_t, \hat{m_t}\}$.
- A play of this game $m_1, o_1, \ldots, m_t, o_t$ defines a stream of payoffs $g_1, g_2, \ldots, g_t$, where $g_t = g(m_t, o_t)$ and each player i has a stream of strategies $\{s_1^i, s_2^2, \ldots, s_t^i\}$, which we call a meta-strategy (SM).
- For a t time-step game, the normalized payoff for player i is $\tilde{g}_t^i = \frac{1}{t} \sum_{n=1}^{t} g_t^i$.

So, an instance of a retail pricing game can be represented by the tuple $(N, S, H, P, \preceq_i, t)$, where each player i seeks to find a meta-strategy in order to maximize their payoff $\tilde{g}_t^i$.

5. GENERALIZED RETAIL MARKET SIMULATOR

We now discuss our general isolated retail simulator components and a set of heuristic strategies, where we can abstractly simulate any retail product market with a given market parameter set. Our experiments used the Power TAC retail market parameters, making our simulator an abstract isolated retail energy market simulator. If needed, we can switch it to any retail product market such as water, gas, other general utilities, etc., given those specific market parameters for each component.

5.1 Customer Model

In the retail market, consumers define the transition function and determine the revenue of a retailer. A consumer can have two basic properties that can affect their behaviors to subscribe to a broker's product. These properties are inertia and rationality.[18]

An *inertia* parameter I defines how frequently the customers should evaluate market tariffs. Customers do not always evaluate tariffs when given the opportunity. For example, they may ignore tariff publications, consider them junk mail, or become lazy. But when a broker changes its tariff price, all subscribed customers reset their inertia parameter. So, the inertia parameter $I \in [0, 1]$ is a probability that determines if a customer will evaluate the tariffs in the market at a particular time slot. To model the market opening at the beginning of a simulation, we expect customers to be paying attention. So, the actual inertia parameter I_a must start with the value of 0 as $I_a = (1 - 2^{-n})I$, where n is a count of tariff publication cycle starting from 0. A broker may not publish tariff at every timeslot. It might need to wait for certain timeslots to pass so that the window opens to publish a tariff. We control this window using a variable named tariff publication cycle, i.e., after how many timeslots a broker can publish a tariff. In our experiments, it is set to 6 timeslots. $(1 - I_a)$ is the portion of the population that will evaluate the market and possibly switch during a particular publication cycle.

Rationality determines how customers evaluate a broker's tariff in the market. If customers are entirely rational, the broker with the lowest tariff price eventually gets all the market share. Otherwise, the lowest price broker sometimes loses some market share. An overall tariff choice does not necessarily follow a deterministic choice of the highest utility value, because customers can be irrational in their decision making. We follow a smoother decision rule based on multinomial logit choice for customer decision-making in choosing tariffs. The logit choice model assigns probabilities to each tariff t_i from the set of evaluated tariffs T as follows: $P_i = \frac{e^{\lambda u_i}}{\sum_{t \in T} e^{\lambda u_t}}$ where u_i is the utility of tariff t_i. The parameter $\lambda = [0, \infty]$ is a measure of how rationally a customer chooses tariffs, where a value of 0 and ∞ represents complete irrationality and perfect rationality, respectively.

In our experiments, we simulate a small simple portion (100 customers) of the factored customer model from the Power TAC retail market. We set I to 0 such that customers are always active and λ to ∞ (in implementation, it is less than ∞ to avoid numeric overflow issues) to make them pure rational. We use a synthetic demand profile per day, which includes two peak demand hours. These peaks are usually in the morning and evening times. We simulate this demand profiles for three-days (i.e., for 72 timeslots). All customers follow the same demand profile. We add a random walk[19] noise to the actual customer demand to simulate a demand predictor for the brokers.

5.2 Product Price Model

To regulate the upper limit of the retail market, we follow Power TAC to set *(i)* the maximum tariff price PP_{max} to \$0.50 per kWh and *(ii)* default starting tariff price to \$0.28 per kWh. We simulate the wholesale cost of the energy using a random walk method[20] where at timeslot t, we denote the per kWh energy cost as C_t. We vary C_t using a trend τ that is updated by a random walk:

$$C_{t+1} = \min(C_{max}; \max(C_{min}; \tau_t C_t))$$
$$\tau_{t+1} = \max(\tau_{min}; \min(\tau_{max}; \tau_t + \text{random}(-0.01, 0.01)))$$

where we set *(i)* C_0 uniformly at random from the interval $[C_{min}, C_{max}]$, *(ii)* C_{max} to \$0.20 per kWh, *(iii)* C_{min} to \$0.10 per kWh, *(iv)* τ_0 to 1.0, and *(v)* reset τ to 1.0 when the random walk exceeds the minimum or maximum boundaries. In other words, if $\tau_d C_t < C_{min}$ or $\tau_d C_t > C_{max}$ then $\tau_{d+1} = 1.0$, which reduces the bimodal tendency of the random walk.[19]

5.3 Heuristic Candidate Strategies

Our pricing game's complex strategies are intuitively similar to cooperation and defection strategies in a Prisoner's Dilemma game. In a two-player 3SM game, non-cooperative strategies cut down the opponent's profit margin, where cooperative strategies increase or keep it the same. We now describe 16 heuristic strategies of different natures that we used by adapting popular policies from auction theory and IPD literature.

5.3.1 Adapted Auction Strategies:

The *Zero Intelligence* (ZI)[21] strategy generates bids at random prices drawn from a uniform distribution. It does not keep a memory of how the agent performed in the previous timeslots. *Zero Intelligence Plus* (ZIP)[22] strategy tries to remember their failed bid prices, and in the next round, they bid with a more competitive price so that they can make more profits. *GD*, proposed by,[23] maintains a belief function representing the probability of its profit on each action and greedily selects the highest probability value action while bidding.

5.3.2 Adapted IPD Strategies:

Tit for Tat (TFT)[24] cooperates on the first move, then copies the opponents last move. We can have two variations of TFT based on their generosity. *Tit for Two Tats* (TF2T) cooperates on the first move and defects only when the opponent defects two times. *Two Tits for Tat* (2TFT) strategy is the same as TFT, except that it defects twice when the opponent defects. *Hard Majority* (HardMj)[25] defects on the first move and cooperates as long as the opponent has cooperated more often than it has defected. Otherwise, it defects. *Soft Majority* (SoftMj)[25] cooperates on the first move and cooperates as long as the opponent has cooperated at least as often as it has defected. Otherwise, it defects. *Grim* (Grim)[26] cooperates until the opponent defects, after which it always defects. *Always Increase* (AlI) always cooperates, *Always Decrease* (AlD) always defects, and *Always Same* (AlS) does not change the current price. *Prober* (Pbr) starts with defect, cooperate, cooperate, and then defects if the opponent has cooperated in the second and third moves. Otherwise, it plays TFT. *Naive Prober* (NvPbr) is like TFT but occasionally defects with a small probability. *Naive Increase* (NvI) is like AlS, but occasionally cooperates with a small probability. *Pavlov* (Pavlov)[27] cooperates on the first move. If the reward or temptation payoff is received in the last round, it repeats the last choice. Otherwise, it chooses the opposite choice.

6. EXPERIMENTAL SETUP

We now have a game and a simulator with heuristic strategies for running experiments to find a better strategy exploration method. This section explains the experimental setup of the broker action-state space, the DQN configurations, and the implementations of CDO-EGTA and four benchmark methods.

6.1 Broker Setup

6.1.1 Action Space:

The brokers who use the candidate strategies can play three actions $a \in A$ to change their tariff prices at a certain time step:

1. **Action D:** Decrease price by \$0.01 (defect)
2. **Action I:** Increase price by \$0.01 (cooperate)
3. **Action NC:** No change in price (cooperate)

The broker's action space is limited between the maximum tariff price PP_{max} and the current cost point C_t. We introduce two new actions **I2** (increases the tariff price twice than action I) and **D2** (decreases the tariff price twice than D) to the Deep RL agent's (DQAgent) action space, which sometimes helps it explore better strategies. This DQAgent is used by all the strategy exploration benchmark methods while learning BR strategies.

Algorithm 1 DO-EGTA

Require: $U, S^H, g(), v()$
$t \leftarrow 0$, $S_t \leftarrow S^H$
$\sigma_t \leftarrow v(U, S_t)$
do
  $t \leftarrow t + 1$
  $\delta_t \leftarrow g(U, \sigma_{t-1})$
  if $U(\delta_t, \sigma_{t-1}) > U(\sigma_{t-1}, \sigma_{t-1})$ **then**
    $S_t \leftarrow S_{t-1} + \{\delta_t\}$
  $\sigma_t \leftarrow v(U, S_t)$
while $S_t \neq S_{t-1}$
return σ_t, S_t

Algorithm 2 HADO-EGTA

Require: $U, K, \gamma, \alpha, g(), g'(), (\sigma_0, ..., \sigma_{t-1})$
$\bar{\sigma} \leftarrow (\sum_{\psi=0}^{t-1} \gamma^{t-1-\psi})^{-1} \sum_{\psi=0}^{t-1} \gamma^{t-1-\psi} \sigma_\psi$
$k \leftarrow 0$
$\sigma_t^k \leftarrow g(U, \sigma_{t-1})$
while $k < K$ **do**
  $k \leftarrow k + 1$
  $\delta_t^k \leftarrow g'(U, \bar{\sigma}, \delta_t^{k-1})$
$\delta_t^* \leftarrow \text{argmax}_{\delta_t^{k'}} \alpha U(\delta_t^{k'}, \sigma_{t-1}) + (1 - \alpha) U(\delta_t^{k'}, \bar{\sigma})$
s.t. $U(\delta_t^{k'}, \sigma_{t-1}) > U(\sigma_{t-1}, \sigma_{t-1})$ [or $\emptyset$ if none]
return δ_t^*

Algorithm 3 CDO-EGTA

Require:
$U(), d(), S^H, S^F, c(), G()$
for $C \leftarrow S^H$ **do**
  for $F \leftarrow S^F$ **do**
    $Score_F = U(C, F)$
    $f \leftarrow \{F, Score_F\}$
    $EFD \leftarrow \{C, f\}$
$\{Clusters, N\} \leftarrow c(EFD)$, $n \leftarrow 0$
do
  $M_{br}^n \leftarrow d(Clusters[n])$
  $n \leftarrow n + 1$
while $n < N$
return $M_{br}, Clusters$

6.1.2 State Features:

We use 37 historical data points as our state features. We have two time-related data (i.e., day and hour), six tariff prices, two profit values, three unit-cost prices, six taken actions, three market share points, and three customer demand profiles. We use the hot-encoding for the actions. So, in total, we have 49 feature values for our states.

6.1.3 DQN Setup:

We have implemented the Double DQN variant by[28] as our best-response oracle. We use the Deeplearning4j framework libraries to implement the DQN in our experiments.[29] A 3-layer ($25 \times 13 \times 7$) neural network works fine for our experiments instead of a large DNN as we have a limited number of features where each of the hidden layers is fully connected with the next hidden layer. We treat each state's 49 feature values as inputs that go into the first hidden layer of 25 nodes. The last layer has seven nodes, which we connect to the output layer of 5 nodes as we have 5 DQAgent actions. We set the experience replay to 1000, batch size to 100, and target DQN update frequency to 500 game's timeslots. We follow the epsilon-greedy policy to select an action in the learning phase where the epsilon starts from 1 and decays linearly to 0.1 to the end of a game. We use this DQN configuration for all the methods in our experiments to learn BR strategies, where we run 2000 games for training and 1000 games for testing.

6.2 Benchmark Methodologies

6.2.1 Minimax Regret:

The minimax regret method selects the candidate strategy that has the lowest maximum regret if it is played against all the candidate strategies. By minimizing the worst-case regret, this method finds the candidate strategy that performs better, in terms of regret, compared to any strategy in the restricted candidate strategy set.[30]

6.2.2 DQSelfPlay:

In the self-play method, the DQAgent learns a strategy by playing against the previous best version of itself in the game without any domain knowledge, except for the game rules.[8,31] By playing a random strategy in the first iteration, the DQAgent converges in 625 iterations. In one iteration, we run one training and one testing round, where 2000 games are played in each training round, 1000 games are played in each testing round.

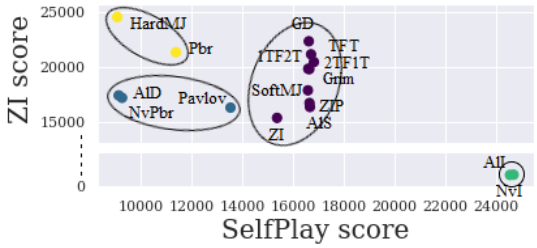

Figure 3. Four Groups Created by Mean Shift on EFD

6.2.3 Double Oracle EGTA (DO-EGTA):

We have a set of heuristic strategies, $S^H = (S_1, S_2, \ldots, S_n)$ for both players. In each round $t = (0, 1, \ldots)$ of the iterative procedure (Alg. 1), the current strategy set is S_t. A mixed-Nash equilibrium over these strategy sets under the game utility function $U()$ is σ_t. The game utility function $U()$ is based on running simulations of the 3SM game in the isolated retail simulator which helps us to estimate the utility values of the strategies played by the players. $g()$ is a DRL algorithm that learns a better response against an opponent's mixed strategy and returns a pure strategy σ_t. $v()$ a Nash-equilibrium solver, which returns any mixed Nash-equilibrium, given a utility function and finite strategy sets. As this is a symmetric game, we provide only one parameter for strategies in $v()$, where both players use the same strategy set.

6.2.4 History Aware Double Oracle EGTA (HADO-EGTA):

HADO-EGTA (Alg. 2) fine-tunes the strategy to maximize payoff against previous opponents, and returns a strategy that deviates beneficially against the current opponent while exploiting old opponents.[3] While HADO-EGTA fine-tunes its strategy, interim DNN weights are periodically recorded at k points, and the weights that are selected yield the highest payoff against a mixed strategy over current and previous opponents, subject to being a beneficial deviation against the current opponent. Following the authors, we used $k = 4$ strategies to select among newly learned BR strategies, including the pre-trained strategy where we discounted old strategies with a decay factor $\gamma = 0.7$ and set weighting $\alpha = 2/7$.

6.3 Clustered Double Oracle EGTA (CDO-EGTA)

In CDO-EGTA (Alg. 3), we apply these subsequent steps:

1. Generate a set of features for each candidate strategy $C \in S^H$ by playing them against a group of feature extraction strategies $F \in S^F$ and build an *Extracted Feature Dataset* (EFD).
2. Use a clustering algorithm $c()$ on EFD to cluster all the candidate strategies S^H into N groups.
3. Apply DO-EGTA algorithm $d()$ on the N groups.
4. Learn N BR strategies and create a mapping of best responses M_{br} for the N groups.

S^F can be a subset of S^H or contain any self-customized strategies to generate the feature payoffs. We call these payoffs as extracted feature data points, where all the data points of S^H build the EFD dataset. In our experiments, we fix two feature extraction strategies, *(i)* the ZI strategy and *(ii)* the candidate strategy itself (i.e., SelfPlay). For each $C \in S^H$, we play the strategy C against ZI and itself 1000 times and use the average payoffs as the ZI and SelfPlay scores. These scores help us to get some sense about the cooperativeness or non-cooperativeness of a candidate strategy. By adding all the scores for each candidate strategy, we build the EFD dataset and apply a clustering algorithm to find groups. Figure 3 shows four groups on the EFD dataset, where the x-axis is the ZI score and the y-axis is the SelfPlay score for all the 16 candidate strategies. Then, we learn the corresponding DQN best responses for each group using DO-EGTA and create a new class of BR strategies.

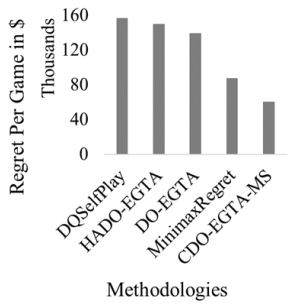

Figure 4. Regret Comparison

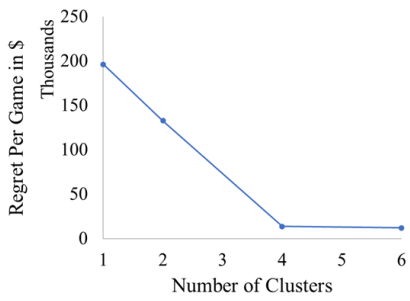

Figure 5. CDO-EGTA Regret by No. of Clusters

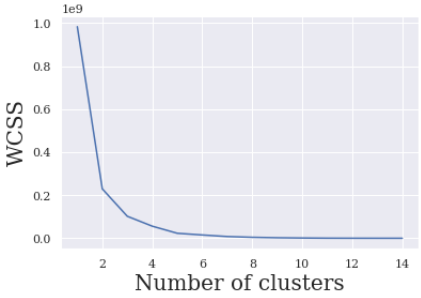

Figure 6. WCSS Error in k-means

7. EXPERIMENTAL RESULTS

To compute regret, we create a two-player meta game where player 2 (column player) has 16 candidate heuristic strategies as its actions. And player 1 (row player) has $N > 16$ strategies as its actions, which includes strategies found by the EGTA and SelfPlay methods. Thus, we create an $N \times 16$ meta game, where we run 1000 test games for each cell to generate the payoffs for both players. After generating the payoffs, we calculate the regret table for player 1. Then, we accumulate all the regret values row-wise to get the regret of a specific strategy.

7.1 Regret Comparison of the Methodologies

In CDO-EGTA, as the appropriate number of clusters for the problem is unknown, we use the *Mean Shift* (MS) clustering algorithm[32] to find the clusters in EFD. Figure 3 shows that MS clusters the 16 candidate strategies into four groups. We thus learn four BR strategies by applying DO-EGTA on these four groups, create M_{br} of those groups, and return it as the output of our CDO-EGTA method. Figure 4 shows the accumulated average regret values for all the five methods per game, where our CDO-EGTA MS method achieves the lowest regret compared to others. The result shows that the clustering algorithm can successfully group the candidate strategies based on EFD, and DO-EGTA becomes more effective in learning new strategies when we use it within CDO-EGTA. In test rounds, CDO-EGTA finds the corresponding BR strategy against the specific opponent agent from M_{br} and plays it to exploit the opponent strategy. In experiments, we get 2TFT strategy as the minimax regret strategy, which does well against cooperative strategies but gets exploited by non-cooperative strategies. The DQSelfPlay agent uses the D2 action exhaustively as it finds that it is beneficial to use the action as much as possible in the self-learning phase and end up having higher regrets in test games. DO-EGTA learns and converges to its ultimate BR strategy while learning against Always Defect (AlD) candidate strategy and, eventually, it becomes the BR strategy of AlD. HADO-EGTA's ultimate BR strategy also converges while learning against AlD, where it could have an improved DNN to maximize revenue. However, in our case, HADO-EGTA has the same performance compared to DO-EGTA, as it selects the same DQN policy as DO-EGTA method while learning k policies against AlD. All these strategies found by DQSelfPlay, DO-EGTA, and HADO-EGTA miss out, earning more revenue opportunities against cooperative strategies because the BR strategy learned by them are non-cooperative strategies. On the other hand, the CDO-EGTA method cooperates while playing against cooperative strategies and defects against non-cooperative strategies, thus outperforming others in terms of regret.

7.2 Varying the Number of Clusters

To test the performance of CDO-EGTA when the number of clusters changes, we use the k-means clustering algorithm by setting different k values and analyze the performance. Figure 5 shows that the regret value decreases with an increasing number of clusters.

The highest number of clusters we can achieve is when we have one strategy per cluster. In this experimental setup, we can have a maximum 16 clusters as there are 16 different heuristic candidate strategies. So when we increase the number of clusters in CDO-EGTA, we are following the correct way to minimize our regret in this game. Because the optimal way to exploit the opponents in this game is to learn specific BR strategies for each

of the candidate strategies and play corresponding BR strategies against them in the test rounds. However, in a large strategy space, learning BR policies for all the retail strategies is infeasible. So, we need to find a reasonable tradeoff between learning a BR strategy against all possible strategies and clustering all strategies into a single group.

The within-cluster sum of errors (WCSS) is the summation of each cluster's distance between that specific clusters each point against the cluster centroid, and figure 6 shows that the WCSS value decreases as we increase the number of clusters. To estimate the correct number of clusters, we seek a point in figure 6 where the cluster size and WCSS value are at a minima to set the number of clusters. Both figures 5 and 6 shows that the regret and WCSS values decrease slowly to the convergence value when the number of clusters increases from 4 to 6. We see that both regret and WCSS value have an inversely proportional relationship with the number of cluster size, which shows the effectiveness of clustering on the candidate strategies while learning new strategies. As a result, we can use WCSS values (figure 6) only to find the correct number of clusters for k-means instead of running CDO-EGTA at different k values, which can avoid learning k different M_{br} maps and running further empirical tests.

8. CONCLUSIONS

We formulated the general retail trading problem as a stochastic simultaneous sequential market game. We created an abstract isolated retail simulator to play the game suitable for any product retail domain. We adopted a set of popular heuristic retail strategies from auction theories and IPD literature and experimented with existing strategy exploration methods to find a better strategy exploration method. Our proposed CDO-EGTA algorithm analyzes the opponent strategy pool and explores better regret minimizing retail strategies than the state-of-the-art algorithm. CDO-EGTA achieves this objective by using clustering methods and learning strategies using DQN with the standard DO-EGTA method. With better clustering methods, feature extraction strategies, and new DO-EGTA variants, CDO-EGTA is likely to have even better performance. One of our limitations is that we have a restricted set of actions for the brokers and a reduced set of initial heuristic strategies. Also, our trading environment is a duopoly market where real-world markets can be an oligopoly. We also simplified our customer model, where real-world customer models can be very complex and rich. Future work involves relaxing these assumptions to better generalize to problems in the real world.

ACKNOWLEDGMENTS

This research is partially supported by NSF grants 1757207, 1914625, and 1812619. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the sponsoring organizations, agencies, or the U.S. government.

REFERENCES

[1] Ponsen, M., Tuyls, K., Kaisers, M., and Ramon, J., "An evolutionary game-theoretic analysis of poker strategies," *Entertainment Computing* **1**(1), 39–45 (2009).

[2] Tuyls, K., Perolat, J., Lanctot, M., Leibo, J. Z., and Graepel, T., "A generalised method for empirical game theoretic analysis," in [*Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*], 77–85 (2018).

[3] Wellman, M. P., Nguyen, T. H., and Wright, M., "Empirical game-theoretic methods for adaptive cyber-defense," in [*Adversarial and Uncertain Reasoning for Adaptive Cyber Defense*], 112–128, Springer (2019).

[4] McMahan, H. B., Gordon, G. J., and Blum, A., "Planning in the presence of cost functions controlled by an adversary," in [*Proceedings of the International Conference on Machine Learning (ICML)*], 536–543 (2003).

[5] Wellman, M. P., "Methods for empirical game-theoretic analysis," in [*Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*], 1552–1556 (2006).

[6] Wellman, M. P., "Putting the agent in agent-based modeling," *Autonomous Agents and Multi-Agent Systems* **30**(6), 1175–1189 (2016).

[7] Schvartzman, L. J. and Wellman, M. P., "Stronger CDA strategies through empirical game-theoretic analysis and reinforcement learning," in [*Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*], 249–256 (2009).

[8] Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al., "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science* **362**(6419), 1140–1144 (2018).

[9] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., et al., "Mastering the game of go without human knowledge," *Nature* **550**(7676), 354–359 (2017).

[10] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al., "Human-level control through deep reinforcement learning," *Nature* **518**(7540), 529–533 (2015).

[11] Watkins, C. J. and Dayan, P., "Q-learning," *Machine Learning* **8**(3-4), 279–292 (1992).

[12] Tampuu, A., Matiisen, T., Kodelja, D., Kuzovkin, I., Korjus, K., Aru, J., Aru, J., and Vicente, R., "Multiagent cooperation and competition with deep reinforcement learning," *PloS one* **12**(4) (2017).

[13] Firoiu, V., Whitney, W. F., and Tenenbaum, J. B., "Beating the world's best at Super Smash Bros. with deep reinforcement learning," *arXiv preprint arXiv:1702.06230* (2017).

[14] OpenAI, "OpenAI five," (2018).

[15] Lample, G. and Chaplot, D. S., "Playing FPS games with deep reinforcement learning," in [*Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*], (2017).

[16] Wang, Y., Shi, Z. R., Yu, L., Wu, Y., Singh, R., Joppa, L., and Fang, F., "Deep reinforcement learning for green security games with real-time information," in [*Proceedings of the AAAI Conference on Artificial Intelligence*], **33**, 1401–1408 (2019).

[17] Mudambi, S. M., "The games retailers play," *Journal of Marketing Management* **12**(8), 695–706 (1996).

[18] Ketter, W., Collins, J., and Reddy, P., "Power TAC: A competitive economic simulation of the smart grid," *Energy Economics* **39**, 262–270 (2013).

[19] Collins, J., Arunachalam, R., Sadeh, N. M., Eriksson, J., Finne, N., and Janson, S., "The supply chain management game for the 2005 trading agent competition," (2004).

[20] Kiekintveld, C., Miller, J., Jordan, P. R., Callender, L. F., and Wellman, M. P., "Forecasting market prices in a supply chain game," *Electronic Commerce Research and Applications* **8**(2), 63–77 (2009).

[21] Gode, D. K. and Sunder, S., "Allocative Efficiency of Markets with Zero-Intelligence Traders: Market as a Partial Substitute for Individual Rationality," *Journal of Political Economy* **101**(1), 119–137 (1993).

[22] Tesauro, G. and Das, R., "High-performance Bidding Agents for the Continuous Double Auction," in [*Proceedings of the ACM Conference on Electronic Commerce (EC)*], 206–209 (2001).

[23] Gjerstad, S. and Dickhaut, J., "Price formation in double auctions," *Games and Economic Behavior* **22**(1), 1–29 (1998).

[24] Robert, A. et al., "The evolution of cooperation," (1984).

[25] Mittal, S. and Deb, K., "Optimal strategies of the iterated prisoner's dilemma problem for multiple conflicting objectives," *IEEE Transactions on Evolutionary Computation* **13**(3), 554–565 (2009).

[26] Friedman, J. W., "A non-cooperative equilibrium for supergames," *The Review of Economic Studies* **38**(1), 1–12 (1971).

[27] Kraines, D. and Kraines, V., "Pavlov and the prisoner's dilemma," *Theory and Decision* **26**(1), 47–79 (1989).

[28] Van Hasselt, H., Guez, A., and Silver, D., "Deep reinforcement learning with double q-learning," in [*Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*], (2016).

[29] Team, E. D. D., "ND4J: Fast, Scientific and Numerical Computing for the JVM," (2016).

[30] Savage, L. J., "The theory of statistical decision," *Journal of the American Statistical Association* **46**(253), 55–67 (1951).

[31] Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al., "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," *arXiv preprint arXiv:1712.01815* (2017).

[32] Cheng, Y., "Mean shift, mode seeking, and clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **17**(8), 790–799 (1995).