2020-01-01

Autonomous Trading Strategies For Dynamic Energy Markets

Moinul Morshed Porag Chowdhury
University of Texas at El Paso

AUTONOMOUS TRADING STRATEGIES FOR DYNAMIC ENERGY MARKETS

MOINUL MORSHED PORAG CHOWDHURY

Doctoral Program in Computer Science

APPROVED:

Christopher Kiekintveld, Ph.D, Chair.

Son Tran, Ph.D.

Mahmud Shahriar Hossain, Ph.D.

Stephen Crites, Ph.D.
Dean of the Graduate School

to my

MOTHER and FATHER

with love

AUTONOMOUS TRADING STRATEGIES FOR DYNAMIC ENERGY MARKETS

by

MOINUL MORSHED PORAG CHOWDHURY

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

Doctoral Program in Computer Science

THE UNIVERSITY OF TEXAS AT EL PASO

May 2020

Acknowledgements

I want to show my deep-felt gratitude to my advisor, Dr. Christopher Kiekintveld. My Ph.D. journey will not be possible without your constant kind support and enduring patience. Chris, I want to say that you have been the best advisor I could have hoped for. Your exceptional professional capabilities and unique personal qualities have made me a unique experience that I will cherish forever. Whenever I was (hopelessly) lost or doubted in my abilities, you always gave me your time whenever I requested and provided clear explanations and directions. Thank you, Professor, for not ceasing belief in me and everything. I also want to thank the other members of my committee, Dr. Son Tran and Dr. Mahmud Shahriar Hossain, for their invaluable suggestions, instructions, remarks, and additional guidance, which helped me to the completion of this work.

My fellow Intelligent Agent and Strategic Reasoning Lab members helped me in many ways during graduate school, including showing me the ropes of the degree process and providing comments, feedback, and suggestions with friendship. I want to thank Jose Perez, Russell Folk, and Ferdinando Fioretto with whom I collaborated directly. I learned a lot by working with each of them. I am also grateful to everyone in the lab who gave me advice, a good example to follow, or friendship over the years, including Sujan, Sunny, Oscar, Marcus, Alonso, and Anthony. Additionally, I thank my department professors and staff for all their hard work and commitment, providing me the means to accomplish my degree. I want to thank Shuvo, Suhail, Galib, and Saif for their moral support, which helped me in my graduate journey.

Finally, I must thank my dear wife for her great continuing, loving support during the development of this work. I also want to thank her for giving me a beautiful daughter, Ilana. This journey was difficult, but you and Ilana were always there to support me and bring joy to me. I can not express all my feelings here; I love you a lot, Tisam & Ilana!

Abstract

With increasing energy demand and an intermittent supply of renewable energy sources, our current energy grid needs a transformation towards a more robust, reliable energy trading architecture. The smart grid promises this architecture as the future of the present energy market, where traders will use digital technologies to automate the management of power delivery. It will improve many issues of the current energy grid such as sustainable, clean, renewable, reliable and secure energy supply, customer participation in markets, distributed generation, and transparency in energy trading. Using autonomous trading agents, we can bridge several dynamic energy markets and ensure an efficient and robust trading environment for all the players in the smart grid. The *Power Trading Agent Competition* (Power TAC) simulation emphasizes the strategic problems that autonomous trading agents, i.e., brokers, will face in managing the economics of a smart grid.

In Power TAC, brokers make trades in multiple parallel markets such as wholesale, tariff, and balancing markets to supply energy from producers to consumers. To be successful, brokers must make reasonable predictions about future supply, demand, and prices in the wholesale and tariff markets to make trading decisions by maintaining a favorable energy imbalance in the balancing market.

Market clearing price prediction is an integral part of the broker's wholesale market strategy because it helps the broker to make intelligent decisions for purchasing energy at low cost in a day-ahead wholesale market. People use machine learning methods to predict prices in the Power TAC wholesale Periodic Double Auction (PDA) market, where the brokers can take advantage of the price predictor to design bidding strategies. PDAs are commonly used in real-world energy markets to trade energy in specific time slots to balance demand on the power grid where multiple discrete trading periods are specified for a single type of good. Strategically, bidding in a PDA is complicated because the bidder must predict and plan for future auctions that may influence the bidding strategy for the current auction.

In our work, we use the RepTree model to predict prices and present a general bidding strategy for PDAs. Our wholesale market strategy uses forecasted clearing prices and *Monte Carlo Tree Search* (MCTS) to plan a bidding approach across multiple time-periods. Additionally, we present a fast heuristic policy that can be used either as a standalone method or as a seeding technique to initialize the search space of the MCTS bidding strategy. We evaluate our bidding strategies using a controlled PDA simulator based on the wholesale market implemented in the PowerTAC competition. We demonstrate that our strategies outperform state-of-the-art champion bidding strategies designed for that competition.

In the retail market, a broker makes sequential decisions simultaneously with other brokers to buy and sell energy through publishing tariffs where a tariff is a contract between a broker and a customer. To be as profitable as possible, a broker needs an effective energy selling retail strategy. Our work includes developing an isolated miniature retail market simulator to control the dynamic and stochastic variables of the vibrant, complex retail market so that we can understand the basic features and strategic dynamics among retail trading strategies. We apply deep reinforcement learning (DQN) to learn the best response (BR) strategy for a specific strategy played in this simulator. Using DQN as a best response learning technique, we propose "Clustered Double Oracle Empirical Game-Theoretic Analysis"(CDO-EGTA), a novel method for minimizing regret (i.e., maximizing revenues) in retail trading. CDO-EGTA method clusters the existing pool of strategies into some groups, learns a new BR strategy for each of the groups using the Double Oracle Empirical Game-Theoretic Analysis method, and outputs a class of BR strategies to play the game. Empirical results show that our method outperforms the existing methods in regret comparison.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

With the increasing electricity demand and intermittent supply of renewable energy sources in the twenty-first century, our current energy grid infrastructure is facing numerous difficult challenges and needs a transformation. It lacks several critical features such as sustainable energy consumption, trading transparency, effective use of pricing and demand response of energy, customer participation, enhanced reliability, and proper distribution management for variable-output renewable energy sources [32]. The Smart Grid is a modern electricity market that promises to provide a more intelligent energy infrastructure [30] than the current traditional energy grid by incorporating more advanced sensing capabilities and artificial intelligence for better decision making in energy trading. It can improve many problems caused by increasing energy demand and intermittent renewable energy supply, where our current energy grid infrastructure is more prone to fail. An advanced feature of this future energy grid is replacing humans with autonomous agents capable of making decisions in homes, businesses, and production facilities to control power delivery, consumption, and trading. To achieve this vision and as the grid becomes more decentralized, automated, and capable of providing much higher volumes of sensor data; we need to develop the economic structures and decision-making algorithms to ensure reliable and efficient energy delivery.

To this end, a key area of research for Smart Grids is to understand the market mechanisms that can coordinate buying and selling decisions in energy markets, and develop automated trading agents that can represent individual people in these markets. One of the significant academic efforts to develop such strategies centers on the *Power Trading Agent Competition* (Power TAC), a competition with over a decade of history [79]. The Power TAC scenario was designed to represent the types of market structures and automated trading problems that would arise in fu-

ture smart grids, focusing on capturing the features of real-world energy markets and emerging smart grid technologies. It implements a sophisticated simulation environment that allows for autonomous trading agents called brokers to compete to maximize their profits by participating in a series of realistic energy markets. It is a trading agent competition [79] designed to advance the understanding of market mechanisms that can coordinate buying and selling decisions in energy markets, and develop strategies for automated trading agents. Power TAC also supports a competitive benchmarking [33] research model building on the experience of competitions in 2012-2017, such as where earlier successful competitions are the Trading Agent Competition for Supply-Chain Management (TAC SCM) [13] and the Trading Agent Competition for Ad Auctions (TAC AA) [27].

Power TAC is a rich energy market simulation consisting of several markets such as the wholesale market, the tariff market, and the load balancing market to trade energy between producers and consumers. The broker operates in parallel in these markets and helps to bridge them successfully [29]. Brokers trade energy from producers to consumers by doing competitive bidding in the wholesale market and publishing tariffs in the retail market.

The wholesale market is one of the primary markets in the Power TAC scenario, which is based directly on similar real-world market designs. This market can be implemented using *Periodic Double Auctions* (PDAs), where a series of double auctions are held for producers and brokers to trade energy for an upcoming period to balance the supply and demand in their energy portfolios. While PDAs can be used to trade any type of good, one prominent use of this style of auction is in short-term energy markets used to balance demand on the power grid (e.g., NordPool, FERC, or EEX [52, 14, 16]). The wholesale market is called a "day ahead" market in which energy trading occurs up to one day in advance by an hourly basis of production/consumption. The energy is a perishable good, and the wholesale market allows brokers to buy and sell quantities of energy for future delivery. Auction market structures like this exist across many different types of perishable goods. So, finding useful, robust, automated strategies for these markets is a significant research challenge.

The retail market is the other primary market where brokers compete mainly to sell energy

to maximize their revenues. The customers in the retail market are households, offices, electric vehicles, and some distributed prosumers. The customers trade energy through a broker where the broker represents their energy demand in the wholesale market as his demand. Brokers buy and sell energy by publishing tariffs in the retail market; thus, it is often called the tariff market. A tariff is a contract between the broker and the customer to trade energy. The primary goal of the broker in this market is to earn revenue by offering competitive tariffs. A broker should publish profitable but attractive tariffs to get consumers/prosumers by maintaining a favorable energy imbalance.

1.1 Main Challenges and Research Questions

In smart grids, an autonomous broker makes decisions continuously from a massive strategy, state, and action space under real-time constraints while processing dynamic and stochastic information. It faces a strategic multi-agent environment in multiple parallel markets. A broker's main task is subsequent decision making in these markets to maximize profit throughout its continuance. While making these decisions, the main challenges brokers face include:

- **Continuous high dimensional action space**: To make bids in the wholesale or retail dynamic markets, brokers need to select a price from a high dimensional continuous price space. Strategically selecting a near-optimal price is challenging as it depends on many dynamic market factors.

- **Time constrained sequential and simultaneous decisions**: A broker selects a price in a fixed limited time, and it makes these decisions simultaneously with other rival brokers in sequential timeslots.

- **Competitive multi-agent environment**: N number of brokers can compete with each other in these dynamic markets, where we see a duopoly environment when $N = 2$ and oligopoly market when $N > 2$

- **High dimensional strategy space**: The continuous action space and time-constrained sequential simultaneous decision making create a high dimensional strategy space for a broker. To compete in different types of market environments, a broker needs to select different types of strategies to maximize its revenue or minimize its cost.

- **High dimensional state space**: Different types of brokers, strategies, actions, and dynamic variables create a high dimensional state space because we need a large feature set to represent a state.

The competitiveness in the smart grid markets depends on the number of participant brokers and their performances where a single broker's performance propagates from one market to other markets for a particular timeslot. By performance propagation in markets, we mean a broker's dependency on the wholesale market in the retail market. For example, one can not publish attractive tariffs, i.e., cheaper price tariffs in the tariff market, if he is unable to buy it at a lower price in the wholesale market.

On a first impression, it seems like finding optimal strategies for the broker is an impossible task. As the task is quite complex, my primary objective in this dissertation is to **"Develop effective autonomous trading strategies"** and investigate them empirically to achieve near-optimal performance for dynamic energy markets.

I have selected the main two markets of the smart grid as a research test-bed.

1. **Wholesale Market** The brokers can procure energy from producers and other brokers through the PDA wholesale market to satisfy their subscribed customers' demand. A broker's main objective in this market is to acquire the energy to meet demand at the lowest possible cost. For this, we need to predict the prices in future auction markets and build a bidding policy around those forecasted prices to minimize the cost. My primary research question for this domain is *"Can we design Monte Carlo Tree Search as a bidding strategy to traverse and select cost-effective prices in periodic double auctions?"* To answer this question, we address the following subquestions:

(a) What are the useful features and methods needed to predict market-clearing prices in a periodic double auction?

(b) Can we effectively initialize a reduced bid price search space?

(c) Can Monte Carlo Tree Search iteratively expand the reduced search space with new promising prices to find the cost-effective bid price?

(d) Will MCTS bidding strategy perform better than other PDA strategies in empirical evaluation?

2. **Retail Market** The broker can earn a substantial amount of revenues from customers by trading in the retail market through tariffs. A broker's objective in this market is to trade energy to achieve the highest possible revenue. There are many fees associated with energy trading in markets such as capacitance fees and balancing fees. In the retail market, the capacitance fee is imposed on the broker as a penalty to trade energy in the peak hours. The balancing fee is imposed by the balancing market to penalize or incentivize the broker based on their energy imbalances. A customer has some inertia to evaluate a tariff from the market, and when it evaluates tariffs, it does not always make rational decisions. Brokers can take advantage of these properties of customers and design retail strategies to maximize their revenue. They should also make strategies to lessen those several fees with their tariffs as much as possible. As the market is full of stochastic and dynamic behaviors, our primary research approach is to recognize the core features of trading with customers and other agents in this market by designing an isolated retail environment. Also, by doing empirical analysis of strategies, we want to find a successful retail strategy. Our research question for this domain is *"Can we harness the strategic dynamics of existing retail trading policies to develop a method to find a regret minimizing retail strategy?"* To answer this question, we can ask the following sub-questions:

   (a) Can we formulate retail market trading as a game using game-theoretic concepts?

   (b) Can we create an isolated small retail simulator to extract the strategic behaviors among retail strategies by playing this game?

(c) What features and game-theoretic solution concepts are needed to explore and select profit-maximizing retail strategies in the isolated simulator?

(d) Will our method perform better than existing literature methods in the empirical analysis?

With these two separate strategies, one to minimize cost at the wholesale market and the other to maximize revenue at the retail market, a broker is expected to be successful in maximizing profit in the smart grid energy trading.

1.2 Contributions

My main contributions in answering these research questions are as follows:

1. Develop a novel wholesale bidding strategy using Monte Carlo Tree Search for periodic double auctions to trade energy, focusing on minimizing cost.

   (a) Learn features and evaluate different types of price predictors to precisely predict PDA market-clearing prices.

   (b) Develop a controlled PDA simulator to test PDA bidding strategies to reduce randomness and high dimensions.

   (c) Design fast heuristic standalone PDA bidding strategies, which can also be used to initiate a reduced and effective action space for MCTS bidding strategy.

   (d) Adapt Monte Carlo Tree Search to simulate hour-ahead auctions and traverse through bid prices to find cost-effective bids.

   (e) Empirically evaluate the performance of the designed PDA bidding strategies.

2. Develop a method to find a retail strategy that can exploit the opponent's strategy while trading energy in the retail market to maximize revenue.

   (a) Formalize the retail trading problem as a new game using game theory.

(b) Develop an isolated retail simulator to extract the core dynamics of retail trading by simplifying the Power TAC retail market and creating empirical meta-games.

(c) Propose a novel CDO-EGTA method that learns a class of new strategies to minimize regret from an existing candidate pool of strategies.

(d) Use empirical game-theoretic regret analysis to evaluate the CDO-EGTA method, where our method outperforms other literature methods.

(e) Have shown ways to improve the performance of the CDO-EGTA method.

1.3 Dissertation Overview

The rest of this document is organized as follows. Chapter 2 describes the autonomous trading problems, i.e., the primary domain of the dissertation and surveys related work. In Chapter 3, we show work on the wholesale market price prediction. Chapter 4 explains the Monte Carlo Tree Search wholesale bidding strategy using price predictors. Chapter 5 shows the results of learning retail strategies in the dynamic Power TAC retail markets. Chapter 6 presents a novel method for exploring and selecting better retail strategies from a strategy set of retail policies. Finally, Chapter 7 present the conclusion and future work.

Chapter 2

Background: Autonomous Trading

This chapter describes the main domain, the Power Trading Agent Competition (Power TAC) simulation environment. It focuses on the essential part of the simulator, which will help to understand the rest of the dissertation. Section 2.1 overviews the competition and simulation environment. Section 2.2 describes the main trading problems of the broker agent. Section 2.3 lists Power TAC's modeling assumptions. Section 2.4 overviews the related work on this domain.

2.1 Power Trading Agent Competition (Power TAC) Overview

The Power TAC [31] models a competitive power market where broker agents buy and sell energy and maintain a portfolio of customers who both consume and supply energy. The brokers compete to maximize their profits over approximately 60 simulated days. Each simulation begins with 14 days of pre-game data (*bootstrap data*), which includes data on customers, the wholesale market, and weather data based on the default broker. The brokers participate in three markets: wholesale, tariff, and balancing market, where the simulation models a regulated distribution utility and a real location-based population of energy customers. Customer models include several entities, such as households, electric vehicles, and various commercial and industrial models. In the simulation, brokers try to make a profit by balancing the energy supply and demand as accurately as possible. By efficiently managing stochastic customer behaviors, weather-dependent renewable energy sources, the broker with the highest bank balance wins the competition. We refer the reader to the Power TAC game specification [31] for more detail. Fig. 2.1 illustrates

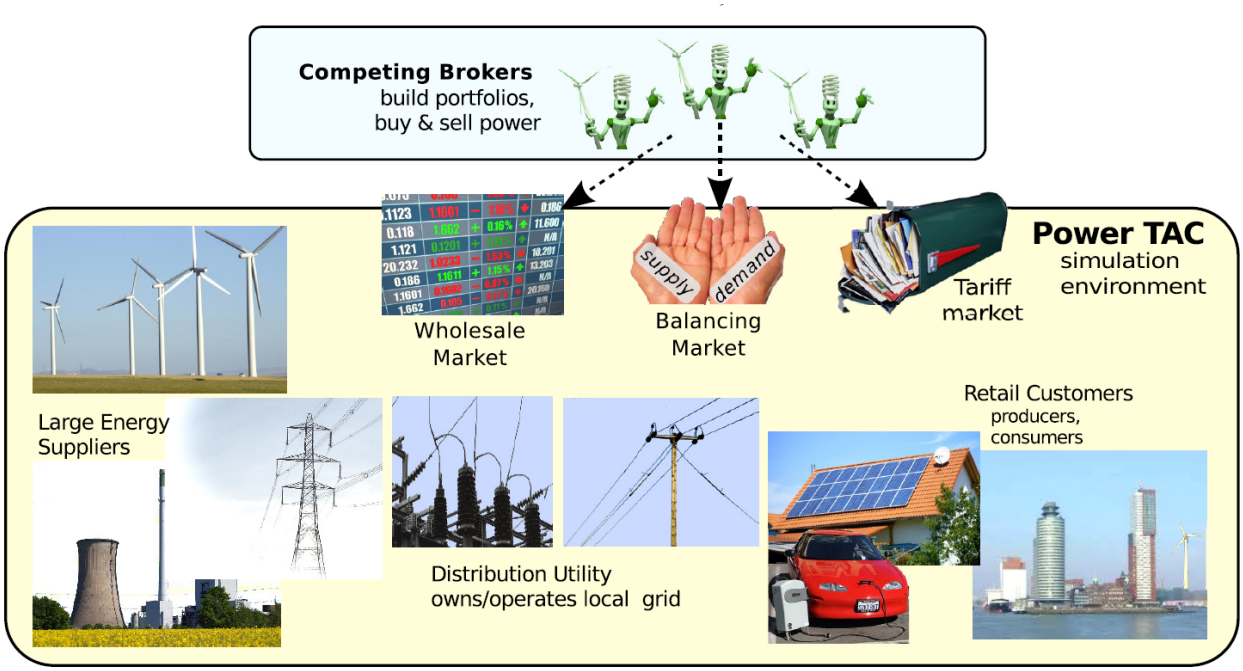


Figure 2.1: Major elements of the Power TAC scenario.

major elements of Power TAC.

2.2 Broker's Power Trading Problem

There are three main markets (Wholesale, Retail, and Balancing) in this simulation, which brings three types of trading problems. The balancing market's goal is to incentivize brokers if they can balance the supply and demand as closely as possible. So the best thing is to avoid participating in the balancing market for now. In this dissertation, we mainly focus on the other two important markets. Fig. 2.2 illustrates a broker's one timeslot activities.

2.2.1 Wholesale Market Trading

In a *double* auction, both buyers and sellers may place bids. An auction is *periodic* if market clearing is triggered based on a specific time interval, so all bids that arrive at an interval are considered in a batch clearing process. A PDA clears bids by matching buy (bid) and sell (ask)

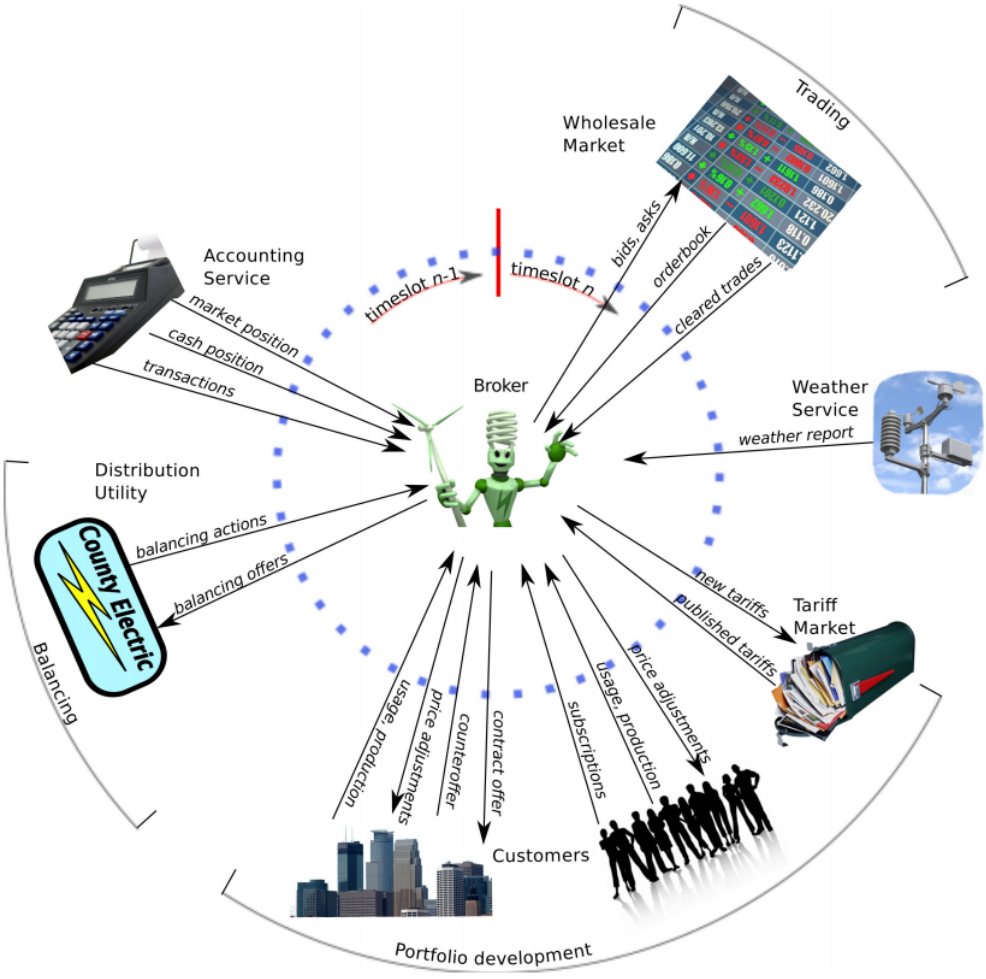


Figure 2.2: Broker's one timeslot activities.

orders and determining the clearing price for each auction [83]. If the minimum ask price has a higher value than the maximum bid price, then the market does not clear. The Power TAC wholesale market functions as a short-term spot market for buying and selling energy commitments in specific time slots, where each time slot represents a simulated hour. Agents can always participate in 24 auctions to trade energy, one auction for each of the next 24 hours. These auctions are *Periodic Double Auctions* (PDAs), similar to those used in European or North American wholesale energy markets [31]. Brokers can submit bids (orders to buy energy) and asks (orders to sell energy), represented by a quantity and an optional limit price. In addition to the brokers' bids, several large "Gencos" also sell energy on the wholesale market. In a *double* auction, both

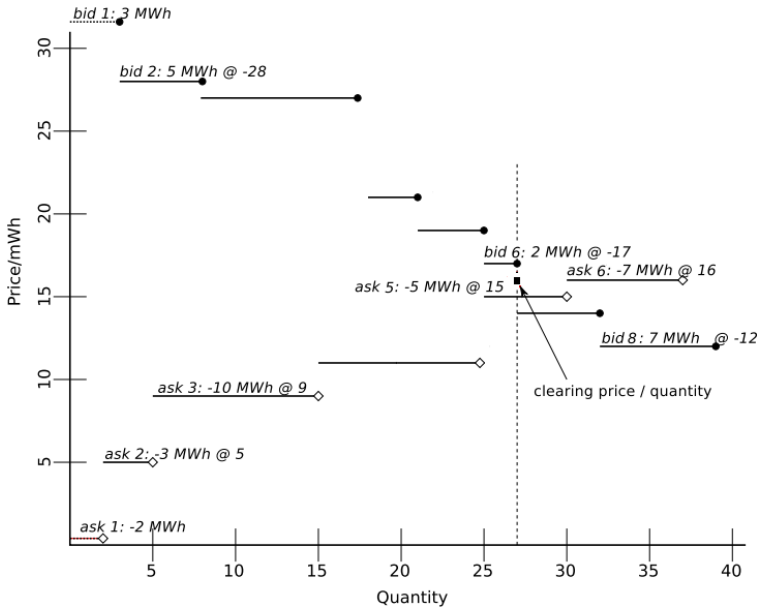


Figure 2.3: Market clearing example: bid 6 and part of ask five are the last to clear.

buyers and sellers may place bids. An auction is *periodic* if market clearing is triggered based on a specific time interval, so all bids that arrive at an interval are considered in a batch clearing process. A PDA clears bids by matching buy (bid) and sell (ask) orders and determining the clearing price for each auction [83]. If the minimum ask price has a higher value than the maximum bid price, then the market does not clear. Fig. 4.4 illustrates an example of an hour ahead auction market clearing. So the wholesale market's primary challenge is to bid at the right price at the right time to purchase energy at the lowest possible price.

2.2.2 Retail Market Trading

In the retail market, the broker trade the energy by publishing tariffs for consumers and distributed renewable energy producers. In Power TAC, it is also called Tariff Market. A tariff can include:

1. Prices which can be flat rate or variable by time or usage
2. Periodic payments
3. Early withdrawal penalties

4. Subscription bonus

Brokers can publish one or more tariffs once every 6 hours, and once a tariff is published, customers can subscribe to it as long as the tariff is active (i.e., not revoked).

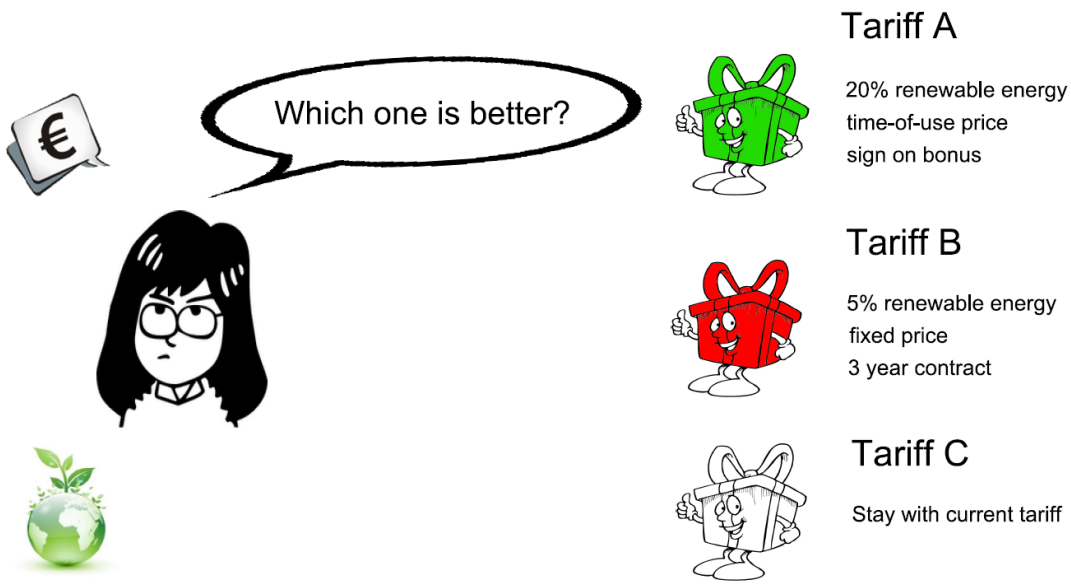


Figure 2.4: Tariff selection problem.

Customers will stochastically subscribe to the attractive, active tariffs in the tariff market. Attractive tariffs are those tariffs where the customer will be able to minimize their cost and discomfort. A discomfort is created when the customer has to shift/curtail their loads to other timeslots. A customer is incorporated with smart meters, which send information to the broker every hour about its consumption and production. Some customers represent a bulk of individuals (e.g., a village of 30,000 people). The whole population does not need to subscribe to a tariff. Different subsets of the same population can subscribe to different brokers' tariffs, but one individual cannot subscribe to the same type of tariffs of multiple brokers. The same type of tariff and individual population have one to one relation. A customer might not do tariff evaluation regularly because of its inertia property. When it evaluates tariffs in the retail market, it might not always make the rational choice. The primary challenge in the retail market is to publish effective tariffs to maximize profit.

2.3 Power TAC Assumptions

The Power TAC Simulator [31] makes these following assumptions:

1. Line capacity limitations are assumed to be non-existent for now but will have to be rethought in the future once more distributed generators and storage facilities are simulated.

2. Power factor effects are ignored for now. A lower power factor means in higher energy losses, and these would possibly affect brokers' decision making on how to charge customers and producers, which is out of scope for now.

3. Power distribution and transformation losses are ignored. These losses are estimated to be 5.5% in North America [25], which we can treat as a constant. Therefore the correctness of the simulation results is not affected by this assumption.

4. In addition to traditional energy generation companies, two more types of producers are simulated. The first type of producer produces energy when they are active and under control of the owners, such as solar panels and wind turbines. The second type of producer is PEV batteries and some CHP units. These are "controllable," i.e., the output can be controlled remotely. Such examples are electric vehicle batteries. These two types of producers are becoming more and more popular in the real world and, therefore, are simulated in the simulation.

5. Real-time technical load balancing operations are out of the broker's action space and are accomplished by using a combination of controllable generators and spinning reserves.

6. The simulation models time as a series of discrete "timeslots" each representing one hour. The regional wholesale market trading intervals are model using this, and this assumption allows the simulator to simulate days rather than minutes or hours.

7. The imbalance between demand and supply for a timeslot is calculated as the difference

of the sum of generation and sum of consumption for that specific timeslot rather than the immediate difference between the two-time series.

8. Using prospective or real-time price signals, some portion of the load can be controlled.

2.4 Related Work

Previous agents in both Power TAC and earlier TAC competitions have considered similar trading problems. In TAC Travel Competitions, boosting-based algorithms have been used to predict hotel prices [81, 22]. Boosting algorithms take weak learning algorithms like regression and boost their performance by training more classifiers and combine them. One agent took advantage of using neural networks for which were trained for each hotel to predict prices [81, 22].

AstonTAC is a Power TAC agent that uses a Non-Homogeneous Hidden Markov Model (NHHMM) to forecast energy demand and price [39]. This was the only agent in that competition that was able to buy energy at a low price in the wholesale market and keep energy imbalance low. TacTex13, the winner of the 2013 Power TAC competition, uses a modified version of Tesauro's bidding algorithm. They modeled the subsequent bidding process as a Markov Decision Process (MDP) for the wholesale market [70]. The wholesale trading module of AgentUDE [48] (the 2014 Power TAC champion) uses an adaptive Q-learning bidding strategy that tracks the past market data. The broker using this technique can understand the market trends regardless of weather conditions and time. In the TAC/SCM game, Deep Maize used a Bayesian model of the stochastic demand process to estimate the underlying demand and trend. It employs a k-Nearest-Neighbors technique to predict the useful demand curves from historical data, self-play games data, and the current game data [35]. TacTex [71], the 2013 and 2015 Power TAC winner, uses a modified version of Tesauro's bidding algorithm, which models the subsequent bidding process as a Markov Decision Process for the wholesale market which is considered to be the current state-of-the-art bidding strategy in PDAs. While there are many other bidding strategies for double auctions [19], most of them are for *Continuous Double Auctions* (CDAs) [64] and would need to be modified for PDAs.

In the tariff strategy, TacTex uses LATTE (Lookahead-policy for Autonomous Time-constrained Trading of Electricity) algorithm [69] to bid in the wholesale market and to publish tariffs in the retail market. It is a general algorithm that can be instantiated in different ways under a variety of environmental conditions. TacTex tries to estimate the long-term utilities of candidates' actions and, in turn, reduces the complexity of searching in a high dimensional searching space by using its look ahead policy. They also did an empirical analysis of the impact of Time-Of-Use (TOU) tariffs in the competitive electric markets as they are proposed for demand-side management both in the literature and the real world. TugaTAC agent [55] uses a fuzzy logic algorithm based on the customer portfolio, where it allows the agent to do adaptive configuration given a complex parameter set in the environment. AgentUDE17 uses an online genetic algorithm to find the best prices for tariffs [49]. This agent is the champion agent in 2017 and 2018. It optimizes the parameters of an electricity consumption tariff, such as unit retail price, periodic payments, signup bonuses, and early withdrawal penalty payments on the fly. Moreover, it also uses a time-of-use (TOU) tariff to reduce the peak demand charges. The authors aim to find a point near the global optimum using a genetic algorithm where the other agents' algorithms fail to explore the full search space because they focus on a small portion of the retail trading problem.

In recent times, deep learning has been successfully worked in learning superhuman strategies for extensive-form games like Go, chess, and shogi [59], [60]. It uses a neural net as a function approximator to generalize the experience, which is more robust than tabular q-learning methods as it can deal with high dimensional state space. A (Deep Neural Net) DNN and associated learning algorithm for deep RL known as a deep Q-network (DQN) [45] has been demonstrated to learn strong policies for playing Atari games using only pixel input, based on the Q-learning method [76] of Watkins and Dayan. Deep reinforcement learning 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 [61] to video games like Super Smash Bros. [17], Dota 2 [47], and Doom [40].

Using specialized algorithms as oracles, McMahan et al. [43] introduced the double-oracle algorithm and applied it to a robot path-planning game. Using a mixed-integer linear program

(MILP) to compute best responses, Jain et al. [26] developed a well-known application of double-oracle for attacker and defender agents in a checkpoint-placement security game. When someone does not have an explicit game model, he/she can develop a simulator and learn best responses through RL. People call it empirical game-theoretic analysis (EGTA) [77] when they solve game models from simulations. Schvartzman and Wellman [57] connected RL with EGTA, applying tabular Q-learning with tile coding to learn better strategies in a continuous double auction trading game. Mason et al. also use Q-learning in a similar work without DNNs [82]. Recently, Wang et al. [75] presented work on applying DQN as the oracle for a zero-sum security game. History Aware Double Oracle EGTA (HADO-EGTA) proposes a modification of DO-EGTA where the author keeps track of previous equilibrium strategies and learn a new DQN strategy over the mix-strategies of current and previous equilibriums [80].

2.5 Chapter Summary

The Power TAC simulator is reasonably rich and realistic and can be viewed as a reasonable substrate domain for studying future power market conditions. Future energy grid will require new market structures [23]. As implementing new power market structures at a large scale in the real world is risky [3], we need to simulate these potential markets first using realistic simulators. Hence Power TAC simulator plays an essential role as a low-risk testbed to test and study general market conditions and agent strategies from the economic point of view.

Chapter 3

Learning Prices in Dynamic Wholesale Market

The main problem we consider in this chapter is learning to predict the clearing prices of periodic double auctions, which can be used by the agent to implement an effective bidding strategy. Our baseline broker used a moving average price prediction based on the price history of the agent. To predict a new price for a week-ahead specific hour price, the baseline agent uses a weighted sum of the current hour's clearing price; yesterday's predicted clearing price for that specific hour and 6 days ahead same hour predicted price. We have experimented with three different machine learning methods to predict clearing prices in the wholesale market: (*i*) REPTree (a type of decision tree) [15], (*ii*) Linear Regression, and (*iii*) Multilayer Perceptron (a type of neural network). We have also investigated a variety of different features for training the predictors. These include eight price features that capture information about the recent trading history, such as the clearing price for the previous hour and the prices for the equivalent time slot in the previous day and week. We also include the weather forecast and time of day because the energy production of renewable energy producers (e.g., solar) depends on these factors. The number of participants in the game is included because the amount of competition affects the market-clearing price. Finally, we include a moving average of the prices as a convenient way to capture an aggregate price history.

To generate training data, we use simulations with a variety of agent binaries from previous tournaments, as well as a variety of different bootstrap initialization files. We train our models using Weka [24] and evaluate their ability to predict market-clearing prices based on the mean absolute prediction error only for auctions that clear (we do not include auctions that do not clear

in the error calculations). In the following experiments, we investigate the performance of the models in several areas, including how well they generalize to new agents, different numbers of agents, and how important the different features are to the performance of the predictors.

3.1 Supervised Price Predictors

We begin with an essential evaluation of the prediction accuracy of the learned models. One of the most significant factors we discovered that influences the accuracy of the models is how we handle auctions that do not clear. In many cases, an auction will have no clearing price due to a spread between the bid and ask prices, which results in the simulation returning null values for these prices. This causes significant problems with the price features we use, as well as the final error calculations. To improve this, we calculate an estimated clearing price for auctions that do not clear by taking the average of the lowest ask price and the highest bid price. Figure 3.1 shows

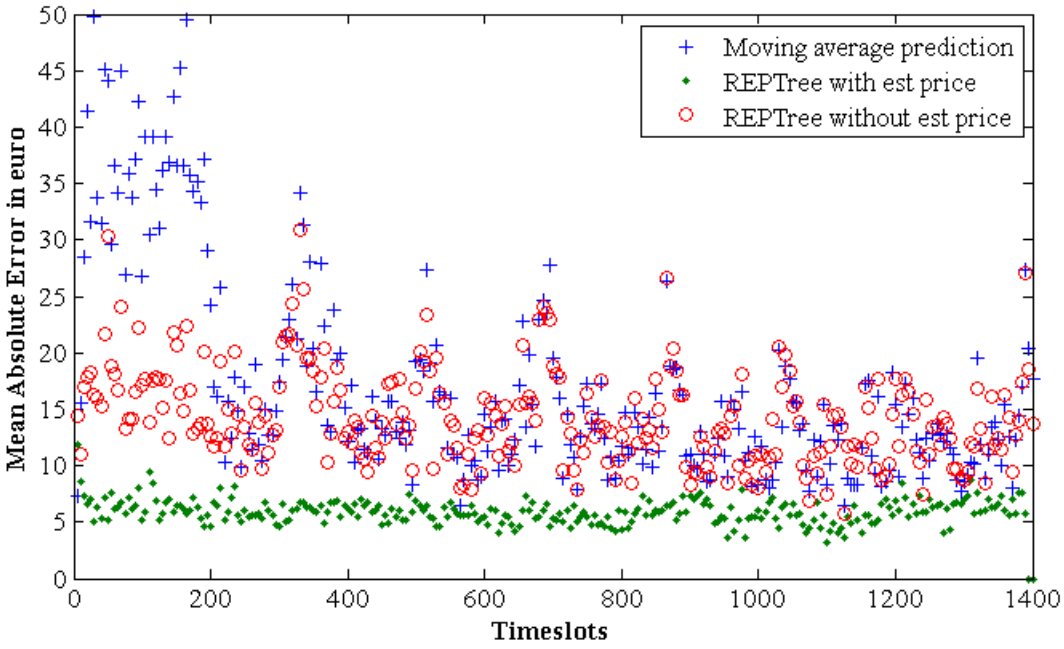


Figure 3.1: Effect of Clearing Price Estimation

the prediction errors during a single simulation for two different REPTree models trained on 20

games, one with estimated clearing prices and the other without. We also include the errors for a simple moving average price predictor as a baseline for comparison. Each data point shows the average error for all auctions in a window of five timeslots. The data show that both REPTree models outperform the moving average predictor. However, the version with estimated clearing prices is dramatically better and produces much more consistent predictions throughout the entire game.

Next, we compare the performance of the three different learning methods with different amounts of training data ranging from 5 to 20 games. We evaluated a variety of different configurations of hidden layers for the Multilayer Perceptron model; only the best one is shown here (MP-20-20, i.e., 2 layers neural network with 20 nodes in each layer). Figure 3.2 shows

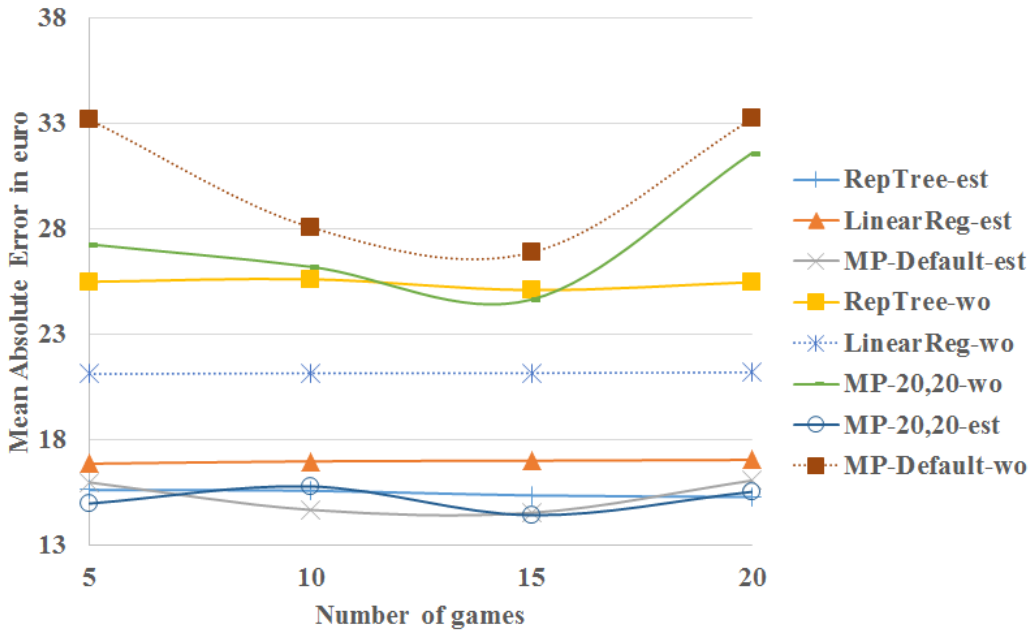


Figure 3.2: Comparison of Several Prediction Models by Number of Games

the average mean absolute error for the different models based on five games of test data. The results show that the decision tree model makes good predictions compared to other models. The decision tree model slowly improves according to the number of games, while other models do not show this trend. The default Multilayer Perceptron (1 layer with 18 nodes) with estimated

prices shows some improvement in the initial number of games than REPTree but finally loses to REPTree in the 20 game model. In all cases, the models with estimated clearing prices are much better than models without estimated prices.

3.2 Evaluation of Price Predictors

3.2.1 Predictor Against Different Types of Agents

In the Power TAC competition, broker agents play many games against different opponents with varying strategies. Here, we test how well our predictors generalize to playing new agents that are not in the training data. We test our models on games of the same size, but varying one of the agents in the game between AgentUDE15, cwiBroker15, and TacTex14. All the predictor models are generated from the training dataset where AgentUDE is used. Figure 3.3 shows the average

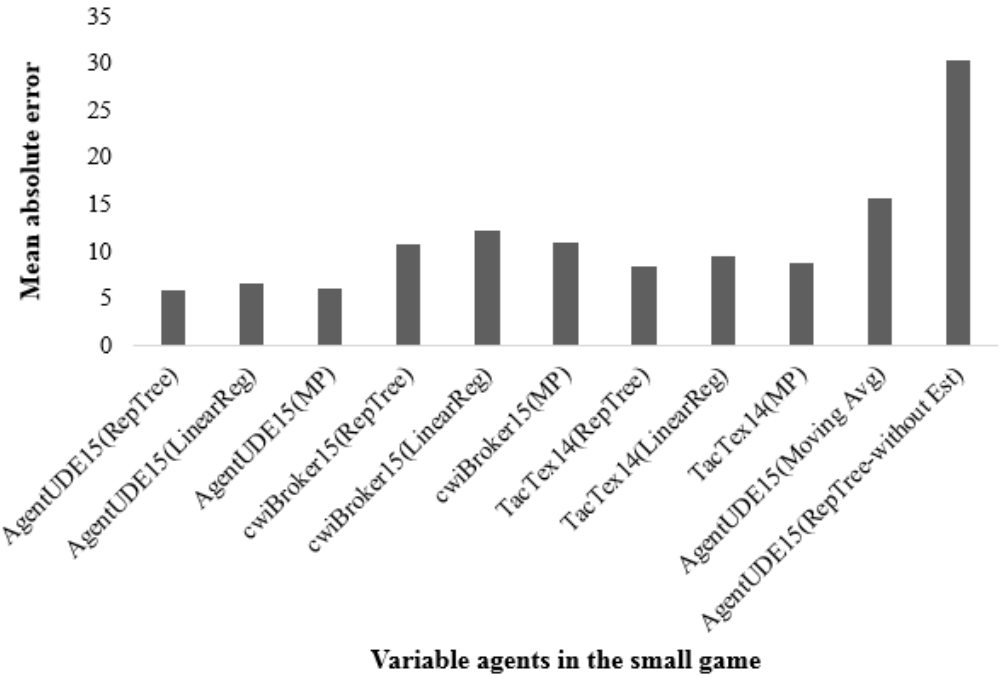


Figure 3.3: Comparison of Several Prediction Models

results for each of the learning methods in the three different agent environments. The REPTree

predictor consistently does better than others, though differences are depending on the pool of opponents. We can also see that the models do best against AgentUDE (which was in the training set), and there is a significant decrease in accuracy when playing either cwiBroker or TacTex.

3.2.2 Predictor Against Different Number of Agents

In the competition or real world, broker agents must play in games with varying numbers of opponents. We experiment with different number of brokers in the games, ranging from 3 to 7 brokers. We focus here on the REPTree predictor since it performs better than the others consistently in previous experiments. The 5 agent predictor models trained on data generated from SPOT(Baseline), AgentUDE15, cwiBroker15, SampleBroker, Maxon14 and the 7 agent predictor models use data from SPOT(Baseline), AgentUDE15, cwiBroker15, SampleBroker, Maxon14, Maxon15, COLDPower and CrocodileAgent15. The test data uses the same agents. We also trained a predictor based on a mixed dataset that included the same number of training games, but with a combination of 3, 5, and 7 agent games. The data in Figure 3.4 shows that, in each case, the model trained on the correct number of agents has the best performance. However, we also note that the mixed model performs very well in all three cases. Table 3.1 shows the average error of the predictor models over the 3 different test game data, and demonstrates that the average error for the mixed model is better than any of the other three models.

Table 3.1: Average Error for the Various Agent Models

7 Agent	**5 Agent**	**3 Agent**	**Mixed Agent**
13.406	13.714	13.958	13.225

3.3 Feature Evaluation

To evaluate which features are the most important for the predictions we used ReliefFAttributeEvaluation [54] and the Ranker method in Weka to rank our 18 features. We also used the

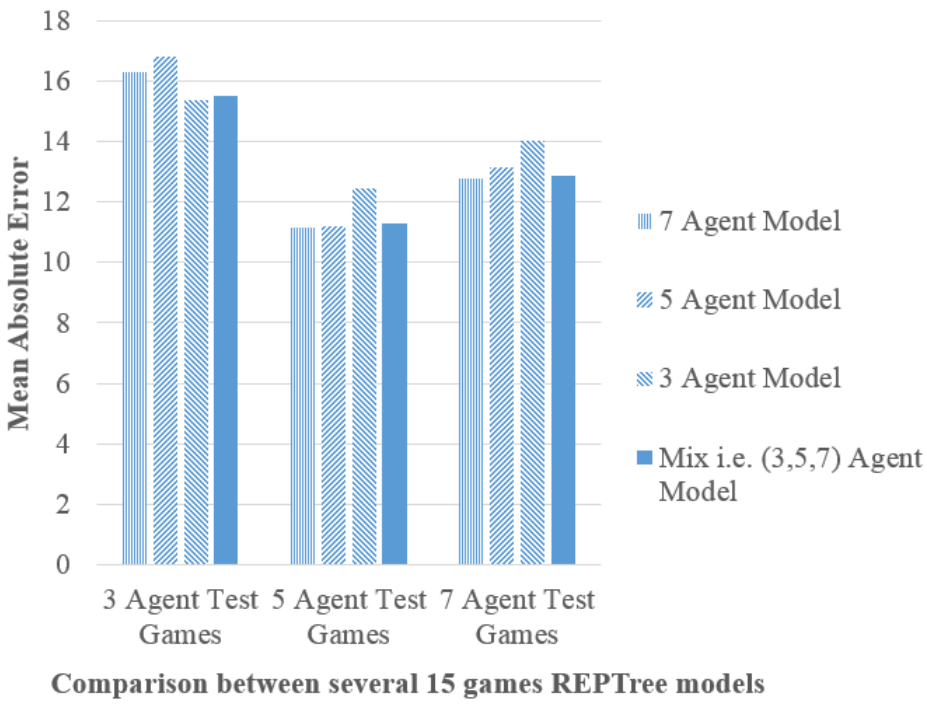


Figure 3.4: Comparison for Different Number of Agents

ClassifierSubsetEval method and best-first search to get the best subset of features from all the features. Table 3.2 shows the top 7 features using the ranker algorithm and the best subset of features using the ClassifierSubsetEval method. We ran the subset evaluation on 5, 10, 15, and

Table 3.2: Feature Evaluation

Ranked Features	**Subset Evaluation**
PreviousHourN_1Price	YesterdayClrPrice
PrevHourClrPrice	PreviousHourN_1Price
PredictedClrPrice	PredictedClrPrice
YesterdayClrPrice	PrevHourClrPrice
AWeekAgoN_1Price	Day
YesterdayN_1Price	HourAhead
PrevOneWeekClrPrice	CloudCoverage

20 games and, for all cases, we found a consistent subset of seven features. The features such as temperature, day of a month, month of a year, the number of participants are ranked low and also out of the best subset. We could potentially discard these types of features while training a predictor model. From the ranked feature column, we see that price features are significant for the REPTree predictor model. So, adding additional features of this type may improve performance.

We use two basic price prediction methods. While these could likely be improved with more sophisticated machine learning methods, the main focus of our work is on the bidding policies, and it would be trivial to adopt better price predictions in any of the policies we propose.

3.4 MDP Price Predictor

We also implement a price predictor used by one of the best agents from previous Power TAC competitions, TacTex [70]. TacTex uses a MDP price predictor where the MDP is defined as follows:

- **State:** $s \in \{0, 1, ..., 24, success\}$, $s_0 := 24$.
- **Action:** limit price $\in \mathbb{R}$.
- **Transition:** A state transitions to one of two states. If a bid is partially or fully cleared, it transitions to the terminal success state. Otherwise, a state s transitions to state $s - 1$.
- **Reward:** In state $s = 0$, the reward is the balancing price per energy unit. In states $s \in 1, \ldots, 24$, it is 0. In state $success$, the reward is the limit price of the successful bid. Both balancing price and limit price are taken as negative, so maximizing the reward results in minimizing costs.
- **Terminal State:** : $\{0, success\}$

Since the MDP is acyclic, solving it requires one back-sweep, starting from state 0 to state 24. The value function is defined as follows [70]:

$$V(s) = \left\{ \begin{array}{c} \text{balancing price if } s = 0 \\ \min_{\text{limit price}} \{p_{cleared} * \text{limit price} + \\ (1 - p_{cleared}) * V(s-1)\} \text{ if } 1 \leq s \leq 24 \end{array} \right\}$$

The transition probability $p_{cleared}(s, \text{limit price})$ for a limit price is computed as follows:

$$\frac{\sum_{tr \in trades[s].tr.\text{clearing price} < \text{limit price}} tr.\text{cleared energy amount}}{\sum_{tr \in trades[s]} tr.\text{cleared energy amount}}$$

Using this MDP's solution, TacTex determines an optimal limit price for each of the 24 states. We compare the accuracy of this online price predictor against the REPTree predictor with a variation of sophisticated bidding strategies where we find that REPTree can predict better prices than online MDP price predictor. A more detailed description is explained in the experiments section 4.8.

3.5 Using Price Predictors for Bidding

We took the best performing predictor from our experiments (REPTree) and tested whether using these predictions could improve performance for a basic bidding strategy. This strategy attempts to target auctions where the clearing price is predicted to be low and to buy a higher volume of the needed energy in those specific auctions. Figure 3.5 shows that using the new predictions and bidding strategy, the agent can buy a high volume of the needed energy when the average clearing price is lowest against the champion agent Maxon15.

3.6 Chapter Summary

It will be a significant advance if intelligent bidding agents can replace humans in wholesale energy markets. We have shown as a first step that we can successfully use machine learning to predict market prices in these auctions in a realistic, smart grid simulation. These predictions are much more accurate than baselines that use moving averages to predict prices, and the error amount is small enough that these should be useful in more sophisticated bidding strategies. The next chapter focuses on designing and evaluating new bidding strategies for these auctions to make use of the price prediction methods described here.

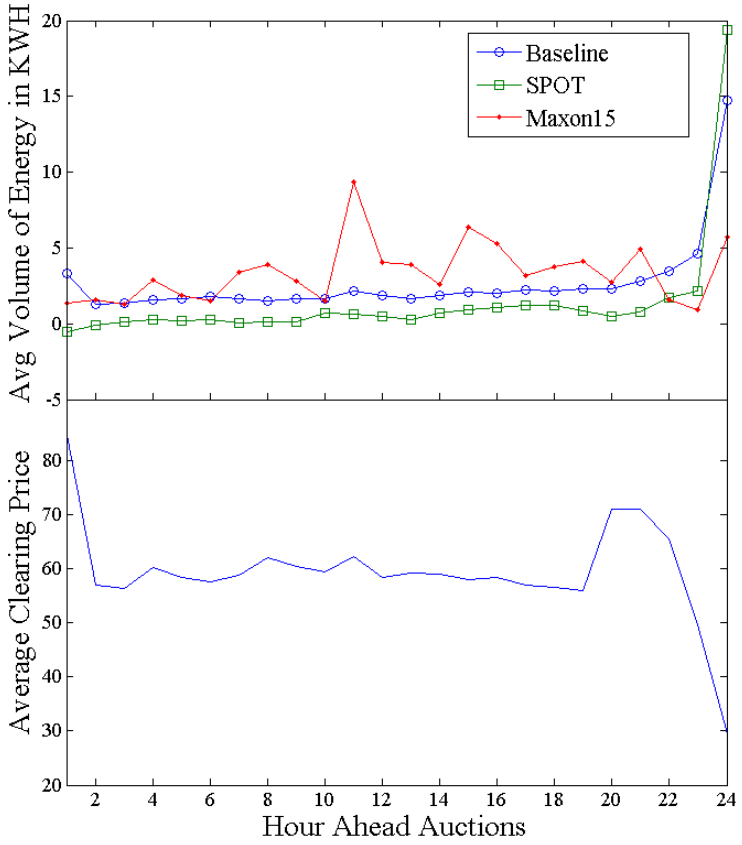


Figure 3.5: Comparison for Wholesale Bidding Strategies

Chapter 4

Wholesale Bidding Strategies

4.1 Introduction

Double auctions are ubiquitous, serving as a general method for buyers and sellers to exchange goods at prices determined by market interactions. Periodic Double Auctions (PDAs) are a specific type of double auction in which bids are cleared periodically in a sequence of pre-defined time periods, as opposed to immediately upon arrival as in a continuous auction. While PDAs can be used to trade any type of good, one prominent use of this style of auction is in short-term energy markets used to balance demand on the power grid (e.g., NordPool, FERC, or EEX [52, 14, 16]). In this chapter, we present general methods for bidding in PDAs that could be applied to any type of market with this structure, but focus our evaluation on energy markets due to the availability of a very realistic simulator and competitive bidding strategies designed by other researchers for this domain. The specific contributions of this chapter are as follows: (1) We develop a controlled simulation environment to test PDA bidding strategies for realistic wholesale energy markets. (2) We present two fast heuristic bidding policies based on machine learning prediction methods for forecasting market-clearing prices. (3) We propose a dynamic MCTS bidding strategy that performs a more comprehensive search of the policy space using an anytime algorithm. (4) We perform empirical evaluations of our PDA bidding strategies using the principles of the Power TAC wholesale market as a platform and show that we significantly improve over both baseline and state-of-the-art bidding strategies from the Power TAC competition.

4.2 Background

4.2.1 Periodic Double Auctions (PDA)

In a *double* auction, both buyers and sellers may place bids. An auction is *periodic* if market clearing is triggered based on a specific time interval, so all bids that arrive at an interval are considered in a batch clearing process. A PDA clears bids by matching buy (bid) and sell (ask) orders and determining the clearing price for each auction [83]. If the minimum ask price has a higher value than the maximum bid price, then the market does not clear. Fig. 4.1 illustrates an

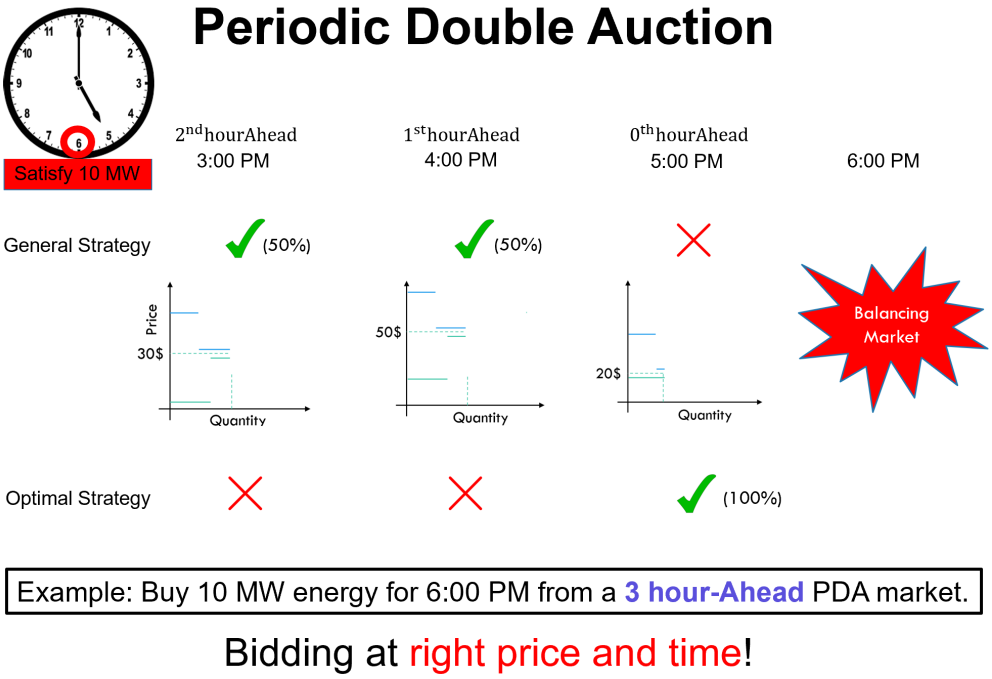


Figure 4.1: A three hour ahead Periodic Double Auction market example

example of a three hour ahead auction market-clearing process. Suppose the current time is 3 PM, and our broker needs to satisfy 10 MW of energy at 6 PM. We can see the optimal strategy to purchase energy from this market is to bid with the right price in the last hour-ahead auction (5 PM market) as it offers the lowest price than the previous two hour ahead auctions.

4.2.2 Monte Carlo Tree Search (MCTS)

MCTS [5] is a tree search algorithm that uses stochastic simulations, as illustrated in Figure 4.2. It incrementally builds a search tree using the following phases:

***(i)* Selection:** The selection of the next state follows the UCT algorithm [36], which balances between exploitation and exploration; ***(ii)* Expansion:** If a state is not in the tree it is added as a new node, so the tree adds one node in each simulation; ***(iii)* Simulation:** After expansion, actions are randomly selected from the action set to perform a rollout to the end of the game; ***(iv)* Backpropagation:** After the simulation, each tree node that was visited during that game is updated by increasing the visit counts and the expected value. These four phases represent one MCTS simulation. After completing the specified number of MCTS simulations, the algorithm selects the current action with the best value.

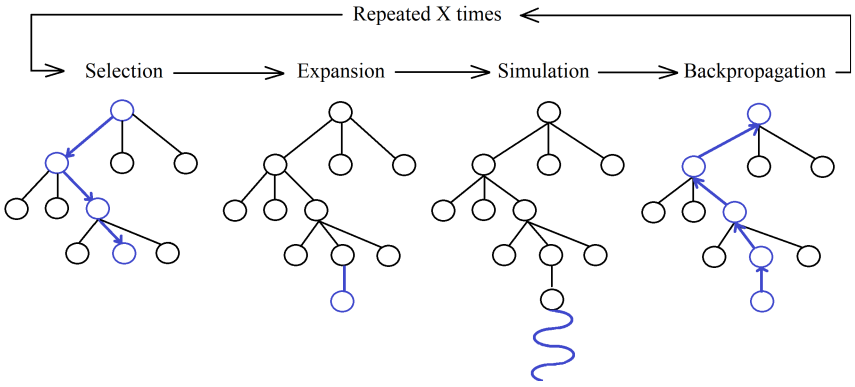


Figure 4.2: Monte Carlo Tree Search

4.3 Proposed Heuristic Bidding Policies

We observe the historical market clearing prices of periodic double auction markets. Figure 4.3 shows that the market clearing prices in an hour ahead periodic double auctions form a gaussian distribution. Using this insight, we design our action space with two variables i.e., a mean value (μ) and a standard deviation (σ). In the previous chapter, we learned price predictors to predict

Figure 4.3: Gaussian distribution in PDA market clearing prices

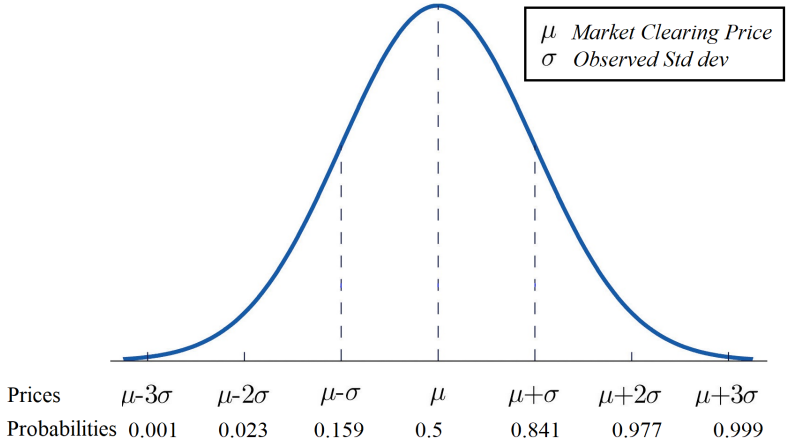


Figure 4.4: Price Multipliers in MCTS action space.

market clearing prices in an hour ahead auction. Here we use these price predictors to estimate the mean value of the distribution (μ). Then we find the average observed standard deviation of the PDA market and use it as (σ). Thus we design our action space, shown in figure 4.4, using both μ and σ to bid in an hour ahead auction. If we increase the bid price, we see from figure 4.4 that the bid becomes more expensive/aggressive, which results in a higher probability of getting the bid cleared in the auction or winning the specific auction. We now introduce two high-speed heuristic bidding strategies for PDAs. These are based on the idea of adjusting the probability of winning each auction, such that the cumulative probability of winning is above a given threshold. We

consider only a single bid for each auction with the total volume that we want to purchase. Both strategies begin by initializing the bidding prices for all auctions with some minimum probability of winning ($P_{min} = 0.025$ in our experiments). They then incrementally raise the bid price in some auctions until the cumulative probability of winning reaches a given threshold.

4.3.1 C1 Strategy

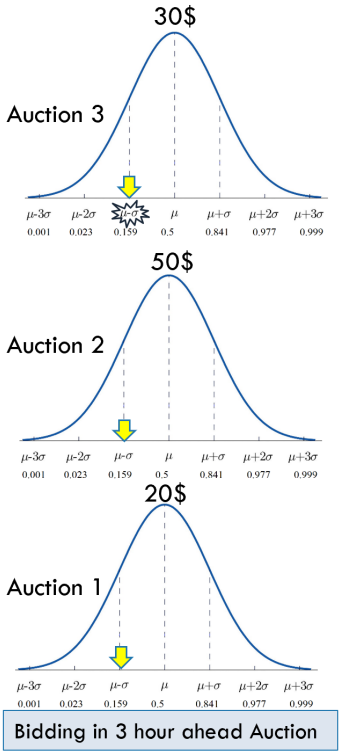


Figure 4.5: C1 Strategy in a 3 hour ahead PDA market

C1 raises the probability of winning (and corresponding bid price) for each auction in a uniform, round-robin pattern such that all probabilities are roughly equal. As the number of bidding opportunities decreases for a specific hour-ahead auction market, the C1 strategy increases its aggressiveness. Figure 4.3.1 shows that all the 3 hour ahead auctions have the same probability value, and the agent selects the price as a limit price of the bid, which maps to the selected probability value by the C1 strategy.

4.3.2 C2 Strategy

C2 increases the bid limit price of the auction, which has the lowest predicted clearing price to increase the probability of winning on that specific auction. This increases the probability of winning the auctions with the lower (predicted) prices. This strategy is a risk-taking strategy that looks for an opportunity to bid in the lowest clearing priced auctions at high limit prices. The idea is to get the bid cleared with the lowest clearing price set by other bidding agents. Figure 4.3.2 shows that all the 3 hour ahead auctions have different probability values because they have different market clearing prices. Auction 1 has the lowest predicted price, so C2 assigns a higher probability value on that auction so that it can bid with a higher price, which can eventually help the agent to clear its bid in the specific auction.

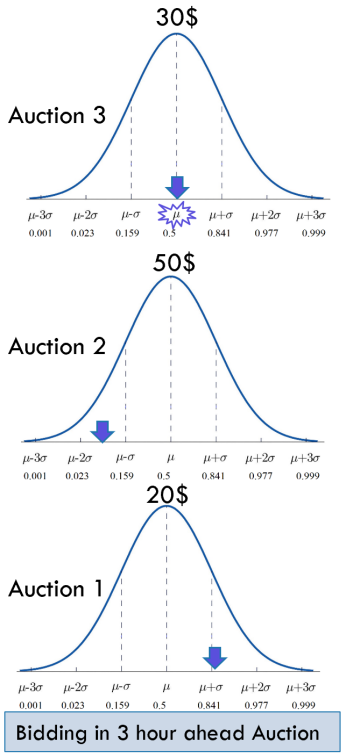


Figure 4.6: C2 Strategy in a 3 hour ahead PDA market

We stop the process when the multiplication of the individual probabilities of winning exceeds the threshold. Both strategies will become more aggressive in later auctions when there are fewer

future opportunities to buy. We show that both of these policies perform well as heuristics, but they can also serve as the starting point for a more comprehensive search.

4.4 Proposed MCTS Bidding Strategy

We now design a more general search policy for finding good bids based on MCTS. **Action:** We represent the main actions of the MCTS strategy as prices relative to the predicted distribution of clearing prices for the current auction. Each action $action_m$ is represented by $\{\mu, \sigma, \{\Delta_{min}, \Delta_{max}\}, \gamma\}$, where μ represents the limit price, σ is the observed standard deviation of the clearing price distribution, $\{\Delta_{min}, \Delta_{max}\}$ is the minimum and maximum price multiplier tuple, and γ is the volume (in %) of the current demand δ. We get the value for μ from one of the price predictors. We estimate σ from 30 four-broker simulations, and record the standard deviations of the errors in the predictions of the auctions. The Δ_{min} and Δ_{max} are used by the agent to create μ_{min}^{mcts} and μ_{max}^{mcts} by varying μ using:

$$\mu_{min}^{mcts} = \mu + \Delta_{min} * \sigma \text{ and } \mu_{max}^{mcts} = \mu + \Delta_{max} * \sigma$$

Using $\{\Delta_{min}, \Delta_{max}\}$, μ, and σ, the MCTS strategy is able to simulate actions in different price ranges. For example, if $\{\Delta_{max}, \Delta_{min}\} = \{1, -1\}$, our MCTS strategy varies its bid prices in the first standard deviation range as illustrated in Figure 4.4. An action is a *NO-BID* action when $\gamma = 0$. For dynamic actions, we use an accurate price instead of price multipliers, where $\mu_{min}^{mcts} = \mu_{max}^{mcts}$. The idea here is that these dynamic actions should be able to hone in on more "pinpoint" prices if they are selected during MCTS simulations.

State: States are nodes in the search tree representing the history of action choices. Each state keeps a memory of its corresponding action-id, visit-count, and $C_{avgUnit}$ (i.e., total avg. unit cost incurred by the agent in this auction and all future auctions). The agent selects the state that has the highest UCT value while doing simulations. Each action leads to a specific state, so an agent with an *action-space*$_m$ size of m actions can go into m different states from a specific state.

Transition: A state S_m^n transitions to one of the states in the next time period, $S_0^{n-1}, \ldots, S_m^{n-1}$.

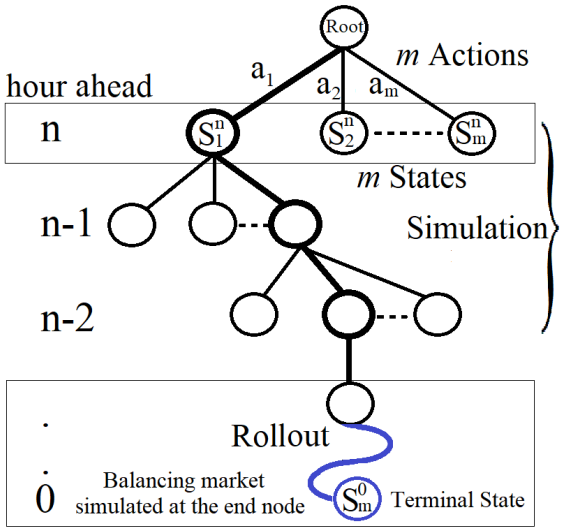


Figure 4.7: MCTS Tree for the n^{th} Hour-Ahead Auction Bidding

Terminal State: The zero hour-ahead states are terminal states, $\{S_0^0, \ldots, S_m^0\}$. If there are m actions and n hour-ahead auctions, the search tree will have m^n final states.

Reward: While doing rollouts/simulations for timeslot t, if the agent reaches a terminal state, it gets the balancing cost $C_{bal,t}$ as a reward, which corresponds to the price the agent would pay for energy in the balancing market. Otherwise, it gets a simulated cost of C_{sim} that is the sum of the cost paid for energy in all of the auctions.

Simulation: While running a MCTS simulation for the n^{th} hour-ahead auction, the agent first gets the current demand (δ_t^n) and tries to clear δ_t^n using a simulation of the market clearing process by selecting action m. It generates a simulated market clearing price $\chi_{m,t}^n$ from a Gaussian distribution where the mean is equal to μ_t^n and standard deviation is σ. The results of non-deterministic choices (i.e., auction clearing prices) are sampled in our MCTS, but these are continuous distributions so we do not model them explicitly as chance nodes with a finite number of outcomes. If the bid's limit price μ^{mcts} is greater than $\chi_{m,t}^n$, the bid gets cleared. If $\upsilon_{m,t}^n$ volume is cleared in this process, the agent updates its δ_t^n for the remaining hour-ahead auction simulation by deducting $\upsilon_{m,t}^n$ from δ_t^n and repeats the same process until it reaches the terminal state or a state where δ_t^n is zero. At each level of the hour-ahead auction, we get $C_{sim,m,t}^n = \chi_{m,t}^n * \upsilon_{m,t}^n$.

Rollout: If an agent reaches a state without children, the agent selects an action randomly, creates a state, and adds it to the tree. Then it runs the random rollout process by picking actions randomly from the action space and traversing from the newly added state to a terminal state. When it reaches a terminal state, it simulates a simplified balancing market and calculates the C_{bal} by multiplying δ_t^n with the unit balancing cost. At time slot t, the unit balancing cost is calculated by doubling the maximum ask price ($Askprice_{max}$) for that specific time slot's hour-ahead auctions. $C_{bal,t}$ is defined by:

$$C_{balUnitPrice,t} = 2 * Askprice^t_{max} \pm noise_{Gaussian}$$

$$C_{bal,t} = C_{balUnitPrice,t} * \delta_t^n$$

Now, if we aggregate all the costs, i.e., $C^n_{sim,m,t}$ and $C_{bal,t}$, we get the total simulation cost C_{sim} for δ_t^n amount of energy:

$$C_{sim} = \sum_{i=0}^{n} C^i_{sim,m,t} + C_{bal,t}$$

If we divide C_{sim} by δ_t^n, we get the unit cost $C_{avgUnit}$ i.e. $C_{avgUnit} = C_{sim}/\delta_t^n$. For each state, we have a normalized value τ which we will use in our UCT formula. Here $\tau = 1 - (C_{avgUnit}/C_{balUnitPrice,t})$.

To select an action, we evaluate the states using the following UCT formula:

$$\lambda_{UCT} = \tau + \sqrt{\frac{2 * \log(\text{parent-visit-count})}{\text{visit-count}}} + \epsilon$$

where ϵ is a small random number to break the ties. The agent selects the state that has the highest λ_{UCT} value while doing simulations. After repeating a selected number of N_{sim} MCTS iterations for the n^{th} hour-ahead auction, the agent builds the tree as illustrated in Figure 4.7. Then it selects the action that leads to the highest τ state S^n_m from the root. After bidding according to the best action, the agent discards the MCTS tree and builds it again from the scratch when it needs to bid for the $(n-1)^{\text{th}}$ hour-ahead auction. We discard the search trees after bidding because of most information changes between auctions, including new weather predictions, broker behaviors, etc., rendering the previous search trees of little value. Our MCTS agent follows Algorithm 1.

Algorithm 1 MCTS Bidding Strategy

procedure MCTS(State cur, TimeSlot t, HourAhead n)
  $\delta_t^n = \delta_t^{n'}$ = GetDemand(t, n);
  while !HasReachedTerminalState(cur) or $\delta_t^{n'} > 0$ **do**
    n' = cur.HourAheadAuction;
    if HasUnvisitedChildStates(cur) **then**
      cur = selectChildRandomly(cur);
      expand(cur);
      $[\sum_{i=0}^{n'} C_{sim,m,t}^i, \sum_{i=0}^{n'} \upsilon_{m,t}^i]$ = rollout($\delta_t^{n'}$);
      C_{sim} += $\sum_{i=0}^{n'} C_{sim,m,t}^i$; $\delta_t^{n'}$ -= $\sum_{i=0}^{n'} \upsilon_{m,t}^i$;
      break;
    else
      cur = selectChildByUCTValue(cur);
      $[C_{sim,m,t}^{n'}, \upsilon_{m,t}^{n'}]$ = simulation($\delta_t^{n'}$);
      C_{sim} += $C_{sim,m,t}^{n'}$; $\delta_t^{n'}$ -= $\upsilon_{m,t}^{n'}$;
    AddToVisited(cur);
  C_{sim} += $C_{bal,t} = C_{balUnitPrice,t} * \delta_t^{n'}$;
  $C_{avgUnit} = C_{sim}/\delta_t^n$;
  backpropagation($C_{avgUnit}$, visitedNodes);

We denote our default MCTS agent as MCTS$_d$. The default action space **Action-Space$_\mathbf{d}$** for MCTS$_d$ has five bid actions and one *NO-BID* action.

- **Number-of-MCTS-simulation** (N_{sim}^d): The default number of MCTS iteration is set to 10,000.
- **Bid-Volume** (γ_d): All actions bid with 100% of the current demand (except for NO-BID).
- **Price-Multipliers** $\{\Delta_{min}^d, \Delta_{max}^d\}$: The five price multiplier tuples for the actions are as follows: $\{-1, 0\}$, $\{0, 1\}$, $\{-1, 1\}$, $\{0, 1\}$, and $\{0, 2\}$.

- **Number-of-Bids** (N^d_{bid}): Submits 10 bids. 9 bids are minimum bandwidth bids ($bidvolume = 0.001$ mW) with limitprice starting from the μ^{mcts}_{min} to μ^{mcts}_{max}. The 10th bid is the main bid which is submitted at μ^{mcts}_{max} price.

4.5 Dynamic MCTS Strategy

An important restriction on MCTS$_d$ is that it searches only a fixed space of possible bidding actions. Here, we consider a simple dynamic MCTS policy [6] that adds promising new actions to the search space over time. In particular, at the time of reaching a specific threshold number of iterations in the MCTS simulation, we add a new action, equal to the lowest simulated unit cost price among all the children states of the root, to the action space. The simulated unit cost price also includes the balancing price. The thresholds are set to 5%, 10%, 20%, and 50% of the total MCTS simulations in the experiment. So, we add a total of 4 dynamic actions to the action space. We denote this agent as MCTS$_{dyn}$. An enhancement of the dynamic MCTS strategy is possible if we can seed the initial action space with good, simple, and fast heuristic strategies.

4.6 Experimental Methods: Testbed

The Power TAC simulation is complex, and there are many factors in the performance of the agents. We decouple the wholesale market from the other features of Power TAC to gain more control and focus just on the performance of the bidding strategies in the wholesale PDA. We implemented a controlled wholesale energy market simulator that imitates a variable number of hour-ahead periodic double auctions (e.g., 24 hour-ahead auctions) as close as possible to Power TAC wholesale energy market [31].

4.6.1 Setting Supply

In Power TAC simulation, the Grid Genco is an abstract entity that simulates a population of generating facilities distributed over a large geographic area, such as the MISO or PJM ISO

organizations in North America. It generates a supply curve that is composed of a succession of ask orders with prices and quantities drawn from distributions that characterize the full price curve observed in the MISO and/or PJM LMP markets. [31]. To set the supply, we collected producer supply data by running 30 Power TAC simulations with 30 different *bootstrap data* and calculate the average energy supply per hour.

4.6.2 Setting demand

Power TAC's wholesale market simulates 11 different city demands. After observing several simulations in Power TAC, we find that the ratio of energy supply and customer demand is nearly 300 for a city. So, to set the demand, we divide the average supply per hour by 300. This is a low demand scenario where the wholesale market provides enough energy to the broker agents to satisfy their demand. We equally distribute the demand to all the brokers in our games for fairness.

4.6.3 Setting Forecasting Features

The demand is modeled on the regional demand observed in the north-central U.S. and is relatively weather-dependent. It simulates a region with cold winters and relatively hot summers, and demand is more dependent on temperatures above the comfort zone than on temperatures below the comfort zone. The Power TAC simulation provides wind speed, wind direction, cloud coverage, and temperature forecasts for every hourAhead auctions. We collected these day-ahead weather forecast data from the 30 simulations' log and used it as forecasting features in our isolated simplified wholesale PDA simulator.

4.7 Benchmark Strategies

4.7.1 Zero Intelligence (ZI)

ZI agent [21] uses a straightforward strategy, generating random bid prices and ignoring the state of the market. For a given unit, prices are drawn from a uniform distribution between the unit's limit price and a minimum allowable price for the buyer. ZI strategy uses the REPTree/MDP price predictor to get a limit price μ. The bid price is drawn from a distribution where the mean is μ, and the standard deviation is \$10. The broker places only one bid with the quantity equal to the demand at that specific time slot and hour-ahead auction.

4.7.2 Zero Intelligence Plus (ZIP)

ZIP agent [65] maintains a scalar variable m denoting its desired profit margin and combines this with a unit's limit price to compute a bid or ask price p. For failed bids, the agent adjusts p in the direction of beating the failed bid. The broker places only one bid according to the demand at that specific time slot and hour-ahead auction. It uses the REPTree/MDP predictor to get a limit price μ. The profit margin m is set to $c\%$ of μ. So, the bid price $p = \mu + \mu * c\%$. (In our experiments, $c = 1\%$). If a bid fails, then an offset value of $q\%$ of the μ is added to p. Otherwise, q is set to 0. So, when a bid fails, bid price $p = \mu + \mu * c\% + \mu * q\%$. (In our experiments, $q = 10\%$).

4.7.3 TacTex

TacTex [70] is a broker in Power TAC. Using an online reinforcement learning (RL) algorithm, TacTex procures to meet demand as cheaply as possible through the sequential bidding in the wholesale market. TacTex models the sequential bidding as an MDP with a finite number of states. The outcome is a sample-efficient online reinforcement learning algorithm that minimizes procurement costs and helps the agent achieve state-of-the-art performance in competitions and controlled experiments.

When TacTex uses its MDP price predictor, it starts a game with no data and learns to bid on-

Algorithm 2 TacTex: Online RL Wholesale Market Strategy

```
1: neededEnergy[1...24]=ComputeNeededEnergy()
2: densities[0...24] ← AddRecentTradesAndBalancing()
3: if HasEnoughData(densities) then
4:     limitPrice[1...24]=SolveMDP(densities)
5: else
6:     limitPrice[1...24]=RandomizedBiddingPolicy()
7: SubmitBids(neededEnergy[1...24], limitPrices[1...24])
```

line, while acting. In each time slot, it solves the MDP, and its estimates are refined periodically. This results in an online RL bidding strategy, which helps the agent to adapt and optimize its bidding in different market conditions. TacTex's bidding strategy is summarized in Algorithm 2.

When we use REPTree as a price predictor, a slight modification is needed in Algorithm 2. REPTree is a supervised learning price predictor, and it predicts prices without densities. Therefore, lines 2 to 6 in the algorithm 2 are ignored, and the REPTree predicted prices are used instead to fill the $limitPrice[1...24]$ array.

4.8 Choosing the Default Price Predictor

From chapter 3, we see that our best candidate price predictor is REPTree price predictor. Now we create training datasets to learn a REPTree price predictor in an environment where the above-stated strategies are present. First, we use four ZI brokers with a default mean price $30 and default standard deviation $10 to generate bidding limit prices. We ran 30 simulations to generate the initial training dataset. After that, we apply cross-validation on the training set to learn our initial REPTree price predictor model, which we call REPTree-V0, using Weka [24]. Then, we run another 50 simulations with the four strategies (ZI, ZIP, TacTex, and MCTS) using the REPTree-V0 price predictor to generate an iteration one training dataset. We apply the same cross-validation method to learn our REPTree-V1 price predictor. We follow the same procedure again to create our REPTree-V2 predictor. The purpose of this to mitigate the dependencies

of initial ZI agents' fixed mean and standard deviation. After the second iteration of learning, the correlation coefficient value does not improve significantly, so we use REPTree-V2 as our REPTree price predictor in our experiments.

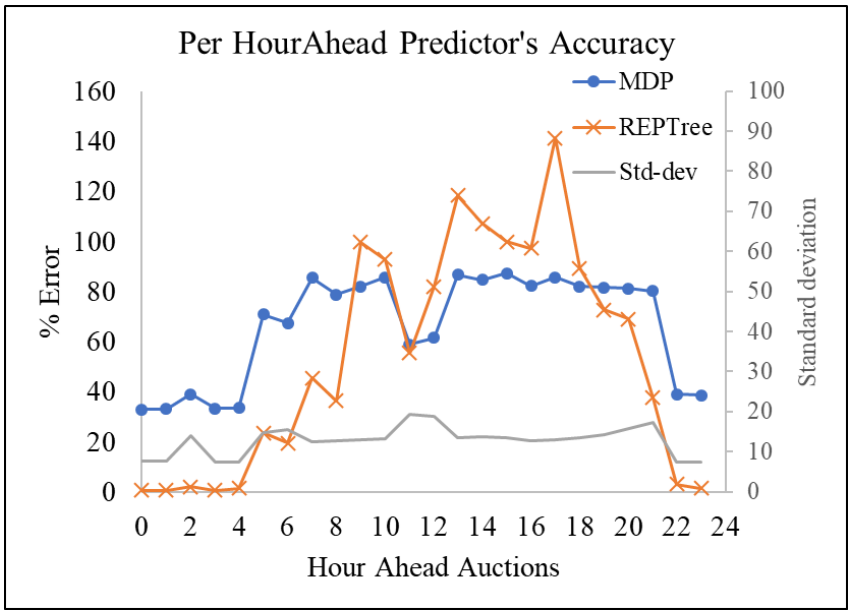


Figure 4.8: Price predictor accuracy comparison.

Figure 4.8 shows the per hour ahead percentage error comparison of both price predictors in a single simulation. We can see that REPTree has better accuracy in predicting per hour ahead of auction prices than the MDP predictor. MDP has an average error of 66.28% for the 24 auctions, where REPTree has 54.05%. The higher standard deviation of market-clearing price in the middle hourAhead auctions results in a higher error rate for the price predictors in the middle. Necessarily, there is much less bidding activity in these middle auctions, so the clearing prices naturally vary much more. After this comparison, we choose REPTree (which is more accurate than MDP predictor) as our default price predictor for the rest of the experiments.

4.9 Experimental Results: MCTS Strategy Variations

MCTS is a statistical anytime algorithm, so more computation time should lead it to better performance [4]. To investigate the effects of additional simulations and the specification of the action space, we did experiments varying three important action space properties ($N_{sim}, \gamma$and$\{\Delta_{min}, \Delta_{max}\}$) of MCTS$_d$ against the three benchmark strategies. First, we vary the number of MCTS iterations (N_{sim}) for the default configuration. Then, we vary the volume percentage (γ) but keeping the price multipliers $\{\Delta_{min}, \Delta max\}$ fixed. Finally, we vary the $\{\Delta_{min}, \Delta max\}$ but keep the γ fixed.

4.9.1 Varying Number of MCTS Simulations

We vary the N_{sim} of MCTS$_d$ by 10, 100, 500, 1K, 2K, 5K, 10K & 15K and run 30 four broker (ZI, ZIP, TacTex, and MCTS) games. As the states are being visited more with a higher number of

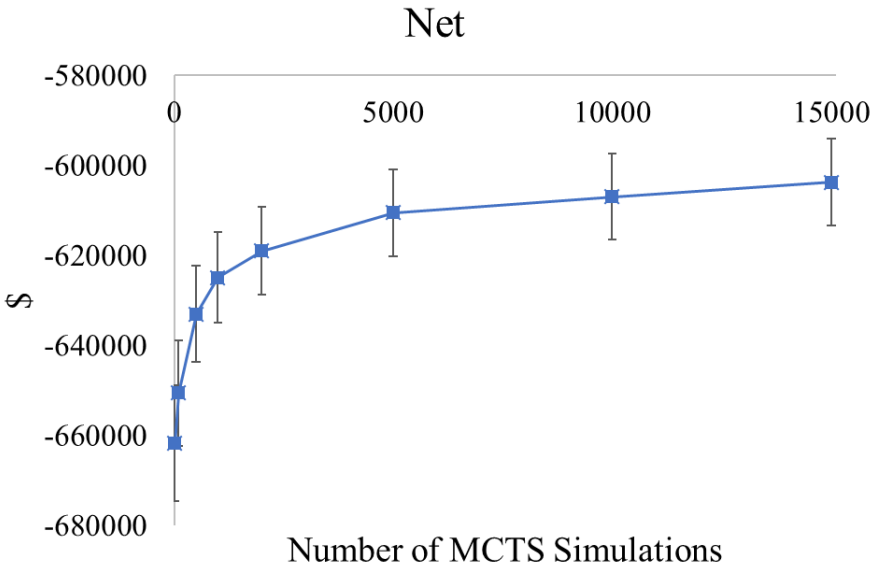


Figure 4.9: Net cost comparison of MCTS$_d$ by varying N_{sim}

MCTS simulations, we can see that the performance of the MCTS$_d$ improves when we increase N_{sim} in Fig. 4.9.

4.9.2 Varying Volume in Action Space

In this experiment, we set $\{\Delta_{min}, \Delta_{max}\}$ to $\{0, 1\}$ (an optimistic range to clear a bid) for MCTS$_d$. By varying the γ only in the action space, we experiment with 3 action-spaces which are as follows:

- **Action-Space$_{\text{vv-2}}$** consists of 2 actions: 1 bid-action with a γ of 100%, and 1 *NO-BID* action.
- **Action-Space$_{\text{vv-3}}$** consists of 3 actions: 2 bid-actions where γ is varied by 100% and 50%, and one *NO-BID* action.
- **Action-Space$_{\text{vv-6}}$** consists of 6 actions: 5 bid-actions where γ is varied by 100%,80%,60%,40%,20%, and one *NO-BID* action.

The main motivation of this experiment is to investigate how MCTS takes advantage of different γ scenarios. We name the agents according to their action-space name. Fig 4.10 shows that

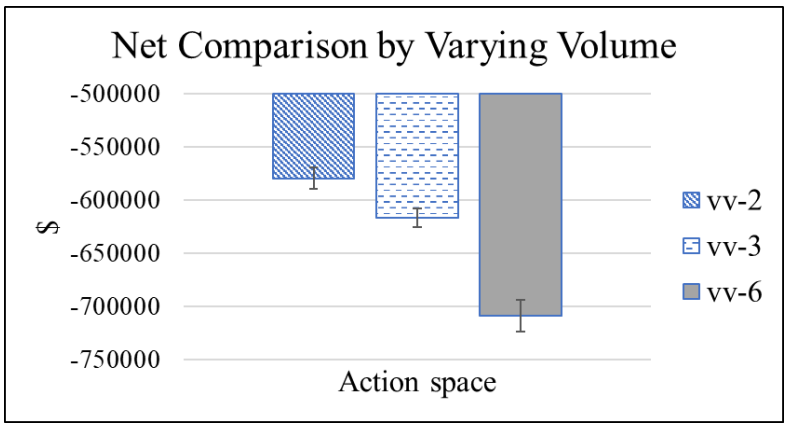


Figure 4.10: Net cost comparison of MCTS by varying volume.

MCTS$_{\text{vv-2}}$ does very well comparing to the other two variations. This is because the number of visits per state is much higher in Action-Space$_{\text{vv-2}}$ as it contains fewer actions than others. It also demonstrates that procuring energy at a full volume at the right moment is a very effective strategy.

4.9.3 Varying Price in Action Space

In this experiment, we want to select a pinpoint price selection and bid with a full volume action space for the agent. So we keep the value of N_{bid} to 1 and γ to 100% fixed for MCTS$_d$. We set $\Delta_{min}, \&\Delta_{max}$ to the same value and vary them with different values for different pin point price selection. We experiment with 3 action-spaces which are as follows:

- **Action-Space$_{\text{vp-2}}$** consists of 2 actions. 1 bid-action with price multiplier of $\{0, 0\}$ and one *NO-BID* action.
- **Action-Space$_{\text{vp-4}}$** consists of 4 actions. 3 bid-actions with price multiplier of $\{-1, -1\}$, $\{0, 0\}$, and $\{1, 1\}$ and one *NO-BID* action.
- **Action-Space$_{\text{vp-6}}$** consists of 6 actions. 5 bid-actions with price multiplier of $\{-2, -2\}$,$\{-1, -1\}$, $\{0, 0\}$, $\{1, 1\}$, and $\{2, 2\}$ and one *NO-BID* action.

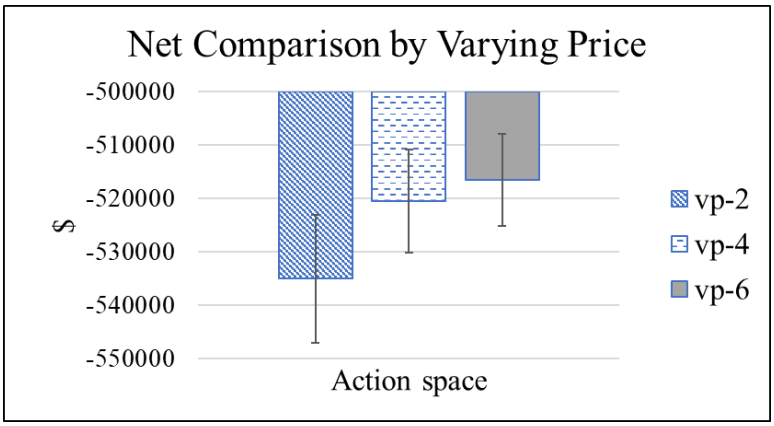


Figure 4.11: Net cost comparison of MCTS by varying price.

From Fig. 4.11, we can see that MCTS$_{\text{vp-6}}$ performs very well compared to other two variations. This shows that a high range of pinpoint price selection option is more effective than having a low range of options.

4.9.4 Final Comparison

We select the two best candidate agents $\text{MCTS}_{\text{vv-2}}$ and $\text{MCTS}_{\text{vp-6}}$ from previous action space experiments and run 30 four broker (ZI, ZIP, TacTex, and MCTS) game for both agents by varying N_{sim}. Fig 4.12 demonstrates that $\text{MCTS}_{\text{vp-6}}$ is doing very well compared to other strategies with

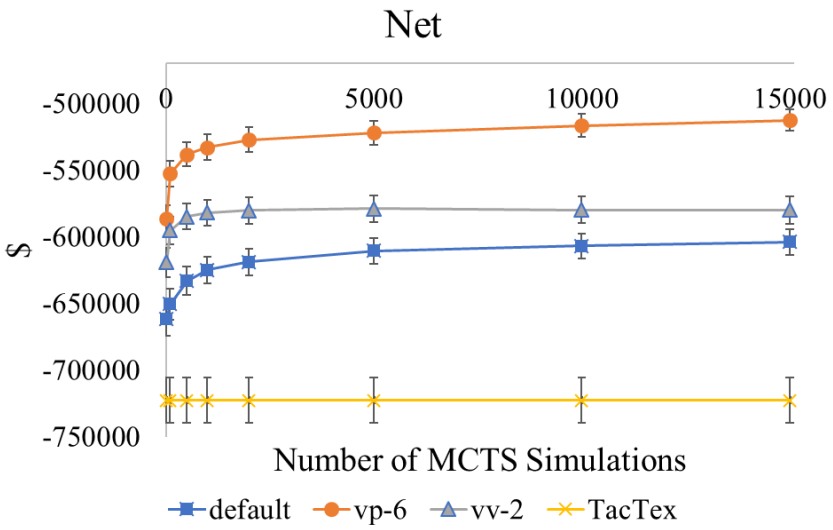


Figure 4.12: Net cost comparison of best candidate agents.

increasing N_{sim} where it is able bid and procure the bid-volume at a lower price. So, the MCTS strategy with a higher number of MCTS simulation and a reasonably wide range of pinpoint price selection option can be considered as the best variation of MCTS. Following these two policies, the MCTS agent simulates the auctions with accurate prices by a higher number of simulations and makes good decisions to bid at the right moment to procure the full demand. We now denote this configuration of MCTS agent as MCTS_{static}, since this agent has a fixed number of actions (5 bid actions and 1 *NO-BID* action).

4.10 Candidate Strategy Comparison

To find the best bidding strategy for the agent, we conducted an empirical analysis for six bidding strategies: ZI, ZIP, TacTex, C1, C2, and MCTS_{static}. We experimented with four-broker games,

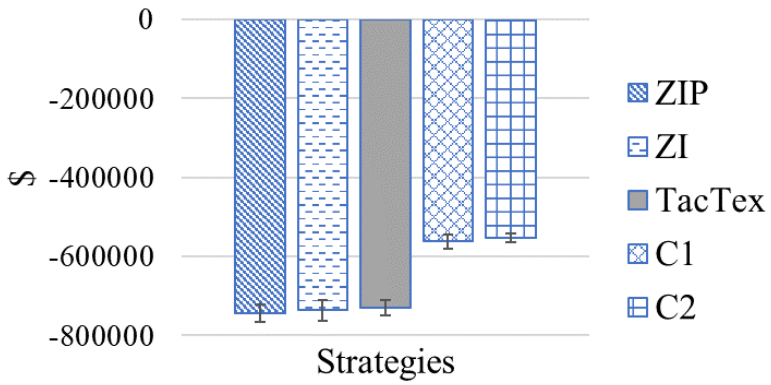


Figure 4.13: Net cost comparison of becnhmark & candidate strategies.

where three brokers were always fixed to be ZI, ZIP, and TacTex. We ran 20 sample games using 20 different boot files that were not used to train the price predictor.

Figure 4.13 shows the performance of the benchmark and candidate strategies. We can see that the C1 and C2 heuristics perform significantly better than ZI, ZIP, and TacTex.

4.11 Dynamic MCTS Comparison

Figure 4.14 demonstrates that MCTS_{dyn} is doing better than MCTS_{static} because of the additional actions that are added to the action space dynamically at several threshold iteration values. To reduce the size of the action space and make the agent more effective in searching the price space, we introduce C1 and C2 as initial action selection strategies inside the MCTS_{dyn} agent. We use them before even starting MCTS to decide which actions will be in the action set considered by MCTS in each state. We name them accordingly as $\text{MCTS}_{dyn\text{-}C1}$ and $\text{MCTS}_{dyn\text{-}C2}$. Figure 4.13 shows that the C2 strategy performs better than the C1 strategy when we run them separately against the three benchmark strategies. C2 tries to exploit the opportunity when the clearing price is low. Figure 4.14 shows that when we seed C2 strategy into the MCTS_{dyn} agent, it performs better than all of the other agents.

Having fewer actions in the action space has the advantage of having a more significant number of visits per state, which is suitable for stabilizing the average values of the MCTS simula-

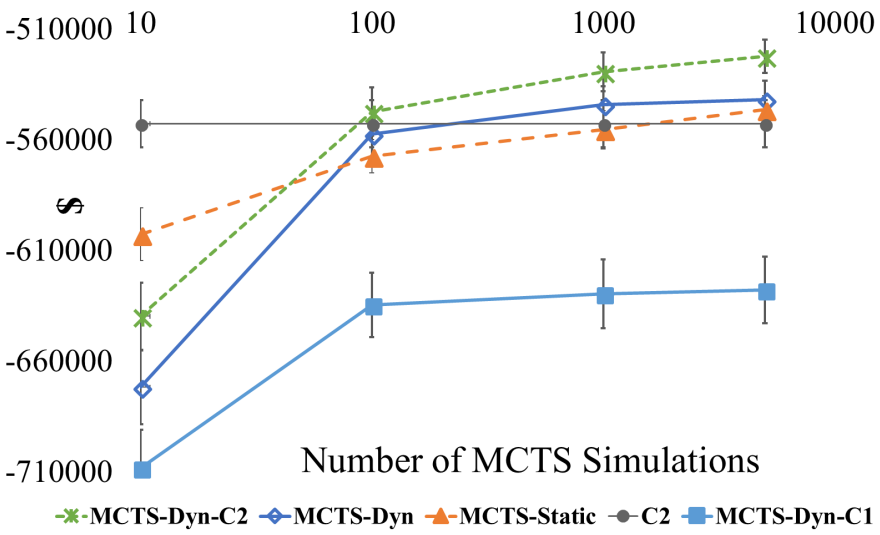


Figure 4.14: Net Cost Comparison of C2 and all MCTS Variations

tion. Having specific actions in small action space is critical. The initial action created by C2 is more accurate to clear the bid in the low priced hour-ahead auctions than C1, and that is why MCTS$_{dyn\text{-}C2}$ is doing significantly better than the other strategies. We found that our MCTS$_{dyn}$ algorithm takes approximately 0.072 seconds to build a 24 hour-ahead MCTS tree with 10k iterations. We have also analyzed the runtime of MCTS$_{dyn\text{-}C2}$ algorithm, and it is 1.23 times faster than the second-best MCTS$_{dyn}$ strategy. This gives an advantage to this strategy in high-frequency trading as it can do more simulations in a limited amount of time.

Finally, our results demonstrate that an MCTS strategy with a more significant number of MCTS simulations, seeded with a reasonably accurate small action space and dynamic action addition, can be considered as the best variation of MCTS. Following these three policies, the MCTS agent simulates the auctions with accurate prices by a higher number of simulations and makes good decisions to bid at the right moment to procure the full demand.

4.12 Chapter Summary

In this chapter, we have mainly proposed a novel approach for bidding in periodic double auctions using Monte Carlo Tree Search (MCTS). We have also proposed a way to seed the initial action

space on the MCTS strategy using some simple heuristics. In the next chapter, we will do an empirical evaluation of this bidding strategy.

We have evaluated the performance of our proposed bidding strategy against two widely known baselines and the current state-of-the-art PDA bidding strategy. Our MCTS bidding strategy shows significant improvement over the baselines and the state-of-the-art strategy. We conducted an empirical analysis to explore ways to improve the MCTS strategy. We also present a fast heuristic strategy that can be used either as a standalone method or as an initial set of bids to seed the MCTS policy. The specification of the action space has a significant effect on MCTS performance, and our experiments guide what features of the actions are most important. Our results show (unsurprisingly) that MCTS performs better with a more significant number of MCTS simulations, and dynamically adding actions during the search process with proper action seeding can significantly improve the agent performance. As part of the future work, there are scopes for finding a kernel regression-based dynamic action adding algorithm with plans to introduce non-determinism.

Chapter 5

Learning Strategies in Dynamic Retail Market

The Power TAC environment offers brokers the ability to issue several different types of tariffs three times per simulation day. Each tariff may be as elaborate or simple as the broker desires though each tariff can only target a single power type such as consumption or production. The most straightforward type of tariff one may issue is a flat rate tariff that offers a single price per kWh to subscribers. From there, the tariff may be augmented with signup bonuses, or a minimum subscription duration and early termination fee. Tariffs may also be customized to offer tiered usage pricing, time of use pricing, or a daily fee in addition to usage pricing. Tariffs may be issued, revoked, or modified at any time, though the new tariffs will only become available for subscription at the designated 6-hour intervals. A tariff is modified by publishing a new tariff with the superseding flag set to the old tariff and then revoking the old tariff. Our broker issued a single, simplified flat-rate tariff at the beginning of the game and modified it throughout the simulation by superseding the past tariff and revoking the past tariff.

To publish effective tariffs so that we gain both the most subscribers, henceforth referred to as market-share, and the highest net balance, we applied standard reinforcement learning techniques to try to learn an effective policy for modifying prices based on observations of the market. This technique is chosen because we want the agent to learn to react in such a way that it gains the best possible reward with little direct input from the agent designers. To achieve this goal, we modeled this problem as a Markov Decision Process (MDP) [37] and utilize the Q-Learning algorithm [1] to learn a good policy. Q-Learning involves an iterative process whereby the SPOT agent plays many simulations, continually updating the Q-Value for a given state, action pair. Q-Learning

will continue to improve the Q-Values until convergence is obtained where the Q-Values for each state, action pair change very little per iteration. In order to expedite the convergence of the Q-Learning algorithm, we implemented a distributed system that allowed many simulations to be run simultaneously.

5.1 Formulating the Problem as an MDP

Recall that in the tariff market, the goal is to design tariffs that will result in the broker agent's highest profit. We investigate a restricted version of the problem, assuming that the broker can only offer flat-rate tariffs, i.e., the price per kWh is uniform across all time steps. However, the broker can vary the price of the flat-rate tariff by having the objective to maximize the profit. This problem can be formulated as a *Markov Decision Process* (MDP) [37], defined by the tuple $\langle \mathbf{S}, s_0, \mathbf{A}, \mathbf{T}, \mathbf{R} \rangle$:

- A set of states $\mathbf{S}$. In our problem, we define the set of states to be all possible pairs of $\langle MS, Bal \rangle$, where MS is the percentage of market share controlled by our agent (i.e., the percentage of customers that are subscribing to our agent) and Bal is the overall profit or loss since the start of the simulation (i.e., the amount of money in the "bank"). We discretized MS from 0% to 100% in increments of 5% and Bal from $-$€2,000,000 to €8,000,000 in increments of €20,000.
- A start state $s_0 \in \mathbf{S}$. In our problem, the start state is always $\langle 0\%, e0 \rangle$ since the agent does not have any subscribers to its tariff and starts with no initial profit or loss.
- A set of actions $\mathbf{A}$. In our problem, the agent's first action is to publish a new flat-rate tariff at €15 per kWh. Subsequent actions are from the following set of actions:

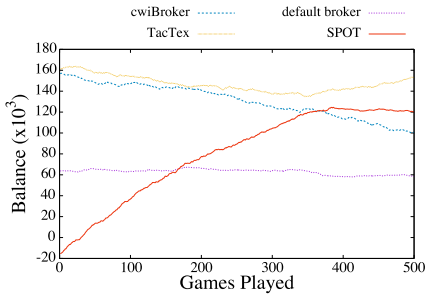
  $\uparrow$: Increase the price of the tariff by €2.00 per kWh. This is implemented by publishing a new tariff at a higher price and revoking the previous lower-priced tariff.

  $\leftrightarrow$: Keep the price of the tariff. This is implemented by not publishing or revoking any tariffs.

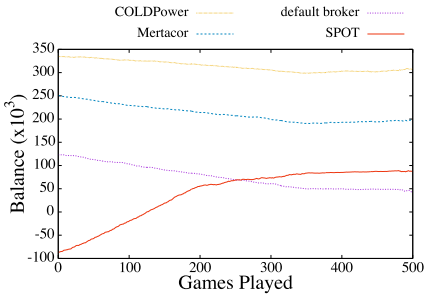
  $\downarrow$: Decrease the price of the tariff by €2.25 per kWh. This is implemented by publishing a new tariff at the lower price and revoking the previous higher-priced tariff.

- A transition function $\mathbf{T} : \mathbf{S} \times \mathbf{A} \times \mathbf{S} \rightarrow [0, 1]$ that gives the probability $T(s, a, s')$ of transitioning from state s to s' when action a is executed. In our problem, the transition function is not explicitly defined and transitions are executed by the Power TAC simulator.
- A reward function $\mathbf{R} : \mathbf{S} \times \mathbf{A} \times \mathbf{S} \rightarrow \mathbb{R}^+$ that gives the reward $R(s, a, s')$ of executing action a in state s and arriving in state s'. In our problem, the reward is the gain or loss in profits of the agent, determined by the Power TAC simulator.

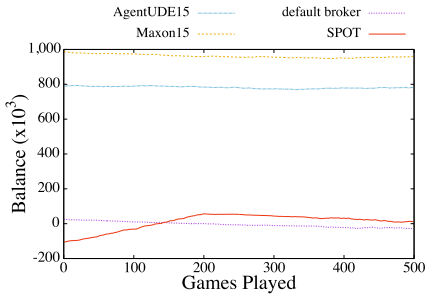
A solution to an MDP is a policy π, which maps states to actions. Solving an MDP is to find an optimal policy, that is, a policy with the most significant expected reward.



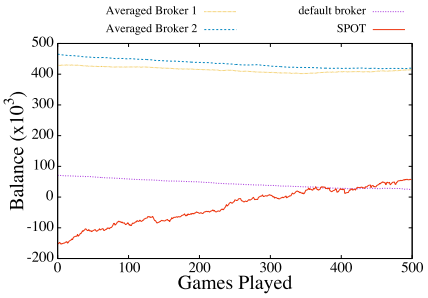
(a) Against Inferior Agents



(b) Against Similar Agents



(c) Against Superior Agents

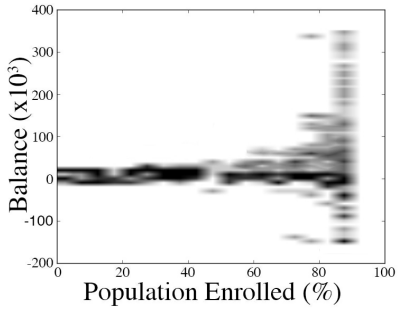


(d) Against Random Agents

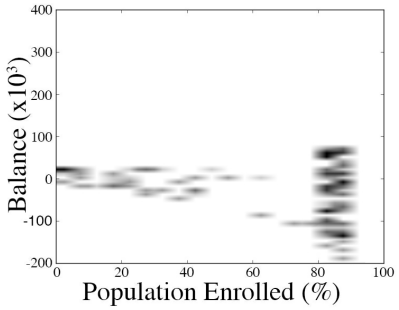
Figure 5.1: Convergence Rates

5.2 Learning a Tariff Pricing Policy

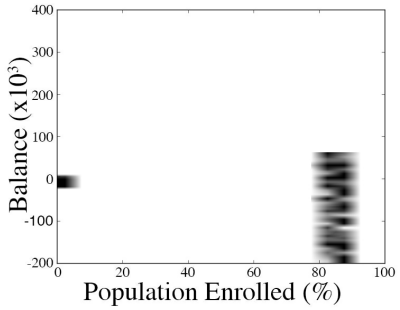
We now describe how to learn an effective policy for the MDP using Q-learning [1]. We initialize the Q-values of all state-action pairs $Q(s, a)$ to 1,000,000 in order to better encourage



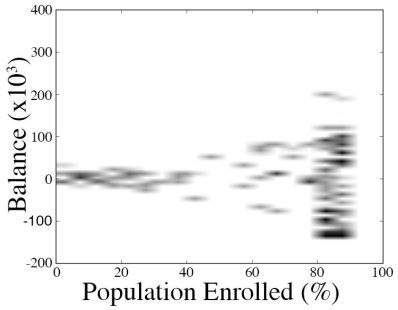
(a) Against Inferior Agents



(b) Against Similar Agents



(c) Against Superior Agents



(d) Against Random Agents

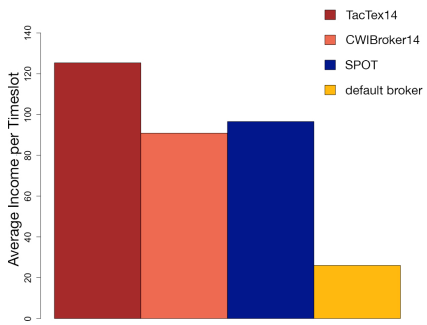
Figure 5.2: Explored States

exploration [1] and use the following update rule to update the Q-values after executing action a from state s and transitioning to state s':

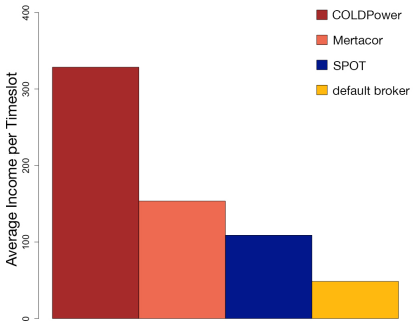
$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha \left\{ R(s,a,s') + \gamma \cdot \max_{a' \in \mathbf{A}} Q'(s',a') \right\} \tag{5.1}$$

where $\alpha = 0.9$ is the learning rate and $\gamma = 1.0$ is the discount factor.

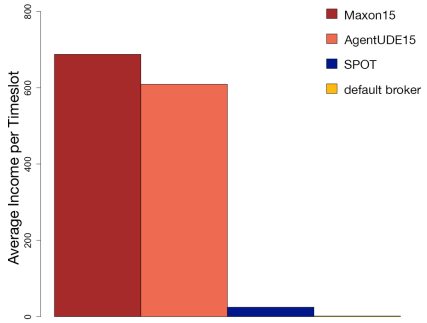
Parallelizing the learning process: In order to increase the robustness of the resulting learned policy, we executed the learning algorithm with ten different simulation bootstrap files [41]. The different bootstrap files may contain different combinations of types of users, with different energy consumption profiles, energy generation capabilities, etc. To speed up the learning process, we parallelize the Q-learning algorithm by running multiple instances of the simulation. We run the simulations in groups of 10 instances, where each instance in the group uses one of the ten unique bootstrap files. Instead of using and updating their local Q-values, all these instances will use and update the same set of Q-values stored on a central database. Once the simulation of



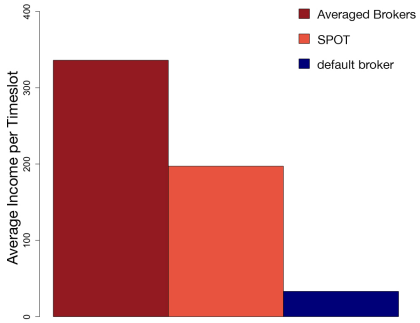
(a) Against Inferior Agents



(b) Against Similar Agents



(c) Against Superior Agents



(d) Against Random Agents

Figure 5.3: Comparison against Opposing Agents

one of the instances ends (the Power TAC simulation can end any time between 1440 to 1800 simulated hours), it will restart with the same bootstrap file from the first time step again.

5.3 Experimental Results

In our experiments, we learn policies against two opposing agents; this scenario corresponds to the 3-agent scenario in the previous Power TAC competition. We characterized possible opposing agents according to their relative competitiveness in the previous years' Power TAC competitions. We learned four different sets of Q-values and, equivalently, four different sets of policies against four different types (in terms of their competitive level) of opposing agents:

- SUPERIOR AGENTS: AgentUDE15 and Maxon15.

- SIMILAR AGENTS: Mertacor and COLDPower.
- INFERIOR AGENTS: TacTex14 and CWIBroker14.
- RANDOM AGENTS: Two randomly chosen agents from the set of 6 agents above.

5.3.1 Convergence Rates

Figure 5.1 shows the convergence rates for all of the scenarios, where the y-axis shows the final balance at the last time step for each iteration. SPOT can learn better policies and improve its final balance with more iterations. To reach convergence, SPOT takes various numbers of iterations, according to opponents. SPOT sees the most variance in games where the opponents are randomized. Against a setlist of opponents, policy convergence is reached in a limited number of iterations. For example, after approximately 200 iterations, convergence is reached against superior brokers.

5.3.2 Explored States

Figure 5.2 illustrates the explored states in the same scenarios. The color of each state represents the number of times the actions for each state were explored, ranging from black, where all three actions were explored the most, to white, where no actions were explored. The figure shows that more states and actions were explored against inferior agents than against superior agents. Additionally, these results also explain the performance of the agent; against superior agents, our agent was very limited in the states it was able to explore, most times being unable to gain more than 5% of the market share. When it did get a significant amount, it was often at a loss. Thus, it took its best actions and maintained a balance of approximately €50,000.

5.3.3 Comparison against Opposing Agents

We evaluated the learned policies against the same set of opposing agents. Figure 5.3 shows the profit in the tariff market alone of each agent over the various time steps. These results are averaged over five different bootstrap files (different from those used in the learning process) and

three runs per bootstrap file. These results are consistent with the final converged results shown in Figure 5.1, where our agent does better against inferior agents than against superior agents.

5.3.4 Profit in the Tariff Market per Timeslot

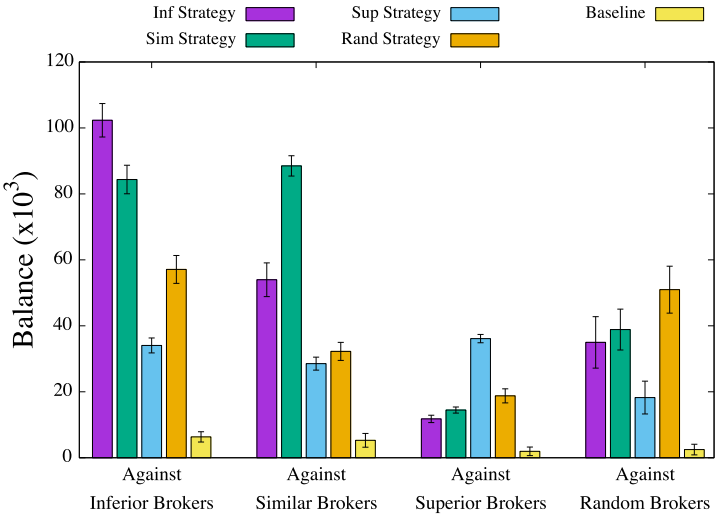


Figure 5.4: Profit in the Tariff Market per Timeslot

Figure 5.4 plots the performance of our agent with each of the four learned policies in addition to an agent with no learned strategy against each pair of opposing agents. The results show that the agent with the policy learned by playing against a specific pair of opponents do best when playing against the same pair of opponents (e.g., the agent with the policy learned through playing against superior agents does better than other policies when playing against superior agents). The policy learned by playing against random agents is more robust against different opponent types, especially compared with the policy learned through playing against superior agents.

5.4 Chapter Summary

The strategy learned by tabular Q-learning fails to achieve the highest profits against the other rival agents. A possible reason for this is that the agent cannot capture the past state accurately because we have a very highly discretized state space and a limited number of market features

(i.e., our states are too generalized). The action space is also minimal compared with the full action space. Both of these restrictions were necessary for defining the MDP because we have a limited amount of data to learn a policy due to the high computational and time cost for running simulations. We see that policy learned against specific opponents do best. It is also interesting to note that other than the superior agent pair with which it trained, the agent with the superior agent training does better only than the baseline agent against all other pairs. This could be because this is the agent with the least explored states and is thus unable to make wise decisions. In contrast, the policy learned by playing against random agents is more robust towards different opponent types, especially random and superior agents.

Chapter 6

Empirical Game Theoretic Methods to Minimize Regret Against Specific Opponents in Retail Markets

6.1 Introduction

We need to be able to learn better strategies than in the previous chapter by getting much more data more quickly and being able to train strategies with richer state and action spaces. We plan to use a simplified simulator to dramatically increase the amount of training data that can access to learn these policies and do strategic analysis. We also want to be able to account for the strategic nature of the game, so we will learn a portfolio of different strategies that will do well against different types of opponents; a one size fits all strategy is not enough for this domain. These motivate the use of empirical game theory to analyze the games, with an improved set of RL methods for learning candidate strategies.

Game theory plays a fundamental role in analyzing multi-agent interactions in complex systems [68]. Economists often use game theory to understand competitive market behavior, where it helps to predict when people engage in certain behaviors, such as price-fixing and price-war [18]. Walsh et al. [73, 74] and Wellman et al. [77] shows that by using a restricted set of heuristic strategies and empirical game theoretic analysis we can examine the agent interactions at a higher meta-level, instead of trying to analyze the interactions between all of the possible detailed strategies. In this analysis paradigm, we create a restricted game with fewer, more abstract strategies that form a game model that is more tractable for analysis. We then iteratively try to

find additional stronger strategies that can be added to this restricted game model to improve the overall analysis. Researchers have applied these methods to no limit Texas hold 'em poker and many types of double auctions [28, 66, 50, 51, 67] and found novel insights in complex systems.

In this chapter, our objective is to find better retail strategies in a complex multi-agent retail market. We decided to apply this promising set of techniques to design new effective strategies. One of the limitations of the empirical game-theoretic approach is that it becomes tough and time-consuming to find the convergence strategy with a vast strategy space it becomes computationally expensive to explore the full game to find true equilibrium strategies or perform other types of analysis. Moreover, it is not guaranteed to perform near optimally with a single strategy when facing different classes of opponents. Here we propose an algorithm that can outperform others in regret analysis and converge faster than conventional EGTA methods to find new strategies.

6.2 Background: Game Theory and Machine Learning

Game theory is the method of representing the strategic interplay between two or more players in a position containing set rules and results.

6.2.1 Normal-form game

A (finite, n-person) normal-form game is a tuple (N, A, u), where:

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

A natural way to represent games is through an n-dimensional matrix.

In a two-dimensional example in table 6.1, each row represents player 1, each column representing an action for player 2, and each cell represents one possible outcome. Each player's

Table 6.1: A normal form game

	C	D
C	-1,-1	-4,0
D	0,-4	-3,-3

utility for an outcome is written in the cell corresponding to that outcome, where player 1's utility listed first. [42]. There are normal form games that have some correlation to the retail trading problem [46].

Iterated Prisoners Dilemma

The issue of publishing tariffs resembles a classic Prisoner's Dilemma. In a standard version of the game, two prisoners are interrogated separately about a bank robbery. If one confesses to the theft and the other does not, the one who confesses is released, and the other receives a stiff sentence. If both confess, they receive moderate sentences. If neither confesses, they receive mild sentences for a lesser crime. Each player follows a dominant strategy, a strategy that is the player's best response, the one with the highest payoff, no matter what strategy the other player chooses. Assuming at an atmosphere of distrust and competition, the dominant strategy equilibrium of the game is for both prisoners to confess, even though the Pareto superior outcome, the one in which neither party can be made better off, is for both to keep quiet [46].

The prisoner's dilemma is a game that shows the reason for two entirely rational individuals not cooperating. Even if we can see that it is in their best interests to support each other, they will not collaborate because of trust issues. In the iterated prisoner's dilemma, two players play this game repeatedly, and they have perfect recall of their taken actions and change their strategy accordingly. Table 6.2 represents a single shot prisoners dilemma if the payoffs support the $T > R > P > S$ and $2R > T + P$ property.

Table 6.2: Iterated Prisoners Dilemma Payoff Matrix

	C	D
C	R, R	S, T
D	T, S	P, P

Repeated Coordination Games

A game is a coordination game, if the following inequalities hold in the payoff matrix for the row player: $A > B$, $D > C$, and for the column player: $a > c$, $d > b$. We can find multiple pure

Table 6.3: Repeated Coordination Games Payoff Matrix

	Left	Right
Up	A, a	B, b
Down	C, c	D, d

strategy nash equilibria in coordination games where players choose the same or corresponding strategies. When this game is played in succession, we call it a repeated coordination game. The retail pricing game is a complex game where agents play different games at different timeslots where opponent agents' actions decide which specific game the agent needs to play. This payoff matrix structure determines which games the agents are playing, and their corresponding taken action can make a transition from one game to another game. At a particular timeslot, other example games retailer can play or transition into are "The Tragedy of the Common", "The Boxed Pigs" etc. [46].

6.2.2 Meta Game

A meta game is a reduced model of a complex interaction. In order to analyze complex games like, e.g., poker, we do not need to consider all possible policies but a set of relevant meta-strategies that are frequently played [51, 68]. Meta strategy means a set of actions that are taken by the agent while playing the complex game. We can model the strategic interaction between two retail traders as a two-player meta game G = (S, U). In the rest of the dissertation, we refer to

the meta-strategy as a strategy, and we do not consider analyzing the atomic level action strategy per timeslot.

The traders simultaneously select one pure strategy, each from a finite strategy set S where U is the utility function. Initially, the retail strategy set S is a set of heuristics taken from prior works.

6.2.3 Empirical Game Theory

We can estimate the payoff of a strategy by simulating a complex game, such as stock market games, cybersecurity games, and trading agent competitions, instead of analyzing the complex game [80]. For example, agents in a stock market place order sequentially, and interact via complicated market rules, such that the only way to determine the outcome of multiple agents' interacting strategies is to simulate their outcome in the market. We often use an environment simulator to determine the payoffs for interacting strategies in such markets. The simulator can take a strategy profile and return a real-valued payoff for each agent i. In this work, I search for Nash equilibria of simulation-based games, which is a part of the field of empirical game-theoretic analysis, or EGTA [77, 78].

To give an introduction of EGTA, suppose there is a simulation-based game $G = (S, U)$ with N players, where the payoffs are available only by sampling a simulator S. As each agent's strategy set S_i is continuous, it may be beneficial to analyze a restricted game $G' = (S', U')$ with 10 strategies per agent. The first step in the EGTA process is to sample payoffs from simulator S for many strategy profiles s. We typically collect many samples for $p_s \sim S(s) \in R^N$ for each profile s. The greater the number of samples collected per profile, the more accurate the resulting model becomes. After collecting payoff samples $p_s \sim S(s)$ for various profiles s, we can estimate the expected payoff for each agent of profile s by taking the sample mean $\overline{\mathrm{p}}_s$. Then we can construct an empirical game $G'' = (S', U'')$ by combining the estimated expected payoffs for many profiles wherein $G'', U''(s) = \overline{\mathrm{p}}_s$ by definition. It is still often possible to find Nash equilibria of G'', even if the empirical game G'' has some missing payoffs. In such a case, the

equilibrium profile must be an equilibrium of the restricted game G' also, in the infinite-sample limit where sampling error vanishes [80]. We terminate the EGTA procedure when an equilibrium profile s is found and returned.

6.2.4 Deep Reinforcement Learning

The Power TAC simulator has a nearly real-world retail market with a high dimensional state and strategy space. In the previous chapter, we applied a standard tabular Q-learning method to learn new retail strategies. The state and strategy space is so vast that tabular Q-learning cannot do well against all strategies where, in recent times, deep learning can overcome this problem. Deep reinforcement learning uses a neural network as a function approximator instead of storing the q-values. It has been successfully solving extensive-form games like Go, chess, and shogi [59, 60] by learning superhuman strategies.

6.2.5 Double-oracle and empirical game-theoretic methods

A double-oracle method solves a two-player game iteratively with a finite set of strategies. By this method, we solve the game and use an oracle for each player to derive the best-response strategy against the current equilibrium opponent. Eventually, the method converges to equilibrium when there is no better response exists. We can use deep learning algorithms as oracles for a player to learn the best-response strategy.

6.3 Proposed Method

In the DO-EGTA method, we seek to get a newly learned strategy that can maximize the payoff against all of the strategies in the current set. History Aware Double Oracle EGTA (HADO-EGTA) proposes a modification of DO-EGTA where the algorithm keeps track of previous equilibrium strategies and learn a new DQN strategy over the mix-strategies of current and previous equilibriums [80]. All of these methods try to learn a single strategy that will maximize the payoff

over all the strategies in the strategy set. However, all these techniques do not help us to exploit some specific set of candidate strategies while we play the game.

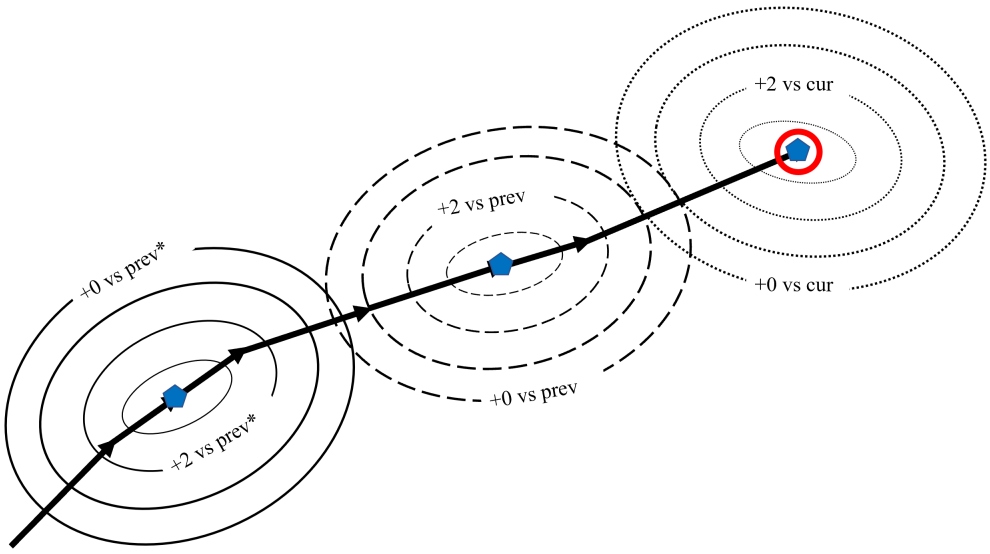


Figure 6.1: Double Oracle EGTA concept

In the standard double oracle EGTA process, we find an equilibrium point from a set of candidate strategies in the restricted game and try to learn the best response strategy to it. We continue this process until we can learn a final strategy, which gives us a positive deviation over all equilibria, or it fails to converge. In figure 6.1, we see a conceptual diagram of DO-EGTA with three types of contours. The solid contour is w.r.t. previous-previous opponent, the dashed contour is w.r.t. previous opponent and the dotted contour is w.r.t. current opponent, which shows how the current strategy is moving forward from learning against different equilibria to the ultimate strategy. Blue pentagons are the points where DO-ETGA converges to a specific set of strategies, and the red circle is the point where it converges to the last strategy.

The problem can arise when we have a large strategy space where we need to do many iterations to converge. In figure 6.2, we can see there is one single red circle, which is the ultimate strategy, and to find the ultimate single strategy which wins over all the equilibrium is time-consuming. Moreover, it is evident that the last learned strategy, in figure 6.2, will be able to do well against all the previous equilibriums. However, it is not guaranteed that it can effectively

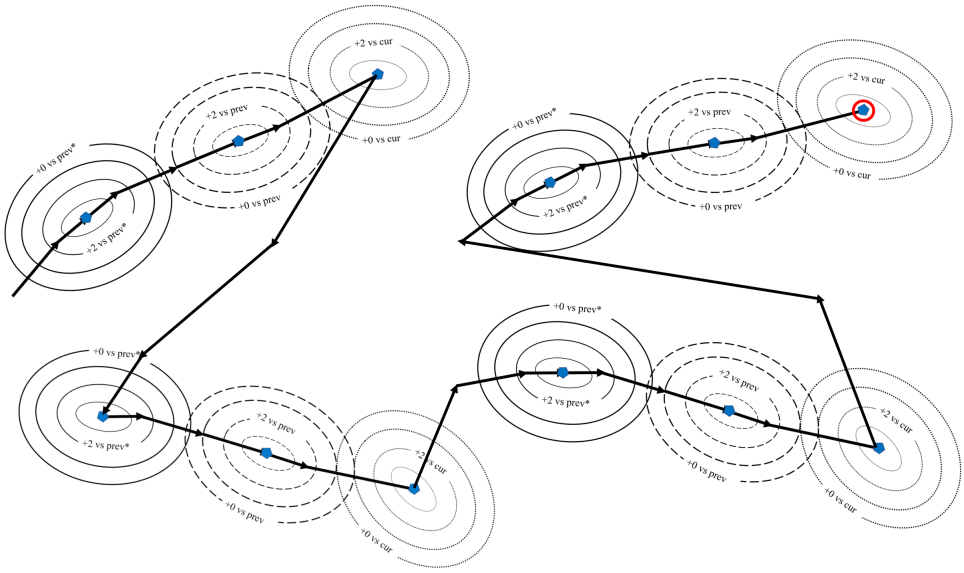


Figure 6.2: Double Oracle EGTA concept in high strategy space

exploit them, which is one of the main limitations of the DO-EGTA method for this domain.

We propose a new method called "Clustered Double Oracle EGTA" (CDO-EGTA), where we try to overcome this limitation. In a duopoly environment, when agents are playing against each other, they can have some action preferences. As the strategies are some probability distribution of actions, there might be some inner correlation between many strategies that may help us to cluster them. We propose an algorithm that finds the candidate strategies' inner correlations to cluster them in some groups and applies the DO-EGTA on each group to find a mapping of BR strategies. In the conceptual diagram of CDO-EGTA in figure 6.3, we can see that we have successfully clustered the strategies into four groups by learning the inner correlation. For each group, we have learned a separate ultimate strategy marked by the red circle. By doing this, we are reducing the strategy space, which results in faster convergence. Moreover, we can expect better exploitation than the DO-EGTA method as we learn different ultimate strategies based on different sets of similar types of strategies.

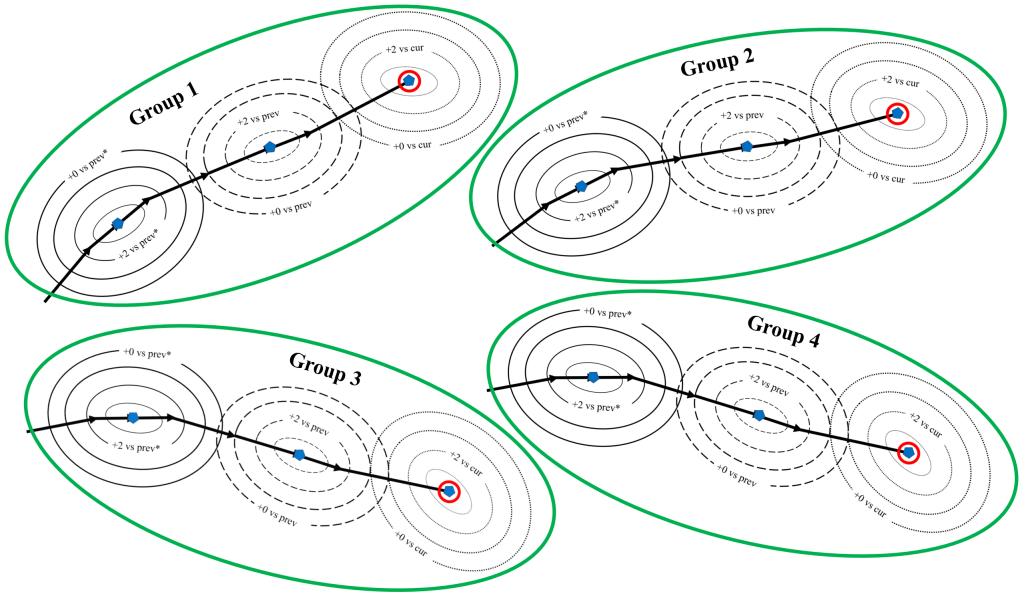


Figure 6.3: Clustered Double Oracle EGTA concept in high strategy space

6.4 Retail Game Model / Simulation Environment

We need a faster abstract simulator that can quickly simulate the Power TAC retail market to produce data as the Power TAC simulator is slow and complex. The newly isolated simulator can run faster and better estimate the core dynamics of retail strategies. Using the abstract retail simulator, we can apply EGTA and create the payoff values for the retail meta game. This section focuses on formulating the retail trading problem as a game using game-theoretic concepts and creating an isolated retail simulator.

6.4.1 Formal Definition of Retail Pricing Game

We define the retail energy trading market as a stochastic simultaneous sequential decision making game (3SDMG) as follows:

- A finite set of players $N = \{1, 2, ...n\}$.
- A state space M where a state m is defined by a set of features $\{f_1, f_2, ..., f_x\}$ using the domain knowledge. The features can be time, players' public and private information etc.

- Each player i has an action set (A^i), where an action (a) is a real number in $+PP_{max} \geq a \geq -PP_{max}$ continuous range. Here PP_{max} is the maximum 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, where a mix strategy is playing several actions with a probability distribution. At time t, each player $i \in N$ chooses a strategy (s_t^i) from their strategy set S^i.

- At time t, each player i plays only one action in the game from their selected strategy which we call move or outcome action ($o_t^i \in O_t$).

- ***Action profiles*** consist of all the outcome actions or moves chosen by individual players defined as $O = A^1 \times A^2 \times ... \times A^n$.

- The payoffs of players are defined by a ***payoff function*** which maps each state and move profile to a real number, $g : M \times O \rightarrow \mathbb{R}^{\mathrm{N}}$ where the i^{th} coordinate of g, g^i, is the payoff to player i as a function of the state m and move profile o ($o \in O$).

- Players have preferences over the payoffs, denoted as $\preceq_i$. For player k, any move profile o_i and o_j, $o_i \preceq_k o_j$ iff $g^k(m, o_i) \leq g^k(m, o_j)$.

- At time $t+1$, players fully observe the move profile o_t and partially observe m_t. A player's cost, profit, market shares etc are the private information which is not shared with others which makes it a imperfect information game.

- At time $t + 1$, players partially observe m_t state where the game starts at some initial state m_1 at time $t = 1$. Players then simultaneously choose a strategy $s_t^i \in S^i$, play moves $o_t^i \in o_t$, and **customers** (nature) selects m_{t+1} based on the current state m_t and move profile o_t according to the probability $P(.|m_t, o_t)$.

- Players have a perfect recall so each player keeps a history (H) of move profiles and visible features ($\hat{m}$) of a state m where H_{t+1} = $\{o_1, \hat{m}_1, o_2, \hat{m}_2..., o_t, \hat{m}_t\}$.

- A play of this game $m_1, o_1, ..., m_t, o_t$ defines a stream of payoffs $g_1, g_2, ..., g_t$ where $g_t = g(m_t, o_t)$ and each player i has a stream of strategies $\{s_1^i, s_2^2, ..., s_t^i\}$ which we call a meta strategy (SM).
- A t time-step game the payoff for player i is $\tilde{g}_t^i = \frac{1}{t}\sum_{n=1}^{t} g_t^i$

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

6.4.2 Customer Model

Consumers are the main components in the retail market alongside other retailers, where they define the transition function and determines the revenue of a retailer. A consumer can have two basic properties that affect their behaviors. The properties are inertia and rationality. These two factors affect how the customer is subscribing to a broker's product.

Customer Inertia factor

The inertia factor defines how frequently the customers should evaluate market tariffs. Customers do not always evaluate tariffs when given the opportunity. They can ignore tariff publications considering them junk mails. When a broker changes the tariff price, a subscribed customer resets its inertia variable. Resetting inertia gives the flavor of customer loyalty until the broker changes the tariff prices in the market, preventing the customer from changing subscription. So the inertia (I) is a probability $I \in [0, 1]$, which determines if a customer will evaluate the tariffs in the market at a particular timeslot. 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.

Customer Rationality

If customers are entirely rational, the agent with the best prices always gets all the market share over time. Otherwise, the best price agent 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:

$$P_i = \frac{e^{\lambda u_i}}{\sum_{t \in T} e^{\lambda u_t}}$$

The parameter λ is a measure of how rationally a customer chooses tariffs: λ = 0 represents random, irrational choice, while $\lambda = \infty$ means perfectly rational. In our experiments we control this using variable R where random is R = 0, perfect rational is R = 1.0

6.4.3 Heuristic Candidate Strategies

A straightforward multi-shot retail market can be compared to an iterative prisoner's dilemma game if the customers have rationality 1 and inertia 0 where the payoffs support the $T > R > P > S$ and $2R > T + P$ property. We adapted the IPD policies from the iterative prisoner's dilemma literature to the retail pricing game. We also extracted policies from other related domains like auction theory and used them as heuristics.

Zero Intelligence (ZI)

The ZI 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)

ZIP strategy tries to remember their failed bid prices, and in the next round, they bid with a more competitive price so that they make more profits.

GD

GD, proposed by Gjerstad and Dickhaut [8], bids based on historical data from the market and maintains a belief function representing the probability of its profit up to current timeframe. For

a buyer, the belief function for all the actions is Pr where

$$Pr_i = \frac{R_{GD}}{\sum_{j \in N} R_j}$$

At a specific timeslot t, Pr_i is defined as the success probability of action i where R_{GD} is the revenue earned by GD agent. This strategy always updates its belief probability by the new values by replacing the old values and selects the best action by a greedy lookup on all probability values of all actions while bidding.

Tit for Tat Class

TFT [53] 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) Same as Tit for Tat except that it defects twice when the opponent defects.

Hard Majority

HardMj [44] non-cooperate on the first move, and non-cooperate if the number of non-cooperations of the opponent is greater than or equal to the number of times it has cooperated, otherwise cooperate.

Soft Majority

SoftMj [44] cooperates on the first move and cooperates as long as the number of times the rival has cooperated higher than or equal to the number of times it has defected, else its defects.

Grim

Grim [20] cooperates until the opponent defects, and after that, always defects.

General Heuristic Strategies Based on Actions

Always Increase (AlI) Cooperates on every move. Always Decrease (AlD) Defects on every move Always Same (AlS) No change on every move.

Prober

Starts with D, C, C, and then defects if the opponent has cooperated in the second and third moves; otherwise, it plays TFT.

Naive Prober

Like Tit for Tat, but occasionally defects with a small probability.

Pavlov

Pavlov [38] cooperates on the first move. If a reward or temptation payoff is received in the last round, then it repeats the last choice, otherwise chooses the opposite choice.

By formulating the retail trading problem as a game and creating a simulator to play this game with all these heuristic strategies, we now have the proper framework for running experiments to find new retail strategies using existing methods. The next section explains the experimental setup of the isolated retail simulator and the configurations of Deep RL.

6.5 Experimental Setup

6.5.1 Isolated Retail Simulator Setup

Candidate Broker Action Space

All the brokers can play three actions to change the prices at a certain time step in the isolated retail simulator.

1. Decrease (-0.01 $ per KWH) (Non-Cooperative action)

2. Increase (+0.01 $ per KWH) (Cooperative action)

3. No change (Cooperative action)

We also have introduced the random walk cost [34] for the brokers to simulate the purchase cost of the energy from the wholesale market.

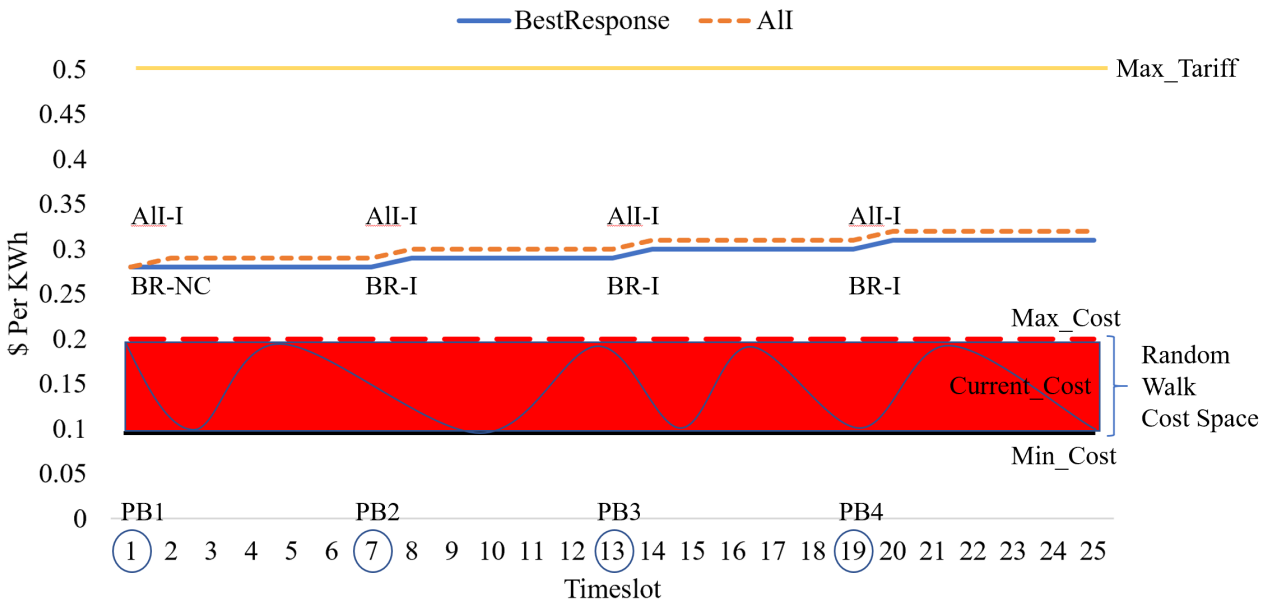


Figure 6.4: Isolated Retail Market Example

C_t is the target cost for timeslot t issued in the market segment. We vary C_t using a trend τ that is updated by a random walk:

$$C_{t+1} = min(C_{max}; max(C_{min}; \tau C_t))$$

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

C_0, the start value of C, is chosen uniformly in the interval $[C_{min}, C_{max}]$. We set C_{max} to 0.20 $ per KWH and C_{min} to 0.10 $ per KWH. We set τ_0 to 1.0 which is the start value of the τ. The trend τ is reset 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 [12].

To regulate the upper limit of the retail market, we set the maximum tariff price of the market to 0.5 $ per KWH. We set the default start price of the tariff to 0.28 $ per KWH. Figure 6.4 shows an instance of the isolated retail market where one broker plays the Always Increase strategy (AlI), and another broker plays the best response to it. The red zone represents the random walk wholesale cost area, which is limited by a maximum and minimum cost value.

Customer Properties

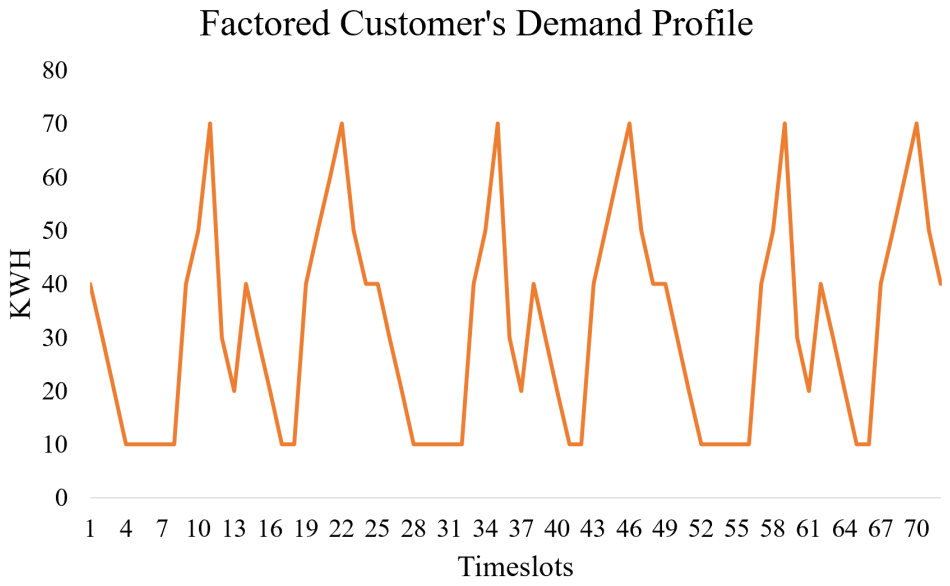


Figure 6.5: Customer Demand Profile

We set the rationality of the customers to 1.0, and the inertia to 0. As a result, all the customers in the market are always active and make 100% rational decisions. We set the customer population to 100. We have decided to simulate a small portion of the factored customer model from the Power TAC retail market. In the factored customer model, all the population has the same demand profile. We use a synthetic demand profile per day, which includes two peak demands. These peaks are usually in the morning and evening times. The below graph shows a three-day demand profile of the customer population. We use these 72 timeslots to simulate a retail market trading environment. We have also introduced a random walk [12] error to the actual customer

demand at a specific timeslot and use that as a demand prediction for the brokers so that they do not possess the actual customer demands.

6.5.2 Baseline Methodologies Implementation

We use three baseline methods in our experiments.

Minimax Regret

In the minimax regret method, we do not learn a new strategy. Besides, we select the candidate strategy, which has the lowest maximum regret if it is played against all the strategies in the candidate set. By minimizing the worst-case regret, the minimax regret method selects the candidate strategy, which performs as closely as to the optimal strategy in the restricted candidate strategy set [56].

Selfplay

In the self-play method, the agent learns a strategy by playing against the previous best version of itself in the game without any domain knowledge except the game rules [59, 58]. The method starts from random play, which is the first iteration, and the number of iteration goes on until it converges. We use the same learning method as in our best response algorithm here. One iteration means one training and testing round, where the Y-axis shows the average performance of the test round. Figure 6.6 shows that our agent reaches the convergence area with few first iterations of the self-play and then sees some variance for the rest of the iteration, which proves that the agent cannot improve more. We stop the learning at 625 iterations and use that version in our experiments. Additional details of the network architecture, training, and testing hyperparameters used are presented in Table A.2 in Appendix A.

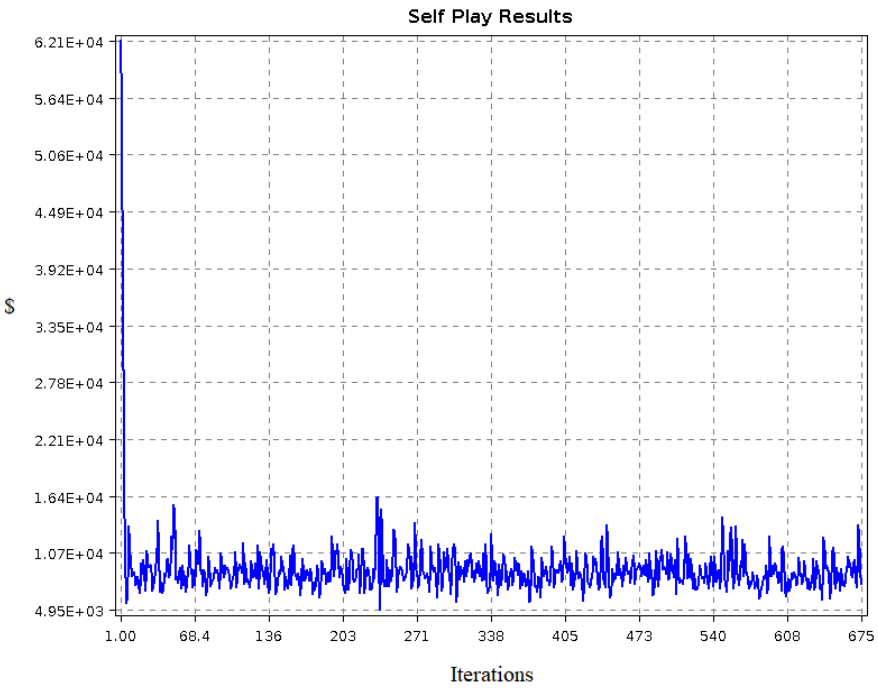


Figure 6.6: Self Play Learning

Double Oracle EGTA (DO-EGTA)

In DO-EGTA method, at each iteration, we solve the game with the current strategy set, compute the nash equilibriums for the current set, pick one equilibrium strategy, and learn the best response strategy using DQN. If the learned strategy has a positive deviation in payoffs comparing to the equilibrium strategy, we add it to the current strategy set, which creates the next strategy set. We iterate this until we reach a convergence point, which is when there is only one equilibrium, and the new equilibrium is a DQN strategy, or DQN fails to get a positive deviation to all the equilibrium strategies of the current set. The below flow chart nicely describes the DO-EGTA technique. We have a set of initial heuristic strategies, $S^H = (S_1, S_2, ...S_n)$ for both players. In each round t = (0,1,.....) of the iterative procedure, the current strategy set is S_t. A mixed-Nash equilibrium over these strategy sets under the game utility fuction $U()$ is σ_t. The game utility function $U()$ is based on running simulations of the 3SDMG game in the isolated retail simulator which helps us to estimate the utility values of the strategies played by the players. $g()$

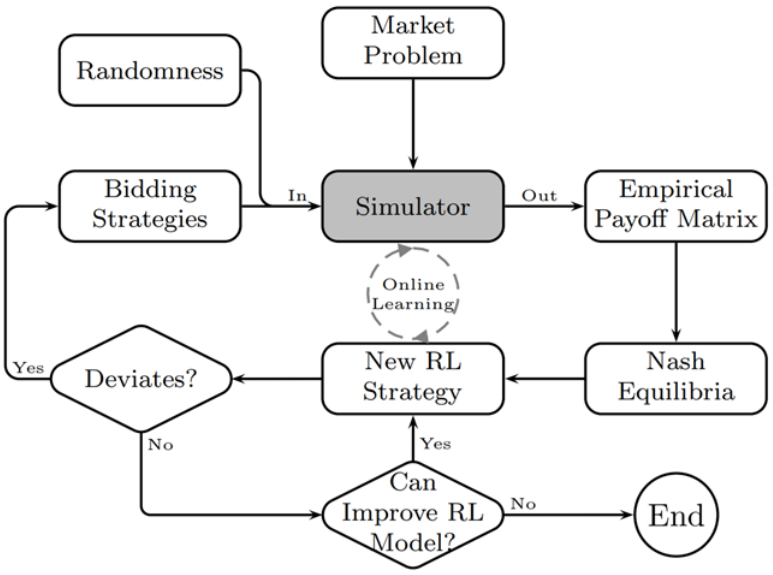


Figure 6.7: DO-EGTA flowchart

is a deep RL algorithm learns a better response against a opponent 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. Here is the DO-EGTA algorithm for each iteration:

DO-EGTA Algorithm

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

Table A.3 shows how we learn the ultimate DQN strategy from the 16 candidate strategies using DO-EGTA.

6.5.3 CDO-EGTA Implementation

We propose the "Clustered Double Oracle EGTA" (CDO-EGTA) algorithm, where we cluster the strategy set into groups using a clustering algorithm and learn a BR strategy for each of the groups using the DO-EGTA method. We find a set of BR strategies instead of one strategy so that we have a larger pool of strategies available that can be used to exploit specific types of opponent strategies when we play the game.

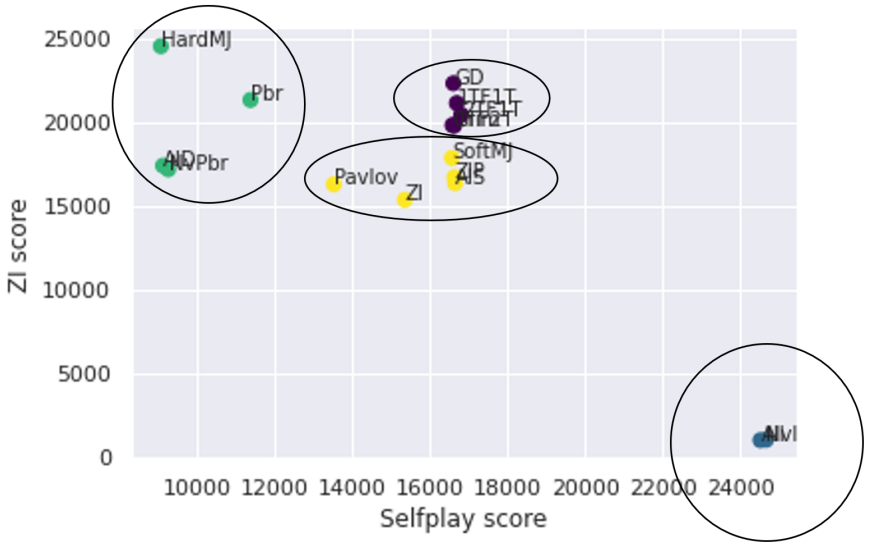


Figure 6.8: Clustering phase in CDO-EGTA

In the clustered double oracle empirical game-theoretic analysis (CDO-EGTA) method, we apply these subsequent steps:

1. Generate a set of features ($Features^H$) for each candidate strategy ($C \in S^H$) by playing them against a group of feature extraction strategies ($F \in S^F$).
2. Using a clustering algorithm, $c()$, cluster all the candidate strategies S^H into K groups.
3. Apply DO-EGTA algorithm, $d()$, on K groups
4. Learn N best response strategies and create a mapping of best responses (M_{br}) for K groups.

CDO-EGTA Algorithm

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

CDO-EGTA has two main components. The first component is the clustering part, where it uses some feature extracting strategies to generate payoffs for the candidate strategies. We call these payoffs as extracted feature data, which we use to group them using some clustering algorithm.

The second component we need to implement when applying the DO-EGTA procedure is a specific method for finding BR strategies. In this work, we use Deep Q-Networks to learn

BR strategies, so we begin by explaining the setup of the states, neural network, and finally, the algorithm itself.

State Features

In the previous chapter, we had only two features in our state the cash balance and market share. With a higher number of features, we can make more accurate predictions by better distinguishing the states. Nevertheless, tabular Q-learning becomes infeasible when the number of features grows. In Deep RL, we do not face this issue because we approximate the Q-values using a function approximator, i.e., a neural network. This allows us to add a more significant number of more useful state features so that we can recognize the states with better accuracy. We have used 37 historical data points as our state features. We have two time-related data (i.e., day, 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 hot-encoding for the action, so in total, we have 37 features for our states. The details of the state features are presented in table A.1 Appendix A.

DQN Setup

We used the deep Q-network (DQN) deep reinforcement learning model familiarized by Mnih et al. [45], and we specifically implemented the Double DQN variant form introduced by van Hasselt et al. [72] We use the Deeplearning4j framework libraries to implement the DQN in our experimentations. [63]

We use a three-layer (25 X 13 X 7) neural network (NN) for our DQN, where each of the hidden layers is fully connected with the next hidden layer. We treat each state's 49 features 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 3 nodes as we have three actions.

Additional details of the network architecture and training hyperparameters used are presented in Table A.2 in Appendix A.

In our experiments, we use two feature extraction strategies (ZI, and the strategy itself) in S^F. By playing each strategy from S^F 1000 times, we get two feature values for each strategy.

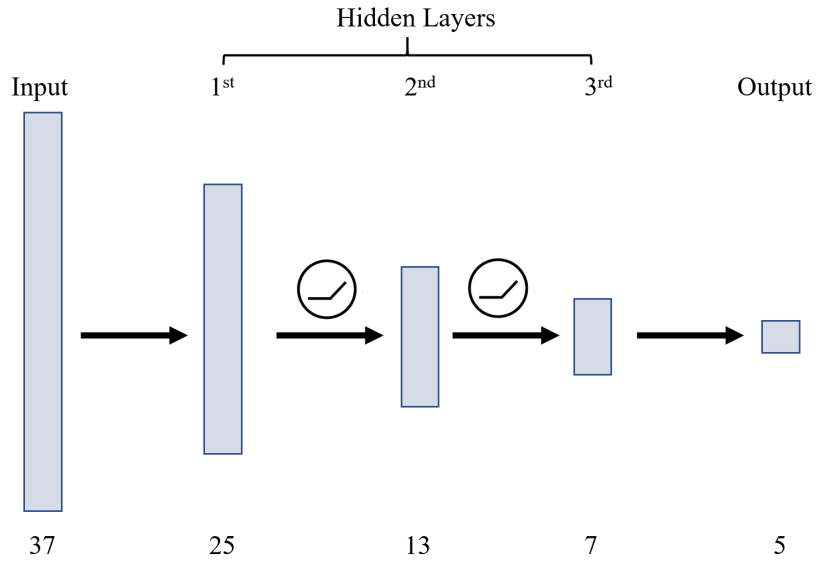


Figure 6.9: Neural Net Architecture for DQN

We have 16 candidate heuristic strategies in S^H. After extracting the feature values for each of the 16 candidate strategies, we apply a clustering algorithm on the extracted feature values to find groups and learn the corresponding DQN best responses for each group using DO-EGTA. Figure 6.8 shows four groups when we apply the k-means clustering algorithm with $k = 4$. Later, DO-EGTA learns an ultimate strategy for each of the groups and creates a map of BR strategies.

6.6 Results

Due to the possibility of non-transitive relationships among several agents [2], [62], it can be challenging to evaluate the quality of learned strategies in games. It might not be possible to fully capture the advantages and disadvantages of a strategy using a single scalar score point. Our goal is to examine how effectively we can exploit the opponent's strategies. Regret analysis helps us to measure how close we were making the optimal choice or exploiting other strategies while deciding under uncertainty. As regret analysis measures the opportunity loss, we use it as a metric to evaluate the strategies found by the CDO-EGTA and baseline methods.

6.6.1 Regret Comparison of the Methodologies

In CDO-EGTA, we plan to cluster the candidate strategies based on the extracted feature data. Initially, we use the mean shift [7] clustering algorithm as we do not know the right number of clusters. The mean shift algorithm finds the number of clusters from the extracted feature data and does the grouping accordingly. In our experiment, this method clusters the candidate strategies into four groups that are listed below.

Table 6.4: Groups created by mean shift clustering algorithm with 2 features

Strategy Set	Strategies
1	ZI, ZIP, GD, 1TF2T, 1TF1T, 2TF1T, SoftMJ, Grim, AlS
2	Pavlov, NvPbr, AlD
3	AlI, NvI
4	HardMJ, Pbr

We learn four final BR strategies by applying DO-EGTA on these four groups and create a map of BR strategies for these groups, which is the output of our CDO-EGTA method.

In our experiments, we create an empirical game where player 1 (row player) has all the strategies as his actions, including four new strategies found by the four methods plus all the 16 candidate heuristic strategies denoted by C1 C16. Player 2 (column player) has only the restricted set of candidate strategies, i.e., 16 in this case as his actions. Thus we create a 20x16 payoff matrix, 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. Figure 6.10 shows the regret table for player 1 where we accumulate all the regrets row-wise for each of the actions of player 1. Figure 6.11 shows the accumulated regret values for all the four methods where our CDO-EGTA method achieves the lowest regret than others.

The result shows that the clustering algorithm is successful in grouping the candidate strategies according to the extracted features, and the DO-EGTA method becomes more effective in learning new strategies when we use it in CDO-EGTA. While playing the retail game, the CDO-

P2

P1	C1	C2	. . .	. . .	C16
C1					
C2					
. . .					
C16					
DO-EGTA					
Self Play					
Minimax					
CDO-EGTA					

Figure 6.10: Regret table for player 1

EGTA method finds the corresponding best response strategy against the opponent agent from the BR mapping and plays accordingly to exploit the opponent. The other methods use only one ultimate DQN strategy, which is not able to exploit all the groups and end up having higher regrets.

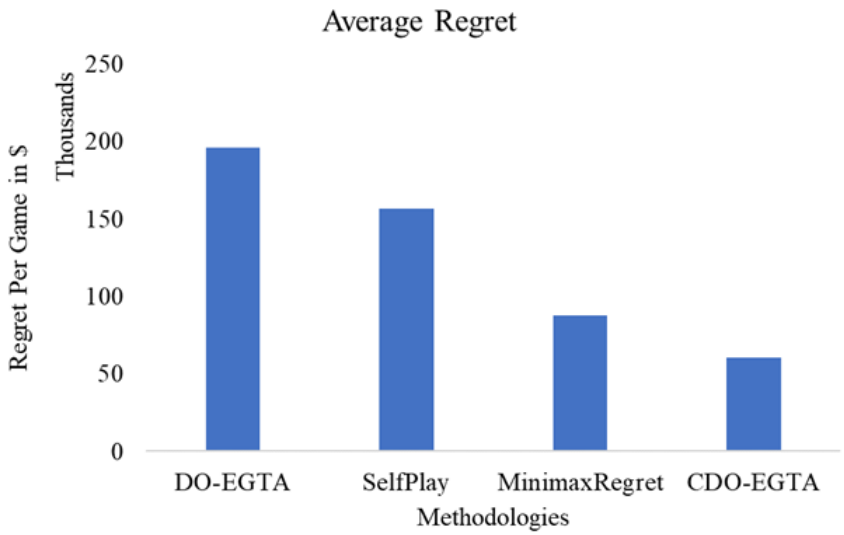


Figure 6.11: Average Regret Comparison

6.6.2 Experiment with different cluster sizes

We also want to see how the performance of the CDO-EGTA changes by the cluster size. For this, we use the k-means clustering algorithm, where we put different values of K and analyze the performance.

Regret Comparision by Cluster Size

We learn different BR maps for different number of clusters and do an empirical analysis of 1000 test games. Figure 6.12 shows that the regret value decreases with the increasing number of

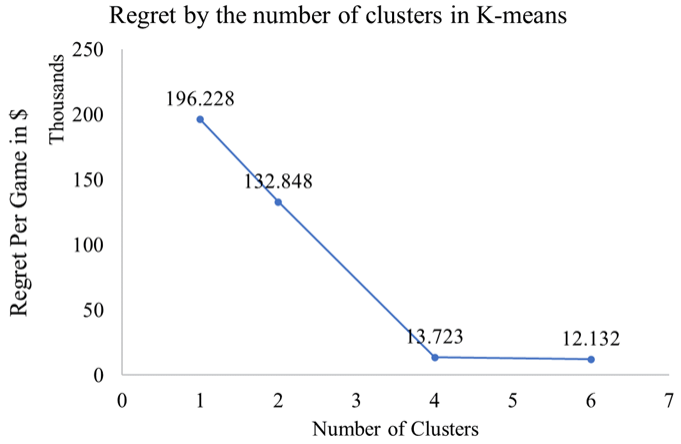


Figure 6.12: Regret Comparision By ClusterSize

clusters. It makes sense because the best way to play this game is to learn best responses to every individual retail strategy and play the best response against those strategies in the game. So when we are increasing the number of clusters, we are diving into the optimal route. However, learning BR policies for all the retail strategies is infeasible for large strategy space. So we need to find a reasonable point where we can do the trade-off of not learning against all single strategies and not treat them all in a single group.

Elbow Method: Finding optimal cluster size

For the k-mean clustering algorithm, we can use the elbow method to find the proper number of clusters to the group the candidate strategies. WCSS is the summation of each cluster's distance between that specific clusters each point against the cluster centroid and figure 6.13 shows that the within-cluster sum of errors (WCSS) decreases as we increase the number of clusters.

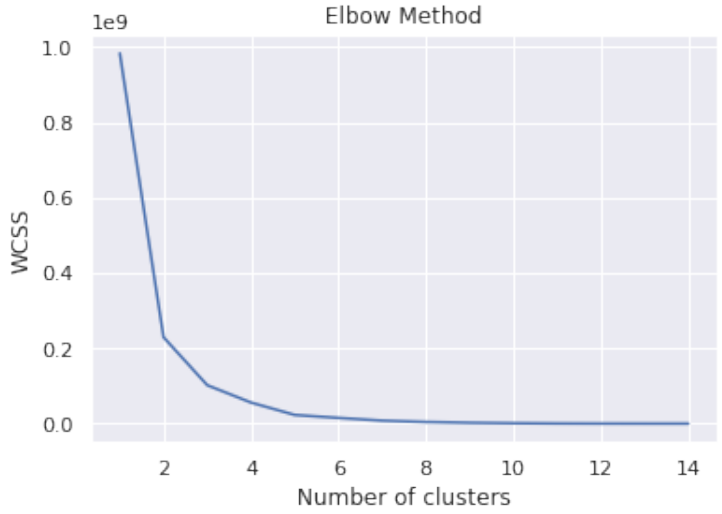


Figure 6.13: Finding the proper number of clusters in K-means algorithm

To find the optimal number of clusters, we try to find a point in figure 6.13 where the cluster size and WCSS value are minima.

Figure 6.12 shows the importance of clustering on the candidate strategies while learning new strategies in this domain. We understand that the optimal way to exploit the opponents in this game is to learn a separate BR strategy for each of the strategies. However, it is infeasible due to high dimensional strategy space. By increasing the number of clusters, we are following that optimal path because the highest number of clusters we can achieve when we have one strategy per cluster. Thus CDO-EGTA method guides us on how to learn better strategies than existing methods.

6.7 Disscussion

The main question we ask in this chapter is how we can design an algorithm to find better regret minimizing strategies by exploiting the opponent strategies. Our proposed CDO-EGTA method achieves this objective by using the standard DO-EGTA method to get a near-optimal performance in regret analysis. One of our limitations is that we have a restricted set of actions for the brokers, along with a restricted set of initial heuristic strategies. Also, our trading environment is a duopoly market where real-world markets are an oligopoly. We also simplified our customer model, where real-world customer models can be very complex and rich. With all these new challenges in this domain, one can do many interesting experiments on this work, which can lead to many exciting future research directions.

Chapter 7

Conclusion and Future Work

Motivated by the contributions of autonomous trading agents in the future energy grid, this dissertation offers autonomous trading algorithms for agents in competitive dynamic energy markets. This chapter reviews the dissertation's main contributions and discusses the promising future work in the challenging domain of autonomous trading.

7.1 Main Contributions

In this work, I present a general bidding algorithm for periodic double auction markets that offers state-of-the-art performance in minimizing cost in different market scenarios. I also offer a novel method for finding effective retail strategies in a high dimensional strategy space for autonomous brokers. We show that our proposed method can help a broker to minimize its regret by exploiting opponent strategies while playing against a set of candidate strategies.

In chapter 3, we show methods of predicting prices in dynamic PDA markets and show that our agent can effectively bid using supervised price predictors. We have learned several price predictors to forecast prices in PDAs but found that REPTree (a type of decision tree) algorithm performed best in empirical evaluations for different agent scenarios. We also learn that by estimating the market-clearing prices for null auctions increases the performance of the predictors significantly. [9, 8]

Chapter 3 sets the background for chapter 4, where we propose a novel PDA bidding strategy, which is the current state-of-the-art strategy in the PDA domain. Here we contribute to the development of an isolated test-bed where we decouple other markets from the wholesale market to design and experiment with PDA bidding strategies [10]. We have designed an effective

wholesale bidding strategy using the Monte Carlo Tree Search algorithm to minimize cost in PDA markets [10]. We have also worked on the enhancement of the MCTS bidding strategy using heuristics and empirical analysis [11]. Our MCTS wholesale bidding strategy has a state-of-the-art performance which can minimize costs better than any other existing strategies in the literature. [9, 8, 10, 11].

Chapter 5 focuses on learning the retail strategies in the vibrant, dynamic retail market of PowerTAC, where we see that the policy learned against specific opponents do best, and the policy learned against random agents is robust. However, we are unable to gain the highest profit against rival agents because our states have a low number of features that are not sufficient enough to capture the whole experience in learning using tabular q-learning efficiently. Another issue is our tariff structure, which is too simple (flat-rate tariff) compared to other agents who are publishing complex tariffs (tiered or TOU tariffs) and exploiting the market.

The insights from chapter 5 motivate us to formalize the retail trading problem as a game (3SDMG) using game-theoretic concepts where we develop a simple isolated retail simulator to play this game. We have simplified the retail market by reducing its complexity from the Power TAC retail market as the Power TAC simulator is very slow to produce training data. We have extracted the core retail market logic from the Power TAC to create a simplified model, where we decoupled PowerTAC retail market from the other markets (wholesale and balancing markets) so that we can better understand the retail trading dynamics. One of our objectives is to extract the strategic behaviors from the retail strategies by using the abstracted simulator to efficiently design new promising retail strategies faster with the help of empirical game-theoretic analysis. We apply deep reinforcement learning (DQN) to learn the best response strategy for a specific strategy played in this simulator. Here we can recognize the essential features and use game-theoretic solution concepts to design a profit-maximizing retail strategy. Using DQN as a best response learning technique, we propose "Clustered Double Oracle Empirical Game-Theoretic Analysis" (CDO-EGTA), a novel method for minimizing regret (i.e., maximizing revenues) in retail trading. CDO-EGTA method clusters the existing pool of strategies into groups, learns a new BR strategy for each of the groups using the DO-EGTA method, and outputs a class of BR

strategies to play the retail game. Empirical results show that our method outperforms the existing methods in regret analysis and successfully exploits the opponent strategies while playing them in the retail game.

7.2 Future Work

Improving the accuracy of the price predictor will increase the performance of our bidding strategy, so working on the improvement of the price predictor is an excellent future research direction. We use supervised price predictors to predict prices in our bidding strategy and make a minimal effort to explore this area. We can ask questions related to the robustness of the supervised PDA price predictors and work on ways to minimize the error effectively online. We can ask if it is possible to use the supervised price predictor as a baseline and learn the errors of it online as offsets, and then add the offset to baseline predicted prices to get more accurate prices.

Another exciting direction can be working on improving the MCTS bidding strategy. In our MCTS bidding strategy, we use a single price predictor where PDAs can have a 24 hour ahead window. We can try using 24 different price predictors for each of the hour-ahead markets and analyze the bidding strategy's performance. One can also try using memory-based MCTS [84] instead of the standard MCTS in our bidding strategy and analyze the results for better performance.

In the retail market, our work is restricted to a set of candidate retail strategies and in a duopoly trading environment. We also assumed that we could identify the opponent agent while trading in the retail market. However, real-world scenarios can be an oligopoly market where the candidate strategy space can contain unknown retail strategies, and brokers keep their identity private. These facts can lead to many promising future research directions; for example, we can ask questions about how our proposed method will perform against unknown retail strategies. One idea is to find the proper group for the unknown strategy by using CDO-EGTA's clustering algorithm and then use the corresponding BR strategy to play the game against that unknown strategy. For better performance, by including the unknown strategy in the candidate strategy

set, we can update the candidate groups by doing a new clustering, and learn corresponding BR policies for all the new groups accordingly. We have used DQN as our BR learning technique. We can use other deep reinforcement learning techniques such as actor-critic, A3C, and analyze the performance for positive deviations in payoffs better than DQN. One can apply HADO-EGTA [80] instead of DO-EGTA in our CDO-EGTA method for better performance.

We learned abstract retail strategies in the retail simulator, which is needed to map back to a more sophisticated retail strategy if we want to test it in the Power TAC simulator. The future work on this part can be how to map back the isolated retail strategy to a real, complex retail strategy. A more comprehensive study is needed to harness the strength of these learning approaches better.

The evaluations in the wholesale and tariff markets are conducted independently of each other. Thus, another future direction includes developing the complete trading strategy by coupling the isolated wholesale and retail strategies and evaluating the complete broker in the Power TAC simulation against other successful brokers. One can also participate in Power TAC's annual tournament and empirically evaluate their brokers against other successful Power TAC brokers from different research groups around the world each year.

7.3 Concluding Remarks

It will be a significant advance if intelligent bidding agents can replace humans in the future dynamic energy markets. Our work shows that we can successfully use machine learning and artificial intelligence to predict market prices and make intelligent trading decisions in dynamic wholesale auctions and retail trading markets. I believe that AI will play a significant role in building the coordination mechanism of smart grid markets, and my dissertation will contribute to future experiments on this domain and help others to understand better the difficult challenges faced by future energy delivery systems.

References

[1] Jinane Abounadi, D Bertsekas, and Vivek S Borkar. Learning algorithms for markov decision processes with average cost. *SIAM Journal on Control and Optimization*, 40(3):681–698, 2001.

[2] David Balduzzi, Karl Tuyls, Julien Perolat, and Thore Graepel. Re-evaluating evaluation. In *Advances in Neural Information Processing Systems*, pages 3268–3279, 2018.

[3] Severin Borenstein. The trouble with electricity markets: understanding california's restructuring disaster. *Journal of economic perspectives*, 16(1):191–211, 2002.

[4] Cameron B. Browne, Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A Survey of Monte Carlo Tree Search Methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, 2012.

[5] Guillaume Chaslot, Sander Bakkes, Istvan Szita, and Pieter Spronck. Monte-Carlo Tree Search: A New Framework for Game AI. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, 2008.

[6] Guillaume M. J. B. Chaslot, Mark H. M. Winands, Jos W. H. M. Uiterwijk, H Jaap van den Herik, and Bruno Bouzy. Progressive Strategies for Monte-Carlo Tree Search. *New Mathematics and Natural Computation*, 4(03):343–357, 2008.

[7] Yizong Cheng. Mean shift, mode seeking, and clustering. *IEEE transactions on pattern analysis and machine intelligence*, 17(8):790–799, 1995.

[8] Moinul Morshed Porag Chowdhury. Predicting Prices in the Power TAC Wholesale Energy Market. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 4204–4205, 2016.

[9] Moinul Morshed Porag Chowdhury, Russell Y. Folk, Ferdinando Fioretto, Christopher Kiekintveld, and William Yeoh. Investigation of Learning Strategies for the SPOT Broker in Power TAC. In *Proceedings of the International Workshop on Agent-Mediated Electronic Commerce and Trading Agents Design and Analysis (AMEC/TADA)*, pages 96–111, 2015.

[10] Moinul Morshed Porag Chowdhury, Christopher Kiekintveld, Tran Cao Son, and William Yeoh. Bidding strategy for periodic double auctions using monte carlo tree search. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1897–1899. International Foundation for Autonomous Agents and Multiagent Systems, 2018.

[11] Moinul Morshed Porag Chowdhury, Christopher Kiekintveld, Son Tran, and William Yeoh. Bidding in periodic double auctions using heuristics and dynamic monte carlo tree search. In *IJCAI*, pages 166–172, 2018.

[12] John Collins, Raghu Arunachalam, Norman M Sadeh, Joakim Eriksson, Niclas Finne, and Sverker Janson. The supply chain management game for the 2005 trading agent competition. 2004.

[13] John Collins, Wolfgang Ketter, and Norman Sadeh. Pushing the limits of rational agents: The trading agent competition for supply chain management. *AI Magazine*, 31(2):63, 2010.

[14] Federal Energy Regulatory Commission, 2017.

[15] Tapio Elomaa and Matti Kaariainen. An analysis of reduced error pruning. *Journal of Artificial Intelligence Research*, pages 163–187, 2001.

[16] European Energy Exchange, 2017.

[17] Vlad Firoiu, William F Whitney, and Joshua B Tenenbaum. Beating the world's best at super smash bros. with deep reinforcement learning. *arXiv preprint arXiv:1702.06230*, 2017.

[18] Franklin M Fisher. Games economists play: a noncooperative view. *The RAND Journal of Economics*, 20(1):113–124, 1989.

[19] Daniel Friedman. The Double Auction Market Institution: A Survey. In *The Double Auction Market*, pages 3–26. Routledge, 2018.

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

[21] Dhananjay K. Gode and Shyam Sunder. 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] Amy Greenwald. The 2002 trading agent competition: An overview of agent strategies. *AI Magazine*, 24(1):83, 2003.

[23] Smart Grid. 2030: A national vision for electricity's second 100 years. *United States of America Department of Energy*, 2003.

[24] Mark A. Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The WEKA data mining software: an update. *SIGKDD Explorations*, 11(1):10–18, 2009.

[25] U. S. E. I. *Annual Energy Review 2010*. Government Printing Office, 2011.

[26] Manish Jain, Dmytro Korzhyk, Ondřej Vaněk, Vincent Conitzer, Michal Pěchouček, and Milind Tambe. A double oracle algorithm for zero-sum security games on graphs. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 327–334. International Foundation for Autonomous Agents and Multiagent Systems, 2011.

[27] P Jordan, Ben Cassell, L Callender, and M Wellman. The ad auctions game for the 2009 trading agent competition. Technical report, 2009.

[28] Michael Kaisers, Karl Tuyls, Frank Thuijsman, and Simon Parsons. Auction analysis by normal form game approximation. In *2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, volume 2, pages 447–450. IEEE, 2008.

[29] Wolfgang Ketter, John Collins, and Carsten A Block. Smart grid economics: Policy guidance through competitive simulation. 2010.

[30] Wolfgang Ketter, John Collins, and Prashant Reddy. Power tac: A competitive economic simulation of the smart grid. *Energy Economics*, 39:262–270, 2013.

[31] Wolfgang Ketter, John Collins, and Mathijs de Weerdt. The 2018 Power Trading Agent Competition. *ERIM Report Series Reference No. 2017-016-LIS*, 2018.

[32] Wolfgang Ketter, Markus Peters, and John Collins. Autonomous agents in future energy markets: The 2012 power trading agent competition. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 1298–1304, 2013.

[33] Wolfgang Ketter, Markus Peters, John Collins, and Alok Gupta. Competitive benchmarking: An is research approach to address wicked problems with big data and analytics. *ERIM Report Series Reference No. ERS-2015-015-LIS*, 2015.

[34] Christopher Kiekintveld, Jason Miller, Patrick R Jordan, Lee F Callender, and Michael P Wellman. Forecasting market prices in a supply chain game. *Electronic Commerce Research and Applications*, 8(2):63–77, 2009.

[35] Christopher Kiekintveld, Jason Miller, Patrick R Jordan, and Michael P Wellman. Controlling a supply chain agent using value-based decomposition. In *Proceedings of the ACM Conference on Electronic Commerce (EC)*, pages 208–217, 2006.

[36] Levente Kocsis and Csaba Szepesvári. Bandit Based Monte-Carlo Planning. In *Proceedings of the European Conference on Machine Learning (ECML)*, pages 282–293, 2006.

[37] Andrey Kolobov. Planning with markov decision processes: An ai perspective. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–210, 2012.

[38] David Kraines and Vivian Kraines. Pavlov and the prisoner's dilemma. *Theory and decision*, 26(1):47–79, 1989.

[39] Rodrigue Talla Kuate, Minghua He, Maria Chli, and Hai H. Wang. An Intelligent Broker Agent for Energy Trading: An MDP Approach. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 234–240, 2013.

[40] Guillaume Lample and Devendra Singh Chaplot. Playing fps games with deep reinforcement learning. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[41] Martin Lauer and Martin Riedmiller. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2000.

[42] Kevin Leyton-Brown and Yoav Shoham. Essentials of game theory: A concise multidisciplinary introduction. *Synthesis lectures on artificial intelligence and machine learning*, 2(1):1–88, 2008.

[43] H Brendan McMahan, Geoffrey J Gordon, and Avrim Blum. Planning in the presence of cost functions controlled by an adversary. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 536–543, 2003.

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

[45] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[46] Susan McDowell Mudambi. The games retailers play. *Journal of Marketing Management*, 12(8):695–706, 1996.

[47] OpenAI. Openai five, 2018.

[48] S. Ozdemir and Rainer Unland. AgentUDE: The Success Story of the Power TAC 2014 Champion. In *Proceedings of the Workshop on Agent-Mediated Electronic Commerce and Trading Agents Design and Analysis (AMEC/TADA)*, 2015.

[49] Serkan Özdemir and Rainer Unland. Agentude17: A genetic algorithm to optimize the parameters of an electricity tariff in a smart grid environment. In Yves Demazeau, Bo An, Javier Bajo, and Antonio Fernández-Caballero, editors, *Advances in Practical Applications of Agents, Multi-Agent Systems, and Complexity: The PAAMS Collection*, pages 224–236, Cham, 2018. Springer International Publishing.

[50] Steve Phelps, Simon Parsons, and Peter McBurney. An evolutionary game-theoretic comparison of two double-auction market designs. In *International Workshop on Agent-Mediated Electronic Commerce*, pages 101–114. Springer, 2004.

[51] Marc Ponsen, Karl Tuyls, Michael Kaisers, and Jan Ramon. An evolutionary game-theoretic analysis of poker strategies. *Entertainment Computing*, 1(1):39–45, 2009.

[52] Nord Pool, 2017.

[53] Axelrod Robert et al. The evolution of cooperation, 1984.

[54] Marko Robnik-Sikonja and Igor Kononenko. An adaptation of relief for attribute estimation in regression. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 296–304, 1997.

[55] Thiago R. P. M. Rúbio, Jonas Queiroz, Henrique Lopes Cardoso, Ana Paula Rocha, and Eugénio Oliveira. Tugatac broker: A fuzzy logic adaptive reasoning agent for energy trading. In Michael Rovatsos, George Vouros, and Vicente Julian, editors, *Multi-Agent Systems*

and Agreement Technologies, pages 188–202, Cham, 2016. Springer International Publishing.

[56] Leonard J Savage. The theory of statistical decision. *Journal of the American Statistical association*, 46(253):55–67, 1951.

[57] L Julian Schvartzman and Michael P Wellman. Stronger cda strategies through empirical game-theoretic analysis and reinforcement learning. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 249–256, 2009.

[58] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

[59] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[60] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

[61] Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. Multiagent cooperation and competition with deep reinforcement learning. *PloS one*, 12(4), 2017.

[62] Anderson Tavares, Hector Azpurua, Amanda Santos, and Luiz Chaimowicz. Rock, paper, starcraft: Strategy selection in real-time strategy games. In *Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2016.

[63] Eclipse Deeplearning4j Development Team. ND4J: Fast, Scientific and Numerical Computing for the JVM. 2016.

[64] Gerald Tesauro and Jonathan L Bredin. Strategic Sequential Bidding in Auctions using Dynamic Programming. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 591–598, 2002.

[65] Gerald Tesauro and Rajarshi Das. High-performance Bidding Agents for the Continuous Double Auction. In *Proceedings of the ACM Conference on Electronic Commerce (EC)*, pages 206–209, 2001.

[66] Karl Tuyls, Ann Nowe, Zahia Guessoum, and Daniel Kudenko. *Adaptive Agents and Multi-Agent Systems III. Adaptation and Multi-Agent Learning: 5th, 6th, and 7th European Symposium, ALAMAS 2005-2007 on Adaptive and Learning Agents and Multi-Agent Systems, Revised Selected Papers*. Springer, 2008.

[67] Karl Tuyls and Simon Parsons. What evolutionary game theory tells us about multiagent learning. *Artificial Intelligence*, 171(7):406–416, 2007.

[68] Karl Tuyls, Julien Perolat, Marc Lanctot, Joel Z Leibo, and Thore Graepel. A generalised method for empirical game theoretic analysis. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 77–85. International Foundation for Autonomous Agents and Multiagent Systems, 2018.

[69] Daniel Urieli. *Autonomous Trading in Modern Electricity Markets*. PhD thesis, Department of Computer Sciences, The University of Texas at Austin, Austin, Texas, USA, December 2015. Code and binaries available at: http://www.cs.utexas.edu/ urieli/thesis.

[70] Daniel Urieli and Peter Stone. TacTex'13: A Champion Adaptive Power Trading Agent. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1447–1448, 2014.

[71] Daniel Urieli and Peter Stone. An MDP-based Winning Approach to Autonomous Power Trading: Formalization and Empirical Analysis. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 827–835, 2016.

[72] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI conference on artificial intelligence*, 2016.

[73] William E Walsh, Rajarshi Das, Gerald Tesauro, and Jeffrey O Kephart. Analyzing complex strategic interactions in multi-agent systems. In *AAAI-02 Workshop on Game-Theoretic and Decision-Theoretic Agents*, pages 109–118, 2002.

[74] William E Walsh, David C Parkes, and Rajarshi Das. Choosing samples to compute heuristic-strategy nash equilibrium. In *International Workshop on Agent-Mediated Electronic Commerce*, pages 109–123. Springer, 2003.

[75] Yufei Wang, Zheyuan Ryan Shi, Lantao Yu, Yi Wu, Rohit Singh, Lucas Joppa, and Fei Fang. Deep reinforcement learning for green security games with real-time information. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1401–1408, 2019.

[76] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[77] Michael P Wellman. Methods for empirical game-theoretic analysis. In *AAAI*, pages 1552–1556, 2006.

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

[79] Michael P. Wellman, Amy Greenwald, Peter Stone, and Peter R. Wurman. The 2001 Trading Agent Competition. *Electronic Markets*, 13(1):4–12, 2003.

[80] Michael P Wellman, Thanh H Nguyen, and Mason Wright. Empirical game-theoretic methods for adaptive cyber-defense. In *Adversarial and Uncertain Reasoning for Adaptive Cyber Defense*, pages 112–128. Springer, 2019.

[81] Michael P Wellman, Daniel M Reeves, Kevin M Lochner, and Yevgeniy Vorobeychik. Price prediction in a trading agent competition. *Journal of Artificial Intelligence Research*, 21:19–36, 2004.

[82] Mason Wright and Michael P Wellman. Evaluating the stability of non-adaptive trading in continuous double auctions. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 614–622. International Foundation for Autonomous Agents and Multiagent Systems, 2018.

[83] Peter R. Wurman, William E. Walsh, and Michael P. Wellman. Flexible Double Auctions for Electronic Commerce: Theory and Implementation. *Decision Support Systems*, 24(1):17–27, 1998.

[84] Chenjun Xiao, Jincheng Mei, and Martin Müller. Memory-augmented monte carlo tree search. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Appendix A

Supplemental Data on Learning Retail Strategy

Table A.1: State features of the DQ Agent

Features	Node Number
Current Timeslot	(1)
Hour of the day	(2)
Previous TS Profit	(3)
Profit Per Timeslot	(4)
Previous TS tariff price	(5)
Previous pub cycle's tariff price	(6)
2nd previous pub cycle's tariff price	(7)
Opponent's previous TS tariff price	(8)
Opponent's previous pub cycle's tariff price	(9)
Opponent's 2nd previous 2 pub cycle's tariff price	(10
Previous timeslot unit cost	(11)
Previous pub cycle's unit cost	(12)
2nd previous pub cycle's unit cost	(13)
Previous timeslot's action	(14–16)
Previous pub cycle's action	(17–19)
2nd previous pub cycle's action	(20–22)
Opponent's previous timeslot's action	(23–25)
Opponent's previous pub cycle's action	(26–28)
Opponent's 2nd previous pub cycle's action	(29–31)
Previous TS market share	(32)
Previous pub cycle market share	(33)
2nd previous pub cycle market share	(34)
Customer's previous TS demand	(35)
Customer's current TS demand	(36)
Customer's next TS demand	(37)

Table A.2: Hyperparameters of DQ Agent Training and Testing

Param	Values
discountfactor	0.99
experience-reply	1000
batch-size	100
target-dqn-update-freq	500
reward-factor	0.01
min-epsilon	0.1
epsilon-nb-steps-perc	1.0
Test Rounds	1000 games
Training Rounds	5000 games

Table A.3: Learning Strategies using DO-EGTA

Candidate Pool (Tot. #NEs)	Selected NE (Learned Strat.)
AlD, NvPbr, Pbr (3)	1.00 AlD (D0)
AlD, NvPbr, Pbr, D0 (1)	1.00 D0
Add AlI to pool	
AlI, D0 (3)	1.00 D0 (D1)
AlI, D0, D1 (5)	1.00 D1 (D2)
AlI D0,D1,D2 (3)	1.00 D2 (D3)
AlI, D2, D3 (1)	1.00 D3
Add ZI to pool	
ZI, D3 (3)	1.00 D3 (D4)
ZI, D3, D4 (1)	1.00 D4
Add HardMJ to pool	
HardMJ, D3, D4 (1)	1.00 D4
Add AlS to pool	
AlS, D4 (3)	1.00 DQ4 (D5)
AlS, D4, D5 (3)	0.54AlS+0.46D5 (D6)
AlS, D5, D6 (3)	AlS is not contributing
Add Pavlov to pool	
Pavlov, D5, D6 (3)	Pavlov is not contributing
Add Grim (G) to pool	
G,D5,D6 (7)	0.32G+0.2D5+0.48D6(D7)
G,D5,D6,D7 (3)	0.77G+0.23D7 (D8)
G,D7,D8 (3)	0.63G+0.37D8 (D9)
G,D9,D7 (5)	1.00D7 (D10)
G,D9,D7,D10 (3)	1.00D10 (D11)
G,D9,D7,D10,D11 (5)	1.00D11 (D12)
G,D9,D7,D11 (5)	0.78G+0.04D9+0.18D7(D13)
G,D9,D7,D11,D13 (7)	0.95G+0.05D13 (D14)
G,D9,D7,D11,D13,D14(5)	0.79G+0.18D7+0.03D13(D15)
G,D9,D7,D11,D13,D15(9)	0.83G+0.09D7+0.08D15(D16)
G,D9,D7,D11,D16,D15 (1)	1.00 D11
Add SoftMJ to pool	
SoftMJ,D11 (3)	0.69SoftMJ+0.31D11 (D17)
SoftMJ,D11,D17 (1)	1.00 D11
Add 2TF1T to pool	
2TF1T,D11 (1)	1.00 D11
Add 1TF1T to pool	
1TF1T,D11 (3)	0.621TF1T+0.38D11 (D18)
1TF1T,D11,D18 (1)	1.00 D11
Add GD to pool	
GD,D11 (3)	0.59GD+0.41D11 (D19)
GD,D11,D19 (3)	0.94GD+0.06D19(D20)
GD,D11,D19,D20 (1)	1.00 D11
Add 1TF2T to pool	
1TF2T, D11 (1)	1.00 D11
Add ZIP to pool	
ZIP, D11 (3)	0.77ZIP+0.23D11 (D21)
ZIP, D11, D21 (3)	ZIP is not contributing
	0.13D11+0.87D21 (D22)
D22, D11, D21 (1)	1.00 D11

Background Information

Vita

Moinul Morshed Porag Chowdhury received a Bachelor of Science in Computer Science and Engineering at the Bangladesh University of Engineering and Technology in Dhaka, Bangladesh, in 2011. He has pursued his Ph.D. in Computer Science at The University of Texas at El Paso under the supervision of Dr. Christopher Kiekintveld. His research interests lie in the intersection between machine learning, multi-agent systems, game theory, and decision making. He is particularly interested in the application of artificial intelligence to solve problems affecting society. He is also interested in bridging the gaps between modern machine learning and decision making communities. His professional experiences include working as a full-time software engineer in several IT firms. He is a recipient of the Graduate Fellowship scholarship from the College of Engineering.