Bidding in Periodic Double Auctions Using Heuristics and Dynamic Monte Carlo Tree Search

Moinul Morshed Porag Chowdhury[1], **Christopher Kiekintveld**[1], **Tran Cao Son**[2], **William Yeoh**[3]
[1] The University of Texas at El Paso
[2] New Mexico State University
[3] Washington University in St. Louis
mchowdhury4@miners.utep.edu, cdkiekintveld@utep.edu, tson@cs.nmsu.edu, wyeoh@wustl.edu

Abstract

In a *Periodic Double Auction* (PDA), there are multiple discrete trading periods for a single type of good. PDAs are commonly used in real-world energy markets to trade energy in specific time slots to balance demand on the power grid. 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. We present a general bidding strategy for PDAs based on forecasting clearing prices and using *Monte Carlo Tree Search* (MCTS) to plan a bidding strategy across multiple time periods. In addition, we 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. We evaluate our bidding strategies using a PDA simulator based on the wholesale market implemented in the *Power Trading Agent Competition* (PowerTAC) competition. We demonstrate that our strategies outperform state-of-the-art bidding strategies designed for that competition.

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 [Pool, 2017; Commission, 2017; Exchange, 2017]). In this paper, 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 smart grid has the potential to improve many problems with our current energy infrastructure, including more effective use of pricing mechanisms to manage demand and supply responses in the grid, greater customer participation and distributed generation capabilities, and proper distribution management for variable-output energy sources [Ketter *et al.*, 2018]. The *Power Trading Agent Competition* (Power TAC) [Ketter *et al.*, 2018] is a trading agent competition [Wellman *et al.*, 2003] designed to advance the understanding of market mechanisms that can coordinate buying and selling decisions in energy markets, and to develop automated bidding agents that can represent individual agents in these markets. It is supported by a sophisticated simulation environment that captures many features of real-world energy markets and allows autonomous broker agents to compete to maximize profits.

The wholesale market is one of the primary markets in the Power TAC scenario, and it is based directly on similar real-world market designs. This market is implemented using PDAs, where a series of double auctions are held for producers and brokers to trade energy for an upcoming time period to balance the supply and demand in their energy portfolios. It operates on a "day ahead" basis for hourly time slots, so energy can be traded up to one day in advance prior to production or consumption. We use the Power TAC simulator for this market as a platform for testing our new PDA bidding strategies.

Bidding in even a single double auction is strategically complex, but PDA bidding strategies have the added complexity of reasoning about future auctions for the same good, including predicting future clearing prices and planning a future bidding strategy. In the Power TAC simulation, an agent needs to be able to compute a bidding strategy simultaneously for 24 concurrent auctions for different future time slots. The agent must submit all decisions within just 5 seconds of wall-clock time, so the strategy computation must be very fast. Using an exhaustive search to find a bidding policy is not feasible given the time limitations and (essentially) continuous nature of the action space (i.e., bid prices).

Our bidding strategies use machine learning methods to predict the distribution of clearing prices for individual auctions. We first propose two novel heuristic methods for bid-

ding based on these predictions that are very fast and outperform existing state-of-the-art bidding strategies for Power TAC. Then, we introduce a more comprehensive approach based on *Monte Carlo Tree Search* (MCTS) search, a statistical anytime algorithm for finding optimal decisions that combines the precision of tree search and the generality of random sampling. We extend this methods to a dynamic version that uses our heuristic policy as an initial action set and dynamically adds additional promising actions over time.

The specific contributions of this paper are as follows: (1) We develop a controlled simulation environment to test PDA bidding strategies for realistic wholesale energy markets. (2) We present machine learning methods for forecasting market clearing prices and two fast heuristic bidding policies based on these predictions. (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 evaluation of our PDA bidding strategies using the principles of 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.

2 Background

Power Trading Agent Competition (Power TAC): The Power TAC [Ketter *et al.*, 2018] 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: the wholesale market, tariff market, and balancing market. The simulation also models a regulated distribution utility and a real location-based population of energy customers during a specific period. Customer models include several entities such as households, electric vehicles, and various commercial and industrial models. Brokers participating in the simulation 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 highest bank balance wins the competition. We refer the reader to the Power TAC game specification [Ketter *et al.*, 2018] for more detail.

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 [Ketter *et al.*, 2018]. 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 bids of the brokers, several large "Gencos" also sell energy on the wholesale market.

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 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 [Wurman *et al.*, 1998]. If the minimum ask price has a higher value than the maximum bid price, then the market does not clear.

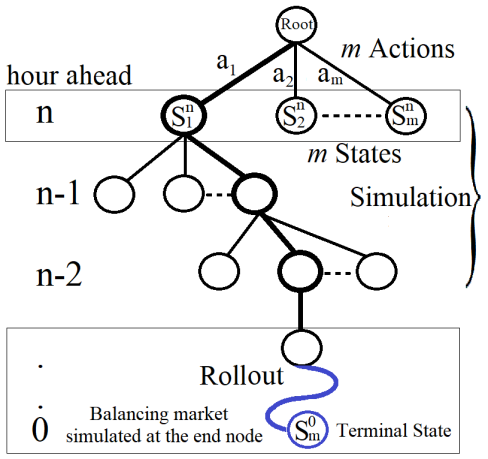

Figure 1: MCTS Tree for the n^{th} Hour-Ahead Auction Bidding

Monte Carlo Tree Search (MCTS): MCTS [Chaslot *et al.*, 2008a] is a tree search algorithm that uses stochastic simulations as illustrated in Figure 1. It incrementally builds a search tree using the following phases: ***(i)* Selection:** The selection of the next state follows the UCT algorithm [Kocsis and Szepesvári, 2006], 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.

3 Related Work

We now briefly review the related work on bidding strategies for Power TAC. AstonTAC [Kuate *et al.*, 2013] uses a non-homogeneous hidden Markov model to forecast energy demand and price for bidding. SPOT [Chowdhury *et al.*, 2015] uses a simple strategy that places 10 bids within a variable price margin depending on the limit price predicted by the REPTree price predictor [Chowdhury, 2016]. The wholesale trading module of AgentUDE [Ozdemir and Unland, 2015] (the 2014 Power TAC champion) uses an adaptive Q-learning bidding strategy that tracks the past market data. Using this technique, the broker is able to understand the market trends regardless of weather conditions and time. TacTex [Urieli and Stone, 2016], the 2013 and 2015 Power TAC winner, uses a

modified version of Tesauro's bidding algorithm, which models the sequential bidding process as a Markov Decision Process.

While there are many other bidding strategies for double auctions [Friedman, 2018], most of them are for *Continuous Double Auctions* (CDAs) [Tesauro and Bredin, 2002] and would need to be modified for PDAs.

4 Bidding Strategies for PDAs

We now describe our proposed bidding strategies for PDAs.

4.1 Price Prediction

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.

MDP Price Predictor: We implement the price predictor used by one of the best agents from previous Power TAC competitions, TacTex [Urieli and Stone, 2014]. A more detailed description is presented in the experiments section.

REPTree Price Predictor: We used a supervised decision tree price predictor that uses *Reduced Error Pruning* (REP). Previous empirical studies show that it performs better in predicting market clearing prices compared to other machine learning strategies [Chowdhury, 2016].

4.2 Heuristic Bidding Policies

We first introduce two very fast 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.

C1 Strategy: 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.

C2 Strategy: Always increase the bid price of the auction with the current minimum predicted clearing price to increase the probability of winning on that specific auction. This focuses on increasing 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.

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.

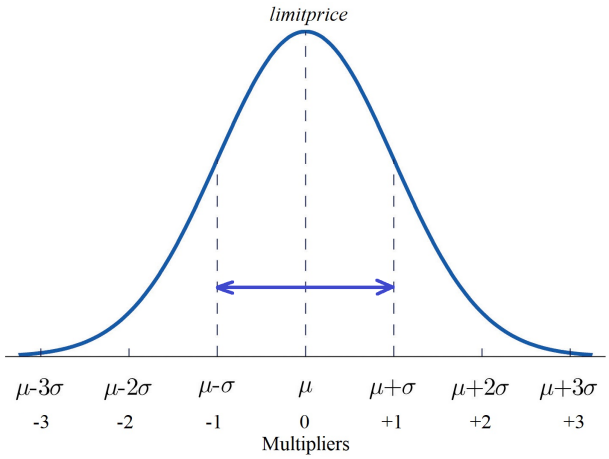

Figure 2: Price Multipliers in MCTS action space.

4.3 MCTS Bidding Policies

We now introduce 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^{mcts}_{min} and μ^{mcts}_{max} by varying μ using:

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

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 2. An action is a *NO-BID* action when $\gamma = 0$. For dynamic actions, we use an accurate price instead of price multipliers, where $\mu^{mcts}_{min} = \mu^{mcts}_{max}$. 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 $action\text{-}space_m$ size of m actions can go into m different states from a specific state.

Transition: A state S^n_m transitions to one of the states in the next time period, $S^{n-1}_0, \ldots, S^{n-1}_m$.

Terminal State: The zero hour-ahead states are terminal states, $\{S^0_0, \ldots, S^0_m\}$. If there are m actions and n hour-ahead auctions, the search tree will have m^n terminal 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 C_{sim} that is the summation 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:

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

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

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

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

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 random small 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 1. Then it selects the action that leads to the highest τ state S_m^n 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 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);

5 Experimental Methods

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 [Ketter *et al.*, 2018].

To set the supply, we collected producer supply data by running 30 Power TAC simulations and calculate the average energy supply per hour. Power TAC's wholesale market simulates 11 different city demands. After observing several simulations in Power TAC, we find the ratio of energy supply and customer demand is near 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. For the weather forecasts (wind speed, wind direction, cloud coverage and temperature forecasts for every hour-ahead auctions), we collected the day-ahead weather forecast data from the simulations and use it as forecasting features in our simulator.

5.1 Benchmark Strategies

Zero Intelligence (ZI): ZI agent [Gode and Sunder, 1993] uses an extremely simple 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 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 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.

Zero Intelligence Plus (ZIP): ZIP agent [Tesauro and Das, 2001] 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, $q = 10\%$).

TacTex: TacTex [Urieli and Stone, 2014] is a broker in Power TAC. Using an online reinforcement learning (RL) algorithm, TacTex procures to meet demand as cheaply as it can 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, which minimizes procurement costs and helps the agent to achieve state-of-the-art performance in competitions and controlled experiments.

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 [Urieli and Stone, 2014]:

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

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

5.2 Training the REPTree Price Predictor

The clearing prices of the previous hour and previous day for the same specific hour-ahead auction capture information about the recent trading history. We selected these two prices as features while training our REPTree price predictor. We also include the four weather forecast (cloud coverage, temperature, wind speed and wind direction) and time of day (current time slot, hour, hour-ahead, date, month, and year) because the energy production of renewable energy producers (e.g., solar) depends on these factors. The game size is also a factor in predicting prices. So, we also consider the number of brokers and producers in the simulation as a feature.

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 [Hall *et al.*, 2009]. 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 1 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.

6 Experimental Results

We first compared the prediction accuracy of both price predictors, and found that the REPTree has a smaller average error (= 54.05%) in predicting per hour ahead auction prices compared to the MDP predictor (= 66.28%). We choose REPTree as our default price predictor for the rest of the experiments.

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. We experimented with four-broker games, 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 3 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. Next, we ran experiments with our MCTS bidding strategy. We denote our default MCTS agent as MCTS_{static}, since this agent has a fixed number of actions (5 bid actions and 1 *NO-BID* action). Based on initial experiments, we have found that bidding with 100% of the current demand is more profitable than

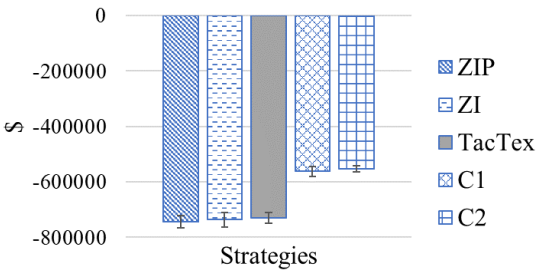

Figure 3: Net cost comparison of becnhmark & candidate strategies.

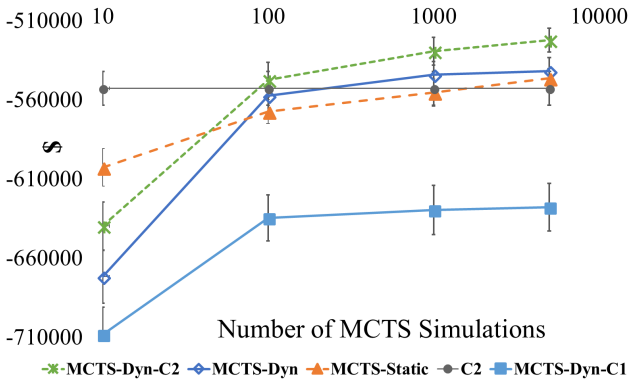

Figure 4: Net Cost Comparison of C2 and all MCTS Variations

bidding with some fraction of the current demand. So in these experiments we always submit 1 bid with full demand. The five bid actions correspond to different prices.

An important restriction on MCTS$_{static}$ is that it searches only a fixed space of possible bidding actions. Here, we consider a simple dynamic MCTS policy [Chaslot *et al.*, 2008b] 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, which is 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$_{dyn}$.

MCTS is a statistical anytime algorithm. So, more computation time should lead it to better performance [Browne *et al.*, 2012]. To investigate the effects of additional simulations and the specification of the action space, we conducted experiments varying important action space properties ($N_{sim}$and$\{\Delta_{min}, \Delta_{max}\}$) of MCTS agents against the 3 benchmark strategies. We vary the N_{sim} of MCTS by 10, 100, 1k, and 5k and run 20 four-broker (ZI, ZIP, TacTex, and MCTS) games. As the states are being visited more with a higher number of MCTS simulation, we can see that the performances of the MCTS agents improve when we increase N_{sim} surpassing the pervious best C2 candidate strategy in Figure 4.

Figure 4 also 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 MCTS$_{dyn\text{-}C1}$ and MCTS$_{dyn\text{-}C2}$. Figure 3 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 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 larger number of visits per state, which is good for stabilizing the average values of the MCTS simulation. It is critical to have accurate actions in a small action space. 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 bounded amount of time.

Finally, our results demonstrate that an MCTS strategy with a larger number of MCTS simulation, 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 right moment to procure the full demand.

7 Conclusions

We propose a novel approach for bidding in Periodic Double Auctions (PDAs) using Monte Carlo Tree Search (MCTS). We evaluate the performance of our 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 great effect on the MCTS performance, and our experiments provide guidance on what features of the actions are most important. Our results show (unsurprisingly) that MCTS performs better with a larger 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, we are now working on finding a kernel regression-based dynamic action adding algorithm with plans to introduce non-determinism in the actions.

Acknowledgments

This research is partially supported by NSF grant 1345232. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the sponsoring organizations, agencies, or the U.S. government.

References

[Browne *et al.*, 2012] 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.

[Chaslot *et al.*, 2008a] 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.

[Chaslot *et al.*, 2008b] 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.

[Chowdhury *et al.*, 2015] 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.

[Chowdhury, 2016] 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.

[Commission, 2017] Federal Energy Regulatory Commission, 2017.

[Exchange, 2017] European Energy Exchange, 2017.

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

[Gode and Sunder, 1993] 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.

[Hall *et al.*, 2009] 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.

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

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

[Kuate *et al.*, 2013] 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.

[Ozdemir and Unland, 2015] 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.

[Pool, 2017] Nord Pool, 2017.

[Tesauro and Bredin, 2002] 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.

[Tesauro and Das, 2001] 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.

[Urieli and Stone, 2014] 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.

[Urieli and Stone, 2016] 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.

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

[Wurman *et al.*, 1998] 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.