A Novel Bidding Strategy for PDAs using MCTS in Continuous Action Spaces

Sanjay Chandlekar[1,2] and Easwar Subramanian[2]

[1] IIIT Hyderabad, India
TCS Innovation Labs, Hyderabad, India
sanjay.chandlekar@research.iiit.ac.in
[2] TCS Innovation Labs, Hyderabad, India
easwar.subramanian@tcs.com

Abstract. Bidding in a periodic double auction (PDA) is challenging due to its sequential nature, where one needs to consider current as well as future auctions to decide the bids. Monte-Carlo Tree Search (MCTS), which is a state-of-the-art online planning algorithm for tackling sequential problems, seems a perfect fit for bidding in PDAs. However, the success stories of MCTS are largely limited to discrete action spaces, and its efficacy diminishes when dealing with continuous actions. Conventional methods often resort to overly simplistic discretizations that limit exploration and fail to provide valuable insights into unexplored actions. In this work, we propose a novel bidding strategy for PDAs, Regression-MCTS, that is built upon MCTS for a continuous action space of bid prices. Unlike conventional methods, our novel MCTS method leverages information obtained from explored actions to enhance the understanding of the larger action set within the continuous domain to place bids in the auctions, thus generalizing the information about action quality between a wider action space for faster learning. To test the efficacy of our proposed method, we design an efficient PDA simulator that closely resembles real-world PDAs. Our analysis verifies that the increase in the number of rollouts improves its performance. Furthermore, our experimental results demonstrate that our approach outperforms existing MCTS-based bidding strategies and the majority of state-of-the-art PDA bidding strategies, showcasing its superior performance in PDAs.

Keywords: MCTS for Continuous Action Space, Online Planning, Bidding Strategy for PDA

1 Introduction

Auctions play a pivotal role in computer science and its associated domains, serving as dynamic mechanisms for the allocation of resources and the facilitation of transactions. Their significance spans a broad spectrum, from the allocation of computational resources in cloud computing to the distribution of spectrum in wireless networks. Double auctions, in particular, are widely used in industries like stock trading and energy markets, with significant economic influence. For

instance, Stock Exchanges facilitate daily transactions worth trillions of dollars, while energy markets in Europe alone witness volumes exceeding 1000 TWh, translating to billions of dollars in cash flow through energy trades [1]. Given this economic impact, the development of bidding strategies capable of optimizing procurement costs holds paramount importance, promising significant improvements in both profitability and operational efficiency.

However, despite the prevalence of double auctions, the design of optimal bidding strategies remains a formidable challenge, particularly in complex domains such as the energy market. In this context, trades often occur through periodic double auctions (PDAs), where participants can procure energy several hours ahead of the delivery hour, necessitating strategic planning across current and future auctions, with each decision influencing subsequent steps. Consequently, the decision-making process for bidding strategies becomes intricate, demanding innovative approaches to navigate the complexities of real-time bidding dynamics effectively. For such sequential decision-making problems, Monte Carlo Tree Search (MCTS) emerges as a fitting framework capable of constructing trees that encompass entire decision-making trajectories, comprehensively capturing process dynamics. Renowned for its efficacy in sequential decision-making tasks, MCTS gained widespread recognition after its pivotal role in AlphaGo's triumph over the world champion in Go [2]. However, MCTS's application predates its Go breakthrough, primarily in game-playing.

Numerous variations aiming to enhance search efficiency and reduce computational overhead have since been proposed. MCTS's advantage lies in its ability to blend the precision of tree search with the generality of random sampling. Nevertheless, its performance in continuous spaces remains a subject of exploration, particularly in real-world problems with continuous action spaces, like determining velocity and acceleration in autonomous driving or setting bid prices in auctions. In these contexts, sequential decision-making is paramount, with each action step affecting the problem state in the future. While MCTS appears well-suited for such tasks, its adaptation to continuous action spaces requires further investigation and refinement.

In the literature, addressing the challenge of applying MCTS to continuous action space problems involves several approaches, including (i) Discretization, (ii) Unpruning or Progressive Widening, (iii) Policy Optimization, and (iv) Action Optimization. Discretization simplifies the continuous action space by discretizing it into a finite set of actions. However, this method constrains MCTS to operate within a fixed action set, limiting its ability to explore all potential outcomes and failing to provide insights into unexplored actions. Progressive Widening addresses the fixed action set limitation by dynamically expanding the action space with more and more simulations. Nevertheless, it also faces constraints due to discretization, restricting thorough exploration of potential outcomes. These techniques share a common limitation: they do not leverage insights gained from explored actions to inform the exploration of unexplored actions or enhance knowledge about previously explored ones.

Given the impracticality of exhaustively exploring all available actions in continuous space, any MCTS algorithm operating in this domain must generalize the action space based on exploration. Policy optimization and action optimization exhibit this generalization capability. Specifically, action optimization (techniques like KR-UCT) aims to enable insights across the entire continuous action space with each action MCTS sample using the environment knowledge, whereas policy optimization methods take a hybrid approach where they build the MCTS tree in a continuous action space and update the policy gradient using the sampled MCTS trajectories to train a parametric model; thereby outperforming discretization and progressive widening methodologies. However, both the above methods are curated for specific problem settings and may not be directly extendable to other problem settings.

In this study, we delve into the realm of continuous action spaces to explore the potential of MCTS, presenting a novel bidding strategy tailored for PDAs. Our strategy, named Regression-MCTS (R-MCTS), harnesses MCTS to derive optimal bidding prices from the continuous action space of prices. Unlike conventional approaches that limit exploration to discrete sets of candidate actions, R-MCTS considers the entirety of the continuous action space, leveraging a predefined set of candidate actions as initialization. Inspired by the KR-UCT method, the core of our strategy lies in generalizing action value estimates across the entire parameter space, facilitating information sharing and enabling exploration beyond the initial candidate set. This adaptability proves invaluable, especially in scenarios with imperfect domain knowledge.

Central to our approach is the generalization of action value estimates over the bid price parameter space for PDAs, achieved by assigning a clearing probability to each action in the action space at each level of the Monte-Carlo tree, reflecting the likelihood of bid clearance. As simulations progress, each exploration updates the clearing probabilities for all previously considered actions at any given level, enhancing MCTS's knowledge incrementally with each iteration. To evaluate the effectiveness of our method, we develop a comprehensive PDA simulation that closely replicates real-world dynamics. This simulation serves as a rigorous testing ground, enabling a comparative analysis of our proposed strategy against state-of-the-art MCTS approaches and top-performing PDA bidding strategies. Below, we outline our contributions:

- We investigate the efficacy of MCTS in continuous action spaces, specifically for the context of placing bids in a periodic double auction.
- Introducing R-MCTS, a novel method designed to navigate the continuous action space of bid prices, harnessing insights gleaned from explored actions to update knowledge about previously visited actions, facilitating enhanced generalization of the action space and accelerating MCTS learning.
- With comprehensive performance analysis, we demonstrate the effectiveness of R-MCTS. Our evaluation encompasses comparisons against several state-of-the-art MCTS methods, as well as various PDA bidding strategies, leveraging a simulated PDA environment as our experimental test bed.

2 Related Work

Some attempts have been made previously to extend the vanilla MCTS to continuous action spaces. One of the early works is done in HOO [5], which deals with continuous arms by utilizing a tree of coverings of the action space and is inspired by the bandit framework. HOOT [3] improvises on vanilla MCTS by replacing the UCB1 action selection rule with HOO. Alternatively, HOLOP [6] embraces a different strategy by modeling this as a continuous bandit problem, where actions correspond to plans, leveraging HOO. Alternative strategies, such as the progressive widening method, have also been expanded to address stochastic continuous state and action planning predicaments by employing double progressive widening [7], which applies progressive widening to both states and actions. To amplify the efficacy of MCTS integrated with progressive widening, cRAVE [8] integrates the RAVE heuristic [10] using Gaussian convolution, thereby promoting information exchange among actions throughout the entire subtree. Additionally, KR-UCT [11] is another notable approach fostering information sharing among actions within the same node through kernel regression, and it generates new actions guided by kernel density estimation. KR-UCT has showcased superior performance compared to cRAVE in simulated curling environments, earning its acknowledgment as state-of-the-art. The work by Couetoux[9] provides an excellent overview of the MCTS literature.

MCTS-based strategies have also found application in the domain of simultaneous ascending auctions [18], negotiations [19] and PDAs, particularly within the framework of the Power Trading Agent Competition (PowerTAC) [13]. This competition, which closely simulates real-world smart grid scenarios, hosts an annual tournament attracting various teams who deploy broker agents equipped with bidding strategies for energy procurement from the wholesale market PDAs. Notably, SPOT [14], a prominent broker agent in the PowerTAC literature, introduced an MCTS-based bidding strategy augmented with heuristic techniques. Inspired by the success of SPOT's MCTS strategy, Tuc_Tac [17] developed its own MCTS bidding strategy, which bears significant resemblance to SPOT's approach. However, both strategies operate within discretized action spaces, crucial for determining bid prices and quantities in auctions. Specifically, they employ bid price predictors to anticipate suitable bid prices for all auctions. These predicted bid prices serve as input for MCTS, which subsequently determines multipliers and quantity fractions from a discrete set of actions. These multipliers are then applied to the predicted bid price to establish lower and upper bounds for bid prices, within which they place multiple bids for the selected bid quantity. However, all these methods are either limited by the discrete nature of action space or curated for specific problem settings and may not be directly extendable to other problem settings.

3 Background

We start by outlining the foundational algorithms that R-MCTS will build upon and introducing the problem domain for which the proposed strategy is designed.

3.1 MCTS for Discrete Action Spaces

MCTS is a powerful algorithm in the realm of artificial intelligence and computer science, particularly acclaimed for its effectiveness in decision-making and problem-solving in complex, strategic environments. Originally developed for game-playing scenarios, MCTS has since found applications in various domains, from robotics to optimization. Unlike traditional tree search methods, MCTS employs a stochastic sampling approach to traverse the search space, effectively balancing exploration and exploitation. By simulating numerous random playouts from each node of a decision tree and updating statistics accordingly, MCTS gradually refines its understanding of the problem landscape, ultimately guiding toward optimal or near-optimal solutions.

MCTS is a simulation-based search approach to planning in finite-horizon sequential decision-making settings. The core of the approach is to iteratively simulate executions from the current state to a terminal state, incrementally growing a tree of simulated states (nodes) and actions (edges). Each simulation starts by visiting nodes in the tree and selecting which actions to take based on a *selection function* and information maintained in the node. Consequently, it transitions to a successor state. When it encounters a node that is not fully expanded, then the node is *expanded* by adding a new leaf to the tree. Then, a *simulation* is performed from the new leaf to a terminal state. The value of the terminal state is then returned as the value for that new leaf and the information stored in the tree is updated.

The most widely used selection function for MCTS is Upper Confidence Bounds Applied to Trees (UCT) [12]. In UCT, each node maintains the mean of the rewards received for each action, $\bar{v}_a$, and the number of times each action has been selected, n_a. It initially tries each action once and then chooses the next action based on the size of the one-sided confidence interval on the reward, computed using the Chernoff-Hoeffding bound as shown below,

$$argmax_a \left[\bar{v}_a + C\sqrt{\frac{log \sum_b n_b}{n_a}} \right] \tag{1}$$

This bound is controlled by the constant C, which determines the exploration-exploitation trade-off and is typically fine-tuned for the specific domain. However, one of the primary limitations of discrete MCTS methods is the fixed size of the action space, which is addressed by progressive widening methods.

3.2 MCTS for Continuous Action Spaces: Simple Progressive Widening (SPW)

In continuous MCTS, the vanilla UCT no longer works since each action should be tried at least once and there are infinitely many actions to be considered. Progressive widening addresses this challenge by maintaining a finite list of actions to be explored and incrementally adding new child action nodes v_c to this

list based on visitation counts. Specifically, a new child node is added whenever the following condition is met:

$$n(v_c)^\alpha \geq |children(v_c)|$$

Here, $\alpha \in (0, 1)$ is a parameter controlling the growth rate, $n(v_c)$ is the visit count of the node v_c's and $|children(v_c)|$ is the size of node v_c children list. This ensures that before adding a new action to the list, the current set of actions gets visited sufficient times. When a new node v_d is created, a new action particle is either sampled from a probability distribution $a \sim \pi_{sampler}(|s)$ or deterministically generated (e.g., to expand the action space coverage). This generated action particle is stored in $action(v_d)$, and the process iterates accordingly.

3.3 Periodic Double Auction

The wholesale market of a smart grid comprises large power-generating companies, also known as GenCos, which produce energy in bulk and offer it at wholesale prices. Energy in this market is traded through an auction mechanism, specifically a *day-ahead periodic double auction* involving GenCos and energy brokers. Here, *day-ahead* signifies that brokers engage in buying or selling energy for future delivery timeslots, typically ranging from 1 to 24 hours ahead. These auctions occur periodically, with clearing taking place after each hour. In most cases, a specific type of double auction known as a *k-double auction* is employed, which is considered for this work as well, as described below.

k-Double Auction: A k-double auction is a type of auction where potential buyers submit bids and potential sellers submit asks to the auctioneer. The auctioneer then aggregates these bids and asks, determining each player's *clearing quantity* and *clearing price* using predefined allocation and payment rules. The clearing price represents the price at which the auctioneer clears the market by matching buyers' bids with sellers' asks. The clearing quantity for each player denotes the number of items they receive (for buyers) or sell (for sellers) after the auction clears, as determined by the allocation rule. Additionally, the payment rule specifies the payment each player makes or receives at clearing time (buyer pays, seller earns). In our context, if a buyer's bid b exceeds the seller's bid s, the clearing price is calculated as $kb + (1 - k)s$ for some fixed $k \in [0, 1]$. Here, we consider $k = 0.5$, meaning the clearing price is the mean of b and s.

Figure 1 provides an illustrative example of market clearing in a typical PDA using the k-double auction mechanism. During each auction instance, buyers and sellers submit their bids and asks to the auction, following the convention where bids contain a positive energy amount and negative price, while asks include a negative energy amount and positive price. Bids, or asks without price details (known as market orders), are sorted first as they are treated as the highest bid amount or lowest ask amount. Demand and supply curves are then constructed from these bids and asks. As illustrated in the figure, asks are sorted in increasing order of prices (forming the supply curve), while bids are sorted in

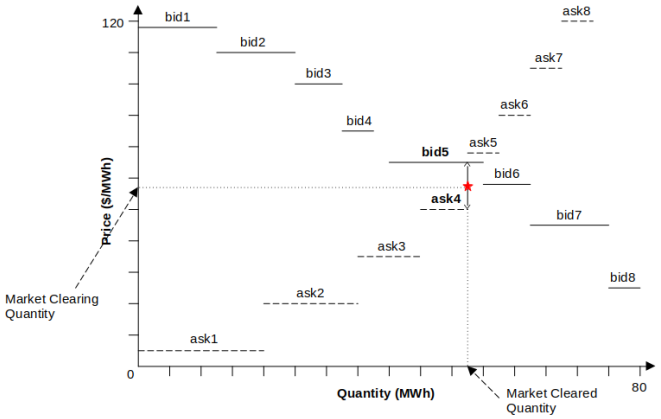

Fig. 1: Market Clearing Example: Partial bid5 and full ask4 are the last to clear

decreasing order of prices (forming the demand curve). The clearing price occurs at the intersection of these curves. If there is no unique price intersection, the clearing price is calculated based on the lowest successful bid (b) and the highest successful ask (s) price, following the definition of a k-double auction.

In this work, we adhere to the *uniform payment rule*, where all successful buyers[sellers] pay[receive] the same clearing price, regardless of their bid[ask] values. The cleared quantity for a buyer[seller] matches its bid[ask] energy amount in the case of full clearing and partially in the case of partial clearance.

4 Regression-MCTS (R-MCTS)

In this section, we present our proposed method, R-MCTS, which operates within a continuous action space by leveraging domain knowledge to generalize explored actions. Our algorithm draws inspiration from KR-UCT's concept of facilitating information sharing among all considered actions, aiding in action space generalization and insights into unexplored actions. Additionally, to accommodate continuous action spaces, we employ the SPW technique. This involves initiating each node with a shallow action space and progressively expanding it as the node is visited repeatedly. Both initially provided candidate actions and progressively added actions, along with their estimated values, collectively form a single dataset at each node for R-MCTS.

Given the fundamental differences in the problem addressed in this work compared to previous approaches like KR-UCT, we do not model execution uncertainty for actions, as there is none in their execution of placing bids in a PDA. Consequently, we abstain from utilizing a kernel function to gain insights into similar unexplored actions. Notably, in the absence of execution uncertainty, KR-UCT essentially operates akin to the standard UCT algorithm with SPW, potentially limiting its efficacy in our problem domain. Instead, we utilize the

knowledge of the market clearing of the bids to strengthen the insights about the actions that have already been visited; here, the actions are the bid prices. The core idea behind our approach is to calculate the clearing probability, $p\_cleared$, for each of the explored actions (bid prices) and keep updating these probabilities with new information about market clearing.

Algorithm 1 delineates the pseudocode for R-MCTS, adhering to the conventional four-step process of MCTS: selection, expansion, simulation, and backpropagation. The primary procedure, R-MCTS, is invoked on the root of the search tree for a fixed number of rollouts determined by a computation budget, returning a list of bid prices for the buyer. The functions $generate\_sellers()$ and $clone\_buyers()$ emulate the sellers and buyers (including R-MCTS and random buyers) of the PDA, respectively. Thus, these modeled sellers and buyers facilitate the simulation of PDA scenarios during the rollout process.

The modeled sellers primarily comprise GenCos, which follow a quadratic supply curve along with some noise. On the other hand, the modeled buyers consist of a clone of R-MCTS alongside ZI buyers that place bids randomly. Throughout the rollout process, we replicate the actual market scenario using these modeled sellers and the same number of modeled buyers as in the original PDA, trading for demands identical to the original demands of buyers. Since we may not have information about the opponents' actual bidding strategies during the rollout, the ZI buyer functions as an opponent of R-MCTS. This setup enables R-MCTS to assess the market scenario and make decisions accordingly. Further insights of the proposed algorithm are provided below.

Calculation of Clearing Probabilities: The function $p_{cleared}(s, action)$ is estimated from the past auction statistics as:

$$p_{cleared}(s, action) = \frac{\sum_{ac \in auction[s], ac.CP < action} ac.cleared\_amount}{\sum_{ac \in auction[s]} ac.cleared\_amount} \quad (2)$$

where $auction[s]$ represents the collection of all past auctions in the current state s, with CP denoting the clearing prices of those auctions. Here, the state is defined by the number of auctions remaining in the PDA, denoted as $rem\_auctions$ and computed based on current and delivery timeslots (Line 2). Essentially, this formula calculates the clearing probability of any $action$, which represents a bid price, by considering what fraction of the total quantity traded in the current state (s) had a clearing price (CP) less than the given $action$. In Algorithm 1, the $get\_pcleared(rem\_auctions, cur.action)$ function executes the above-mentioned calculations for each action or bid price selected during the rollouts (Line 11).

4.1 Selection and Expansion Phase

The selection phase locates the leaf node in the current tree (the node that has not been expanded yet) and then we perform the expansion phase to expand the tree from that node (Lines 9-15). Our algorithm invokes the *select* method (Line

Algorithm 1 R-MCTS(rem_quant, cur_ts, $delv\_ts$)

```
bids ← [], root ← Node()                                  # initialise bids list and root node
rem_auctions ← delv_ts − cur_ts    # number of auctions remaining in a PDA
if rem_quant > 0 then
    while i in NUMBER_OF_ROLLOUTS do
        visited ← [], rewards ← [], cur ← root    # initialise lists of visited, rewards
        visited ← [visited; root]
        list_of_sellers ← generate_sellers()
        list_of_buyers ← clone_buyers()
        while not cur.is_leaf(rem_quant) do
            cur ← select(rem_quant)                                  # see algorithm 2
            cur.p_cleared ← get_pcleared(rem_auctions, cur.action)
            cp, cq, rem_quant ← perform_auction(cur.action, rem_quant,
            list_of_sellers, list_of_buyers)  # clearing the current auction round
            rewards ← [rewards; cp]
            visited ← [visited; cur]
        end while
        cur_cost, cur_quant ← cur.simulation(rem_quant, rem_auctions)
        root ← backpropogation(rewards, visited, cur_cost, cur_quant)
    end while
    lp ← root.best_action()
    bids ← [bids; Bid(buyer_ID, lp, rem_quant)]                  # limit order
else
    bids ← [bids; Bid(buyer_ID, NULL, rem_quant)]                # market order
end if
return bids
```

Algorithm 2 select($node$, rem_quant)

```
if number_of_visits(node)^α ≤ number_of_children(node) then
    A ← actions considered in node
    action ← argmax_{a∈A} E(v|a) + C sqrt( log Σ_{b∈A} number_of_visits(b) / number_of_visits(a) )   # UCB-select
else
    new action ← child of node by taking an action using p_cleared data
    node.children ← [node.children; action] # SPW-select: expanding action space
end if
new_state = node.action
return new_state
```

10), which determines the next node to traverse by utilizing two selection modes: UCB-select and SPW-select, as depicted in Algorithm 2. UCB-select adheres to the UCB formula at each decision node, aiding the algorithm in selecting from previously explored actions. At each node, crucial metrics such as the number of visits, the action that led to the current node, the average unit purchase cost (mean utility $\mathbb{E}(v|a)$), and a set of child nodes b are maintained. As iterations progress and outcomes are revisited, their mean utilities and visit counts are updated accordingly. These updated estimates are then integrated into their

respective roles within the UCB formula, guiding the selection of actions for further refinement. The scaling constant C plays a pivotal role, governing the tradeoff between exploring less-visited actions and refining the value of more promising actions.

SPW-select is triggered when the condition for SPW 3.2 is satisfied (Line 1), indicating that a node has been visited sufficiently many times compared to its number of children. This condition signifies the necessity to expand the action space for the current node. This decision is based on maintaining the number of outcomes in a node bounded by some exponential function of the number of visits to the node. To execute the SPW step, a random policy is employed to select an action beyond the initial action space. Alternatively, domain knowledge can be utilized for selecting a new action.

During the SPW stage, we leverage the knowledge of $p_{cleared}$ data in the current state of the node to introduce a new action into the action space. Given that PDA allows a buyer to bid for 24 rounds per auction, our strategy varies based on risk tolerance. In early rounds, we opt for actions with lower $p_{cleared}$, taking higher risks to procure the required quantity at lower prices. As the auction progresses towards its final rounds, we adopt a more conservative approach, prioritizing actions with higher $p_{cleared}$ to ensure procurement certainty. This strategy enables R-MCTS to explore risky actions (or bid prices) initially and transition towards more assured options as the auction progresses. Once the new action is chosen, it is added to the list of children of the current node, and the function returns the next state after executing the selected action.

Once the algorithm reaches a leaf node in the tree, it expands the tree by selecting an action and conducting auction clearing ($perform\_auction()$) for that chosen action. This process results in transitioning to the next state, which becomes a child of the current node. Subsequently, the simulation phase is conducted from this new state.

4.2 Simulation and Backpropagation Phase

When a new outcome is integrated into the tree, a complete search ensues to reach a terminal state using a default policy, which can be a random policy as well. We simulate the remaining rounds of the auction from the state generated during the expansion phase and record the procurement cost along with the quantity purchased by R-MCTS (Line 16). During the simulation phase, at each of the remaining auction states ($rem\_auctions$), we conduct auction clearing ($perform\_auction()$) and record intermediate procurement costs. Additionally, we update the remaining quantity of all the brokers for the subsequent state based on the market clearing in the current state. Subsequently, all the intermediate procurement costs and cleared quantities of R-MCTS are aggregated and returned as the output of the simulation phase. This output is then utilized to update $\mathbb{E}(v|a)$ and the visit count of the visited nodes along the selected path through the tree during the backpropagation phase (Line 17).

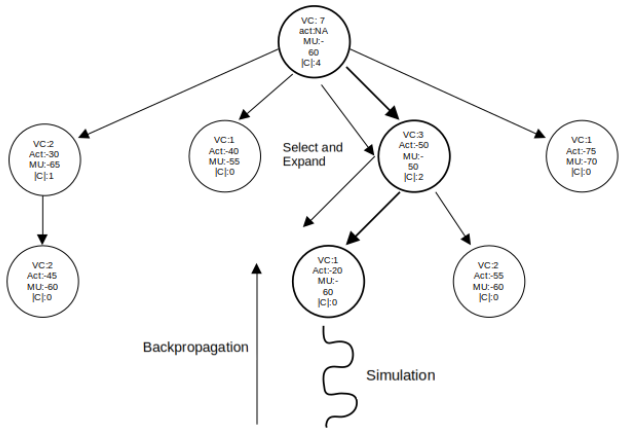

Fig. 2: Pictorial Representation of R-MCTS Displaying all the Phases

4.3 Final Selection Phase

Once the computational budget is exhausted, determined either by the number of iterations or by a pre-allocated time limit, we utilize the search results to select an action for execution. The chosen action is the one at the root with the highest mean utility, indicating the lowest unit purchase cost.

Figure 2 shows an intermediate state of R-MCTS. Each node contains VC, act, MU, |C| denoting visit count of the node, action or limitprice that lead to the node, mean utility, number of children, respectively.

5 Experimental Evaluation

In this section, we conduct experiments to assess the performance of R-MCTS within continuous action spaces. As R-MCTS deals with a continuous action space, it can effectively handle any number of actions by expanding the tree in breadth. Specifically, we apply R-MCTS to the domain of PDAs, where it generates bid prices from a continuous range of feasible prices. To facilitate our evaluation, we have developed an efficient PDA simulator that faithfully replicates real-world PDA dynamics. This simulator serves as our testing platform to thoroughly examine R-MCTS and compare it against baseline strategies. We begin by detailing the experimental setup, followed by descriptions of each of the opponent baseline strategies. Finally, we present results alongside discussions.

5.1 Experimental Setup

To thoroughly evaluate the proposed R-MCTS, we conduct a comprehensive analysis alongside state-of-the-art MCTS strategies and efficient PDA bidding

strategies. Our experiments are organized into three sets. Set-1 focuses on examining the learning capabilities of R-MCTS by analyzing its performance relative to the number of rollouts. Specifically, we compare the average purchase cost of R-MCTS across varying rollout numbers to ascertain whether increased rollouts lead to improved performance. Set-2 involves comparing the purchase cost of R-MCTS against various other MCTS-based bidding strategies in 1v1 games (only two buyers in PDAs, R-MCTS and one of the opponent strategies) under different demand scenarios (low, medium, high, and extreme). By performing these experiments, we aim to demonstrate the efficacy of our proposed approach against state-of-the-art MCTS methods. Similarly, Set-3 entails a comparison of R-MCTS against potent bidding strategies from the PowerTAC literature in 1v1 games across all four demand scenarios. This experiment assesses the performance of our strategies against some of the best PDA bidding strategies.

By conducting these comparisons across different demand levels, we aim to test the efficacy of our strategies under diverse circumstances. Each experiment is executed 100 times to ensure robust results, with mean values presented in the results. Below, we provide detailed explanations for each set of experiments.

Experiment Set-1: In this set of experiments, we engage R-MCTS as the sole participant in a PDA. Each iteration involves randomly generating demand for the strategy from a Gaussian distribution. To fulfil this demand, R-MCTS conducts a series of rollouts to familiarize itself with the environment and determine optimal bidding strategies. The purpose of these experiments is to assess the learning proficiency of R-MCTS across varying numbers of rollouts, ranging from low to high. Specifically, we explore the following rollout numbers: $\{1, 5, 10, 50, 100, 500, 1000\}$. For each rollout quantity, we record the unit purchase cost of R-MCTS, and the corresponding graph is depicted in Figure 3. This analysis provides insights into how the performance of R-MCTS evolves with increasing rollout numbers.

Experiment Set-2: This set of experiments aims to assess the performance of the R-MCTS bidding strategy against various other MCTS-based strategies. To achieve this, we leverage several state-of-the-art MCTS methods to design bidding strategies. Specifically, we implement bidding strategies based on Vanilla-MCTS and MCTS-SPW [7]. Additionally, we incorporate the SPOT [14] agent, renowned as one of the top-performing strategies in PowerTAC tournaments. Each of these strategies engages in 1v1 competition against R-MCTS. Following each individual bidding round, we record the average unit purchase costs of each opponent, including R-MCTS. These experiments are conducted across four different demand levels: (i) Low, (ii) Medium, (iii) High, and (iv) Extreme. R-MCTS's results are computed based on 500 rollouts. The resulting graph is presented in Figure 4.

Experiment Set-3: The third set of experiments mirrors Set-2, albeit with a broader scope. Here, we compare the R-MCTS bidding strategy against a

comprehensive array of efficient bidding strategies drawn from the PowerTAC literature. These strategies embody some of the most effective approaches for PDAs, encompassing diverse attributes such as learning-based strategies using Reinforcement Learning, Heuristics-based methods, and more. Specifically, we include ZI, ZIP, VV21 and SPOT strategies in our evaluation. Similarly to Set-2, each of these strategies engages in 1v1 competition against R-MCTS across all four demand levels. We meticulously record the average unit purchase costs of each opponent alongside R-MCTS following individual bidding rounds. R-MCTS's results are derived from 500 rollouts. The comparative performance graph is displayed in Figure 5.

5.2 Baseline Strategies

Below, we provide concise descriptions of the baseline strategies employed as opponents in our experiments. These strategies are categorized into two groups: MCTS-based strategies and PowerTAC bidding strategies.

MCTS-based Strategies

MCTS-Vanilla: MCTS-Vanilla is an MCTS method utilizing a discrete and fixed-size action space, constructed iteratively through a search tree as detailed in Section 3.1. For designing a bidding strategy for PDAs, MCTS-Vanilla discretizes the action space of bid prices. Each action within this space is defined by two multipliers, α_1 and α_2, applied to the lower and upper bounds of feasible bid prices, respectively. The resulting bid is then placed in the PDA, with the remaining quantity placed as bid quantity across all auction instances. Procurement cost, determined by the bid price, serves as a reward propagated back through the tree.

MCTS-SPW: MCTS-SPW (Simple Progressive Widening) extends MCTS-Vanilla to accommodate continuous action spaces better. While still maintaining a discrete action space, MCTS-SPW dynamically grows its size with the number of rollouts. In the context of designing a bidding strategy for PDAs, MCTS-SPW initializes the action space with randomly sampled bid prices. With more and more rollouts, it explores actions outside the initial action space while striking a balance between exploration and exploitation. Like MCTS-Vanilla, it allocates the remaining quantity as a bid quantity across all auction instances.

SPOT: SPOT [14] employs an MCTS-based bidding strategy; additionally, SPOT integrates several heuristic techniques to optimize bid prices and strategically place multiple bids in auctions. Specifically, it calculates the standard deviation of clearing prices σ offline and incorporates an external limit price predictor that provides limit-price μ. Utilizing a discrete action space, each action includes two multipliers for the limit-price (Δ_{min} and Δ_{max}) and a fraction γ for bid quantity. MCTS selects an appropriate action, and multiple bids are strategically placed within the price range of $\mu + \Delta_{min} * \sigma$ and $\mu + \Delta_{max} * \sigma$ to procure a total fraction of γ of the remaining quantity.

PowerTAC Bidding Strategies

ZI: Instead of considering market conditions, the ZI strategy randomly chooses a bid price within a set range (between a minimum and maximum allowed value) for each auction in a PDA. This strategy involves submitting a single bid with a randomly chosen price for each auction, along with the remaining desired quantity as the bid quantity in all instances.

ZIP: The ZIP agent [15] keeps a scalar variable m representing the profit it aims to achieve, which gets combined with a unit limit price to calculate a bid price p. Small increments adjust the price for each trade with the help of a δ by comparing the submitted bid price and the clearing price. The initial bid price μ is decided randomly at the start of the game and the profit margin is set to -1% of μ, resulting in the initial bid price being $p = \mu * 0.99$. Both δ and μ are updated after each auction to improve future bids. Like ZI strategy, ZIP agent submits the entire remaining quantity as the bid quantity for each auction.

VV21: VV21 [16] is a heuristic strategy that models the cost curve of GenCos derived from uncleared ask information available in PDAs. This strategy aims to identify the price corresponding to the buyer's bid quantity (based on demand forecasts of both the buyer and the market) on the cost curve. It then sets this price as the upper bound on limit prices and places multiple bids below it. This approach facilitates procurement of the majority of the quantity from other buyers' asks in the market, potentially at lower prices, with GenCos considered as suppliers of last resort. VV21 places bids for all the remaining quantity by dividing it into multiple bids.

5.3 Results and Discussion

Below, we present the results of each of the three experiments mentioned above, averaging over 100 random rollouts.

Experiment Set-1: Figure 3 depicts the results from Set-1 experiments, illustrating the impact of the number of rollouts on the average unit procurement cost of R-MCTS. As evident in the graph, there exists an inverse relationship between the number of rollouts and the procurement cost: as the number of rollouts increases, R-MCTS enhances its performance and procures the demand at lower prices. Consequently, the efficacy depends on the number of rollouts feasible within the system.

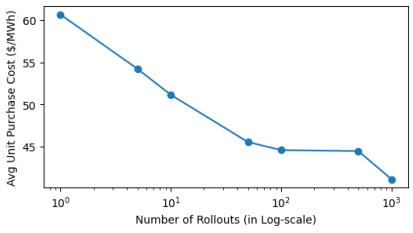

Fig. 3: Rollouts vs Average Unit Purchase Cost for R-MCTS

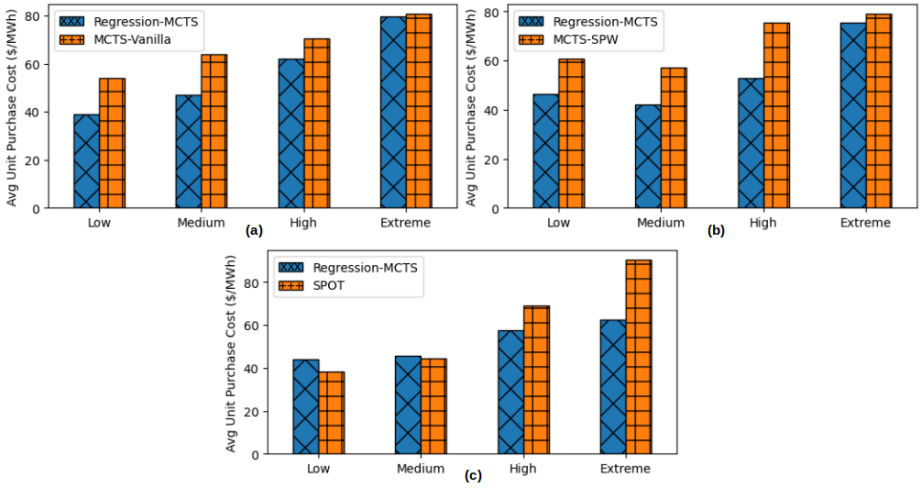

Fig. 4: Comparing R-MCTS vs MCTS-based Bidding Strategies in Low, Medium, High and Extreme Demand Scenarios (((a) vs MCTS-Vanilla (b) vs MCTS-SPW (c) vs SPOT))

Experiment Set-2: Figure 4 illustrates the results of Experiment Set-2, comprising three graphs that compare R-MCTS against three opponents across four different demand scenarios. Specifically, Figure 4(a) compares the R-MCTS against MCTS-Vanilla, Figure 4(b) against MCTS-SPW, and Figure 4(c) against SPOT. As depicted in Figures 4(a) and 4(b), R-MCTS consistently outperforms MCTS-Vanilla and MCTS-SPW across all demand levels—low, medium, high, and extreme. With the exception of extreme demand, R-MCTS achieves significantly lower procurement costs compared to MCTS-Vanilla and MCTS-SPW across all scenarios. Similarly, as evident in Figure 4(c), except for low-demand scenarios, R-MCTS matches SPOT's performance in medium-demand scenarios and outperforms SPOT by a substantial margin in high and extreme-demand scenarios.

Experiment Set-3: Figure 5 presents the results of Experiment Set-3, comprising three graphs comparing R-MCTS against three opponents across four different demand scenarios. Specifically, Figure 5(a) compares R-MCTS against MCTS-ZI, Figure 5(b) against MCTS-ZIP, and Figure 5(c) against VV21. As depicted in Figure 5(a), R-MCTS consistently outperforms ZI across all four demand levels. Similarly, as shown in Figure 5(b), except for extreme demand scenarios, R-MCTS outperforms ZIP by a substantial margin and nearly matches its performance in extreme demand scenarios. Finally, as illustrated in Figure 5(c), while VV21 outperforms R-MCTS in low and medium-demand scenarios, R-MCTS maintains its performance in high and extreme-demand scenarios, surpassing VV21 by a considerable margin. Notably, VV21 is regarded as the best

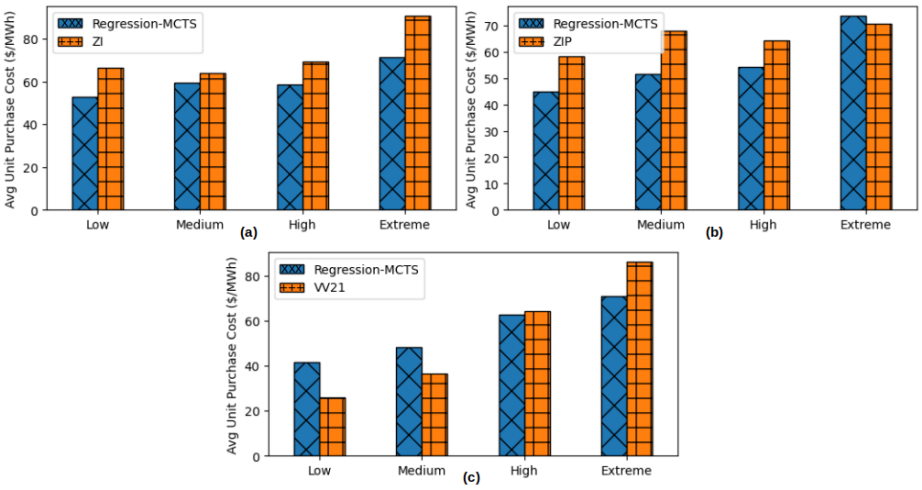

Fig. 5: Comparing R-MCTS vs PowerTAC Bidding Strategies in Low, Medium, High and Extreme Demand Scenarios (((a) vs ZI (b) vs ZIP (c) vs VV21))

strategy for PowerTAC PDA, and our proposed strategy proves to be more robust across different demand scenarios.

The series of experiments conducted above underscore the effectiveness of our proposed R-MCTS method in the continuous action space of bid prices for PDAs. It shows performance enhancement with an increasing number of rollouts. Furthermore, our method consistently outperforms several top-tier MCTS-based and PowerTAC PDA bidding strategies across different demand levels, underscoring its robustness in adapting to varying demand scenarios.

6 Conclusion

In this work, we delve into the underexplored realm of Monte Carlo Tree Search (MCTS) for continuous action spaces, presenting a novel bidding strategy named R-MCTS that extends MCTS to the continuous action space of bid prices, aiming to improve auction bidding strategies. By leveraging information from explored actions, R-MCTS enhances understanding of the larger action space within the continuous domain, facilitating more informed bidding decisions. Specifically, we utilize clearing probabilities of bid prices calculated based on MCTS's previous auction bids, thereby generalizing information about action quality. To evaluate our method's effectiveness, we developed a realistic PDA simulator closely resembling real-world scenarios. Our analysis reveals that increasing the number of MCTS rollouts enhances performance. Moreover, R-MCTS outperforms existing MCTS-based baseline bidding strategies and many state-of-the-art PDA bidding strategies, showcasing its superior performance in PDAs.

References

1. Nord Pool AS: Anual Report, https://www.nordpoolgroup.com/49eea7/globalassets/download-center/annual-report/annual-review-2020.pdf, last accessed 2024/02/26
2. Silver, D., Huang, A. and Maddison, C.: Mastering the game of Go with deep neural networks and tree search. In: Nature 529, 484–489 (2016).
3. Mansley, C., Weinstein, A. and Littman, M.: Sample-Based Planning for Continuous Action Markov Decision Processes, In: ICAPS, vol. 21, no. 1, pp. 335-338, Mar. (2011).
4. Bubeck, S., Munos, R., Stoltz, G. and Szepesvari, C.: X-Armed Bandits, In: Journal of Machine Learning Research, pp. 1655–1695 (2011)
5. Bubeck, S., Munos, R., Stoltz, G. and Szepesvari, C.: Online optimization in X-armed bandits. In: NIPS, pp. 201–208 (2008)
6. Weinstein, A. and Littman, M. L.: Bandit-based planning and learning in continuous-action markov decision processes. In: ICAPS, 2012.
7. Couetoux, A., Hoock, J., Sokolovska, N., Teytaud, O. and Bonnard, N.: Continuous Upper Confidence Trees. In LION'11: Proceedings of the 5th International Conference on Learning and Intelligent Optimization, Italie (2011)
8. Couetoux, A., Milone, M., Brendel, M., Doghmen, H., Sebag, M. and Teytaud, O.: Continuous Rapid Action Value Estimates. In: The 3rd Asian Conference on Machine Learning (ACML2011), volume 20, pages 19–31, Taoyuan, Taiwan, Province De Chine (2011) JMLR.
9. Couetoux, A.: Monte Carlo Tree Search for Continuous and Stochastic Sequential Decision Making Problems. Data Structures and Algorithms [cs.DS]. Université Paris Sud - Paris XI, (2013)
10. Gelly, S. and Silver, D.: Monte-Carlo tree search and rapid action value estimation. In: computer Go. Artificial Intelligence, 175(11): pp. 1856–1875 (2011)
11. Yee, T., Lis´ylis´y, V. and Bowling, M.: Monte Carlo Tree Search in Continuous Action Spaces with Execution Uncertainty. In: The Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, pp. 690-696 (2016)
12. Kocsis, L. and Szepesvári, C.: Bandit Based Monte-Carlo Planning. In: Machine Learning: ECML 2006. Lecture Notes in Computer Science(), vol 4212. Springer, Berlin, Heidelberg (2006).
13. Ketter, W., Collins, J. and Reddy, P.: Power TAC: A competitive economic simulation of the smart grid. In: Energy Economics, pp. 262-270 (2013)
14. Chowdhury, M., Kiekintveld, C., Tran, S. and Yeoh, W.: Bidding in Periodic Double Auctions Using Heuristics and Dynamic Monte Carlo Tree Search. In: International Joint Conferences on Artificial Intelligence Organization, pp. 166–172 (2018)
15. Cliff, D.: Minimal-Intelligence Agents for Bargaining Behaviors in Market-Based Environments. In: Technical Report HPL-97-91, Hewlett Packard Labs (1997)
16. Chandlekar, S., Pedasingu, B.S., Subramanian, E., Bhat, S., Paruchuri, P. and Gujar, S.: VidyutVanika21: An Autonomous Intelligent Broker for Smart-grids. In: Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, pp. 158–164 (2022)
17. Orfanoudakis, S., Kontos, S., Akasiadis, C. and Chalkiadakis, G.: Aiming for Half Gets You to the Top: Winning PowerTAC 2020. In: European Conference on Multi-Agent Systems, pp. 144-159 (2021)
18. Pacaud, A., Marceau, C., and Aurelien, B.: Monte Carlo Tree Search Bidding Strategy for Simultaneous Ascending Auctions. In: 20th International Symposium

on Modeling and Optimization in Mobile, Ad hoc, and Wireless Networks (WiOpt). IEEE, 2022.
19. Buron, C., Zahia, G., and Sylvain, D.: MCTS-based Automated Negotiation Agent. In: PRIMA 2019: Principles and Practice of Multi-Agent Systems: 22nd International Conference, Turin, Italy, October 28–31, 2019, Proceedings 22. Springer International Publishing, 2019.).