# Artificial Intelligence for Multi-Unit Auction design[*]

Peyman Khezr[†] Kendall Taylor[‡]

**Abstract**

Understanding bidding behavior in multi-unit auctions remains an ongoing challenge for researchers. Despite their widespread use, theoretical insights into the bidding behavior, revenue ranking, and efficiency of commonly used multi-unit auctions are limited. This paper utilizes artificial intelligence, specifically reinforcement learning, as a model free learning approach to simulate bidding in three prominent multi-unit auctions employed in practice. We introduce six algorithms that are suitable for learning and bidding in multi-unit auctions and compare them using an illustrative example. This paper underscores the significance of using artificial intelligence in auction design, particularly in enhancing the design of multi-unit auctions.

*Keywords*: multi-unit auction; reinforcement learning; auction design.

## 1 Introduction

Multi-unit auctions play a fundamental role in allocating goods and services across diverse markets, including treasury bills, emission permits, spectrum licenses, and electricity (Khezr and Cumpston, 2022; Bichler and Goeree, 2017). These auctions are often characterized by asymmetric information and, due to the existence of

[*]This research is supported by the Translation Investment Fund (TIF23015), RMIT University. We would like to acknowledge various seminar participants for insightful comments. The usual disclaimer applies.

[†]School of Economics, Finance and Marketing, RMIT University, Melbourne, Australia. Corresponding author, email: peyman.khezr@rmit.edu.au.

[‡]School of Computing Technology, RMIT University, Melbourne, Australia.

multiple units, result in a complex bidding environment. Although multi-unit auctions are commonly employed in practice, there is a lack of consensus in the literature regarding their performance in terms of revenue and efficiency. Moreover, economic theory often lacks clear guidance on the outcomes of many multi-unit auctions (Krishna, 2009).

A range of disciplines, including economics and computer science, have employed various methodologies to investigate multi-unit auctions. Auction theory, data analytics, experimental approaches, and simulations represent some of the predominant methods used in these studies (Ausubel et al., 2014; Roughgarden et al., 2017; Morgenstern and Roughgarden, 2016; Hailu and Thoyer, 2007). Each method offers distinct advantages but also comes with limitations. A common shortfall across these approaches is their inability to comparatively assess the performance of different multi-unit auction variations under similar conditions. Additionally, traditional simulation methods often lack clear guidance as they require an existing model or theoretical prediction to feed into the simulation. In this article, we use artificial intelligence, specifically introducing several reinforcement learning (RL) algorithms, to simulate bidding behavior in various multi-unit auctions.

Reinforcement learning (RL) has been recognized as an effective means of simulating learning processes in uncertain environments, such as auctions (Silver et al., 2021). The capability of RL-based algorithms to learn in a model-free manner makes them particularly well-suited to contexts like multi-unit auctions, where predictive models are often unclear or unavailable. This paper concentrates on three primary RL approaches: Q-Learning, Policy Gradient, and Actor-Critic. We discuss six algorithms that are suitable for simulating the learning and bidding processes in auctions. Our findings detail the principal advantages and disadvantages of each algorithm in the context of bidding behavior simulation. This paper represents one of the initial attempts to apply artificial intelligence to analyze multi-unit auctions.

Our approach leverages artificial intelligence as a complementary tool to previous methodologies, aiming to emulate the human thought process and decision-making in complex bidding scenarios within multi-unit auctions. Using a standard private value model, we assume that buyers' valuations for each unit up for auction are privately known. Our investigation focuses on three sealed-bid multi-unit auctions: the Discriminatory Price (DP), the Generalized Second-Price (GSP), and the Uniform-Price (UP) auctions. All three of these auctions have been employed extensively in real-world markets to sell various goods and services.

We first provide a metric for learning and demonstrate through an example, in which each bidder demands two units, that almost all RL algorithms perform very well in terms of learning to bid. Although they vary in terms of speed and learning patterns, some top-performing algorithms surprisingly converge to very similar strategies in the final episodes of learning. One of the six algorithms, namely Proximal Policy Optimization, not only performs the best by earning the highest payoffs in all cases but also is the most stable in terms of learning. We then compare the revenue and efficiency of the three auctions across three different scenarios.

Our results indicate that the uniform-price Auction performs very well in terms of efficiency, followed by the generalized second-price Auction. The discriminatory price auction, however, dominates in terms of revenue in most cases, except when the number of units available is the lowest.

## Background

Since the seminal work by Vickrey (Vickrey, 1961), it is known that bidding behavior in multi-unit auctions does not necessarily follow those in single unit auctions. Despite this, most of conventionally used multi-unit auctions are extensions of a standard single-unit format. For instance, the generalized second-price auction, as appears from its name, is a format that extend the second-price rule to a multi-unit case. There are several studies that show these multi-unit extensions do not necessarily have the ideal properties of the single unit format (e.g. Back and Zender, 1993; Edelman et al., 2007; Ausubel et al., 2014). Additionally, due to multiple equilibria and lack of closed form solutions for the bidding functions, theoretical studies have no clear conclusion about the revenue and efficiency ranking of commonly used multi-unit auctions (Krishna, 2009; Ausubel et al., 2014).

Recently, the use of RL approaches, particularly Q-learning algorithms, has become more prevalent in auction theory literature. For example, Banchio and Skrzypacz (2022) employ a simple Q-learning algorithm to simulate bidding behavior in two different single-unit auction formats. Khezr et al. (2024) is another study that uses Q-learning to simulate a more complex type of auction, namely *the knapsack auction.* While various other RL approaches could potentially be used to simulate bidding in auctions, current studies are mainly limited to simple Q-learning algorithms. Therefore, our main aim is to explore other, more advanced algorithms that could effectively simulate such complex bidding environments.

## 2 Model

A seller owns $K > 2$ units of a homogeneous object. There are $n > 1$ potential buyers, each demanding up to $k > 1$ units of the object. Every buyer $i$ has a distinct marginal value, $v_k^i > 0$, for the $k$th unit of the object. We use $\mathbf{v}^i = (v_1^i, v_2^i, ..., v_k^i)$ to denote the vector of values for buyer $i$ with diminishing marginal values, that is, $v_1^i \geq v_2^i \geq ... \geq v_k^i$. Each buyer privately knows these values. However, it is publicly known that these values are independently and identically distributed according to a distribution function $F_i(.)$ with finite bounds. Finally, to avoid trivial cases, we assume that the sum of the units demanded by all buyers exceeds the number of available units, that is, $\sum_i k_i > K$.

### The allocation mechanism

The seller uses a sealed-bid multi-unit auction to allocate the objects. Each bidder submits a vector of bids $\mathbf{b}^i$ for all the units they demand. The auctioneer ranks the bids from the highest to the lowest, and allocate object to the $K$ highest submitted bids. We explore three types of payment rules as follows.

### Discriminatory Price Auction (DP)

In this auction, every buyer $i$ submits up to $k$ bids, represented by $\mathbf{b}^1 = (b_1^i, b_2^i, ..., b_k^i)$. The auctioneer arranges all the $n \times k$ bids from the highest to the lowest. The top $k$ bids each win a unit and pay the amount they bid for that particular unit. For example, if the first bid from bidder three, $b_1^3$, holds the top rank, she secures the first unit and pays $b_1^3$ for it.

### Generalized Second-Price Auction (GSP)

Much like the DP auction, each buyer $i$ submits up to $k$ bids. The top $k$ bids each win a unit.However, in this case, winners pay the amount of the bid that is ranked immediately below their own (excluding their own subsequent bids). For instance, if bidder three's first bid, $b_1^3$, holds the highest rank, she secures the first unit but pays the amount of the second-highest bid.

### Uniform-Price Auction (UP)

In this auction every winning bidder pays a uniform price for each unit equal to the highest losing bid, that is the $K + 1$th highest bid.

## 3 Reinforcement Learning approaches

We use Reinforcement Learning (RL) methods to model bidder behavior in multi-unit auctions with multiple bidders. These algorithms are inspired by behavioral psychology, where an *agent* learns to achieve a goal by receiving feedback through rewards or penalties. An agent, in the context of this work, is a simulated human bidder operating within an *environment*, an auction, in which they interact with other agents via bidding for available items. The environment encompasses everything the agent interacts with and can change over time, either due to the agent's actions or independently. Interactions between an agent and the environment are defined by the terms: *state*, *action*, and *reward*.

A *state* is a complete description of the agent's situation within the environment. An *observation*, which might be a partial view of the state, provides the agent with information about the current situation. *Actions* are the set of all possible decisions the agent can make. A *reward* is a feedback signal indicating the immediate effect of the agent's previous action. The agent's objective is to maximize the cumulative reward, or return, over time. This objective and the associated decision-making strategy are encapsulated in an agent's *policy*, a mapping of states (or observations) to actions.

The RL problem is fundamentally about learning the optimal policy that maximizes the expected return when starting from any given state. The agent gradually improves its decision-making abilities by exploring and exploiting different actions. These methods are adapted to the complexity and uniqueness of each auction scenario presented in this paper. Our approach involves multiple agents (bidders) in a dynamic, discrete environment. We evaluate three main approaches for this study:

1. **Q-Learning:**
   Traditional tabular and deep learning approaches are assessed, where an agent learns from their own actions, states, and rewards. In the tabular version, each agent has a Q-table (developed over many auction episodes), which guides them to the best action for a given state that maximises rewards. Initially, exploration is emphasised, but over time, exploiting known actions becomes dominant. The deep Q-learning version (DQN) uses a neural network instead

of a Q-table. This improves the handling of large state-action spaces and enables better generalisation in complex environments.

2. **Policy Gradient:**
   Again, traditional and deep learning versions are considered. The policy gradient approach models action probabilities for given states, aiming to maximise rewards. The policy is parameterised and updated based on the reward gradient. A neural network represents the policy using deep learning, enhancing generalisation and scalability and allowing for more complex policy learning.

3. **Actor-Critic:**
   This method combines value-based estimates (in ways similar to Q-learning) and policy-based functions (similar to those used in Policy Gradient algorithms). The *actor* proposes actions, while the *critic* evaluates them and guides policy updates using neural networks. Two popular and successful methods are evaluated: the Advantage Actor-Critic (A2C) and Proximal Policy Optimization (PPO) algorithms. These methods differ in their approaches to estimating the value of actions and updating the policy.

## 3.1 Environment setup and evaluation

A typical RL problem trains an agent using multiple independent sets of agent and environment interactions called an *episode*. An episode is a complete sequence of states, actions, and rewards that ends with a terminal state. It consists of a series of steps, where at each step, the agent receives an observation of the environment's current state, takes an action based on that observation, and receives a reward from the environment. For example, in a chess game, an episode would be one complete game from the initial board setup until checkmate, resignation, or a draw is reached. The agent and environment, therefore, interact in a loop:

1. The environment presents a state to the agent
2. The agent takes an action based on the state
3. The environment transitions to a new state and provides a reward to the agent
4. The process repeats from step 1 (until the termination condition is met)

In this paper's multi-agent, multi-unit auction scenario, an episode includes only a single step. The number of bids an agent can submit is the same for all agents for all episodes in a training or evaluation session. The state is unique for each agent (but may be identical) and represents the agent's valuation of each item on offer. Each agent's item valuations are drawn from a random distribution (with replacement), for each episode. An episode's state is independent of all other episodes and, possibly, may not be repeated during a session. The single-step episodes differ from typical RL situations and can confound the learning process further.

The reward function used for all auction types is as follows:

$$R = \begin{cases} {}^{p}/_{max(v,\ 1.0)}, & \text{if } s = 1 \text{ and } p > 0 \\ {}^{-(v-p)}/_{max(v,\ 1.0)}, & \text{if } s = 1 \text{ and } p \leq 0 \\ -0.01, & \text{if } s = 0, \end{cases} \tag{1}$$

where $R$ is the reward function, $s$ is a binary variable representing whether the bid is successful, $p$ is the agent's payoff given their bid, and $v$ is the item value associated with a particular bid.

The reward function, therefore, returns values that can be either positive or negative, rewarding positive payoffs and punishing overbidding more severely than unsuccessful bidding. Specifically, when $p > 0$, the agent receives a positive reward as determined by Equation 1. However, when $p < 0$, it indicates that the agent's payment—whether it is their own bid (in DP) or some other lower bid (in GSP and UP)—exceeds their value, thus the numerator of the reward function in Equation 1 becomes negative. In this case, the agent receives a penalty in the form of a negative reward. Both reward and punishment are proportional to the value, which means that for higher values, any given positive or negative payoff is less rewarded or punished, respectively. For instance, if an agent's payoff is 2, it is more significantly rewarded when their value is 3 as opposed to when their value is 9. This design ensures that the utility function remains concave for all agents.

Next we outline six RL algorithms used in the auction simulations: two Q-learning methods, two Policy Gradient, and two Actor-Critic approaches. Each algorithm is implemented as a *bidder* who places a bid in relation to the utility or *value* they place on each of the items on offer. The algorithms are all established and proven methods, each possessing unique strengths and weaknesses (see Table 1).

### 3.2 Q-Learning

Q-Learning is an off-policy model-free RL method developed by Watkins (1989). The method learns the value of an action in a given state without requiring a model of the environment. A key device used by Q-Learning is a data store called a Q-table. The Q-table records expected rewards for all possible actions in a given state. The Q-table is updated using a formulation developed by Bellman (1957). The Bellman equation considers several factors, including the current state, the reward obtained from the action taken, the maximum expected reward for the new state, and a discount rate that determines the importance of future rewards. This process is performed iteratively to help the agent learn the optimal policy by choosing actions with the highest expected rewards.

For each iteration, the Q-value for a state-action combination $Q(s,a)$ is updated using the Bellman equation:

$$Q^{new}(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma \max_{a} \left(Q(s',a') - Q(s,a)\right), \tag{2}$$

where $Q^{new}(s,a)$ is the updated Q-value for the state-action pair, $Q(s,a)$ is the current Q-value for the same state-action pair, $\alpha$ is the learning rate[1] which determines how much new information overrides old information, and $r$ is the immediate reward received after taking action $a$ in state $s$. To balance the influence of immediate versus future rewards, a discount factor $\gamma$ is used, and the largest Q-value for the next state $s'$ $(\forall a \in s)$ is given by $\max_a \left(Q(s',a') - Q(s,a)\right)$.

A key term in Bellman equation is $r + \gamma \max_a \left(Q(s',a') - Q(s,a)\right)$, which represents the difference between the predicted Q-values of the current state-action pair $Q(s,a)$, and the sum of the immediate reward $r$ combined with the discounted value of the 'best' action in the next state $Q(s',a')$. This difference, known as the *temporal difference* is added to the existing Q-value in the Q-table (derived in a previous iteration) during the update process. The resulting adjusted Q-value promotes a more accurate estimation of expected future rewards.

Q-learning also requires a search strategy that facilitates the exploration of new actions while exploiting the knowledge gained through previous actions. Typically, this takes the form of an action-selection strategy, whereby a random action, or the 'best' action (for a given state), is taken with some probability. Taking a random action encourages exploration by evaluating new action-state pairs and possibly discovering improved policies. Taking the 'best' known action aims to maximise the expected reward (Tijsma et al., 2016). In this paper, the search strategy employed implements a decaying $\epsilon - greedy$ mechanism where the exploration rate $\epsilon$ decreases over time:

[1]When referring to RL algorithms, a 'learning rate' is a hyperparameter used to regulate the rate of updates within the algorithm. This should not be confused with the term 'learning ratio' used in this paper, which is used to refer to the ratio of a simulated bidder's bid amount to the item's value.

$$a_t = \begin{cases} \arg\max_a Q(s,a) & \text{with probability } 1-\epsilon_t \\ \text{random action} & \text{with probability } \epsilon_t, \end{cases} \tag{3}$$

where $Q(s,a)$ is the Q-value function that estimates the expected future reward for taking action $a$ in state $s$, $t$ is the current time step, and $\arg\max_a Q(s,a)$ is the action that maximizes the Q-value function for the current state $s$. The exploration rate at time $t$ is determined by $\epsilon_t = \epsilon_{\max} \cdot \text{decay\_rate}^t$, where $\epsilon_{\max}$ is the maximum exploration rate (i.e., the initial value of $\epsilon$), decay_rate is a constant $[0,1]$ that determines the rate of decay, and $t$ is the current time step.

While the scenarios presented in this paper (i.e., auctions with a single bidding round and one-step episodes) differ from traditional RL problems (with numerous steps per episode), the use of Q-Learning and the Bellman equation remains valid. Even though the traditional interpretation of the Bellman equation involves considering future rewards and actions over multiple steps, in a single-step scenario, the equation simplifies to focus on the immediate reward and action taken.

In a one-step episode, the Bellman equation evaluates the immediate reward for a specific action in a given state. It calculates the value of taking that action by considering both the immediate reward and any potential future rewards that the action might influence. Although the concept of future rewards and optimal actions over multiple steps may not be directly applicable in a one-step scenario, evaluating actions based on their immediate consequences is still consistent with the principles of the Bellman equation.

The Q-Learning algorithm faces challenges in multi-unit auction simulation due to large search spaces and non-deterministic and dynamic environments. Storing and processing Q-table can be computationally intractable. Convergence towards an optimal policy is slow, and exploration methods become less effective. The non-deterministic environment leads to Q-value instability and divergence, and the limited state information of agents compounds the problem.

#### 3.2.1 Deep Q-Learning Network

A Deep Q-learning network (DQN) combines the principles of Q-learning with the capabilities of Deep Neural Networks (DNN) (Mnih et al., 2015). It addresses the limitations of traditional Q-learning, especially in environments with large or continuous state spaces where maintaining a Q-table becomes impractical. A DNN refers to a subset of machine learning algorithms that use multi-layered structures of nodes or neurons to simulate the decision-making capabilities of the human brain.

The use of multiple layers in the network allows for the processing of complex, high-dimensional data.

In place of a Q-table, DQN uses DNNs as function approximators that estimate the Q-values for each state-action pair. The Q-value function is initialised with a primary neural network and a target network. It selects actions using an $\epsilon$-*greedy* policy and stores experience tuples in a replay memory buffer. The Q-network is updated periodically to minimize the loss between predicted and target Q-values. Through repeated interaction and learning, the Q-network gradually converges to the true Q-values, resulting in an optimal policy that maximizes expected returns.

## 3.3 Policy Gradient

Known as 'Vanilla' Policy Gradient (VPG) or 'REINFORCE', this RL algorithm optimises policies directly rather than using a value function as Q-Learning does. Policy gradient methods aim to learn a policy $\pi : S \rightarrow A$ that maps states to actions. It is optimised by adjusting policy parameters in the direction that increases the expected return. This is achieved through gradient ascent on the expected return, with the policy represented as a probability distribution over actions.

The algorithm uses a loss function that guides the update of policy weights based on the expected return function $J(\theta)$. The loss function involves the log probabilities of actions taken in each state multiplied by the discounted rewards, averaged over time steps (Sutton and Barto, 2018). The REINFORCE algorithm is synonymous with Policy Gradient approaches and is typically expressed as the following gradient update rule (Williams, 1992):

$$\theta \leftarrow \theta + \alpha\gamma^t r_t \nabla_\theta \ln \pi_\theta(a_t|s_t), \tag{4}$$

where $\theta$ represents the parameters or the policy $\pi$, $\alpha$ is the learning rate, $\gamma$ the discount factor, $r_t$ the return from time step $t$, $a_t$ is the action taken at time $t$, and $s_t$ is the state at time $t$. The term $\nabla_\theta \ln \pi_\theta(a_t|s_t)$ represents the gradient of the logarithm of the policy's probability of taking action $a_t$ in state $s_t$, with respect to the policy parameters $\theta$.

VPG and REINFORCE both suffer from high variance in gradient estimates, which delays convergence and negatively impacts exploration of the search space. Generally, the principal source of variability in policy gradient approaches stems from the variability of accumulated rewards over long multi-step episodes. While the RL scenarios in this paper comprise single-step episodes, the non-deterministic and dynamic environments used ensure the problem of high variance continues to degrade results.

### 3.3.1 Deep Policy Gradient

Similar to how DQN extends traditional Q-Learning through the use of DNNs, deep policy gradient methods replace the parameterized function of VPG with DNN function approximations. The increased representation power of DNNs allows for efficient search of larger state and action spaces and facilitates the use of techniques to reduce variance and encourage convergence.

The DNN used in Deep Policy Gradient approaches takes the current state of the environment as an input and outputs a probability distribution over actions (for discrete action spaces) or parameters of a distribution from which actions are sampled (for continuous action spaces). The DNN's weights and biases are the policy parameters that are then iteratively adjusted during training. Sophisticated optimisers can also be used with Deep Policy methods, which handle noisy gradient estimates much better than VPG approaches. An entropy term is commonly used with the loss function to assist convergence and avoid sub-optimal policies.

While an improvement over VPG, Deep Policy Gradient is very sensitive to the hyperparameter selection. The choice of learning rate, discount factor, and neural network architecture can result in considerable result variation. Unfortunately, choosing the settings is a non-trivial task requiring repeated experimentation and adjustment.

## 3.4 Actor-Critic

The *Actor-Critic* approach combines aspects of both policy-based (the "Actor") and value-based (the "Critic") learning. The actor component is responsible for selecting actions based on the current policy, that is, a mapping from states to actions. The critic evaluates the actions taken by the actor by estimating the value function, which measures the expected return from a given state using the current policy. The critic's evaluation is then used to update the actor's policy towards more rewarding actions. One key advantage of actor-critic methods is their ability to handle continuous and discrete action spaces. In addition to this versatility, separating the policy and value function estimates can stabilize the learning process when compared to methods using only policy or value function estimation.

### 3.4.1 Advantage Actor-Critic

The *Advantage Actor-Critic* (A2C) method is a variant of the Actor-Critic approach that improves the actor's policy by using the *advantage function.* The advantage function measures the relative quality of an action in a given state compared to the average action in that state. The A2C algorithm addresses some of the key issues with both policy gradient and Q-learning-based methods. Specifically, in using the advantage function, A2C reduces the variance in policy updates, leading to more stable learning; both continuous and discrete action spaces can be handled efficiently (unlike Q-learning approaches); and the method scales well in complex environments due to the separate handling of policy and value function estimations.

The operation of the A2C algorithm proceeds as follows:

- During the interaction with the environment phase, the actor assesses the current state of the environment and decides on an action according to its policy. This chosen action is then implemented within the environment, which in turn generates a new state and awards a reward.
- The critic appraises the selected action by determining the value of the current and subsequent states. The advantage is then computed by taking the difference between the actual reward received and the sum of the discounted value of the forthcoming state and the value of the present state.
- The actor modifies its policy parameters by utilizing gradients obtained from the advantage function, thereby favouring actions that yield positive advantages. At the same time, the critic adjusts its value function parameters with the aim of reducing the discrepancy between the estimated values and the actual returns.

Mathematically, the advantage function, $A(s, a)$, is defined as the difference between the action-value function, $Q(s, a)$, and the state-value function, $V(s)$:

$$A(s, a) = Q(s, a) - V(s). \tag{5}$$

The policy gradient update rule in A2C can be represented as follows, where $\theta$ represents the parameters of the policy network, $\alpha$ is the learning rate, and $\nabla_\theta \log \pi_\theta(a|s)$ is the gradient of the log-probability of taking action $a$ in state $s$ under policy $\pi$ parameterized by $\theta$:

$$\Delta\theta = \alpha \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) A(s_t, a_t). \tag{6}$$

The value function (critic) is updated to minimize the difference between the estimated value and the actual return, which can be represented using a loss function, $L(w)$, where $w$ represents the parameters of the value function:

$$L(w) = \frac{1}{2} \sum_{t=0}^{T} \left(R_t - V_w(s_t)\right)^2 . \tag{7}$$

Here, $R_t$ is the return (cumulative discounted reward) from time $t$.

A2C excels in scenarios where future rewards must be estimated, and decisions must consider long-term outcomes. Consequently, the advantage function plays a crucial role in A2C methods, facilitating the balance between immediate and future rewards. In the single-bid auction scenarios examined in this paper, such features are underutilised and possibly unnecessary. Each auction episode is effectively a single decision with an immediate reward, and the learning process primarily focuses on associating states directly with their resulting rewards without considering future states. Nonetheless, experimental results using A2C (see section 4) reveal a substantial improvement over simpler policy gradient and Q-learning methods, especially regarding learning stability.

### 3.4.2 Proximal Policy Optimisation

Proximal Policy Optimization (PPO) was introduced in 2017 by John Schulman and colleagues in their paper "Proximal Policy Optimization Algorithms" (Schulman et al., 2017). This paper presented PPO as a simpler alternative to Trust Region Policy Optimization (TRPO) while achieving similar or better performance. PPO works by limiting the change in policy during each update to avoid disruptively large policy updates.

In the context of reinforcement learning (RL) and specifically Proximal Policy Optimization (PPO), a policy refers to an agent's strategy to make decisions. It defines the mapping from perceived states of the environment to actions to be taken. In the case of PPO, the algorithm trains a stochastic policy in an on-policy way, which means that it explores by sampling actions according to the latest version of its stochastic policy

The policy in RL can be deterministic or stochastic. A deterministic policy directly maps states to actions, while a stochastic policy outputs a probability distribution over actions for a given state. PPO focuses on training a stochastic policy, enhancing exploration and robustness when applied to complex environments. The PPO algorithm's objective is to improve the policy in a stable and sample-efficient manner, enabling the agent to make better decisions over time.

While PPO and A2C are both state-of-the-art Actor-Critic variants, PPO has several advantages over A2C, particularly in terms of stability, sample efficiency, and flexibility (Cheng et al., 2021; Corecco et al., 2023). Specifically, such advantages include:

- Stability: PPO's clipped surrogate objective function (equation 8) prevents large updates to the policy, ensuring smoother convergence and reducing the likelihood of training instability.
- Simplicity: complex hyperparameter tuning and sophisticated optimization techniques are unnecessary.
- Sample Efficiency: can learn effective policies from fewer interactions with the environment.
- Robustness and Scalability: Handle discrete and continuous action spaces, multiple agents, and parallel environments.
- Balanced sample and computational complexity: lower variance and no expensive second-order optimization.
- Exploration-Exploitation trade-off: uses the same policy for exploration and exploitation along with entropy regularisation to encourage the former when needed.
- Empirical Performance: Empirical results have shown that PPO outperforms A2C in various benchmarks and applications Huang et al. (2022).

The key idea behind PPO is to improve the training stability of the policy by limiting the change made to the policy during each update. This is achieved by constraining the policy updates to a 'trust region', which prevents the new policy from deviating too far from the old policy. In an auction environment with private information and non-deterministic outcomes, minimising the size of policy updates is an important feature. Unpredictable by other auction participants (agent or human) during the training of an RL agent can severely compromise learning by misleading an algorithm's search for the best decision. PPO achieves this via a clipped surrogate objective function. The objective of this function is defined as:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[ \min(r_t(\theta)\hat{A}_t, \mathrm{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t) \right] \tag{8}$$

where $L^{CLIP}(\theta)$ is the clipped surrogate objective, $\hat{\mathbb{E}}_t$ is the empirical expectation over a batch of training data, $r_t(\theta)$ is the ratio of the probability of taking an action under the new and old policies, $\hat{A}_t$ is the estimated advantage of taking an action at time step $t$, and $\text{clip}(x, a, b)$ clips the value of $x$ to be between $a$ and $b$.

The clipped surrogate objective limits the policy update to a range determined by $1 - \epsilon$ and $1 + \epsilon$, where $\epsilon$ is a hyperparameter that controls the size of the trust region. This ensures that the policy update is not too large, which improves training stability.

Like A2C, using PPO in single-bid auctions does not fully exploit the algorithm's potential. PPO is designed for environments where actions have long-term consequences over multiple steps; however, as displayed in section 4 below, PPO performs exceptionally well in all auction types examined in this paper.

In summary, Table 1 provides an account of the advantages and disadvantages of each RL algorithm used in this paper. The following section then presents the results from a series of empirical experiments with each of the six algorithms performing as bidders in the three different auction types: Discretionary Price, Generalised Second-Price, and Uniform-Price.

Table 1: Main advantages and disadvantages of the six reinforcement learning algorithms considered in the paper.

| **Algorithm** | **Advantages** | **Disadvantages** |
|---|---|---|
| Q-Learning | • Simple to implement for discrete action spaces<br>• Finds optimal policy given enough time | • Struggles with large or continuous state spaces<br>• Requires discretization for continuous actions |
| Deep Q-Learning (DQN) | • Handles high-dimensional state spaces using deep neural networks<br>• Off-policy learning | • Struggles with continuous action spaces<br>• Can be sample inefficient |
| Vanilla Policy Gradient | • Directly optimizes the policy<br>• Can handle continuous action spaces | • High variance in gradient estimates<br>• Can be sample inefficient |
| Deep Policy Gradient | • Can solve complex problems with high-dimensional inputs<br>• Flexible with continuous action spaces | • Requires careful tuning of hyperparameters<br>• High computational cost |
| Advantage Actor-Critic (A2C) | • Reduces variance of policy gradient estimates<br>• Can handle continuous actions<br>• Balances bias and variance | • More complex to implement than simpler methods<br>• Requires tuning of advantage estimation |
| Proximal Policy Optimization (PPO) | • Stable and robust performance<br>• Simplifies implementation compared to other advanced methods<br>• Effective in various environments | • More complex than basic policy gradients<br>• May require more computational resources for large-scale problems |

## 4 Simulations

In this section, we conduct various simulations using the six algorithms described above. Since the primary aim of this paper is to assess various RL algorithms that are suitable for simulating bidding behavior, we maintain a consistent number of bidders across all simulations and only vary the total number of units available to allow for a tractable analysis. We assume each auction features six bidders, each demanding two units. We explore three scenarios with regard to the total units available for each auction: one with four total units, one with six units, and one with eight units.[2]

The main variable used to assess the learning process in each simulation is called the 'learning ratio', which is calculated as the value minus the bid, divided by the value. Intuitively, one expects an agent following an optimal strategy to submit bids that proportionally differ from their value in a stable manner. Therefore, the learning ratio is a good metric to test such performance. Once we confirm that the learning patterns demonstrate reasonably consistent learning, we then use revenue and efficiency as the metrics to evaluate auction performance. Revenue is simply the sum of payments made by all winning bidders, and efficiency is calculated as the sum of the $K$ highest values (irrespective of the bids), where $K$ is the number of units available minus the sum of the values of those who won an item. When this term is zero, the auction has allocated the objects in a fully efficient manner.

### 4.1 Implementation

All simulations were programmed using Python 3.10 combined with the "Gymnasium" library (for developing and comparing reinforcement learning algorithms)Towers et al. (2023), and "Stable Baselines 3" (for the deep learning and actor-critic algorithm implementations) Raffin et al. (2021). All algorithms are implemented with default settings, and no hyper-parameter tuning was conducted.

Custom Gymnasium environments were created for the simulated auctions, and algorithms from Stable Baselines were modified for use in multi-agent auction scenarios. All simulations were performed on the same computer, running the Linux-based Ubuntu 22.04 operating system with an Intel Core i9-9900KS 4.00GHz CPU, 64Gb RAM, and a Nvidia RTX 3080 GPU with 10Gb RAM (all deep-learning algorithms were run using the GPU).

[2]These numbers are chosen to represent three cases: one where the total number of units available is below half of the total demand, one where it is equal to half, and one where it is more than half but less than the total demand (Engelbrecht-Wiggans and Kahn, 1998; Khezr and Menezes, 2017).

## 4.2 Preliminary simulations

To compare the performance of each algorithm, we aim to run them in an environment where each method competes against the others. Prior to this, we pre-train each algorithm individually against agents with a random bidding strategy. The objective of this initial training is to facilitate each agent's exploration of the search space and learning the basic mechanics of the auction environments. In addition, training against agents that make random decisions helps prevent overfitting and co-adaption to other agent's bidding strategies (Zhu et al., 2024; Zang et al., 2023).

All six RL approaches were initially trained against five agents who made random bids at or below their item values. A total of 54 training sessions were performed with the resultant model (or Q-table) saved. For the six algorithms, training runs were performed using 100,000 episodes for each combination of auction type (DP, GSP and UP) and number of items on offer (4, 6 and 8).

## 4.3 Simulation results

We begin with the learning results for the three auctions in three different scenarios. Figure 1 shows the learning ratios of the six algorithms for the Discriminatory Price auction across three scenarios. As shown in this figure, Proximal Policy Optimization (PPO) is the most stable algorithm in terms of the learning ratio and is, in some sense, the fastest. However, all six algorithms demonstrate reasonable convergence to similar strategies by the final episodes. It should be noted that since bids are ranked from highest to lowest, the order in which an agent submits their highest bids (first or second) does not impact the outcome. In all cases, agents bid below their values, a straightforward conclusion from the theory (Krishna, 2009). Moreover, when more units are available, agents tend to reduce their second bids further, which is also very intuitive.

A similar pattern of learning is observed in Figure 2 for the Generalized Second Price auction. Again, in all cases, the agents bid below their values, aligning with theoretical predictions (Edelman et al., 2007). The DPN algorithm struggles to learn an optimal strategy, and as the number of items increases, its performance worsens. This is mainly because DPN is significantly influenced by stochastic outcomes, and in the GSP format, where the payoff is determined by someone else's bid, DPN struggles to find a consistent learning path. This issue is most pronounced in the Uniform Price auction, where computing an expectation for the highest losing bid from a probabilistic perspective is much more challenging.

Finally, Figure 3 illustrates the learning patterns for the Uniform Price auction.

In PPO, Actor-Critic (A2C), and Q-learning, when the number of items is low, bids are less truthful. While Q-learning maintains this strategy even as the number of items increases, PPO and A2C shift towards more truthful bids with more items available. We later show that this adjustment allowed these two algorithms to achieve the highest payoffs among all agents.

*Payoffs*

Having shown the learning patterns of all agents, we next demonstrate their performance in terms of payoffs, which is the most important metric for their behavior. As illustrated in Tables 2, 3, and 4, PPO consistently achieves the highest payoff among the six algorithms without any exceptions. A2C is the second-best performing algorithm in almost all cases, except for two instances where VPG ranks second. The other algorithms vary in terms of their payoff rankings across the three different scenarios and auctions. Notably, Q-learning, despite its simplistic structure, ranks among the top three algorithms in several scenarios, including DP and GSP with four units and DP with eight units.

*Auction performance*

Next, we show how each auction performed in terms of revenue and efficiency. Figure 4 depicts the revenue and efficiency for each auction across the three scenarios. Focusing on later episodes, where the learning patterns stabilize, we observe that with four items available, all three auctions generate similar revenues, with the Uniform Price (UP) auction slightly outperforming the others. However, as the number of units increases, this ranking changes; with eight items available, the Discriminatory Price (DP) auction generates the highest revenue, followed by the Generalized Second Price (GSP) and then UP. In terms of efficiency, UP dominates the other two auctions in all scenarios, followed by GSP and DP.

### 4.3.1 Simulation results with all PPO agents

Given the results above, it is clear that the PPO agent significantly outperforms the others in terms of learning to bid optimally. However, observations of the learning patterns indicate that the auction ranking in terms of revenue and efficiency is influenced by other sub-optimal learning behaviors. It is worthwhile to explore how each auction performs when all six agents are PPO. Figure 5 shows the results regarding the revenue and efficiency rankings of the three auctions when all agents are PPO. The Uniform-Price auction remains the most efficient among the three, closely followed by GSP. However, the revenue rankings are reversed, with GSP now generating the highest revenue, followed by UP.

Two clear conclusions can be drawn from these observations: first, the Uniform-Price auction consistently performs well in terms of efficiency; second, GSP is the most stable auction in terms of both revenue and efficiency, regardless of whether the agents bid optimally or not.

## 5 Conclusions

In this article, we introduced six reinforcement learning algorithms to simulate bidding behavior in three different multi-unit auctions. While this paper extensively investigates RL-based approaches for simulating auctions, the task is far from complete. There is a pressing need for future studies to explore additional modeling assumptions for agents, including the possibility of allowing more than two bids per agent and increasing the number of items available. We note that such expansions require significant computational resources and time commitment, which were beyond the scope of this research.

## 6 Appendix: Figures and Tables

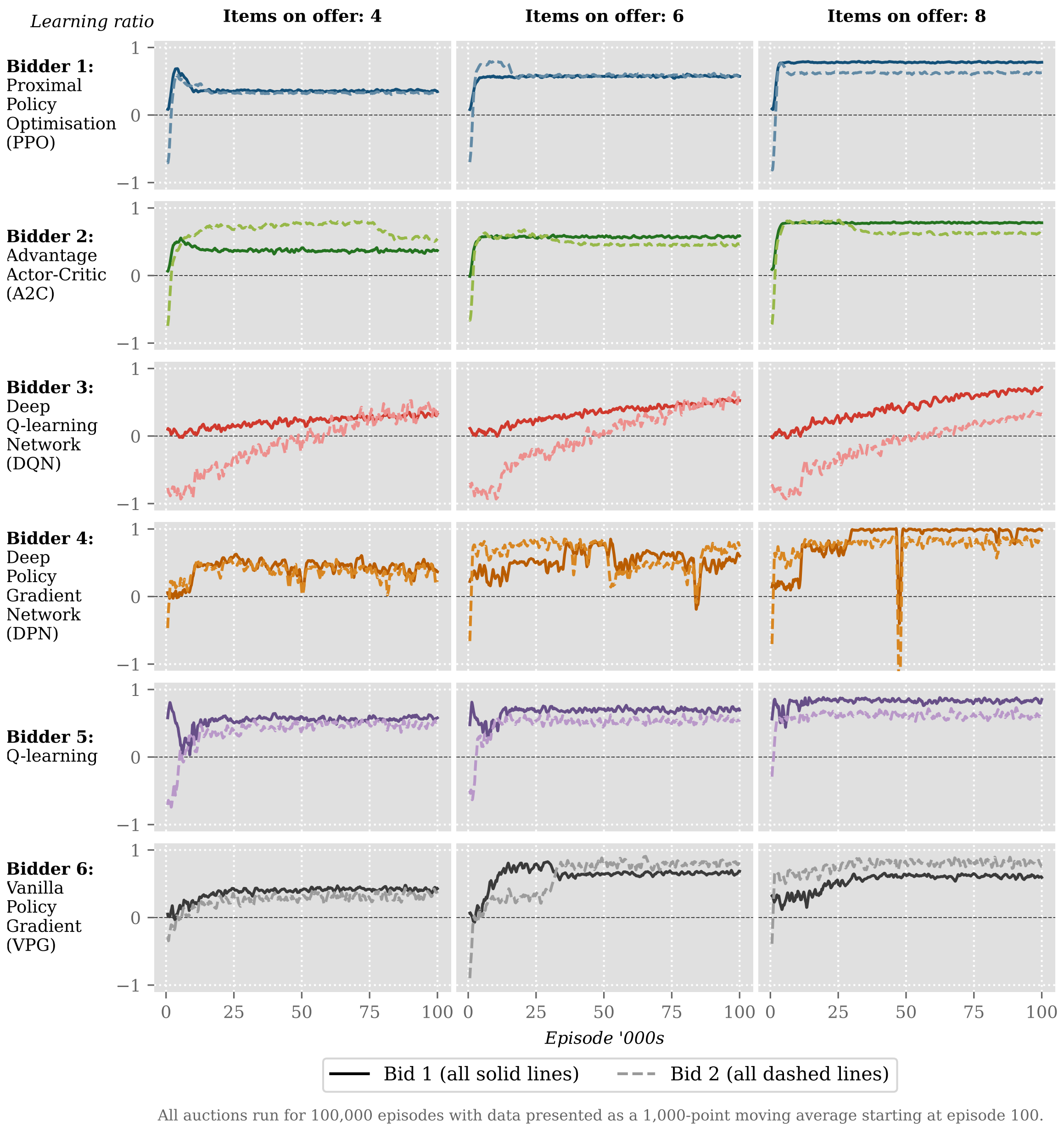


Figure 1: Bidder learning ratios for the Discriminatory Price (DP) auction comparing four, six and eight items on offer with six bidders, each demanding two items.

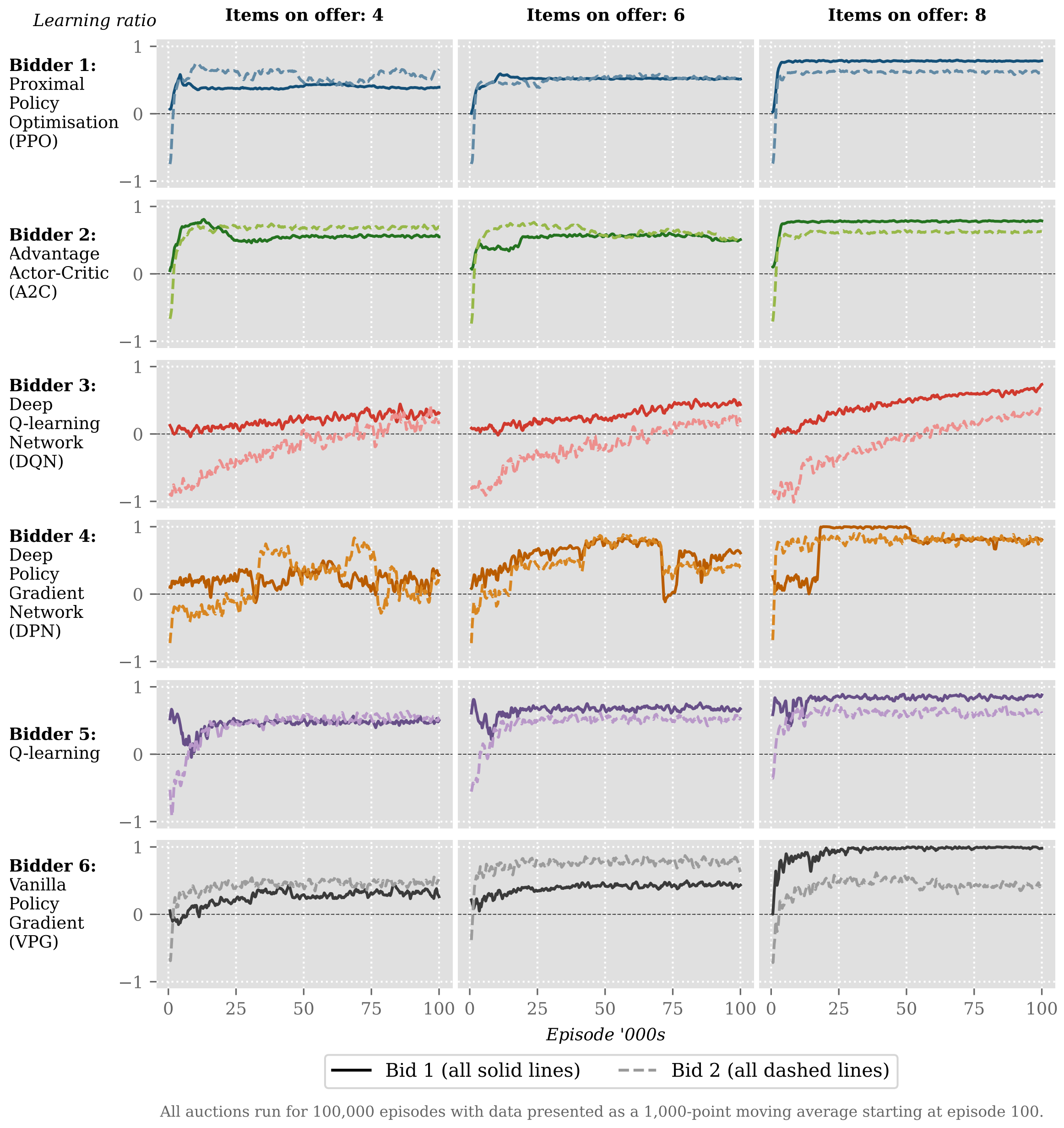
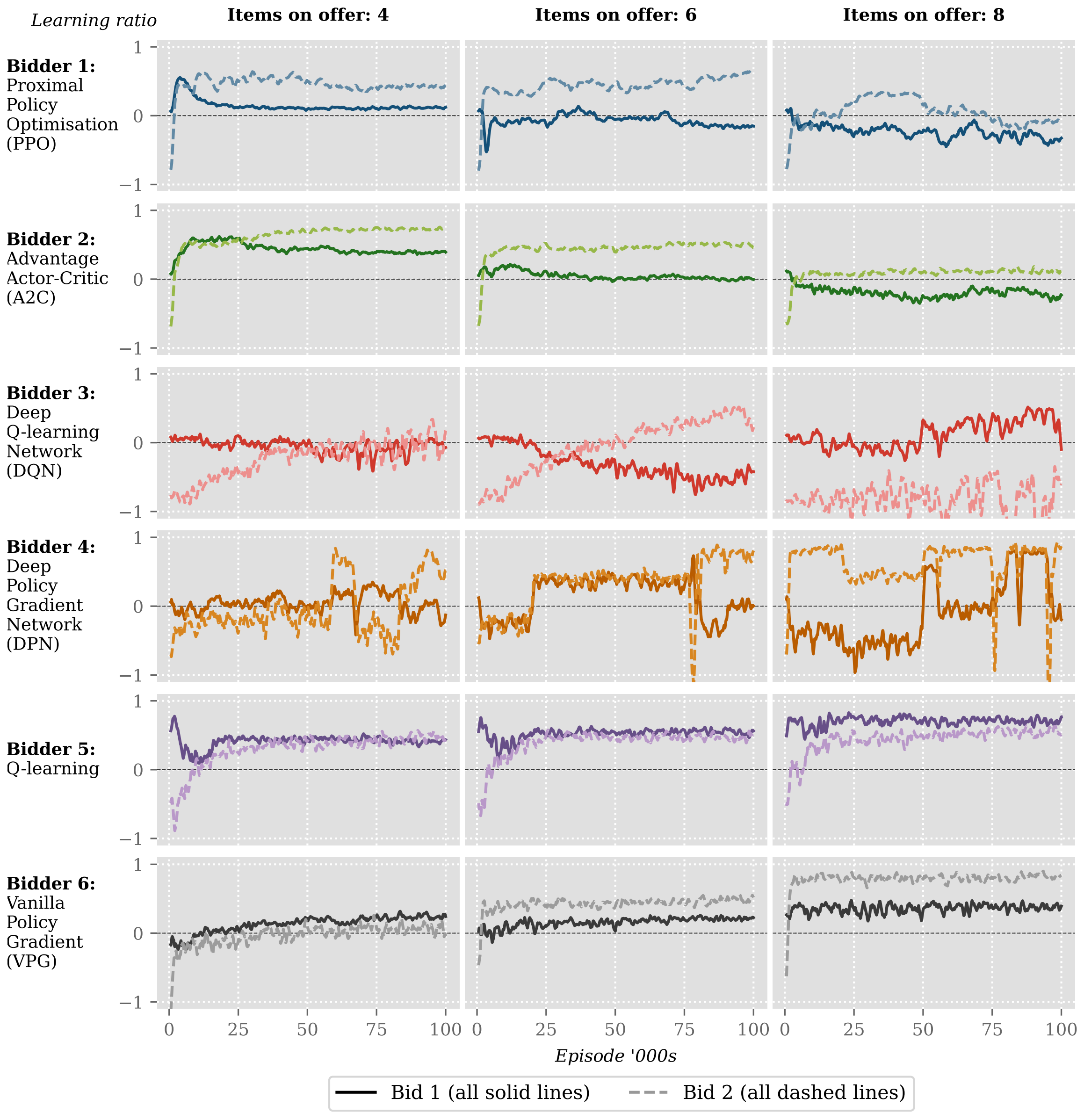


Figure 2: Bidder learning ratios for the Generalised Second-Price (GSP) auction comparing four, six and eight items on offer with six bidders, each demanding two items.



All auctions run for 100,000 episodes with data presented as a 1,000-point moving average starting at episode 100.

Figure 3: Bidder learning ratios for the Uniform-Price (UP) auction comparing four, six and eight items on offer with six bidders, each demanding two items.

Table 2: Auctions with four items on offer: Bidder results by auction type over 100,000 episodes, ranked by total payoff in each auction type.

| | **Bidder** | | | **Payoff** | | **Cost** | |
|---|---|---|---|---|---|---|---|
| **Auction type** | **Rank** | **ID** | **Type** | **Total** | **Mean** | **Mean** | **Items won** |
| Discretionary Price (DP) | 1 | 1 | PPO | 253,189 | 2.9 | 4.0 | 86,886 |
| | 2 | 6 | VPG | 141,213 | 2.0 | 5.1 | 69,335 |
| | 3 | 5 | QL | 137,913 | 2.2 | 4.0 | 63,710 |
| | 4 | 2 | A2C | 128,023 | 3.0 | 3.1 | 43,038 |
| | 5 | 4 | DPN | 109,916 | 2.3 | 3.9 | 47,873 |
| | 6 | 3 | DQN | -9,577 | -0.1 | 5.2 | 89,158 |
| Generalised Second-price (GSP) | 1 | 1 | PPO | 225,346 | 3.1 | 4.8 | 72,910 |
| | 2 | 2 | A2C | 174,360 | 3.0 | 5.0 | 57,324 |
| | 3 | 5 | QL | 154,940 | 2.6 | 4.6 | 60,610 |
| | 4 | 4 | DPN | 147,031 | 2.2 | 4.6 | 66,691 |
| | 5 | 6 | VPG | 134,208 | 2.0 | 6.9 | 68,439 |
| | 6 | 3 | DQN | 38,024 | 0.5 | 5.6 | 74,026 |
| Uniform price(UP) | 1 | 1 | PPO | 209,210 | 2.9 | 7.7 | 70,926 |
| | 2 | 6 | VPG | 192,545 | 2.6 | 6.8 | 74,819 |
| | 3 | 4 | DPN | 177,993 | 2.4 | 5.9 | 75,657 |
| | 4 | 2 | A2C | 161,215 | 3.0 | 6.9 | 53,947 |
| | 5 | 5 | QL | 119,337 | 2.7 | 5.4 | 44,518 |
| | 6 | 3 | DQN | 90,500 | 1.1 | 6.4 | 80,133 |

Table 3: Auctions with six items on offer: Bidder results by auction type over 100,000 episodes, ranked by total payoff in each auction type.

| | **Bidder** | | | **Payoff** | | **Cost** | |
|---|---|---|---|---|---|---|---|
| **Auction type** | **Rank** | **ID** | **Type** | **Total** | **Mean** | **Mean** | **Items won** |
| Discretionary Price (DP) | 1 | 1 | PPO | 453,766 | 3.7 | 2.1 | 123,852 |
| | 2 | 2 | A2C | 366,916 | 3.9 | 2.0 | 94,967 |
| | 3 | 6 | VPG | 286,966 | 3.6 | 2.4 | 80,754 |
| | 4 | 4 | DPN | 269,956 | 3.3 | 2.7 | 82,612 |
| | 5 | 5 | QL | 253,881 | 2.7 | 2.8 | 92,859 |
| | 6 | 3 | DQN | 139,945 | 1.1 | 4.0 | 124,956 |
| Generalised Second-price (GSP) | 1 | 1 | PPO | 463,650 | 3.8 | 2.9 | 122,098 |
| | 2 | 2 | A2C | 358,361 | 4.0 | 2.3 | 89,536 |
| | 3 | 6 | VPG | 309,214 | 3.5 | 4.1 | 88,471 |
| | 4 | 3 | DQN | 290,137 | 2.0 | 4.0 | 142,628 |
| | 5 | 5 | QL | 262,489 | 3.0 | 3.0 | 88,643 |
| | 6 | 4 | DPN | 245,347 | 3.6 | 2.7 | 68,624 |
| Uniform price(UP) | 1 | 1 | PPO | 448,183 | 4.0 | 6.6 | 113,126 |
| | 2 | 2 | A2C | 437,020 | 3.9 | 5.7 | 112,489 |
| | 3 | 6 | VPG | 378,679 | 4.0 | 4.9 | 93,767 |
| | 4 | 3 | DQN | 339,120 | 2.9 | 6.7 | 116,875 |
| | 5 | 4 | DPN | 329,467 | 3.7 | 4.6 | 90,033 |
| | 6 | 5 | QL | 285,283 | 3.9 | 4.2 | 73,710 |

Table 4: Auctions with eight items on offer: Bidder results by auction type over 100,000 episodes, ranked by total payoff in each auction type.

| | **Bidder** | | | **Payoff** | | **Cost** | |
|---|---|---|---|---|---|---|---|
| **Auction type** | **Rank** | **ID** | **Type** | **Total** | **Mean** | **Mean** | **Items won** |
| Discretionary Price (DP) | 1 | 1 | PPO | 691,272 | 4.1 | 1.0 | 168,251 |
| | 2 | 2 | A2C | 649,851 | 4.2 | 1.0 | 154,506 |
| | 3 | 5 | QL | 415,899 | 3.6 | 1.5 | 116,527 |
| | 4 | 6 | VPG | 357,015 | 3.3 | 2.5 | 107,338 |
| | 5 | 3 | DQN | 332,151 | 1.7 | 2.8 | 190,720 |
| | 6 | 4 | DPN | 226,261 | 3.6 | 1.5 | 62,658 |
| Generalised Second-price (GSP) | 1 | 1 | PPO | 711,596 | 4.3 | 1.0 | 164,451 |
| | 2 | 2 | A2C | 701,923 | 4.3 | 1.0 | 164,603 |
| | 3 | 3 | DQN | 576,871 | 3.1 | 2.8 | 188,601 |
| | 4 | 5 | QL | 450,890 | 4.0 | 1.5 | 111,465 |
| | 5 | 4 | DPN | 389,694 | 4.6 | 1.4 | 84,582 |
| | 6 | 6 | VPG | 262,028 | 3.0 | 1.7 | 86,298 |
| Uniform price (UP) | 1 | 1 | PPO | 725,297 | 4.5 | 5.6 | 160,439 |
| | 2 | 2 | A2C | 715,136 | 4.6 | 5.5 | 156,397 |
| | 3 | 3 | DQN | 671,326 | 3.7 | 4.8 | 182,148 |
| | 4 | 4 | DPN | 547,658 | 4.9 | 5.2 | 111,755 |
| | 5 | 6 | VPG | 532,158 | 5.3 | 3.0 | 101,297 |
| | 6 | 5 | QL | 389,310 | 4.4 | 2.9 | 87,964 |

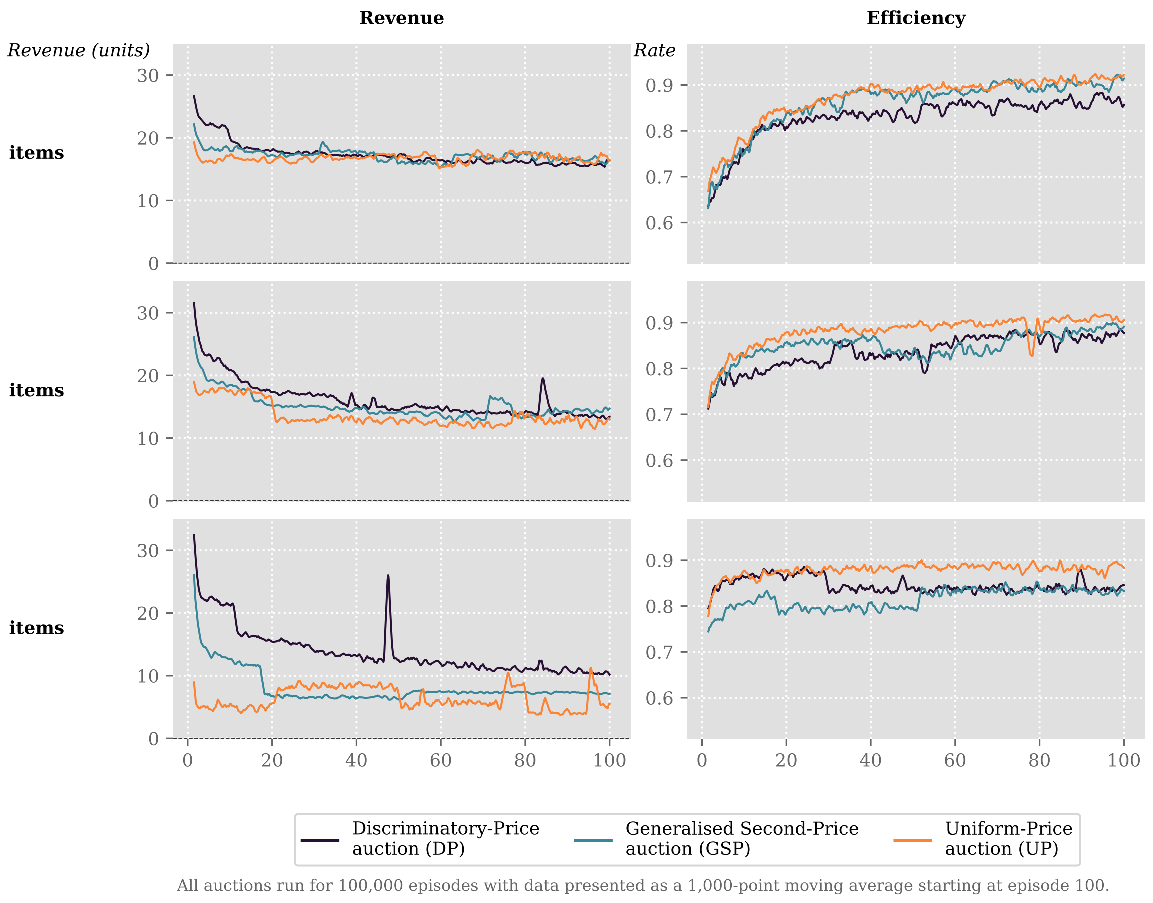


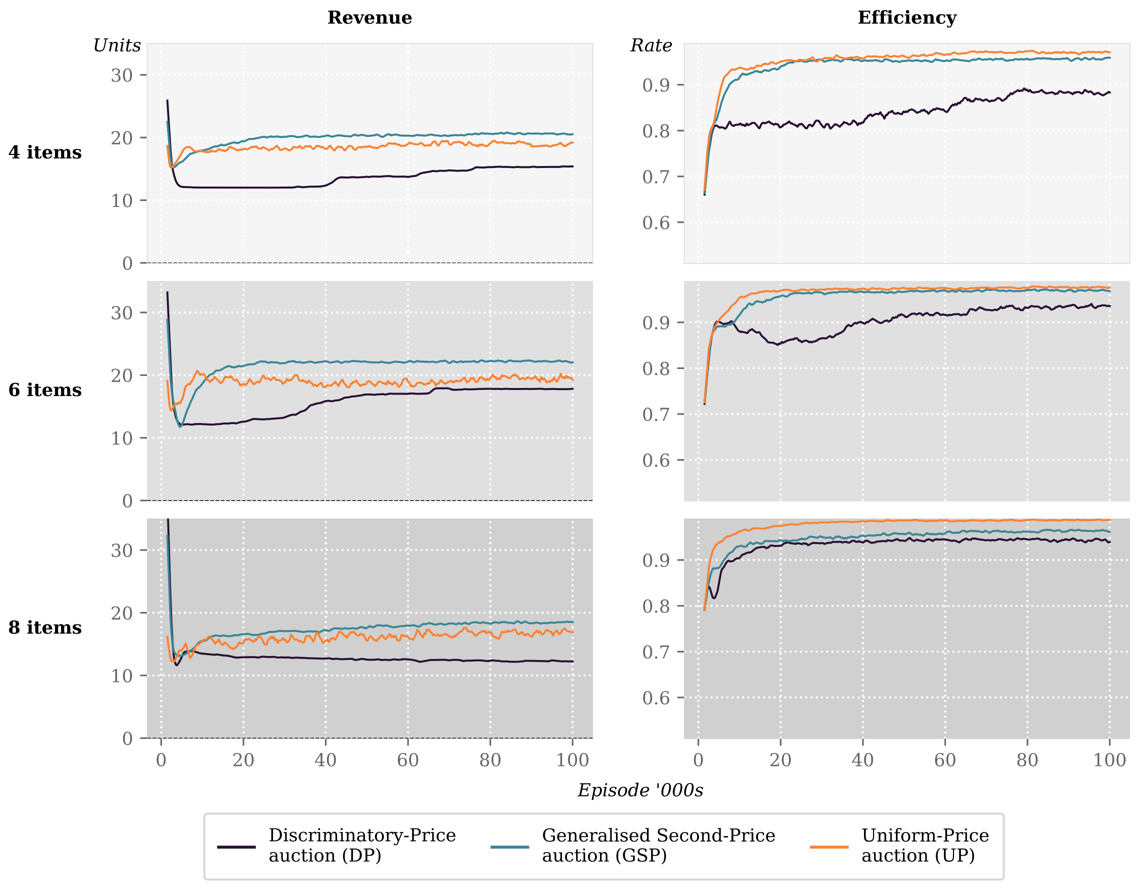
Figure 4: Comparison of auction performance metrics for the Discriminatory Price (DP), Generalised Second Price (GSP), and Uniform Price (UP) auctions with four, six and eight items on offer. Each auction comprised six bidders (six different reinforcement learning algorithms), each demanding two items.

Table 5: Auction revenue and efficiency by items on offer and auction type over 100,000 episodes

| **Items** | **Auction type** | **Revenue** | | | | **Efficiency** | | |
|---|---|---|---|---|---|---|---|---|
| | | **Total** | **Mean** | **Min** | **Max** | **Mean** | **Min** | **Max** |
| 4 | Discretionary Price (DP) | 1,738,683 | 17.39 | 8 | 36 | 0.83 | 0.04 | 1.00 |
| | Generalised Second-price (GSP) | 1,721,494 | 17.21 | 3 | 36 | 0.86 | 0.04 | 1.00 |
| | Uniform price (UP) | 1,679,288 | 16.79 | 0 | 36 | 0.87 | 0.00 | 1.00 |
| 6 | Discretionary Price (DP) | 1,628,580 | 16.29 | 7 | 53 | 0.84 | 0.15 | 1.00 |
| | Generalised Second-price (GSP) | 1,520,078 | 15.2 | 4 | 52 | 0.85 | 0.22 | 1.00 |
| | Uniform price (UP) | 1,366,080 | 13.66 | 0 | 48 | 0.88 | 0.20 | 1.00 |
| 8 | Discretionary Price (DP) | 1,400,719 | 14.01 | 3 | 63 | 0.84 | 0.34 | 1.00 |
| | Generalised Second-price (GSP) | 831,480 | 8.31 | 1 | 65 | 0.81 | 0.21 | 1.00 |
| | Uniform price (UP) | 642,584 | 6.43 | 0 | 56 | 0.88 | 0.30 | 1.00 |

Table 6: PPO: Auction revenue and efficiency by items on offer and auction type over 100,000 episodes

| **Items** | **Auction type** | **Revenue** | | | | **Efficiency** | | |
|---|---|---|---|---|---|---|---|---|
| | | **Total** | **Mean** | **Min** | **Max** | **Mean** | **Min** | **Max** |
| 4 | Discretionary Price (DP) | 1,354,492 | 13.68 | 9 | 35 | 0.88 | 0.22 | 1.00 |
| | Generalised Second-price (GSP) | 1,970,021 | 19.90 | 5 | 33 | 0.97 | 0.06 | 1.00 |
| | Uniform price (UP) | 1,833,460 | 18.52 | 4 | 32 | 0.98 | 0.35 | 1.00 |
| 6 | Discretionary Price (DP) | 1,567,221 | 15.83 | 7 | 47 | 0.95 | 0.36 | 1.00 |
| | Generalised Second-price (GSP) | 2,123,696 | 21.45 | 7 | 42 | 0.99 | 0.32 | 1.00 |
| | Uniform price (UP) | 1,875,216 | 18.94 | 0 | 48 | 0.99 | 0.43 | 1.00 |
| 8 | Discretionary Price (DP) | 1,267,547 | 12.80 | 6 | 56 | 0.97 | 0.50 | 1.00 |
| | Generalised Second-price (GSP) | 1,723,650 | 17.41 | 9 | 56 | 0.98 | 0.50 | 1.00 |
| | Uniform price (UP) | 1,585,248 | 16.01 | 0 | 56 | 0.99 | 0.60 | 1.00 |



All auctions run for 100,000 episodes with data presented as a 1,000-point moving average starting at episode 1,000.

Figure 5: PPO: Comparison of auction performance metrics for the Discriminatory Price (DP), Generalised Second Price (GSP), and Uniform Price (UP) auctions with four, six and eight items on offer. Each auction comprised six bidders (six PPO algorithms), each demanding two items.

## References

Ausubel, Lawrence M, Peter Cramton, Marek Pycia, Marzena Rostek, and Marek Weretka (2014) “Demand reduction and inefficiency in multi-unit auctions,” *The Review of Economic Studies*, 81 (4), 1366–1400.

Back, Kerry and Jaime F Zender (1993) “Auctions of divisible goods: On the rationale for the treasury experiment,” *The Review of Financial Studies*, 6 (4), 733–764.

Banchio, Martino and Andrzej Skrzypacz (2022) “Artificial intelligence and auction design,” in *Proceedings of the 23rd ACM Conference on Economics and Computation*, 30–31.

Bellman, Richard (1957) *Dynamic Programming*: Princeton University Press.

Bichler, Martin and Jacob K Goeree (2017) *Handbook of spectrum auction design*: Cambridge University Press.

Cheng, Wen-Chung (Andy), Zhen Ni, and Xiangnan Zhong (2021) “Experimental Evaluation of Proximal Policy Optimization and Advantage Actor-Critic RL Algorithms using MiniGrid Environment,” in *34th Florida Conference on Recent Advances in Robotics (FCRAR 2021)*, Florida Atlantic University, Boca Raton, United States.

Corecco, Samuel, Giorgia Adorni, and Luca Maria Gambardella (2023) “Proximal Policy Optimization-Based Reinforcement Learning and Hybrid Approaches to Explore the Cross Array Task Optimal Solution,” *Machine Learning and Knowledge Extraction*, 5 (4), 1660–1679, 10.3390/make5040082.

Edelman, Benjamin, Michael Ostrovsky, and Michael Schwarz (2007) “Internet advertising and the generalized second-price auction: Selling billions of dollars worth of keywords,” *American economic review*, 97 (1), 242–259.

Engelbrecht-Wiggans, Richard and Charles M Kahn (1998) “Multi-unit auctions with uniform prices,” *Economic theory*, 12, 227–258.

Hailu, Atakelty and Sophie Thoyer (2007) “Designing Multi-unit Multiple Bid Auctions: An Agent-based Computational Model of Uniform, Discriminatory and Generalised Vickrey Auctions,” *Economic Record*, 83, S57–S72.

Huang, Shengyi, Anssi Kanervisto, Antonin Raffin, Weixun Wang, Santiago Ontañón, and Rousslan Fernand Julien Dossa (2022) “A2C is a special case of PPO.”

Khezr, Peyman and Anne Cumpston (2022) "A review of multiunit auctions with homogeneous goods," *Journal of Economic Surveys*, 36 (4), 1225–1247.

Khezr, Peyman and Flavio M Menezes (2017) "A new characterization of equilibrium in multiple-object uniform-price auctions," *Economics letters*, 157, 53–55.

Khezr, Peyman, Vijay Mohan, and Lionel Page (2024) "Strategic Bidding in Knapsack Auctions," *arXiv preprint arXiv:2403.07928*.

Krishna, Vijay (2009) *Auction Theory, 2nd Edition*, San Diego: Academic Press.

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver et al. (2015) "Human-level control through deep reinforcement learning," *Nature*, 518 (7540), 529–533, 10.1038/nature14236.

Morgenstern, Jamie and Tim Roughgarden (2016) "Learning simple auctions," in *Conference on Learning Theory*, 1298–1318, PMLR.

Raffin, Antonin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann (2021) "Stable-Baselines3: Reliable Reinforcement Learning Implementations," *Journal of Machine Learning Research*, 22 (268), 1–8, `http://jmlr.org/papers/v22/20-1364.html`.

Roughgarden, Tim, Vasilis Syrgkanis, and Eva Tardos (2017) "The price of anarchy in auctions," *Journal of Artificial Intelligence Research*, 59, 59–101.

Schulman, John, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov (2017) "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*.

Silver, David, Satinder Singh, Doina Precup, and Richard S Sutton (2021) "Reward is enough," *Artificial Intelligence*, 299, 103535.

Sutton, Richard S and Andrew G Barto (2018) *Reinforcement Learning: An Introduction*: MIT Press, 2nd edition.

Tijsma, Arryon D., Madalina M. Drugan, and Marco A. Wiering (2016) "Comparing exploration strategies for Q-learning in random stochastic mazes," in *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, 1–8, 10.1109/SSCI.2016.7849366.

Towers, Mark, Jordan K. Terry, Ariel Kwiatkowski et al. (2023) "Gymnasium," March, 10.5281/zenodo.8127026.

Vickrey, William (1961) "Counterspeculation, auctions, and competitive sealed tenders," *The Journal of finance*, 16 (1), 8–37.

Watkins, Christopher John Cornish Hellaby (1989) *Learning from Delayed Rewards* Ph.D. dissertation, King's College, Cambridge.

Williams, Ronald J. (1992) "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, 8 (3), 229–256, 10.1007/BF00992696.

Zang, Yifan, Jinmin He, Kai Li, Haobo Fu, Qiang Fu, and Junliang Xing (2023) "Sequential Cooperative Multi-Agent Reinforcement Learning," in *Adaptive Agents and Multi-Agent Systems*, `https://api.semanticscholar.org/CorpusID:258845637`.

Zhu, Changxi, Mehdi Dastani, and Shihan Wang (2024) "A survey of multi-agent deep reinforcement learning with communication," *Autonomous Agents and Multi-Agent Systems*, 38 (1), 4, 10.1007/s10458-023-09633-6.

# Artificial Intelligence for Multi-Unit Auction design

Peyman Khezr

RMIT

## Introduction

- Multi-unit auctions are fundamental for many markets
  - e.g. Treasury, Wholesale electricity, Emissions permit, Spectrum, etc.
- Understanding bidding behavior in these auctions remains an ongoing challenge
- Limited theoretical insights
- Lack of available data (bidder's side)
- Experiments are helpful but with limitations

- Economists and computer scientists study multi-unit auctions using various methods
- Auction theory, data analytics, experimental approaches, and simulations
- Inability to comparatively assess different multi-unit auctions under similar conditions
- Traditional simulation methods require existing model or theoretical prediction
- We explore Reinforcement Learning as a model-free method to simulate behavior in multi-unit auctions

## Previous literature

- Banchio and Skrzypacz (2022): Q-learning, single item
- Hailu and Thoyer (2007): agent base, multi-unit
- Khezr et al. (2024): Q-learning, knapsack auction
- Silver et al., (2021): RL is effective in simulating learning processes in uncertain environments

## Our approach

In this paper:

- Three main RL-based approaches

1. Q-Learning
2. Policy Gradient
3. Actor-Critic

We develop six algorithms based on the above approaches and simulate the following three auctions:

1. Discriminatory price (DP)
2. Generalised Second-price (GSP)
3. Uniform-price (UP)

## The three approaches

- Q-learning: uses Q-tables to ...
- Policy gradient: $\nabla_\theta E[R|\pi_\theta]$
- Actor-Critic

Six algorithms:

1. Q-learning
2. Deep Q-learning Network
3. Vanilla Policy Gradient
4. Deep Policy Gradient
5. Actor-Critic (A2C)
6. Proximal Policy Optimization

## Comparison

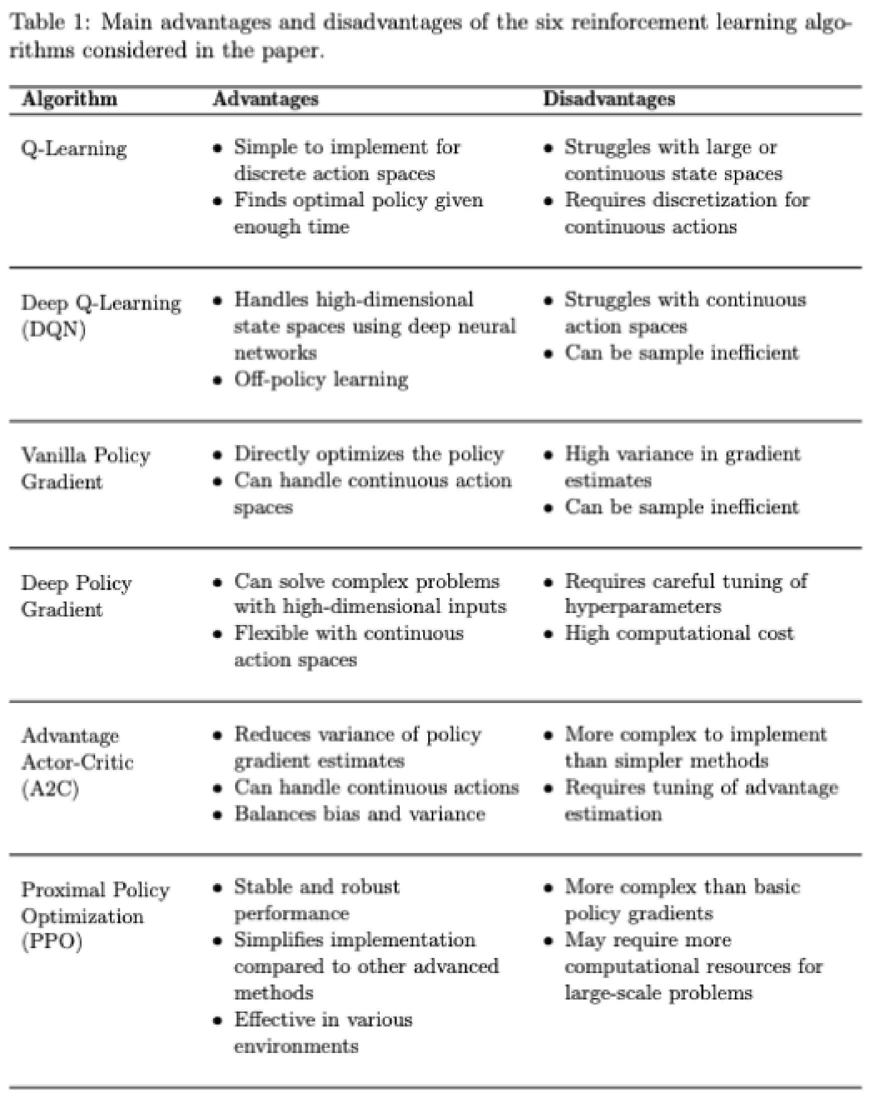


Table 1: Main advantages and disadvantages of the six reinforcement learning algorithms considered in the paper.

| Algorithm | Advantages | Disadvantages |
|---|---|---|
| Q-Learning | • Simple to implement for discrete action spaces<br>• Finds optimal policy given enough time | • Struggles with large or continuous state spaces<br>• Requires discretization for continuous actions |
| Deep Q-Learning (DQN) | • Handles high-dimensional state spaces using deep neural networks<br>• Off-policy learning | • Struggles with continuous action spaces<br>• Can be sample inefficient |
| Vanilla Policy Gradient | • Directly optimizes the policy<br>• Can handle continuous action spaces | • High variance in gradient estimates<br>• Can be sample inefficient |
| Deep Policy Gradient | • Can solve complex problems with high-dimensional inputs<br>• Flexible with continuous action spaces | • Requires careful tuning of hyperparameters<br>• High computational cost |
| Advantage Actor-Critic (A2C) | • Reduces variance of policy gradient estimates<br>• Can handle continuous actions<br>• Balances bias and variance | • More complex to implement than simpler methods<br>• Requires tuning of advantage estimation |
| Proximal Policy Optimization (PPO) | • Stable and robust performance<br>• Simplifies implementation compared to other advanced methods<br>• Effective in various environments | • More complex than basic policy gradients<br>• May require more computational resources for large-scale problems |

## Simulations

- Learning ratio: $R = \frac{v}{k} - \frac{B}{k}$
- Revenue: some of the payments by packed agents
- Efficiency $E = S - C$; $S$ achievable, $C$ auction surplus

## AI and Reinforcement Learning

- Experiments have limitations
- Agents that follow similar learning patterns
- Allow creation of millions of data points
- Has several valuable lessons about auction performance
- Real-world auctions with automated algorithms (bots)

In this paper we use Reinforcement Learning:

- Shown to be an effective way of learning
- We use Q-Learning algorithm, a model free learning

## Conclusions

Thank you!