Learning to Mitigate AI Collusion on Economic Platform

Gianluca Brero*
Harvard University
Cambridge, Massachusetts, United States
gbrero@g.harvard.edu

Nicolas Lepore*
Harvard University
Cambridge, Massachusetts, United States
nlepore33@gmail.com

Eric Mibuari*
Harvard University
Cambridge, Massachusetts, United States
mibuari@g.harvard.edu

David C. Parkes*
Harvard University
Cambridge, Massachusetts, United States
parkes@g.harvard.edu

ABSTRACT

Algorithmic pricing on online e-commerce platforms raises the concern of tacit collusion, where reinforcement learning algorithms learn to set collusive prices in a decentralized manner and through nothing more than profit feedback. This raises the question as to whether collusive pricing can be prevented through the design of suitable "buy boxes," i.e., through the design of the rules that govern the elements of e-commerce sites that promote particular products and prices to consumers. In previous work, Johnson et al. [17] designed hand-crafted buy box rules that use demand-steering, based on the history of pricing by sellers, to prevent collusive behavior. Although effective against price collusion, these rules effect this by imposing severe restrictions on consumer choice and consumer welfare. In this paper, we demonstrate that reinforcement learning (RL) can also be used by platforms to learn buy box rules that are effective in preventing collusion by RL sellers, and to do so without reducing consumer choice. For this, we adopt the methodology of *Stackelberg MDPs*, and demonstrate success in learning robust rules that continue to provide high consumer welfare together with sellers employing different behavior models or having out-of-distribution costs for goods.

KEYWORDS

Multi-agent Systems; Reinforcement Learning; Platform Design

1 INTRODUCTION

The last decade has witnessed a dramatic shift of trading from retailers to online e-commerce platforms such as Amazon and Alibaba. In these platforms, sellers are increasingly using algorithms to set prices for their products. On one hand, algorithmic pricing can be beneficial for market efficiency, enabling sellers to quickly react to market changes and also in enabling price competition. At the same time, concerns that algorithmic pricing may facilitate collusive behaviors have been raised by many government authorities, including the U.S. Federal Trade Commission (FTC) [31] and the European Commission [30]. These concerns are also finding support in computational and empirical studies. Calvano et al. [11] study pricing agents in a simulated platform economy, and show in simulation that commonly used reinforcement-learning algorithms will learn to initiate and sustain collusive behaviors. Assad et al. [4] also provide empirical support for algorithmic collusion in a study of Germany's retail gas stations, showing an association between the delegation of pricing to algorithms and an increase in the markup of stations' prices over gas cost. As highlighted by Calvano et al. [10], these collusive behaviors are unlikely to be a violation of antitrust law, as they are learned responses to profit signals and not the result of explicit agreements.

As a solution to this problem, one can try to prevent algorithmic collusion by introducing suitable rules by which platforms can choose which sellers to promote to buyers. For example, might it be possible to promote competition, even in the face of sellers with reinforcement learning (RL) algorithms for pricing, by choosing a suitable set of rules that govern how different products are ranked or displayed to consumers? A leading example is the *Amazon Buy Box* algorithm, which determines, for a given consumer search, which products and prices to highlight to a consumer. This has been noticed by Johnson et al. [17], who have designed and studied hand-crafted platform interventions that implement buy-box policies based on price choices of sellers in order to hinder collusion between RL algorithms. At the same time, the rules that they study also introduce undesirable effects in the way the market operates, by limiting consumers to a single seller, and there remains potential for more effective interventions.

The novel approach that we take in the present paper is to understand whether machine learning can also be used defensively by a platform, with the platform learning buy box rules that are effective in mediating the behavior of RL-based sellers. We demonstrate for the first time how machine learning, specifically RL, can be used to automatically design platform rules that promote consumer welfare and prevent collusive pricing in an ecosystem where sellers are also using RL to set prices.

We model the interaction between the platform and sellers as a *Stackelberg game* [16], where the leader is the platform designer and sets the platform rules and the sellers respond, in our case by using RL to set prices given these rules. To solve for an optimal platform rule, we use a variation on the *Stackelberg Markov decision process* (MDP) methodology [6]. This Stackelberg MDP technique carefully defines the episode structure of an MDP such that the RL algorithm representing the leader will learn to optimize its reward (here, consumer surplus) given that its rules cause re-equilibration on the part of the followers (here, the sellers who make use of Q-learning algorithms to set prices). The Stackelberg MDP framework is well formed as long as the re-equilibration behavior of sellers

*Author order is alphabetical.

Appears at the 1st Workshop on Learning with Strategic Agents (LSA 2022). Held as part of the Workshops at the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), N. Bishop, M. Han, L. Tran-Thanh, H. Xu, H. Zhang (chairs), May 9–10, 2022, Online.

can be modeled through Markovian dynamics, as is the case with Q-learning. We introduce the class of *threshold platform rules* and formally show that it contains rules that approximately maximize consumer surplus when sellers play according to a subgame perfect equilbrium of the game induced by the platform rules. We show that this Stackelberg MDP framework and associated RL methodology can indeed be used to derive effective platform policies that outperform the handcrafted rules suggested by Johnson et al. [17]. We also show that the learned platform rules should continue to be effective when market conditions change, for example as the result of a change to the cost structure of sellers. Finally, we test the framework in settings where the sellers' restart their learning dynamics randomly and asynchronously, this resulting in highly non-stationary dynamics. We demonstrate that the Stackelberg MDP framework and implied learning methodology remains effective even in these scenarios.

Further related work. Abada and Lambin [1] have highlighted collusive behavior between RL algorithms for selling and buying electric power, and consider the use of machine learning as a mitigation, in their case learning a regulator agent that can sell and buy energy. But this approach was ultimately unsuccessful in their setting, leading to lower welfare than that one obtained under collusive behavior. A key difference is that we work in the Stackelberg framing, and by leveraging commitment power are able to derive successful interventions.

There is a broader agenda of *automated mechanism design* (AMD) [12], which proposes to use algorithms rather than analytical methods for the design of incentive mechanisms. Especially relevant is work on *empirical mechanism design* [3, 8, 32], which applies the method of *empirical game theory* to search for the equilibria of induced games by building out a suitable set of candidate strategies [18, 19, 34]. Bünz et al. [9] also use computational methods for the design of combinatorial auction rules while considering bidders' strategic response. Recent work has also studied RL methods for the design of incentive-compatible *sequential price mechanisms* (SPMs) [7]. In a followup work, Brero et al. [6] consider SPMs with an initial message passing round, and introduce the *Stackelberg MDP framework* to design SPMs along with bidders who are modeled through no-regret learning.

Shen et al. [26], Tang [29], Zheng et al. [35] also make use of RL to optimize different market mechanisms (including matching markets, internet advertising, and designing tax policies) under strategic agents' responses. Unlike our work, these methods do not leverage the designer's commitment power or consider the Stackelberg structure of the induced game with market participants. There is also growing interest in the use of deep learning for optimal economic design [13–15, 20, 22, 24, 25, 27, 28, 33].

2 PRELIMINARIES

2.1 Economic Platform Model

We follow Johnson et al. [17], and consider sellers $\mathcal{N} = \{1, \ldots, n\}$, each of whom sells a differentiated product on an economic platform. Also, each seller has the same *marginal cost*, $c > 0$, for producing one unit of its product for sale. Sellers interact with each other repeatedly over time in setting prices and selling goods to consumers. At each time period, $t = 0, 1, \ldots$, each seller i observes all past prices and platform behavior, and sets a *price* $p_{i,t} \geq 0$ for its product. We let $p_t = (p_{1,t}, \ldots, p_{n,t})$ denote a generic price profile quoted at time t.

There is also a platform, which acts to set the rules of a buy box that governs which sellers are displayed to buys. In each period t, the effect of the platform's buy box rules is to display some set $\mathcal{N}_t \subseteq \mathcal{N}$ of the sellers to consumers. Consumers in this period can only buy from this set of sellers so that those outside the set forfeit sales. There is also an outside option, indexed by 0, which provides each consumer with a fallback choice that may be more desirable than the products and prices corresponding to displayed sellers.

Competition between sellers for consumer demand is modeled through the standard *logit model* for consumer choice. The product of seller i has *quality index* $\alpha_i > 0$, this providing horizontal differentiation across products, and the outside good has quality index $\alpha_0 > 0$.

Products are offered to a new unit mass of consumers at each period t. Each of these consumers wishes to buy at most one product. To obtain a consumer's utility for each product, we first draw $n + 1$ consumer-specific variables $\zeta_0, \zeta_1, \ldots \zeta_n$, independently from a type I extreme value distribution with *scale parameter* $\mu > 0$, for each of the n products and the outside option. We then let the consumer's utility be $\alpha_i + \zeta_i - p_{i,t}$ for product i, and $\alpha_0 + \zeta_0$ for the outside option. Considering the continuum of consumers, the effect is that firm $i \in \mathcal{N}_t$ receives fraction $D_i(p_t; \mathcal{N}_t)$ of demand in period t, given by

$$D_i(p_t; \mathcal{N}_t) = \frac{\exp\left(\frac{\alpha_i - p_{i,t}}{\mu}\right)}{\sum_{j \in \mathcal{N}_t} \exp\left(\frac{\alpha_j - p_{j,t}}{\mu}\right) + \exp\left(\frac{\alpha_0}{\mu}\right)}. \tag{1}$$

Each firm not in $\mathcal{N}_t$ receives zero demand in period t. From Equation (1), we see that scale parameter $\mu > 0$ serves to control the extent of horizontal differentiation of the products, with no differentiation and perfect substitutes obtained in the limit, as $\mu \to 0$. According to the logit model, the total *consumer surplus* based on this demand system is,

$$U(p_t; \mathcal{N}_t) = \mu \cdot \log \left[\sum_{j \in \mathcal{N}_t} \exp\left(\frac{\alpha_j - p_{j,t}}{\mu}\right) + \exp\left(\frac{\alpha_0}{\mu}\right) \right]. \tag{2}$$

From Equation (2), we see that consumer surplus is maximized for low prices and with all sellers being displayed, so that $\mathcal{N}_t = \mathcal{N}$ and consumers have the complete choice of products. Seller i's *profit* ρ_i in period t is given by its per-unit profit multiplied by demand, i.e.

$$\rho_i(p_t; \mathcal{N}_t) = (p_{i,t} - c) \cdot D_i(p_t; \mathcal{N}_t). \tag{3}$$

As in Calvano et al. [11] and Johnson et al. [17], we only let sellers pick their price from prices on a discretized interval. In defining an upper bound for this interval, it is useful to consider the symmetric *monopoly price* p^m, which is the price that, when set by every seller and assuming that all sellers are displayed, maximizes their total profit, i.e.,

$$p^m \in \operatorname{argmax}_{p \in \mathbb{R}_+} \sum_{i=1}^{n} \rho_i(p \cdot \mathbf{1}_n; \mathcal{N}), \tag{4}$$

where $\mathbf{1}_n = (1, \ldots, 1)$. Following Johnson et al. [17] and Calvano et al. [11], we consider here a symmetric solution to this monopoly pricing problem, where each seller sets the same price for its product. We note that this price is also optimal, in particular, in the case of perfect collusion, where a single seller is able to set the price to offer for each of the set of products. We use this price and the unit cost c to guide the choice as to the interval of prices that are made available to the RL pricing agents.

2.2 Markov Decision Processes

In a single-agent, Markov decision process (MDP), an agent faces a sequential decision problem under uncertainty. At each step t, the agent observes a state variable $s_t \in S$ and chooses an action $a_t \in A$. When taking action a_t in state s_t, the agent obtains a reward $r(s_t, a_t)$, and the environment moves to state s_{t+1} according to a distribution $p_t(s_{t+1}|s_0, \ldots, s_t, a_0, \ldots, a_t)$. When

$$p_t(s_{t+1}|s_0, \ldots, s_t, a_0, \ldots, a_t) = p_t(s_{t+1}|s_t, a_t)$$

we have the Markov property, and when

$$p_t(s_{t+1}|s_t, a_t) = p(s_{t+1}|s_t, a_t)$$

the environment is stationary. Given a Markovian, stationary environment it is without loss of generality to consider policy $\pi : S \to A$ (stationary, and depending only on current state s_t), and we let $\tau = (s_0, a_0, ..., s_T, a_T)$ denote a state-action trajectory determined by executing policy π, and $p_\pi(\tau)$ denote the probability of trajectory τ as induced by the MDP and policy π. The optimal policy π^* maximizes the expected total reward, i.e.,

$$\pi^* \in \text{argmax}_\pi E_{\tau \sim p_\pi(\tau)} \left[\sum_{t=0}^{T} \delta^t r(s_t, a_t) \right], \tag{5}$$

where $\delta \in [0, 1]$ is the discount factor. The time-horizon T can be finite or infinite. In some scenarios, the policy π cannot access states s_t to determine its actions, but only observations o_t sampled from probability distributions $p(o_t|s_t)$. In this case the MDP becomes a *partially-observable MDP* (POMDP).

The existing literature on tacit collusion has assumed *Q-learning* on the part of sellers, which is an RL algorithm that learns an estimate of the *action-value function* $Q^*(s, a)$. This action-value function gives the expected total reward of taking action a at state s and using the optimal policy function in the future. Once $Q^*(s, a)$ has been learned, the optimal policy is

$$\pi^*(s) \in \arg\max_{a \in A} Q^*(s, a). \tag{6}$$

A Q-learning algorithm maintains an $|S| \times |A|$ Q-matrix, Q_t, representing the estimate of $Q^*(s, a)$ at step t. Usually, this matrix is randomly initialized. At each step t, the algorithm takes action a_t that with probability $1 - \varepsilon_t$ is optimal according to its current Q-matrix Q_t, and with probability ε_t chosen uniformly at random from the set of avalilable actions. We call ε_t the *exploration rate*. The entry (s_t, a_t) of Q_t is updated based on feedback via a convex combination of its previous value and the reward $r(s_t, a_t)$ attained from the action plus the discounted value of the state s_{t+1}:

$$Q_{t+1}(s_t, a_t) = (1 - \alpha) Q_t(s_t, a_t) + \alpha [r(s_t, a_t) + \delta \max_{a \in A} Q_t(s_{t+1}, a)]. \tag{7}$$

Parameter $\alpha \in [0, 1]$ is the *learning rate*. When the environment is stationary and Markovian, and under suitable assumptions on the learning rate and exploration rate, the Q-matrix is guaranteed to converge in the limit to the action-value function $Q^*(s, a)$ and the policy to the optimal policy.

Multi-agent Reinforcement Learning. In settings with multiple agents, for example the multiple seller agents in our economic platform model, we can formulate a *multi-agent MDP* [5]. In a multi-agent MDP for n agents we have a set of states S common to all agents, and a set of actions A_i for each agent i. When each agent i picks action $a_{i,t}$ in state s_t, the environment moves to state s_{t+1} according to a distribution $p(s_{t+1}|s_t, a_{1,t}, .., a_{n,t})$ and each agent i obtains a reward $r_i(s_t, a_t)$ which depends on the underlying state and joint actions. To derive optimal sellers' policies in our multi-agent MDP, we follow Calvano et al. [11] and Johnson et al. [17] and assume decentralized learning amongst sellers, modeling each seller through its own Q-learning algorithm. One key challenge is that each agent independently updates its policy, which makes the environment non-stationary from the view point of any single agent, violating the assumptions required for convergence of Q-learning. However, both Calvano et al. [11] and Johnson et al. [17] nearly always obtain convergence in their experiments, and we find the same in our experiments. Moreover, Q-learning is taken in these works as a positive theory for how seller agents might actually work, operationally, in learning to set prices on an e-commerce platform. Checks for convergence are based on policy stability over a defined time horizon as detailed by Johnson et al. [17].

2.3 The Platform Design Problem

To formalize the platform design problem, we model the interaction between the platform, which sets the rules of the buy box, and the sellers as a Stackelberg game. The platform designer is the leader and fixes the platform rules. The sellers are the followers, and play an infinitely repeated game according to these rules. As with Calvano et al. [11], Johnson et al. [17], we model the sellers' behavior through decentralized Q-learning. As a result, the platform design problem considers a kind of *behavioral Stackelberg framework*, with followers modeled through Q-learning rather than playing an equilibrium of the induced game.

The sellers. In this model, we fix the states that comprise the MDP of a seller to include the prices set by all sellers in the last period, i.e., $s_t = p_{t-1}$. We initialize s_0 to be a randomly selected price profile. The action of a seller is modeled as one of m equally-spaced points in the interval ranging from just below the sellers' cost c to just above the monopoly price p^m. At each step $t \geq 0$, each seller i selects a price $p_{i,t}$ and is rewarded by its per-period profit $\rho_i(p_t; \mathcal{N}_t)$, which depends on $p_t = (p_{1,t}, \ldots, p_{n,t})$ and the choice of which sellers $\mathcal{N}_t$ are displayed by the platform.

To formalize the platform design problem, let $\sigma^* = (\sigma_1^*, .., \sigma_n^*)$ denote a strategy profile selected by Q-learning on the part of sellers, in response to the platform rule, and in the long run, after a suitably large number of steps. The platform must decide in each period which sellers to display to consumers. For this, we denote the platform rule as *policy* π, and we adopt for the state of this platform policy the prices quoted by sellers in step t, p_t, so that the

policy uses these prices to select a set of agents to display, with $\mathcal{N}_t$ selected according to $\pi(p_t)$.

Let $p_t^* = \sigma^*(s_t)$ denote a price profile chosen under seller strategies σ^*, i.e., in response to the platform rules, and at some large enough time step t^*, and $\tau^* = (p_{t^*}^*, p_{t^*+1}^*, ..)$ denote a trajectory of these prices. As Q-learning is a random process whose dynamic is affected by the platform rules, price trajectory τ^*, will itself be randomly distributed, and we denote this distribution as $p_\pi(\tau^*)$.

We can now define the behavioral Stackelberg problem facing the platform, which is the problem of finding a policy π that maximizes consumer surplus given the effect of this policy on the induced strategy profile of sellers.

Definition 1 (Behavioral Stackelberg Problem). *Let $CS(\pi)$ denote the expected discounted sum of consumer surplus when sellers follow strategy σ^* forward from period t^*, i.e.,*

$$CS(\pi) = \mathbb{E}_{\tau^* \sim p_\pi(\tau^*)} \left[\sum_{t=t^*}^{\infty} \delta^t U(p_t^*; \pi(p_t^*)) \right]. \tag{8}$$

The rules of an optimal platform are any rules such that $\pi^* \in argmax_\pi CS(\pi)$.

Here, we assume that the platform knows or is able to estimate the consumer surplus $U(p_t^*; \pi(p_t^*))$, so that this is available as reward feedback in learning a suitable platform rule. In practice, we envision that consumer surplus can be estimated from proxies of consumer satisfaction, for example those available from a feedback system provided to consumers, or by directly estimating the parameters of the demand system given the prices and revealed demand of consumers. Last, we note that, given that we model our followers' behavior via Q-learning, each equilibrium price trajectory τ^* consists of price cycles. Thus, we can reformulate 1 using a finite time horizon large-enough to capture these cycles. In our experiments, we will use this alternative formulation together with a discount factor $\delta = 1$.

3 THRESHOLD PLATFORM RULES

In this section, we consider the special class of *threshold platform rules*. These are rules that set a price threshold below which a seller will be displayed, the same threshold for all sellers, and with this threshold depending on the current prices offered by sellers.

Definition 2 (Threshold Platform Rule). *A* threshold platform rule *sets a threshold $\tau(p_t) \geq 0$, for each price profile p_t, such that $\mathcal{N}_t = \{i \in \{1, .., n\} : p_{i,t} \leq \tau(p_t)\}$, i.e., any seller whose price is no greater than the threshold is displayed to consumers.*

Although simple, this class of threshold rules is instructive because it leads to a simple optimality result: there is a threshold rule that makes the market competitive, with all sellers displayed and consumer surplus maximized in the subgame perfect Nash equilibrium (SPE) of the induced continuous pricing game. Even though the pricing behaviors that arise from Q-learning dynamics need not converge to a SPE (and we have only a discrete set of prices), this provides useful theoretical support for the choice we make to adopt this family of threshold platform rules in our experimental work.

Proposition 1. *For any $\epsilon > 0$, there exists a threshold platform rule π such that $CS(\pi) \geq CS(\pi^*) - \sum_t \delta^t \epsilon$ under a subgame perfect Nash equilibrium (SPE) of the infinitely-repeated continuous pricing game induced by platform rule π.*

Proof. For any $\eta > 0$, we study the stage game with continuous prices induced by the threshold platform rule that sets threshold $\tau = c + \eta$ for each price profile p. We show that this stage game has a unique Nash equilibrium in which each seller sets a price $p_i = p^*$, for some $p^* \in (c, c+\eta]$. Given this, we have that every seller pricing at $p^* \in (c, c + \eta]$ in every period is a SPE of the infinitely repeated game, since this is an open-loop Nash equilibrium profile (and thus SPE by the single-deviation principle). Moreover (as it is a continuous function over price profiles) the consumer surplus comes arbitrarily close, for a small enough $\eta > 0$, to the maximum consumer surplus, which coincides with every seller pricing at cost.

Left to prove is that every seller pricing at p^* is a Nash eq (NE) of the stage game. First, pricing $p_i > \tau(p_t) = c + \eta$ provides zero profit to a seller because the seller is not part of the displayed set of sellers. Similarly, pricing $p_i = c$ provides zero profit. Now dropping the time period t, because we study a generic stage game, and considering prices $p = (p_1, \ldots, p_n) \in \times_{i=1}^n (c, c + \eta]$, so that all sellers are displayed, and with seller profit $\rho_i(p; \mathcal{N})$ for price profile p, we have

$$\frac{\partial}{\partial p_i} \rho_i(p; \mathcal{N}) = -(p_i - c) \frac{D_i(p; \mathcal{N})(1 - D_i(p; \mathcal{N}))}{\mu} + D_i(p; \mathcal{N}).$$

By first-order optimality conditions, we have $\frac{\partial}{\partial p_i} \rho_i(p; \mathcal{N}) = 0$, for each i, only when $p_i = \hat{p}$, for some $\hat{p} > c$ [2]. Furthermore, $\rho_i(p_i, p_{-i}; \mathcal{N})$ is concave. Thus, if $c + \eta \geq \hat{p}$, there is only one Nash equilibrium of the stage game, where each seller i sets price $p^* = \hat{p} \leq c+\eta$. On the other hand, if $c+\eta < \hat{p}$, we have that $\rho_i(p; \mathcal{N})$ is strictly increasing when $p \in \times_{i=1}^n (c, c + \eta]$, and there is a unique Nash equilibrium where each seller quotes price $p^* = c + \eta$. □

Although instructive, the platform intervention used in Proposition 1 is severe, as it only displays products with prices close to the sellers' costs. In particular, this platform rule is fragile, and would lead to market failure if these costs vary. By letting the threshold τ also vary with the price profile p_t, we can hope for milder interventions that still mitigate collusion but remain robust to variations in the costs faced by sellers in the marketplace. We will study this effect in our experiments, showing this additional robustness by comparing rules that are restricted to using the same threshold for all price states with those that can choose a different threshold for different states. In the next section, we show how optimal platforms in this larger class can be identified as the solution to a suitably formulated RL problem.

4 LEARNING OPTIMAL PLATFORM RULES

In this section, we formulate the platform design problem through the *Stackelberg MDP* framework [6]. This creates a suitably defined MDP in which the optimal policy maximizes our platform design objective (8), solving the behavioral Stackelberg problem (Definition 1).

Definition 3 (Stackelberg MDP for platform design). *The* Stackelberg MDP for platform design *is a finite-horizon MDP, where each episode has the following two phases:*

(1) *An* equilibrium phase*, consisting of $n_e \geq 1$ steps. In this phase, each state s_t includes the step counter t, the sellers' current Q-matrices, and the prices p_t quoted by the agents. Policy actions determine the set of agents displayed (in their more general version, $a_t = \mathcal{N}_t$). State transitions are determined by Q-learning, where each seller i updates its Q-matrix after being rewarded by $\rho_i(p_t; \mathcal{N}_t)$. The policy has zero reward in every time step ($r(s_t, a_t) = 0$, for $t \leq n_e$).*
(2) *A* reward phase*, consisting of $n_r \geq 1$ steps, each with the same actions and states as the equilibrium steps. The reward phase differs in two ways. First, the Q-matrices of sellers are not updated, and second, the platform policy now receives a non-zero reward, and this is set in each step to be equal to the consumer surplus in that step ($r(s_t, a_t) = U(p_t; \mathcal{N}_t)$, for $t > n_e$).*

This Stackelberg MDP formulation is an adaptation of the one that had been earlier provided by Brero et al. [6] to learn sequential price mechanisms (SPMs) in the presence of communication from bidders. Here, our stage games replace SPMs, and the followers respond through Q-learning dynamics rather than no-regret algorithms. Following Brero et al. [6], we show the Stackelberg MDP formulation is well-founded by showing that an optimal policy will also solve the Behavioral Stackelberg design problem of Definition 1. Specifically, when the number of reward steps n_r is large enough and when $n_e \geq t^*$, the optimal policy, denoted $\pi^*_{n_e,n_r}$, for the Stackelberg MDP with n_e equilibrium and n_r reward steps maximizes the objective in Equation (8).

Proposition 2. *The optimal Stackelberg MDP policy $\pi^*_{n_e,n_r}$, for an equilibrium phase with $n_e \geq 1$ steps and a reward phase with $n_r \geq 1$ steps, maximizes $CS(\pi)$, for seller behavior induced after n_e steps, in the limit for $n_r \to \infty$.*

Proof. Let $\tau \sim p_\pi(\tau)$ denote a generic trajectory determined by executing policy π in the Stackelberg MDP environment. We have

$$\pi^*_{n_e,n_r} \in \text{argmax}_\pi \mathbb{E}_{\tau \sim p_\pi(\tau)} \left[\sum_{t=n_e+1}^{n_e+n_r} \delta^t r(s_t, a_t) \right], \tag{9}$$

recognizing $r(s_t, a_t) = 0$ if $t \leq n_e$. After replacing $r(s_t, a_t)$ with $U(p_t; \pi(p_t))$, we can rewrite the objective in (9) as

$$\mathbb{E}_{\tau \sim p_\pi(\tau)} \left[\sum_{t=n_e+1}^{n_e+n_r} \delta^t U(p_t; \pi(p_t)) \right],$$

which approximates (8) as n_r grows to infinity. □

In practice, n_r does not need to go to infinity, but rather be large enough to capture the discounted reward associated with the pricing policies of sellers induced by the platform's policy. In an extreme case, when the platform policy and seller policy are both deterministic, and in the absence of cycles, then a single time step is sufficient for the reward phase given the logit demand model and continuum model of consumers.

As in Brero et al. [6], we work with a partially-observable variation on the Stackelberg MDP, since the MDP state includes the Q-matrix of each seller, which is private to the sellers. Rather, the platform can only observe the price profile, p_t. Brero et al. [6] handle this additional challenge in their setting by training the leader policy via an *actor-critic* deep RL algorithm, and leveraging the paradigm of *centralized training and decentralized execution* [21, e.g.,]. In particular, they give the critic network, which estimates the sum of rewards until the end of the episode, access so the full state during training. Only the actor network, which represents the policy, is restricted to the partial state information. This helps to stabilize training while still providing a policy that makes use of only observable state information.

Here, we want to study the use of the Stackelberg MDP framework to train useful leader policies "in the wild," where the learning algorithm of the platform can only access the kind of information that an economic platform would expect to have in regard to deployed seller algorithms. For this reason, we deviate from Brero et al. [6], and do not allow even the critic network access to the Q-matrix of a seller, since this is internal to a seller agent and private.[1] In particular, we make use of actor-critic networks (and the A2C algorithm), but restrict both the critic- and actor-networks to access only the price profile p_t quoted by the sellers.

We see in experiments that the deep RL algorithms used for platform learning are nevertheless able to derive optimal platform policies, when only giving the critic network access to price profiles, along with a binary flag determining whether the environment is in an equilibrium or reward phase. We allow this binary flag because when used "in the wild," the learning system can control for itself when to attend to rewards.[2] Our experiments also show learning robustness in settings where sellers adopt a highly non-stationary behavior, restarting their learning rates randomly and asynchronously.

5 EXPERIMENTAL RESULTS

In this section, we evaluate our learning approach via three main experiments. We first consider training performance in terms of consumer surplus, bench-marking our RL interventions against the ones introduced by Johnson et al. [17]. We find that our learning approach is able to learn optimal leader strategies in the Stackelberg game with the followers across all the seeds we tested, significantly outperforming the other interventions. We then test the robustness of our interventions by evaluating our platforms in environments where sellers have different costs from those assumed during training. Here, we consider a slightly-modified training approach where agents use random prices in the reward phase with some probability. We find that, when using "ad hoc" thresholds that depend on the prices quoted, our platform policies are much more flexible, achieving good performances also under different costs. In the experiments described so far, we let the sellers adopt a behavior model similar to the one used in Calvano et al. [11] and Johnson et al. [17],

[1] Brero et al. [6] have different motivations and situate their work as one of using the Stackelberg MDP framework as an offline computational method for the design of optimal mechanisms, with a mechanism then deployed in the real world once trained. For this reason, they give the critic network during training access to private information of the followers, which in their context is the value of buyers and their internal, no-regret learning state.

[2] Whereas we freeze the Q-matrices during a short reward phase, this could be equally well achieved by running a long-enough equilibrium phases, so that rather than freezing then, the Q-matrices are relatively stable when collecting rewards.

with sellers using the same exploration rate ε_t, and decaying this over time. In our third experiment, we also test the ability of the Stackelberg MDP framework to adapt to a seller behavior model in which each of them restarts its exploration rate randomly and asynchronously, as would be more typical. Even in this scenario, our platform interventions significantly outperform all the baselines, with threshold platform policies using prices approaching optimal performances.

Experimental set-up. As in Calvano et al. [11] and Johnson et al. [17], we consider settings with two pricing agents with cost $c = 1$, quality indexes $\alpha_1 = \alpha_2 = 2$, and $\alpha_0 = 0$, and we set parameter $\mu = 0.25$ to control horizontal differentiation. The seller Q-learning algorithms are also trained using discount factor $\delta = 0.95$, exploration rate $\varepsilon_t = e^{-\beta t}$ with $\beta = 1e-5$, and learning rate $\alpha = 0.15$. We deviate from prior work in considering the choice of one of five possible prices for the action of a seller, these prices ranging from just below the sellers' cost to the monopoly price, whereas previous work gave sellers a choice of fifteen different prices (over a similar range). We need to use a smaller grid in order to satisfy our computational constraints given the more exacting computational work in this paper; earlier work studied the effect of different, hand-designed platform rules, and did not also use RL for the automated design of suitable platform rules. This coarsened price grid allows us to train our policy for 50 million steps in 18 hours using a single core on a Intel(R) Xeon(R) Platinum 8268 CPU @ 2.90GHz machine.

Learning algorithm. To train the platform policy, we start from the A2C algorithm provided by *Stable Baselines3* [23]. Given that our policy is only rewarded at the end of our Stackelberg MDP episodes, we configure A2C so that neural network parameters are only updated after this reward phase. In this way we guarantee that policies inducing desired followers' equilibria are properly rewarded. Furthermore, to reduce variance due to non-deterministic policy behavior throughout episodes, we maintain an observation-action map throughout each episode: When a new observation is encountered during the episode, the policy chooses an action following the default training behavior and stores this new observation-action pair in the map. This reduces variance, so that if the observation is encountered again during the same episode, the policy will take the same action.

As discussed in Section 2.2, the Q-learning as used by seller agents is guaranteed to converge only if the underlying environment is stationary, which is not the case in this multi-agent setting. In their work, Calvano et al. [11] and Johnson et al. [17] find that, despite these potential non-stationarities, Q-learning dynamics converge most of the time. In this earlier work the platform rules are fixed. In order to avoid introducing additional non-stationarity due to the platform rules changing, we assume sellers restart the Q-learning process by re-initializing exploration rates every time the platform rules change (i.e., at the beginning of every Stackelberg MDP episode). We relax this assumption in Section 5.3.

5.1 Learning Performance

In this section, we evaluate the training performance of our policies. For this, we train our policies for 50 million steps in total. We set up the Stackelberg MDP environment using 50k equilibrium steps and 30 reward steps. We selected these hyperparameters using the following criteria: First, we want to make sure that the equilibrium phase of our Stackelberg MDP is long enough such that sellers' dynamics produce best responses to platform policies. At the same time, we want to avoid too long equilibrium phases, as this makes rewards too sparse. With 50k steps we have a good trade-off between these two desiderata. Regarding the reward phase, we want to make sure that this reward is representative of the converged policy reached by Q-learners, and their converged behavior is usually a single price profile or an loop of two or three price profiles. We adopt 30 reward steps to be conservative.

We consider the following kinds of interventions on behalf of the platform designer:

- *No intervention:* Here the platform does not intervene. Sellers are always displayed, no matter the price they quote. To derive this baseline, we run our Q learning dynamics until convergence (as described in Johnson et al. [17]) for each seed and then average the surplus at final strategies.
- *PDP:* We test *price-directed prominence*, the first platform intervention introduced by Johnson et al. [17]. Here, the platform only displays the seller who quotes the lower price (breaking ties at random), thus enhancing competition. As for *no intervention*, we compute the performance of *PDP* by averaging consumer surplus after Q learning dynamics converge.
- *DPDP: Dynamic price-directed prominence* is the second intervention introduced by Johnson et al. [17], which also conditions the choice of the (unique) displayed seller on past prices. Under this intervention, quoting prices equal to cost is a subgame perfect equilibrium of the induced game (under suitable discount factors). As for the previous baselines, we compute the performance of *DPDP* by averaging consumer surplus after Q learning dynamics converge.
- *No state RL:* Here we use our Stackelberg MDP methodology to train a platform policy that does not use prices p_t to determine the threshold at which to admit each seller to the buy box (thus, no state).[3] Here, Q learning is restarted whenever a Stackelberg MDP episodes begins.
- *No Stackelberg state-based RL:* Here we use our Stackelberg MDP methodology to train a platform policy that sets a threshold at which to admit each seller as a function of the price profile quoted by the sellers (thus, state-based). This is the full class of threshold platform rules. As above, Q learning is restarted whenever a Stackelberg MDP episodes begins.

Figure 1 shows the consumer surplus realized under these different interventions. First, we confirm the results of Johnson et al. [17], and see consumer surplus improvements from both *PDP* and *DPDP* compared to *No intervention*, with *DPDP* outperforming *PDP*. Our RL interventions based on our Stackelberg framework dramatically improve consumer surplus, driving it to (approximately, in the state-based scenario) its maximal level (which, in our setting, is approximately 0.94). This is confirmed by the fact that, for both *No state* and *State-based* RL, all sellers are displayed and they almost

[3] This class of policies already includes the approximately optimal policy described in the proof of Proposition 1.

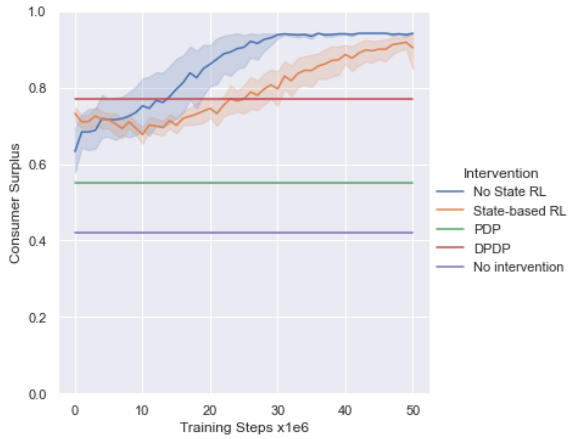

Figure 1: Learning performance of no state RL and state-based RL, and compared with different baselines. The results are averaged over 10 runs and shaded regions show 95% confidence intervals.

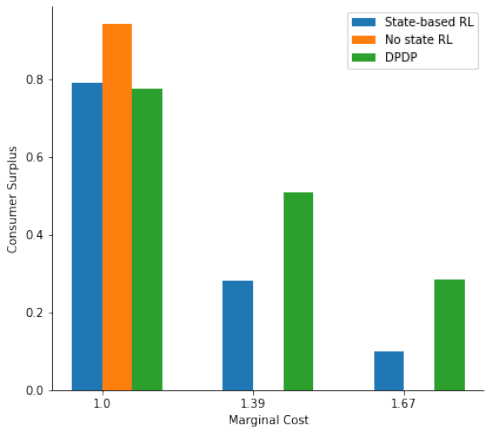

Figure 2: Robustness test when the platform's buy box policy is trained with *random-price probability*=0.4. The results are averaged over 10 runs.

always quote their minimal prices at the end of training. Given that this is the optimal (i.e., surplus maximizing) seller behavior, we can conclude that our Stackelberg-based learning methodology almost always finds an optimal leader strategy given the Q-learning behavior of sellers. We note that, as expected, it is easier for *no state RL* to reach the maximal performance given that that its class of policies is much smaller than the ones considered by *state-based RL*. However, as we will see in the next set of experiments, the state-based policy is much more flexible, and achieves better robustness to the case that the cost basis changes for sellers, such that the test environment is very different from that assumed during training.

5.2 Robustness Tests

As observed in our previous experiments, the Stackelberg-based RL algorithms are effective in learning interventions that maximize consumer surplus for a given economic setting. However, as they are tailored to the economic setting at hand, these interventions can perform poorly when facing settings different from those faced at training time. To learn more robust platform rules, we also train with a modified version of the Stackelberg MDP: at each reward step, with some *random-price probability*, sellers quote prices sampled uniformly at random from the price grid. In this way, the platform is rewarded during training for robust performance when presented with prices that are not produced by the Q-learning equilibrium dynamics of the sellers given the seller cost structure during training. Having trained with this random price perturbation, we test our policies in settings with different sellers' costs: in addition to the default cost $c = 1.0$, we also test with sellers that have cost $c = 1.38$ (falling in between the second and the third price from the set of discretized prices between 0.95 and 2.1) and sellers with cost $c = 1.67$ (falling in between the third and the fourth price of the price grid). As we can see from Figure 2, when using random prices during training (and in particular with random-price probability 0.4), the state-based policy is more flexible, displaying sellers with higher prices due to their higher costs, while continuing to outperform DPDP when seller costs are the same as during training. This is also confirmed by the policy visualizations in Figure 3, which show how the buy box tends to be much more open under this modified training regime. At the same time, our no state policy performs very poorly when tested at costs that differ from those assumed during training, generating zero consumer surplus even under this modified training regime. Despite this, the state-based RL policy has lower performance than DPDP in environments with out-of-training seller costs, suggesting that the platform interventions introduced by Johnson et al. [17] are more effective when the market is highly dynamic (and in the absence of continued learning by the platform, as conditioned change).

5.3 Learning in the Wild

In our previous experiments, we assumed that both sellers restart their learning processes any time the platform rules change. This is consistent with the original experiments run by Calvano et al. [11], which demonstrated seller collusive behavior. However, this assumption may not hold in real-world settings, where sellers can restart their learning processes asynchronously and at any time. This behavior can present new challenges to learning an effective platform policy. Indeed, in this scenario, changes in the sellers' behavior may be caused not only by different platform interventions, but also as a result of learning restarts.

In this section, we evaluate the performance of the Stackelberg MDP framework in scenarios where sellers randomly restart their exploration rate during training. Specifically, we assume that, at each step of the platform's learning process, each seller restarts its exploration rate with some probability. We set this probability such that, in expectation, each seller restarts its exploration once per Stackelberg MDP episode (which corresponds to the number of steps between platform updates). A particular concern is that this behavior may result in effectively random prices if a restart occurs close to the nominal reward phase of the Stackelberg MDP episode. A sensible response of a platform to this would be to

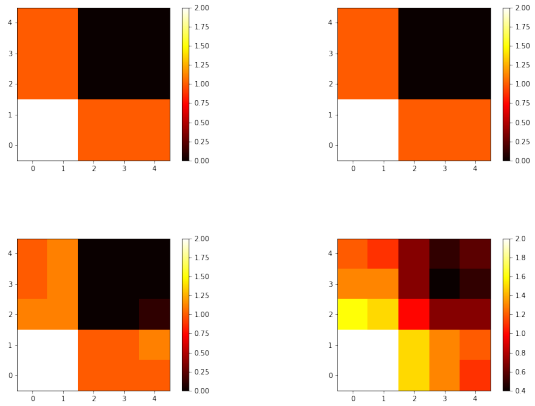

Figure 3: Policy visualization - number of displayed agents given price selection, averaged over 10 seeds. (Top-left): No state RL with no price perturbation in the reward step. (Top-right): No state RL with 40% random price probability in the reward step. (Bottom-left): State-based RL with no price perturbation in the reward step. (Bottom-right): State-based RL with 40% random price probability in the reward step. The heatmap shows white when the average number of agents displayed is 2, and black when the average number of agents displayed is 0. The axes show indices representing the ordering price pairs. E.g., (0, 0) represents a selection of the two lowest prices by both agents 0, while (4, 4) represents the highest prices by both agents.

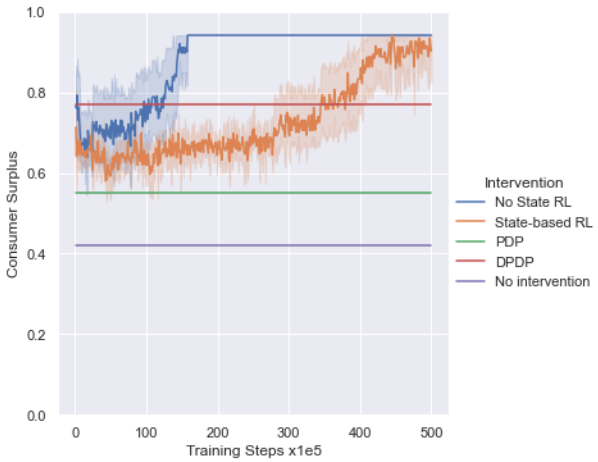

Figure 4: Learning in the wild performance. The results are averaged over 10 runs and shaded regions show 95% confidence intervals.

monitor sellers' prices, and isolate stages when price profiles are more stable to audit rewards. For a stylized version of this, we allow sellers' exploration to restart as outlined above, but pause any exploration during the reward phase of the Stackelberg MDP. Furthermore, if restarts occur close to the reward phases, rewards may reflect an out-of-equilibrium behavior of the sellers (even if exploration is paused). To avoid this problem, we generate our plots by logging rewards in an evaluation Stackelberg MDP episode we run every 100k training steps. These evaluation episodes use the current platform policy and operate it executing the action with the highest weight given each observation. In these episodes, the Q learning processes are run as in the previous sections, without intra-episode restarts. As we can see from Figure 4, the Stackelberg learning framework allows us to derive close-to-optimal policies even in this "in the wild" setting. Given that rewards are collected in evaluation episodes (where policies are operated via highest-weighted actions), the optimal intervention under no State RL is executed much earlier than in the simulations of Figure 1, reaching the maximum reward after only 15M training steps.

6 CONCLUSION

This work has demonstrated that effective platform interventions can be learned via machine learning methodologies that recognize and make use of the platform's commitment power. Specifically, we introduced the class of *threshold policies*, showing that it contains policies that optimize consumer surplus. We introduced a learning methodology that can effectively learn optimal leader policies in this class. The interventions we learned significantly outperform the interventions introduced in prior work. We also showed how our learned platform interventions can be made more robust when settings are dynamic, with varying seller cost structures, by adopting a suitably-modified training problems. In addition, we demonstrated that this approach to platform learning is robust to still more realistic assumptions about sellers' learning behavior.

Despite we focused on preventing price collusion in a simple setting, we believe that our approach is much more general. For example, it can be used to design interventions for the electricity markets studied by Abada and Lambin [1] or more general interventions (not only the threshold ones) in the setting we considered. In order to scale to larger settings, one could leverage symmetries among market participants.

We note our approach can also be used to derive "ad-hoc" interventions throughout the learning process. These interventions can be derived by letting the platform policy observe longer price histories or entire Q matrices (even though this would significantly increase the learning complexity).

ACKNOWLEDGMENTS

This research is funded in part by Defense Advanced Research Projects Agency under Cooperative Agreement HR00111920029. The content of the information does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred. This is approved for public release; distribution is unlimited. The work of G. Brero was also supported by the SNSF (Swiss National Science Foundation) under Fellowship P2ZHP1 191253. We thank Emilio Calvano and Justin Johnson for their availability to answer questions about their work and for guidance in replicating some of their results. We also thank Alon Eden, Matthias Gerstgrasser, and Alexander MacKay for for helpful discussions and feedback.

REFERENCES

[1] Ibrahim Abada and Xavier Lambin. 2020. Artificial Intelligence: Can Seemingly Collusive Outcomes Be Avoided? *Available at SSRN 3559308* (2020).

[2] Simon P Anderson and Andre De Palma. 1992. The logit as a model of product differentiation. *Oxford Economic Papers* 44, 1 (1992), 51–67.

[3] Enrique Areyan Viqueira, Cyrus Cousins, Yasser Mohammad, and Amy Greenwald. 2019. Empirical Mechanism Design: Designing Mechanisms from Data. In *Proceedings of the Thirty-Fifth Conference on Uncertainty in Artificial Intelligence (Proceedings of Machine Learning Research, Vol. 115)*. 1094–1104.

[4] Stephanie Assad, Robert Clark, Daniel Ershov, and Lei Xu. 2020. Algorithmic pricing and competition: Empirical evidence from the German retail gasoline market. (2020).

[5] Craig Boutilier. 1996. Planning, learning and coordination in multiagent decision processes. In *TARK*, Vol. 96. Citeseer, 195–210.

[6] Gianluca Brero, Darshan Chakrabarti, Alon Eden, Matthias Gerstgrasser, Vincent Li, and David Parkes. 2021. Learning Stackelberg Equilibria in Sequential Price Mechanisms. (2021).

[7] Gianluca Brero, Alon Eden, Matthias Gerstgrasser, David Parkes, and Duncan Rheingans-Yoo. 2021. Reinforcement Learning of Sequential Price Mechanisms.

[8] Erik Brinkman and Michael P. Wellman. 2017. Empirical Mechanism Design for Optimizing Clearing Interval in Frequent Call Markets. In *Proceedings of the 2017 ACM Conference on Economics and Computation*. 205–221.

[9] Benedikt Bünz, Benjamin Lubin, and Sven Seuken. 2020. Designing core-selecting payment rules: A computational search approach. *Available at SSRN 3178454* (2020).

[10] Emilio Calvano, Giacomo Calzolari, Vincenzo Denicolò, Joseph E Harrington, and Sergio Pastorello. 2020. Protecting consumers from collusive prices due to AI. *Science* 370, 6520 (2020), 1040–1042.

[11] Emilio Calvano, Giacomo Calzolari, Vincenzo Denicolo, and Sergio Pastorello. 2020. Artificial intelligence, algorithmic pricing, and collusion. *American Economic Review* 110, 10 (2020), 3267–97.

[12] Vincent Conitzer and Tuomas Sandholm. 2004. Self-interested automated mechanism design and implications for optimal combinatorial auctions. In *Proceedings of the 5th ACM conference on Electronic commerce*. ACM, 132–141.

[13] Michael J. Curry, Ping-Yeh Chiang, Tom Goldstein, and John Dickerson. 2020. Certifying Strategyproof Auction Networks. In *Proc. 33rd Annual Conference on Neural Information Processing Systems*.

[14] Michael J. Curry, Uro Lyi, Tom Goldstein, and John Dickerson. 2022. Learning Revenue-Maximizing Auctions With Differentiable Matching. In *Proc. 25th International Conference on Artificial Intelligence and Statistics*. Forthcoming.

[15] Paul Duetting, Zhe Feng, Harikrishna Narasimhan, David C. Parkes, and Sai Srivatsa Ravindranath. 2019. Optimal Auctions through Deep Learning. In *Proceedings of the 36th International Conference on Machine Learning*. 1706–1715.

[16] Drew Fudenberg and Jean Tirole. 1991. *Game theory*. MIT press.

[17] Justin Johnson, Andrew Rhodes, and Matthijs R Wildenbeest. 2020. Platform design when sellers use pricing algorithms. (2020).

[18] Patrick R. Jordan, L. Julian Schvartzman, and Michael P. Wellman. 2010. Strategy exploration in empirical games. In *9th International Conference on Autonomous Agents and Multiagent Systems*. 1131–1138.

[19] Christopher Kiekintveld and Michael P. Wellman. 2008. Selecting strategies using empirical game models: an experimental analysis of meta-strategies. In *7th International Joint Conference on Autonomous Agents and Multiagent Systems*. 1095–1101.

[20] Kevin Kuo, Anthony Ostuni, Elizabeth Horishny, Michael J. Curry, Samuel Dooley, Ping-Yeh Chiang, Tom Goldstein, and John P. Dickerson. 2020. ProportionNet: Balancing Fairness and Revenue for Auction Design with Deep Learning. *CoRR* abs/2010.06398 (2020).

[21] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems* 30 (2017).

[22] Neehar Peri, Michael J. Curry, Samuel Dooley, and John P. Dickerson. 2021. PreferenceNet: Encoding Human Preferences in Auction Design with Deep Learning. In *Proceedings of the 35th Conference on Neural Information Processing Systems*. Forthcoming.

[23] Antonin Raffin, 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 (2021), 1–8. http://jmlr.org/papers/v22/20-1364.html

[24] Jad Rahme, Samy Jelassi, Joan Bruna, and S. Matthew Weinberg. 2021. A Permutation-Equivariant Neural Network Architecture For Auction Design. In *Proc. Thirty-Fifth AAAI Conference on Artificial Intelligence*. 5664–5672.

[25] Jad Rahme, Samy Jelassi, and S. Matthew Weinberg. 2021. Auction Learning as a Two-Player Game. In *Proc. 9th International Conference on Learning Representations, ICLR*.

[26] Weiran Shen, Binghui Peng, Hanpeng Liu, Michael Zhang, Ruohan Qian, Yan Hong, Zhi Guo, Zongyao Ding, Pengjun Lu, and Pingzhong Tang. 2020. Reinforcement mechanism design: With applications to dynamic pricing in sponsored search auctions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 2236–2243.

[27] W. Shen, P. Tang, and S. Zuo. 2019. Automated Mechanism Design via Neural Networks. In *Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems*.

[28] Andrea Tacchetti, DJ Strouse, Marta Garnelo, Thore Graepel, and Yoram Bachrach. 2019. A Neural Architecture for Designing Truthful and Efficient Auctions. *CoRR* 1907.05181 (2019).

[29] Pingzhong Tang. 2017. Reinforcement mechanism design.. In *IJCAI*. 5146–5150.

[30] The Organisation for Economic Co-operation and Development. 2017. Algorithms and Collusion– Note from the European Union. www.oecd.org/competition/algorithms-and-collusion.htm

[31] U.S. Federal Trade Commission. 2018. *The Competition and Consumer Protection Issues of Algorithms, Artificial Intelligence, and Predictive Analytics*, Hearing on Competition and Consumer Protection in the 21st Century, U.S. Federal Trade Commission. https://www.ftc.gov/policy/hearings-competition-consumer-protection

[32] Yevgeniy Vorobeychik, Christopher Kiekintveld, and Michael P. Wellman. 2006. Empirical mechanism design: methods, with application to a supply-chain scenario. In *Proceedings 7th ACM Conference on Electronic Commerce (EC-2006)*. 306–315.

[33] Jakob Weissteiner and Sven Seuken. 2020. Deep Learning—Powered Iterative Combinatorial Auctions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 2284–2293.

[34] Michael P. Wellman. 2006. Methods for Empirical Game-Theoretic Analysis. In *Proceedings, The Twenty-First National Conference on Artificial Intelligence*. 1552–1556.

[35] Stephan Zheng, Alexander Trott, Sunil Srinivasa, Nikhil Naik, Melvin Gruesbeck, David C Parkes, and Richard Socher. 2020. The ai economist: Improving equality and productivity with ai-driven tax policies. *arXiv preprint arXiv:2004.13332* (2020).