Algorithmic Collusion Without Threats

Eshwar Ram Arunachaleswaran
Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA, USA

Natalie Collina
Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA, USA

Sampath Kannan
Simons Institute for the Theory of Computing, University of California, Berkeley, CA, USA

Aaron Roth
Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA, USA

Juba Ziani
ISyE, Georgia Tech, Atlanta, GA, USA

Abstract

There has been substantial recent concern that automated pricing algorithms might learn to "collude." Supra-competitive prices can emerge as a Nash equilibrium of repeated pricing games, in which sellers play strategies which threaten to punish their competitors if they ever "defect" from a set of supra-competitive prices, and these strategies can be automatically learned. But threats are anti-competitive on their face. In fact, a standard economic intuition is that supra-competitive prices emerge from *either* the use of threats, *or* a failure of one party to correctly optimize their payoff. Is this intuition correct? Would explicitly preventing threats in algorithmic decision-making prevent supra-competitive prices when sellers are optimizing for their own revenue?

No. We show that supra-competitive prices can robustly emerge even when both players are using algorithms which do not explicitly encode threats, and which optimize for their own revenue. Since deploying an algorithm is a form of commitment, we study sequential Bertrand pricing games (and a continuous variant) in which a first mover deploys an algorithm and then a second mover optimizes within the resulting environment. We show that if the first mover deploys *any* algorithm with a no-regret guarantee, and then the second mover even approximately optimizes within this now static environment, monopoly-like prices arise. The result holds for any no-regret learning algorithm deployed by the first mover and for *any* pricing policy of the second mover that obtains them profit at least as high as a random pricing would – and hence the result applies even when the second mover is optimizing only within a space of non-responsive pricing distributions which are incapable of encoding threats. In fact, there exists a set of strategies, neither of which explicitly encode threats that form a Nash equilibrium of the simultaneous pricing game in algorithm space, and lead to near monopoly prices. This suggests that the definition of "algorithmic collusion" may need to be expanded, to include strategies without explicitly encoded threats.

2012 ACM Subject Classification Theory of computation → Algorithmic game theory

Keywords and phrases Algorithmic Game Theory, Algorithmic Collusion, No-Regret Dynamics

Digital Object Identifier 10.4230/LIPIcs.ITCS.2025.10

Related Version *Full Version*: `https://www.arxiv.org/pdf/2409.03956` [3]

Supplementary Material *Software (Source Code)*: `https://github.com/eshwarram/Non_Myopic_Pricing`, archived at `swh:1:dir:8ecaca48118cc1984e5a41878d828d807920a98b`

Funding ERA was supported by NSF grants CCF 1910534 and 2045128. NC and AR were partially supported by NSF grants FAI-2147212 and CCF-2217058, the Hans Sigrist Prize, and the Simons Collaboration on Algorithmic Fairness.

Acknowledgements We thank Rakesh Vohra and Deke Hill for valuable discussions on the subject of algorithmic collusion. SK was on leave from the University of Pennsylvania and serving as the Associate Director of the Simons Institute for the Theory of Computing at the time of writing of this paper. Finally, we would like to thank the ITCS reviewers for their helpful comments and feedback.

1 Introduction

Consider a market in which there are two sellers of some commodity. We hope and expect that competition should drive the price of the good in the market down to its production cost – for if seller A was consistently selling above cost, then there would be an opportunity for seller B to slightly undercut seller A, capturing the whole market and increasing their profit. In fact, this is the Nash (and correlated) equilibrium outcome in several classical formalizations of such competition including the Bertrand Duopoly model. We refer to pricings that are at (or near) the Nash equilibrium prices as *competitive prices*.

As the name suggests, competition is essential to maintaining competitive prices: a monopolist could freely set the price of a good as high as the market could support (which, for goods with inelastic demand, could be quite high indeed) without fear of being undercut by a competitor. So, if in a market with multiple participants, we see prices that are closer to the monopoly price than the competitive price, we might suspect "anti-competitive" behavior on the part of the sellers.

Yet, defining anti-competitive behavior turns out to be a difficult exercise. The Sherman act has been interpreted to require express agreement to coordinate on prices in the form of overt communication as a pre-requisite for liability. However, a recent line of work has focused on the ability of *pricing algorithms* to arrive at supra-competitive prices without explicit communication – concerns rooted in real world observations [4] and a growing academic literature [2, 20, 15]. The mechanism underlying most results in this literature ultimately are rooted in "folk-theorem" style equilibria of repeated pricing games [6, 22]. Broadly speaking, folk theorems establish that all pairs of prices that guarantee the sellers at least the profit they obtain with competitive pricing can be realized as equilibria of the repeated game. Crucially, the strategies which are shown to lead to these anti-competitive equilibria have explicitly encoded threats aimed at punishing the other seller if they deviate from a proscribed pricing path. Further, reinforcement learning algorithms can automatically discover these strategies; recent work [8, 18] has shown both empirically and analytically that these algorithms can converge to supra-competitive prices, and they do so by the emergence of punishments and threats, even without being explicitly programmed to learn such strategies. Using such algorithms may not violate current anti-trust law as they do not involve explicit communication, but one can imagine future legislation and case law that forbids the use of algorithms that make use of threats (appropriately defined). An algorithm is after all an explicit commitment to action as a function of its observations, and committing to an algorithm might be interpreted as communicating the content of the threat encoded within the algorithm.

To formally rule out "anti-competitive" behavior, an approach one might take is to define algorithms that do not encode threats. In fact, the standard economic stance [16, 8] has been that threats are an *explicit* requirement for collusion, distinguished from other situations in which supra-competitive prices can arise due to a "failure to optimize:"

> "To us economists, collusion is not simply a synonym of high prices but crucially involves "a reward-punishment scheme designed to provide the incentives for firms to consistently price above the competitive level" (Harrington 2018, p. 336). The reward-punishment scheme ensures that the supra-competitive outcomes may be obtained in equilibrium and do not result from a failure to optimize." [8]

A first attempt might be to say that (at least if there are other optimizers in the market) "non-responsive" algorithms which set prices completely independently of competitor prices (and consequences of competitor prices) should be permitted – indeed, if they cannot react to their competitors, then they cannot threaten them. But this is far too limiting as it does not allow for algorithms that react to market conditions. Recent work by [17] and [10] set out a principled definition of what constitutes non-collusive behavior of a pricing algorithm by defining algorithms that satisfy some internal consistency properties (calibrated regret, also known as "swap regret") and then shows that if all algorithms in the market satisfy these conditions then the outcome will be competitive prices. Can this condition be relaxed? What if the first entrant to a market deploys (and hence commits to) an algorithm that satisfies these internal consistency conditions, and then the second entrant plays a simple policy that does not satisfy the swap regret condition, but also does not encode threats (e.g. because it is entirely non responsive)? Might we still see competitive prices? Would we, in accordance with the intuition of [16, 8] see competitive prices if the follower not only did not deploy threats, but also succeeded in "optimizing"?

In this paper we show that the answer is robustly "no" by showing how a broad class of "reasonable" behavior on the part of all parties must lead to supra-competitive (near monopoly) prices. In particular, with the view that deploying an algorithm is inherently a "commitment" (at least in the short term), we study two seller pricing games in a leader-follower model, in which the leader first deploys an algorithm, and then the follower deploys an algorithm which may optimize within the market environment that is implicitly defined by the leader's pricing algorithm. If both sellers viewed this scenario as a Stackelberg game, and optimized over the set of all pricing algorithms, it would hardly be surprising if anti-competitive behavior emerged. Indeed, the Stackelberg equilibria in algorithm space (See [11]) in a repeated pricing game would involve the leader committing to an algorithm that prescribes monopoly prices and commits to punish the follower if she deviates from this prescription.

However, we instead study the behavior that emerges if sellers attempt to play reasonable algorithms. In particular, we assume the leader (following the principle set out by [17, 10]) commits to setting prices using an arbitrary *no-regret* algorithm. No-Regret algorithms are widely understood to be a "natural" class of algorithms for use in competitive environments [24], and our results will hold for any algorithm in this class, which includes the no-swap regret algorithms studied by [17, 10].

Once the leader deploys a no-regret learning algorithm, the other seller (whom we call the follower, or the optimizer) deploys an algorithm to optimize their own revenue in the environment defined by the leader's no-regret learning algorithm. Once again, we do not assume that the follower is best responding in algorithm space: they may, e.g., use a heuristic reinforcement learning algorithm to optimize over a limited set of policies – even *non-responsive* policies that are explicitly independent of the leader's prices, and hence cannot encode threats. We show that however they choose to optimize their own revenue, if they are even marginally successful (choosing any policy that guarantees them at least the revenue that a random pricing strategy would), then in combination, the two algorithms will inevitably lead to supra-competitive prices in a variety of settings. When this happens, it is not just the follower who enjoys the revenue of the high prices, but as we prove, also the leader (if the leader deployed a no-regret learning algorithm). Thus the leader will have no strong incentive to deviate from their deployed algorithm, even if they deployed it without initially being aware of the competitive environment. In fact, within the class of "reasonable" strategies we study for the leader and follower (a no-swap-regret algorithm for the leader

and a non-responsive algorithm for the follower), there exists a Nash equilibrium of the simultaneous pricing game in algorithm space, which gives *no party* any incentive to deviate, supports near-monopoly prices, and does not involve threats. This suggests that it may be worth revisiting the standard economic viewpoint [8, 16] that *threats* are a necessary precondition for collusion.

Summary of Results

We study two types of models: a simple Bertrand duopoly model, where the firm with the lowest price captures the entire demand, and a smoother multinomial-logit-based model where firms with higher prices can still capture some of the demand. Our multinomial logit model includes a temperature parameter τ that controls how sensitive the demand for each firm is to prices, and converges to a Bertrand competition when the parameter τ grows large. Within these models, we prove the following results:

1. The second entrant into the market (whom we call the optimizer) can extract a constant fraction of the monopoly price revenue against any no-regret algorithm by playing a non-responsive strategy (i.e. a fixed distribution of prices across all rounds, that does not depend on or respond to the history of play). As a result, if they approximately optimize their pricing policy (in any set of policies that includes natural fixed price distributions), they will be guaranteed a constant fraction of the monopoly revenue. Observe that this holds even if the class they optimize over *only* includes non-responsive pricing strategies, which are incapable of encoding threats.
2. Simultaneously, by deploying any no-regret algorithm, the first entrant (whom we call the learner) extracts a constant fraction of the revenue obtained by any optimizer strategy that obtains a constant fraction of the monopoly price revenue. Taken in conjunction with the previous result, this implies that if the first entrant into the market deploys (and hence commits to) a no-regret learning algorithm, and the optimizer deploys any strategy that approximately optimizes over any class of policies that includes fixed price distributions, *both* parties will extract a constant fraction of the monopoly price revenue. This has important implications: not only do supra-competitive prices emerge, but the benefits are shared across both sellers, giving neither a substantial incentive to deviate. In other words, the optimizer is not "exploiting" the no-regret learner, but rather implicitly cooperating with them, despite the absence of threats.
3. In fact, we show that if the first entrant into the market deploys a no-swap regret algorithm, then there is a non-responsive (i.e. fixed) price distribution for the optimizer that forms a *Nash* equilibrium of the simultaneous pricing game in algorithm space: i.e. *neither* player has any incentive to deviate from their selected algorithm. This demonstrates the existence of a Nash equilibrium of the game that supports supra-competitive prices and yet does not involve threats, contra to the economic intuition [8, 16] that supra-competitive prices result from *either* threats or a failure to optimize.

When we refer to supra-competitive prices in the above theoretical results, we mean prices that are within a constant factor of the monopoly price. The constants in our theorems are small, but we provide numerical evidence that the constant is in fact no smaller than $2/e \approx 0.74$.

We also prove an additional result about the Bertrand duopoly model:

- Our results apply to *any* no regret learning algorithm deployed by the first entrant into the market. This is a super-set of the no-swap regret learning algorithms proposed to be definitionally "competitive" by [17, 10]. Should no regret algorithms beyond no-swap

regret algorithms also be considered competitive? This is not the case for all no-regret dynamics – [23] show the existence of coarse correlated equilibria supported on supra-competitive prices in Bertrand pricing games, implying the existence of no-regret dynamics that result in such prices. However, we show that when both sellers use Mean-based No-Regret (MBNR) algorithms (a class which includes common no-regret learning algorithms such as multiplicative weights and follow the perturbed leader) to set prices, the empirical distribution of play always converges to the competitive price. This suggests that at least in the Bertrand Duopoly model, mean based no-regret learning algorithms might also be considered reasonable/competitive pricing strategies, in addition to the no-swap regret learners proposed by [17, 10]. We note that this result does not necessarily carry over to the more complex pricing models studied by [17].

Due to space constraints, we defer some proofs to the full version of the paper [3].

1.1 Related Work

The seminal work by [8] experimentally shows that Q-learning algorithms, a simple and central type of reinforcement learning algorithm, can learn to collude with each other via threats in a repeated, simultaneous-action pricing game. These algorithms learn to deviate to lower prices in response to another algorithm doing so, and slowly rise back to higher prices. [18] attain similar results in the sequential repeated pricing game. The authors in these works are specifically concerned with the ability of the algorithms to learn clear threats, as opposed to simply reaching supra-competitive prices. This is because, from the perspective of many economists, such threats are a requirement for collusion. The prevailing view, as espoused by [16] and [8], is that any algorithms which reach supra-competitive prices without threats must be suboptimal, and thus uninteresting as candidate pricing algorithms.

This perspective is supported by [1] (experimentally) and [2] (theoretically). These works consider a different class of reinforcement learning algorithms and show that, while they do not converge to supra-competitive prices when they have full computational capacity, they can reach supra-competitive prices if these algorithms are limited in their ability to explore. These supra-competitive prices arise even though these algorithms do not retain enough state information to encode threats. Therefore, while these works do show supra-competitive prices without threats, they are only shown arising from algorithms which are specifically designed to be suboptimal.

Another line of research has explored the emergence of supra-competitive prices as a byproduct of algorithmic symmetry. [15] show that, if two identical, deterministic algorithms are played against each other, their "exploration" will be symmetric, and hence they will be unable to ever observe the benefits of undercutting their competitors (an inherently asymmetric outcome).

[5] study algorithms that attempt to estimate the payoff of actions through stochastic experiments. They show that in certain settings in which players update the reward for the action they play, but not other actions, then the experiments can correlate the actions of the players, leading to supra-competitive prices. In contrast, the algorithms we study operate in the full information setting (because agents can observe the price played by the other agent, they can estimate the profit they would have made had they played a different price at a given time period), and are designed for adversarial settings, which are well specified in competitive environments: they are designed with the understanding that there are no reward distributions to estimate because profits depend on the evolving actions of an opponent.

In contrast to these previous works, which argue variously that the property of algorithms which leads to supra-competitive pricing are threats and sub-optimality, we argue that the core is in the fact that deploying an algorithm is a form of commitment. This is most in line

with the perspective of [20]. This work treats algorithms as a form of commitment which contributes to collusion. However, the algorithms explored in that work are constrained to simple, two-state automata, which severely limits the strategy space. By contrast, we consider the space of all possible algorithms mapping histories to future prices, and show results for broad families of "reasonable" algorithms – not just equilibrium strategies.

There is also a recent line of work on regulating pricing algorithms. Both [10] and [17] suggest that observed *swap regret* might be taken as a proxy for collusive behavior. They show, in a related but more general model than our own, that if both sellers in a duopoly have vanishing swap regret, the time-average price must converge to the competitive price. Our work highlights the importance of *both* sellers having no-swap regret; as we show, if the first entrant into a market deploys a no swap regret algorithm, then many natural behaviors for the second entrant lead to supra-competitive prices.

Finally, there is a recent line of experimental work demonstrating anti-competitive behavior in large language models, including market division in the Cournot competition model [21] and, more similarly to this work, price fixing in the Bertrand competition model [14]. These works provide evidence for the emergence of algorithmic collusion in the increasingly ubiquitous realm of LLMs, underscoring the importance of understanding the foundations of algorithmic collusion.

The emergence of anti-competitive behavior in no-regret dynamics is also explored in [19], though in a somewhat different setting – this work considers agents deploying no-regret algorithms to bid for them in an auction, and shows that agents misreporting their true item value to the algorithms can lead to collusive outcomes.

2 Model

2.1 The Single-Stage Pricing Game

We will first formally define the payoff model of the stage game, in which both sellers select a price only once and specify the payoff they obtain.

We work in a duopoly model, where two sellers are competing to sell to a single buyer. Both sellers pick from a discrete set of prices $\mathcal{P} = \{\frac{1}{k}, \frac{2}{k}, \cdots 1\}$. Here, k is a discretization parameter, and we assume that $k \geq 20$.

Game definition

We consider two sellers, who we will refer to as the Sellers 1 and 2. The action space of each seller is the set of possible prices in $(0, 1]$, up to some discretization parameter $1/k$. Both sellers can choose to distributions of prices from the set $\mathcal{P}$, and we denote the set of all distributions by $\Delta^{\mathcal{P}}$.

To define our game, we first need to define the concept of an allocation rule. An *allocation rule* is a (potentially randomized) rule that divides demand across the two sellers as a function of their chosen prices:

▶ **Definition 1** (Allocation Rule $C_i(p_1, p_2)$)**.** *An* allocation rule $C_i : \mathcal{P} \times \mathcal{P} \mapsto [0, 1]$ *is a mapping from a price pair* (p_1, p_2) *from the two sellers, respectively, to a fraction of the demand that will be allocated to seller* $i \in \{1, 2\}$*.*

Given an allocation rule that defines and controls market outcomes as a function of prices, we can now define the payoff of a seller in our single-stage game as a function of said allocation rule C.

▶ **Definition 2** (Seller Payoff in the Stage Game). *Given an allocation rule $C(.,.)$, the payoff of seller i is given by*

$$u_i(p_1, p_2) = p_i \cdot C_i(p_1, p_2). \tag{1}$$

When the sellers play distributions, the payoff of the seller is understood to be the expected payoff.

We define the buyer price as the price charged by the seller that is the output of the allocation rule.

▶ **Definition 3** (Average Buyer Price in the Stage Game). *For an allocation rule C and (mixed) strategies p_1, p_2, the average price paid by the buyer is $\mathbb{E}[p_1 C_1(p_1, p_2) + p_2 C_2(p_1, p_2)]$. Note that this means that the average buyer price is equal to the (expected) sum of payoffs of the two sellers.*

Throughout this paper we assume that the *total* demand (added across both sellers) for the item being sold is fixed and independent of the prices. Without loss of generality, this total demand is set to be normalized to 1. As a result, the "monopoly price" in these models is always 1. Within this fixed-demand model, we consider two allocation rules: a Bertrand model and a multinomial-logit-based model, both formalized in Section 2.1.1. Both allocation rules are *symmetric* and therefore the resulting games are *symmetric* as well.

We can now formally define the one-stage game studied in this paper, often informally referred to simply as the "stage game":

▶ **Definition 4** (Pricing Stage Game). *A two-player pricing stage game $G(k, C)$ is defined as a combination of:*

1. *Two sellers, respectively named 1, 2;*
2. *An action space $A = \mathcal{P} = \{\frac{1}{k}, \frac{2}{k}, \ldots, 1\}$ corresponding to possible prices that can be played by 1 and 2. We denote by p_1 the action of Seller 1 and p_2 the action of seller 2. These actions may be picked from the set of distributions over $\mathcal{P}$, which we call $\Delta^{\mathcal{P}}$.*
3. *Payoff functions $u_i(p_1, p_2)$ for $i \in \{1, 2\}$, defined as a function of the competitive allocation rule C as in Definition 2.*

A particular pricing game is defined by the allocation rule C and the discretization parameter k. We will work with symmetric allocation rules (the allocation does not depend on the labels of the sellers), which will naturally induce a symmetric finite game between the two sellers.

The one-stage game described is this section is the starting point of our study, one that helps us define competitive prices as the Nash equilibria outcome of the game. However, the primary object of our study is *repeated* interactions between sellers, described in Section 2.2.

Equilibria of the Stage Game

We will refer to two kinds of equilibria of the stage game which differ in their assumed timing. The *Nash* equilibria characterize outcomes in simultaneous play, whereas *Stackelberg* equilibria characterize outcomes of optimal play when one player may first commit to a strategy, and the other player then responds.

▶ **Definition 5** (Nash Equilibrium of the Stage Game). *A pair of (independent) distributions (p_1, p_2) for the two sellers is said to be an ε-Nash equilibrium if $u_1(p_1, p_2) \geq \max_{p \in \mathcal{P}} u_l(p, p_2) - \varepsilon$ and $u_2(p_1, p_2) \geq \max_{p \in \mathcal{P}} u_2(p_1, p) - \varepsilon$. When $\varepsilon = 0$, this pair is a Nash equilibrium.*

▶ **Definition 6** (Stackelberg Equilibrium of the Stage Game). *A pair of (independent) distributions p_1, p_2 for the two sellers is said to be an ε-Stackelberg equilibrium of the stage game with seller 1 as the leader if*

$$u_2(p_1, p_2) \geq \max_{p \in \mathcal{P}} u_2(p_1, p),$$

and

$$u_1(p_1, p_2) \geq \max_{(q_1, q_2) \in \Delta^{\mathcal{P}}} u_1(q_1, q_2) - \varepsilon$$
$$s.t. \quad u_2(q_1, q_2) \geq \max_{q \in \mathcal{P}} u_2(q_1, q).$$

When $\varepsilon = 0$, this pair is a Stackelberg equilibrium.

2.1.1 Competition Models

Now that we have defined the pricing stage game in full generality, we will instantiate it with two different specific competition rules, which result in two well-known economic pricing models.

2.1.1.1 Bertrand Competition

In the Bertrand Competition model, the seller picking the lower price captures all of the demand. The seller picking the higher price earns nothing. For cases in which sellers chose the same price p_L, they split the market equally.

▶ **Definition 7** (Bertrand Competitive Allocation Rule C_i^B). *C_i^B is a competition rule such that, given player $j \neq i$,*

$$C_i^B(p_1, p_2) = \begin{cases} 1 & p_i < p_j, \\ \frac{1}{2} & p_i = p_j, \\ 0 & p_i > p_j \end{cases} \tag{2}$$

▶ **Definition 8** (Bertrand Pricing Stage Game $G^B(k)$). *The* Bertrand Pricing Game *is a pricing game instantiated with the competition rule C^B and with a pricing discretization of $\frac{1}{k}$.*

The Bertrand model is discontinuous in that the seller who selects the lower price captures the entire market. We also study a generalization of Bertrand competition in which the fraction of the market captured by a seller is a smooth function (a logit function) of the difference in prices set by the two sellers. This model has a temperature parameter τ, and as τ tends to infinity, it recovers the Bertrand competition model as a special case.

2.1.1.2 Multinomial-logit-based Price Competition

This competition model is parameterized by a temperature parameter $\tau \geq 0$

▶ **Definition 9** (Logit Competitive Allocation Rule $C_i^{L,\tau}$). *$C_i^{L,\tau}$ is a competition rule such that for $j \neq i$:*

$$C_i^{L,\tau}(p_1, p_2) = \frac{e^{\tau p_j}}{e^{\tau p_i} + e^{\tau p_j}} \tag{3}$$

▶ **Definition 10** (Multinomial-logit-based Pricing Stage Game $G^B(k, \tau)$)**.** *The* Multinomial-logit-based Pricing Game *is a pricing game instantiated with the competition rule* $C^{L,\tau}$ *and with a pricing discretization of* $\frac{1}{k}$*.*

We prove similar results as in the Bertrand model in this logit model, assuming that the temperature parameter is sufficiently large, i.e., the allocation rule is sufficiently sensitive to differences in price. As $\tau \to \infty$, we recover the Bertrand model in the limit, since the lower price seller wins the demand with probability 1. Likewise, when $\tau = 0$, the sellers each win with probability $1/2$ regardless of their price, so it is to be expected that we need τ to be reasonably large to prove any meaningful results.

This model is a special case of the logit model of [8] and is obtained by setting all "product indices" in their model to be equal, and by removing the outside good from consideration (by setting its product index a_0 to $-\infty$), the latter action fixing the total demand.

For simplicity, in the main body of the paper we prove our results in the Bertrand game (Section 4). We show that they generalize to the Multinomial-Logit-Based pricing game in the full version of the paper.

2.2 The Repeated Pricing Game

We now move from the stage game to the repeated pricing interaction that will be our main setting of interest. The pricing game is played repeatedly over T rounds. During each round t, the two sellers simultaneously pick mixed strategies $p_{1,t},\ p_{2,t}$ over the set of prices $\mathcal{P}$ (these are independent distributions over $\mathcal{P}$). The sellers receive the corresponding utilities equal to the expected payoff of pricing $p_{1,t}$ against $p_{2,t}$ in the stage game. Both sellers observe the full mixed strategy of their opponent. Here, the pricing behavior of each player will be defined by algorithms which map history to the pricing distribution at the next round, and we study this larger interaction as a game in which the strategy space for each player is the algorithm that they will deploy. We formalize this setting below by defining the strategy space and the utility functions.

Algorithms as Strategies

The sellers' strategy space is the space of all algorithms, mapping histories to price distributions at the following round in arbitrary ways.

▶ **Definition 11.** *An algorithm* $\mathcal{A}$ *for the* T *round repeated game is a mapping from the history of play (without loss of generality of only the other seller) to the next round's action, denoted by a collection of* T *functions* $A_1, A_2 \cdots A_T$*, each of which deterministically map the transcript of play (up to the corresponding round) to a mixed strategy to be used in the next round, i.e.,* $A_{1,t}(p_{2,1}, p_{2,2}, \cdots, p_{2,t-1}) = p_{1,t}$ *for the algorithm* $\mathcal{A}_1$ *used by seller 1.*

We define a restricted subclass of algorithms below – algorithms that do not use nor respond to the history of play. Such algorithms, in particular, cannot make use of or respond to threats as they are oblivious to the behavior of their opponent.

▶ **Definition 12.** *If* A_t *is independent of its input for all* t*, we call it* non-responsive*. If in addition* $\mathcal{A}_t$ *outputs the* same *distribution for all* t *we call it static.*

If the two sellers pick algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$ respectively, this induces a transcript of T price distribution pairs $((p_{1,1}, p_{2,1}), \cdots (p_{1,t}, p_{2,t}) \cdots (p_{1,T}, p_{2,T}))$ where $A_{1,t}(p_{2,1}, p_{2,2}, \cdots, p_{2,t-1}) = p_{1,t}$ and, similarly, $A_{2,t}(p_{1,1}, p_{1,2}, \cdots, p_{1,t-1}) = p_{2,t}$ for each round t.

Game definition

We again consider two sellers, who we will now refer to as the *learner* and the *optimizer*. The action space of each seller is now the set of possible *algorithms* mapping the history of play to a pricing distribution at the current round.

▶ **Definition 13** (Seller Payoff in the Repeated Game). *Let $p_l^{1:T}$ and $p_o^{1:T}$ represent the sequence of T distributions over prices by the learner and the optimizer, respectively, given $\mathcal{A}_l$ and $\mathcal{A}_o$. Then, the payoff of seller $i \in \{l, o\}$ is*

$$U_i(\mathcal{A}_l, \mathcal{A}_o) = \sum_{t=1}^{T} u_i(p_l^t, p_o^t) \tag{4}$$

We can now formally define the central game studied in this paper:

▶ **Definition 14** (Repeated Pricing Game). *A two-player repeated pricing game $G(k, C, T)$ is defined as a combination of:*

1. *Two players, named the "learner" and the "optimizer";*
2. *A time horizon T specifying the length of the repeated game;*
3. *An action space $A = \mathcal{A}^T$ corresponding to possible sequential pricing algorithms for games of length T;*
4. *Payoff functions $U_l(\mathcal{A}_l, \mathcal{A}_o)$ for the learner and $U_o(\mathcal{A}_l, \mathcal{A}_o)$ for the optimizer, defined as in Definition 13.*

A particular repeated pricing game is defined by the allocation rule C and the discretization parameter k.

Equilibrium Notions in Algorithm Space

Once again, we can study two kinds of equilibria of the repeated game in "algorithm space", depending on the timing of the game: Nash equilibria if play is simultaneous, and Stackelberg equilibria if the learner has commitment power and moves first.

▶ **Definition 15** (Nash Equilibrium in Algorithm Space). *A pair $(\mathcal{A}_l, \mathcal{A}_o)$ of algorithms for the two sellers is said to be an ε-Nash equilibrium in "algorithm space" if $U_l(\mathcal{A}_l, \mathcal{A}_o) \geq \max_{\mathcal{A} \in \mathcal{A}^T} U_l(\mathcal{A}, \mathcal{A}_o) - \varepsilon$ and $U_o(\mathcal{A}_l, \mathcal{A}_o) \geq \max_{\mathcal{A} \in \mathcal{A}^T} U_o(\mathcal{A}_l, \mathcal{A}) - \varepsilon$. When $\varepsilon = 0$, we call this pair a Nash equilibrium in algorithm space.*

▶ **Definition 16** (Stackelberg Equilibria in Algorithm Space). *A pair $(\mathcal{A}_l, \mathcal{A}_o)$ of algorithms for the two sellers is said to be an ε-approximate Stackelberg equilibrium in "algorithm space" if*

$$U_o(\mathcal{A}_l, \mathcal{A}_o) \geq \max_{\mathcal{A} \in \mathcal{A}^T} U_o(\mathcal{A}_l, \mathcal{A}),$$

and

$$U_l(\mathcal{A}_l, \mathcal{A}_o) \geq \max_{(\mathcal{B}_l, \mathcal{B}_o) \in \mathcal{A}^T} U_l(\mathcal{B}_l, \mathcal{B}_o) - \varepsilon$$
$$s.t. \;\; U_o(\mathcal{B}_l, \mathcal{B}_o) \geq \max_{\mathcal{B} \in \mathcal{A}^T} U_o(\mathcal{B}_l, \mathcal{B})$$

When $\varepsilon = 0$, we call this pair a Stackelberg equilibrium in algorithm space.

We investigate a setup in which one seller, who we call the Learner, commits to an algorithm $\mathcal{A}_L$, and then the other seller, who we call the Optimizer, responds with their own algorithm $\mathcal{A}_O$ [1]. This is reminiscent of a Stackelberg game in algorithm space; however, in contrast to standard Stackelberg games, we do not necessarily assume that the Optimizer exactly best-responds to the Learner's algorithm, or that the Learner necessarily deploys the algorithm representing the optimal commitment strategy. Instead, we examine which pricing outcomes arise under various Optimizers of different capacities, and show that, when the Optimizer is responding to a no-regret algorithm (defined in Section 3), a robust set of responses, including exactly optimal but also including approximately optimal non-responsive strategies, lead to supra-competitive prices.

Competitive vs supra-competitive prices

In the economics literature, competitive prices are defined as prices that can emerge from the static Nash equilibria of the pricing game (i.e. Nash equilibria of the stage game, where prices, rather than algorithms, form the strategy space). We formalize this below:

▶ **Definition 17** (Competitive and Supra-competitive prices). *For some pricing model G, let p be the highest price that is in the support of any Nash equilibrium of the stage pricing game. Then, p' is a competitive price if $p' \leq p$. Furthermore, p' is a supra-competitive price if $p' = \Omega_k(1)$* [2].

Supra-competitive prices are sometimes defined as any prices that are strictly above the competitive price. In this work, we use and guarantee a stronger definition, and require supra-competitive prices to be a constant fraction of the maximum possible price (1), even when the competitive price tends to 0 as $k \to \infty$. Thus our definition of supra-competitive prices really refers to "near monopoly" prices.

Throughout the paper we will regularly use high seller payoff as proof of supra-competitive prices. We make that connection explicit here, showing that in both the Bertrand and Logit models, the average price over time is exactly equal to the sum of the average utilities of the two sellers:

▶ **Definition 18** (Average Buyer Price in the Repeated pricing Game). *For an allocation rule C and seller algorithms $\mathcal{A}_l$, $\mathcal{A}_o$, the average buyer price over the rounds is simply the time average of the buyer prices over each round induced by the seller's prices and the allocation rule in the resulting transcript of play (see Definition 3).*

We make a simple observation connecting the average payoffs of the sellers with the average prices over the rounds.

▶ **Observation 19.** *The average buyer price in a repeated pricing game exactly equals the sum of the average payoffs of the sellers.*

[1] We follow the convention in prior work in renaming the leader and follower as the learner and optimizer where one of the key cases considered is the leader playing a no-(swap-)regret algorithm. This is helpful in clarifying the roles of the players, as we will use the fact that the optimizer, playing against a no-swap-regret algorithm, can get their Stackelberg *leader* value in the stage game

[2] The $\Omega_k(1)$ notation means that this quantity remains a constant between 0 and 1 regardless of the growth of k, in contrast to an expression such as $\frac{1}{k}$.

3 Online Learning Preliminaries

Classes of Algorithms

We are interested in three specific classes of algorithms that the learner might commit to: *no-regret algorithms*, *no-swap-regret algorithms*, and *mean-based algorithms*. The learner and optimizer repeatedly play a single-stage game G for T rounds. We refer to the learner's payoff function (in the stage game) as u_L and the actions chosen by the learner and optimizer in round t as $x_t \in \Delta^n$ and $y_t \in \Delta^m$ respectively (the action spaces are distributions over finite sets of actions).While we describe these algorithms from the perspective of the learner, they are also valid algorithms for the optimizer and afford them the same properties against algorithms picked by the learner.

No-regret definitions

We consider a setting in which a learner faces an adversary, here called an "optimizer" that can choose an arbitrary sequence of actions, with knowledge of the learner's algorithm. The learner obtains a payoff $u_L(x_t, y_t)$ as a function of his own action x_t and the optimizer's action y_t. A desirable property is that in hindsight, after the sequence of learner and optimizer actions have been realized, the learner does not *regret* not having played a simple strategy (like consistently playing whichever turned out to be the best fixed action in hindsight). We say that an algorithm $\mathcal{A}$ has *regret* $r(T)$ on some sequence of length T if, regardless of the sequence of optimizer actions, the learner is guaranteed that in hindsight, they obtained payoff at least as high as they could have with the best fixed action, minus $r(T)$.

▶ **Definition 20** ($r(T)$-Regret Algorithm)**.** *An algorithm has worst-case regret $r(T)$ if, for any sequence of actions $(y_1, y_2, \ldots, y_T)$ taken by the optimizer, the total payoff of the learner can be lower bounded by*

$$\sum_{t=1}^{T} u_L(x_t, y_t) \geq \left(\max_{x^* \in [n]} \sum_{t=1}^{T} u_L(x^*, y_t) \right) - r(T).$$

An algorithm is a ***no-regret algorithm*** *if it is an $r(T)$-regret algorithm with $r(T) = o(T)$.*

We also consider a strict subset of the class of no-regret algorithms, the class of *no-swap regret* algorithms. An algorithm $\mathcal{A}$ is a no-swap-regret algorithm if it has the no regret property not just marginally, but *conditionally* on each action it played. This can be formalized by requiring that the learner's cumulative payoff is at least as high as it would have been had they, retrospectively, been able to apply some *swap function* $\pi : [n] \to [n]$ to their sequence of actions in hindsight.

▶ **Definition 21** ($r(T)$-Swap Regret Algorithm)**.** *An algorithm has worst-case swap regret $r(T)$ if, for any sequence of actions $(y_1, y_2, \ldots, y_T)$ taken by the optimizer, the total payoff of the learner can be lower bounded by*

$$\sum_{t=1}^{T} u_L(x_t, y_t) \geq \max_{\pi:[n]\to[n]} \sum_{t=1}^{T} u_L(\pi(x_t), y_t) - r(T).$$

where $\pi(x_t)$ refers to the linear extension of π acting on the support of x_t.

An algorithm is a ***no-swap regret algorithm*** *if it is an $r(T)$-swap regret algorithm with $r(T) = o(T)$.*

Here the maximum is over all swap functions $\pi : [n] \to [n]$ (extended linearly to act on elements y_t of Δ_n). It is a fundamental result in the theory of online learning that both no-swap-regret algorithms and no-regret algorithms exist (see [9]).

Some no-regret algorithms have the property that at each round, they approximately best-respond to the historical sequence of losses interpreted as a mixed strategy. Following [7] and [13], we call such algorithms *mean-based algorithms*. Formally, we define mean-based algorithms as follows.

▶ **Definition 22** (Mean-Based Algorithm). *An algorithm $\mathcal{A}$ is* $\gamma(t)$-mean-based *if whenever $j, j' \in [m]$ satisfy*

$$\frac{1}{t}\sum_{s=1}^{t} u_L(j', y_s) - \frac{1}{t}\sum_{s=1}^{t} u_L(j, x_s) \geq \gamma(t),$$

then $x_{t,j} \leq \gamma(t)$ (i.e., if j is at least $\gamma(t)$ worse than some other action j' against the historical average action of the opponent, then the total probability weight on j must be at most $\gamma(t)$). A learning algorithm is mean-based *if it is $\gamma(t)$-mean-based for some $\gamma(t) = o(1)$.*

Many standard no-regret learning algorithms are mean-based, including Multiplicative Weights, Hedge, Online Gradient Descent, and others (see [7]).

Up to low order terms, best responses to no-swap regret algorithms in algorithm space are well understood: the optimizer can do no better than to play a static non-responsive strategy which at every round plays an identical action distribution that is very close (in any reasonable norm) to their optimal Stackelberg leader strategy in the stage game:

▶ **Lemma 23** ([13]). *If the learner plays a no-swap-regret algorithm, then there exists $\varepsilon = o_T(1)$ and a $o(T)$-additive best-response of the optimizer that involves playing a distribution D' in each round such that $||D - D'||_\infty \leq \varepsilon$ where D is the optimizer's leader Stackelberg strategy of the underlying stage game.*

4 Results in the Bertrand Model

4.1 Equilibria of the Stage Game

In this section, we focus on Nash and Stackleberg Equilibria of the stage game. We remind the reader that the Nash Equilibria of the stage game define which prices are competitive. In turn, this section identifies what the baseline competitive prices are in the Bertrand model (the equivalent result for the logit model can be found in the full version).

We begin by showing that in any Nash equilibrium of the Bertrand stage game, the resulting competitive prices must be at most $\frac{2}{k}$, implying in particular that competitive prices yield in total a vanishing fraction of the monopoly revenue. To prove this, we need to show that pricing high relative to the other seller's price distribution is a dominated strategy, in a robust sense. Intuitively, it is clear that the best-response to a deterministic price is to undercut your opponent. We show that the same idea works against distributions in the following Lemma that we use throughout this section. The proof is deferred to the full version.

▶ **Lemma 24.** *Consider one seller (say seller 2) in the Bertrand stage game playing a distribution d where the total weight placed on prices above x (where $x \geq \frac{3}{k}$) is equal to b, for $b \leq \frac{1}{24k}$. Then, for the other seller (seller 1), there is some price $x' < x$ such that $u_1(x', d) \geq u_1(x, d) + \frac{1}{24k^2}$.*

The above lemma has a simple corollary which will also be useful:

▶ **Corollary 25.** *In a Bertrand stage game in which one seller's distribution is d, pricing at* $p \geq max(supp(d))$ *is a strictly dominated strategy when* $\max(supp(d)) \geq \frac{3}{k}$.

We are now ready to show what constitutes competitive prices in the Bertrand game.

▶ **Lemma 26.** *All Nash Equilibria of the Bertrand stage game are supported on prices that are at most* $\frac{2}{k}$.

Proof. We will show this by induction:
Base Case: No Nash Equilibrium (NE) of this game have support on price 1. To see this, note that by Corollary 25, the price 1 is dominated for both sellers, and thus cannot be in the support of any NE.
Induction Hypothesis: If no NE has support on price $\frac{m}{k}$ or higher and m is an integer s.t. $m \geq 4$, then no NE can have support on price $\frac{m-1}{k}$. To see this, consider any NE, (d_1, d_2) such that neither distribution is supported on prices $\frac{m}{k}$ or higher. Since $max(supp(d_1)) \leq \frac{m-1}{k}$, where $\frac{m-1}{k} \geq \frac{3}{k}$, the best-response to it cannot be $\frac{m-1}{k}$ (the proof of this is deferred to the full version). Therefore d_2 cannot have support on price $\frac{m-1}{k}$. A similar argument holds for d_1, extending the induction hypothesis.
Therefore, no Nash Equilibria of the game have support on any prices above $\frac{2}{k}$. ◀

We observe a gap between the equilibrium prices of the stage game between the Nash and the Stackelberg equilibrium. The following lemma relies upon results we will show in Section 4.2.1, but we state it here for ease of organization.

▶ **Lemma 27.** *The Stackelberg Equilibria of the stage game leads to supra-competitive prices*

Proof. By Lemma 30, the optimizer can achieve utility $\Omega_k(T)$ against a no-swap regret learner (by playing a static strategy). [13] establish that the maximum possible average payoff achievable against a no-swap regret learner is upper bounded by the stage game Stackelberg leader value. Therefore, the stage game Stackelberg leader value is $\geq \Omega_k(1)$, and thus the resulting price is $\geq \Omega_k(1)$ (see Observation 19 which lower bounds the average price by the average payoff of either seller). ◀

4.2 Results for the Repeated Bertrand Game

We begin by considering the Stackelberg equilibria of the repeated Bertrand game. As we observed earlier, allowing unrestricted commitment to any possible algorithm opens the door for explicit usage of threats to maintain collusive prices. The following lemma shows that supra-competitive prices emerge out of such commitments, and is unsurprising.

▶ **Lemma 28.** *The Stackelberg Equilibrium of the repeated Bertrand game induces supra-competitive prices.*

The proof, which is based on the algorithm of [11] to find optimal algorithmic Stackelberg strategies in repeated games, is deferred to the full version. It is known that the resulting leader algorithm is "obviously" anti-competitive, in that it encodes an explicit threat for the follower. However, we show that supra-competitive prices arise from a vast array of algorithms which are not facially anti-competitive. In particular, we consider the class of mean-based no-regret algorithms and the class of no-swap regret algorithms, which are well-behaved in the following sense: for both classes of algorithms, when both sellers use any algorithm in the class, the prices converge to competitive prices.

4.2.1 Committing to No-Regret Algorithms Induces Supra-Competitive Prices

In this section, we will prove two key results pertaining to the repeated version of the game:

1. While playing against any no-regret learner, the optimizer can always guarantee themselves at least $\Omega_k(T)$ net payoff, even when restricted to static non-responsive strategies (recall: strategies which are oblivious to their opponent's behavior and simply play the same pricing distribution every day). In fact, they attain $\Omega_k(T)$ net payoff even by playing a uniform distribution over prices each day.
2. If an optimizer attains $\Omega_k(T)$ net payoff against a no-regret Learner, that no-regret Learner also obtains $\Omega_k(T)$ net payoff.

Taken together, these two results imply that, if the optimizer, optimizing in the environment defined by the learner's no-regret algorithm, does at least as they could do using a simple static and non-responsive strategy, the induced prices are supra-competitive. Furthermore, *both* the learner and the optimizer benefit from these high prices. Finally, we will show that, when the learner is playing a no-swap regret algorithm and the optimizer plays a static non-responsive best response, this is in fact an approximate *Nash* Equilibrium in algorithm space (where the approximation is a sublinear additive factor). We emphasize that despite the fact that the optimizer happens to be playing a static, non-responsive algorithm, this is a best response for them in all of (unrestricted) algorithm space. This result in fact holds true in all games not just pricing games – but in the context of pricing games, it gives an example of a Nash equilibrium with supra-competitive prices between two algorithms which do not explicitly encode threats.

To achieve these results, we must reason about an optimizer's payoff when playing against a no-regret learner. Consider a no-regret algorithm playing against a seller who plays the same distribution in each round. This fixes the expected cumulative payoff of each fixed action for the no-regret learner; and thus, by the no-regret guarantee, the learner must play a best-response for all but a diminishing fraction of rounds. This, in turn, has implications for the follower's aggregate payoff. We capture this relationship in the following lemma. Note that this lemma applies to all bimatrix repeated games, not just pricing games.

▶ **Lemma 29.** *Consider any T-round repeated game defined by a stage game G with non-negative payoffs. Against any learning algorithm with the no regret property, if the optimizer plays a static strategy s each round, the optimizer's expected payoff is at least $u_o(BR^*_\ell(s), s) \cdot T - o(T)$, where u_o is the optimizer's value in G and $BR^*_\ell(s)$ is the element of the learner's best response to s in G that minimizes $u_o(BR^*_\ell(s), s)$.*

Proof. Let the total regret of the learning algorithm be $r(T)$, and let $BR_\ell(s)$ be the set of all best responses to s from the learner. Furthermore, let $g = u_\ell(BR^*_{\ell(s)}, s) - \max_{y \in [m] \setminus BR_{\ell(s)}} u_\ell(y, s)$.

Let p^* be the average probability distribution played by the learning algorithm against s. Since there is no correlation between the two seller's randomness, the learner's average payoff is

$$u_\ell(p^*, s) \leq \mathbb{P}[p^* = x \in BR_\ell(s)] \cdot u_\ell(BR_\ell(s), s) + \mathbb{P}[p^* = x \notin BR_\ell(s)] \cdot (u_\ell(BR_\ell(s), s) - g)$$

As the learner's algorithm has average regret $\frac{r(T)}{T}$, we have that

$$\mathbb{P}[p^* = x \in BR_l(s)] \cdot u_\ell(BR_\ell(s), s) + \mathbb{P}[p^* = x \notin BR_\ell(s)] \cdot [u_\ell(BR_\ell(s), s) - g] \geq u_\ell(BR_\ell(s), s) - \frac{r(T)}{T}.$$

Therefore, $\mathbb{P}[p^* = x \notin BR_\ell(s)] \cdot g \leq \frac{r(T)}{T}$, which implies $\mathbb{P}[p^* = x \notin BR_\ell(s)] \leq \frac{r(T)}{Tg}$, and finally $\mathbb{P}[p^* = x \in BR_\ell(s)] \geq 1 - \frac{r(T)}{Tg}$.

This allows us to lower bound the payoff of the optimizer (where the first inequality implicitly uses the fact that the both sellers always get non-negative payoff) –

$$\begin{aligned}
u_o(p^*, s) &\geq \mathbb{P}[p^* = x \in BR_\ell(s)] \cdot u_o(BR_\ell^*(s), s) \\
&\geq \left(1 - \frac{r(T)}{Tc}\right) u_o(BR_\ell^*(s), s) \\
&= u_o(BR_\ell^*(s), s) - \frac{r(T)}{T} \cdot \frac{u_o(BR_\ell^*(s), s)}{g} \\
&= u_o(BR_\ell^*(s), s) - \frac{o(T)}{T} \cdot \frac{u_o(BR_\ell^*(s), s)}{g} \\
&= u_o(BR_\ell^*(s), s) - \frac{o(T)}{T}.
\end{aligned}$$

◀

We are now ready to show that if an optimizer's only goal is maximizing their own payoff, and they do even a passable job, they will get high payoff against a no-regret learner. In fact, they can do this even by performing the extremely simple static strategy of pricing uniformly randomly every day. Recall that by Observation 19, the average payoff of the optimizer lower bounds the average price. Therefore, as long as the optimizer is optimizing in the environment defined by the learner's no regret algorithm, there is a wide range of optimizer behaviors which induce supra-competitive prices. This includes all optimizers that perform better than static random pricing, ranging from approximate optimization over static responses to exact optimal dynamic responses.

▶ **Lemma 30.** *The optimizer can get payoff at least $\Omega_k(T)$ against any no-regret learner in the Bertrand model by playing a static strategy of uniformly random pricing. Formally, for any no-regret algorithm $\mathcal{A}^{noreg}$ and the static uniformly random algorithm $\mathcal{A}^r$, $U_o(\mathcal{A}_l^{noreg}, \mathcal{A}_o^r) \geq \Omega_k(T)$.*

Proof. Let r be the uniformly random distribution over prices. By Lemma 29, the optimizer's expected payoff is at least $u_2(BR_\ell^*(r), r) \cdot T - o(T)$.

First, we will prove that any best-response of the learner $BR_\ell(r) \geq \frac{1}{5}$. To see this, note that by pricing at $\frac{1}{2}$ against r, the learner's payoff would be at least

$$\begin{aligned}
\mathbb{P}\left[r > \frac{1}{2}\right] \cdot \frac{1}{2} = \mathbb{P}\left[r \geq \lfloor \frac{1}{2} + \frac{1}{k} \rfloor\right] \cdot \frac{1}{2} \\
&= \frac{1 - \lfloor \frac{1}{2} + \frac{1}{k} \rfloor}{1 - \frac{1}{k}} \cdot \frac{1}{2} \\
&\geq \left(1 - (\frac{1}{2} + \frac{1}{k})\right) \cdot \frac{1}{2} \\
&\geq \frac{\frac{1}{2} - \frac{1}{k}}{2} \\
&\geq \frac{1}{4} - \frac{1}{2k} \\
&\geq \frac{1}{4} - \frac{1}{20} = \frac{1}{5}
\end{aligned}$$

As the learner can get payoff at least $\frac{1}{5}$, their best response price will always be at least $\frac{1}{5}$, as otherwise they would always get payoff strictly less than $\frac{1}{5}$.

We now invoke Lemma 29 to lower bound the payoff of the optimizer. The worst learner best-response to the uniform distribution, from the perspective of the optimizer, must be bounded below by $\frac{1}{5}$ i.e. the optimizer must select a price that is at least $p^* := \frac{\lfloor \frac{k}{5} \rfloor}{k}$ [3]. Additionally, the optimizer's payoff is monotone increasing in the learner's price. Thus the optimizer gets total payoff at least $T \cdot u_o(r, p^*) - o(T)$. Analyzing this gives us the desired result –

$$\begin{aligned} u_o(r, p^*) \cdot T - o(T) &\geq \mathbb{E}[r | r < p^*] \cdot \mathbb{P}[r < p^*] \cdot T - o(T) \\ &\geq \mathbb{E}\left[r | r \leq p^* - \frac{1}{k}\right] \cdot \mathbb{P}\left[r \leq p^* - \frac{1}{k}\right] \cdot T - o(T) \end{aligned}$$

$\mathbb{E}[r | r <= p^* - \frac{1}{k}]$ is exactly $\frac{p^*}{2}$ (the average of $\frac{1}{k}$ and $p^* - \frac{1}{k}$). Substituting p^* results in $\frac{p^*}{2} = \frac{\lfloor \frac{k}{5} \rfloor}{10}$, this is lower bounded by $\frac{2}{25}$ (assuming $k > 20$). On the other hand, we can rewrite $\mathbb{P}\left[r \leq p^* - \frac{1}{k}\right]$ (by substituting p^*) as $\frac{\lfloor \frac{k}{5} \rfloor - 1}{k}$. This term is monotone increasing in k and lower bounded by $\frac{3}{25}$ (assuming $k > 20$). Thus, the optimizer payoff is lower bounded by $\frac{6T}{625} - o(T)$, completing the proof. ◀

Since by Observation 19 the average payoff of the optimizer lower bounds the average price, we have already shown that any no-regret algorithm induces supra-competitive prices against any optimizer who performs better than random pricing. We will now go on to show that the learner deploying the no-regret algorithm benefits from these supra-competitive prices as well.

▶ **Theorem 31.** *For any no-regret learner algorithm $\mathcal{A}_l^{noreg}$, and for any optimizer algorithm $\mathcal{A}_o$, if*

$$U_o(\mathcal{A}_l^{noreg}, \mathcal{A}_o) \geq \Omega_k(T)$$

then

$$U_l(\mathcal{A}_l^{noreg}, \mathcal{A}_o) \geq \Omega_k(T)$$

Proof. Let $U_o(\mathcal{A}_l^{noreg}, \mathcal{A}_o) \geq c \cdot T$ for some constant c between 0 and 1, independent of k.

Note that, in order to achieve c payoff on average, the optimizer must have priced at or above $\frac{c}{2}$ with average probability (over the rounds) at least $\frac{c}{2}$. Assume for contradiction that the optimizer does not do this. Then, we can upper bound their utility by assuming they capture demand in all rounds:

$$\begin{aligned} U_o(\mathcal{A}_l^{noreg}, \mathcal{A}_o) &\leq \mathbb{P}\left[p_o \geq \frac{c}{2}\right] \cdot 1 + \left(1 - \mathbb{P}\left[p_o \geq \frac{c}{2}\right]\right) \cdot \left(\frac{c}{2}\right) \\ &< \frac{c}{2} + \left(1 - \frac{c}{2}\right) \cdot \frac{c}{2} \\ &= c - \frac{c^2}{4} \end{aligned}$$

This is strictly less than c, and thus by contradiction, the optimizer must have priced at or above $\frac{c}{2}$ with frequency at least $\frac{c}{2}$.

[3] By construction, p^* is in the set of prices available to the seller

Given this, one fixed action that the learner could have taken is to price at $\frac{c}{2}$. Then,

$$U_l(\mathcal{A}_l^{noreg}, \mathcal{A}_o) \geq \frac{1}{2}\mathbb{P}\left[p_o \geq \frac{c}{2}\right] \cdot \left(\frac{c}{2}\right)$$
$$\geq \frac{1}{2} \cdot \frac{c}{2} \cdot \left(\frac{c}{2}\right) = \frac{c^2}{8}$$

By the no-regret guarantee of the learner, therefore, we have that $U_l(\mathcal{A}_l^{noreg}, \mathcal{A}_o) \geq \frac{c^2 T}{8} - o(T) = \Omega_k(T)$. ◀

We can now combine these results, to show that, if the learner deploys any no-regret algorithm in the Bertrand model and the optimizer responds via any strategy which gets them payoff at least that of static random pricing, then prices are supra-competitive and both the learner and the optimizer get a constant fraction of the profits.

▶ **Theorem 32.** *In a Bertrand repeated game, for any no-regret learner algorithm $\mathcal{A}_l^{noreg}$ and any optimizer algorithm $\mathcal{A}_o$ such that*

$$U_o(\mathcal{A}_l^{noreg}, \mathcal{A}_o) \geq U_o(\mathcal{A}_l^{noreg}, \mathcal{A}_o^r), \tag{5}$$

where $\mathcal{A}_o^r$ is the static uniformly random algorithm, we have

$$U_o(\mathcal{A}_l^{noreg}, \mathcal{A}_o) = \Omega_k(T) \tag{6}$$

$$U_l(\mathcal{A}_l^{noreg}, \mathcal{A}_o) = \Omega_k(T) \tag{7}$$

Further, the average price is $\Omega_k(1)$.

Proof. By Lemma 30, the optimizer gets payoff at least $\Omega_k(1) \cdot T$ against any no-regret algorithm by playing static random pricing. Therefore, the optimizer's payoff utilizing any strategy which is better than static random pricing also gives them a payoff of at least $\Omega_k(1) \cdot T$. Furthermore, by Theorem 31, this implies that the learner also gets a payoff of at least $\Omega_k(1) \cdot T$. Finally, by Observation 19, this implies that the average price is $\Omega_k(1)$. ◀

We show an equivalent result for the more general Multinomial-logit model in the full version of the paper. Note that despite our treatment of the problem in a sequential play setting, neither sequential play nor commitment power is necessary for our results. In fact, in the game where leader and optimizer pick their algorithm simultaneously, if one player plays any no-swap regret algorithm and their opponent plays a static strategy corresponding to the Stackelberg leader distribution of the stage game, then this forms a Nash equilibrium in algorithm space:

▶ **Theorem 33.** *There exists an $o(T)$-approximate Nash Equilibrium in algorithm space in any repeated game consisting of a no-swap regret algorithm for the leader and a static, non-responsive algorithm for the follower.*

Proof. In particular, such an equilibrium exists between any algorithm with total swap regret $r(T) = o(T)$ and the static algorithm which plays (something very close to) the Stackelberg leader strategy of the stage game each round. To show that this is true, we will show that each is an $\frac{o(T)}{T}$-approximate best response to the other.

First, we will show that a particular distribution (which is ε-close to the Stackelberg leader strategy of the stage game) is near-optimal against any No-Swap Regret algorithm (NSR) when played on every round. By Lemma 23, there exists a distribution D' such that $||D - D'||_\infty \leq \varepsilon$, where D is the static Stackelberg leader strategy and D' is a $o(T)$-best response to any algorithm with sublinear swap regret $r(T)$.

It remains to show that NSR is optimal against this distribution. However, NSR is optimal against any static distribution played on every round. The best response against any algorithm playing a static distribution is to play (one of) the best-response action(s) each day. Therefore, the gap between the average payoff of the optimal response and the average payoff attained by the NSR algorithm is bounded by the average external regret of the NSR algorithm, which is at most $r(T) = o(T)$. ◀

Here, notably, both players are "succeeding in optimizing" since they are playing best responses in algorithm space against one another, neither is using an algorithm which encodes threats, and yet the outcome is near monopoly prices.

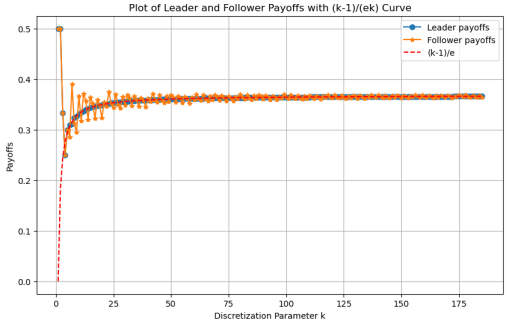
5 Numerical Investigation of Constants

We showed in Lemma 30 that the optimizer can get a payoff of $\Omega_k(T)$ against any no-regret learner, just by playing the uniform distribution over their prices. An implication is that they would do at least as well with their optimal static strategy, which was enough for our results which are stated in asymptotic notation. While we did not get attractive constants via this simple approach, we numerically verified that the optimal static optimizer distribution against a no-regret algorithm, which is the Stackelberg leader strategy for a number of values of k, results in average prices of more than $\frac{2}{3}$ (in fact approximately $\frac{2(k-1)}{ek}$). [12] give an efficient algorithm for computing the Stackelberg equilibrium of a two player game, which we use to numerically compute the Stackelberg leader strategy as well as the payoff of the Stackelberg leader, which will approximately (up to subconstant in T additive error) be the average payoff of the optimizer repeatedly playing the Stackelberg leader strategy (using Lemma 29). Our numerical results show that the Stackelberg value of the leader (as well as the follower) is approximately $\frac{k-1}{ek}$ for a range of values of k in Figure 1a. By Observation 19, if the follower optimizes over the space of static strategies, the average price for the buyer will be $\frac{2(k-1)}{ek} \geq \frac{2}{3}$. We note that while our experiments show these prices are achievable by non-responsive optimizers for all values of k from 1 to 200, it remains interesting future work to prove that this holds for all k. Finding a closed form analytical expression for the Stackelberg leader strategy of the stage game is an open problem and would likely help in proving the conjecture for all k.

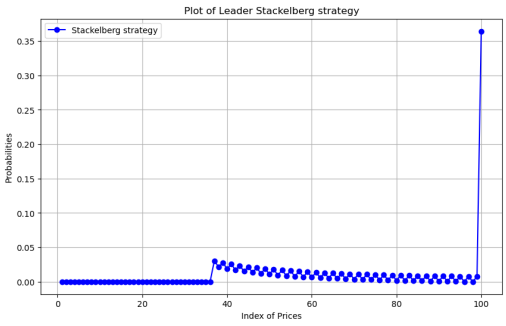
We also show the Stackelberg strategy for $k = 100$ in Figure 1b. Our numeric results also indicate that every price between 36/100 to 98/100 is (tied for being) a best response for the follower given the optimal leader commitment[4].

The code for these experiments can be found at Code Repository.

[4] In the Stackelberg Equilibrium problem, the follower typically tie-breaks in favor of the leader, in our application the optimizer adds some infinitesimal extra probability mass on price 99/100 to ensure that no-regret learner learns a unique best-response of 98/100 (which is the best outcome for the optimizer).



(a) Stackelberg Leader/Follower Payoffs for varying k.



(b) Stackelberg Leader Strategy for $k = 100$.

6 Discussion and Conclusion

Defining anti-competitive behavior is delicate. Monopoly like prices can arise in a number of scenarios: a *failure to optimize* (if e.g. players are simply not best responding to one another) as well as through *collusion* (which has been interpreted as requiring explicit *threats* when speaking of algorithmic collusion [8, 16]). Past work [17, 10] have proposed no-swap-regret algorithms as reasonable competitive algorithms in pricing scenarios, on the basis that they converge to competitive prices when both parties use them, and they seem both to successfully optimize and not to encode threats.

Our results complicate this picture. We show that if the first entrant into a market deploys a pricing algorithm with the no-swap-regret guarantee, then this very strongly incentivizes the next entrant to deploy an algorithm that will lead to supra-competitive prices. In fact, *anything the second entrant does* that obtains them profit at least that of a random pricing strategy will inevitably lead to supra-competitive prices. Moreover, this will not be at the expense of the first entrant – the no(-swap) regret learner will *also* enjoy supra-competitive revenue. And the algorithm of the second entrant can be entirely static and non-responsive, and therefore unable to encode threats. In fact, this phenomenon does not hinge on the sequential nature of play that we focus on and does not hinge on either player having commitment power. As we show, there is a Nash equilibrium of the game – in algorithm space – maintaining supra-competitive prices, which involves one player playing a no swap regret algorithm and the other playing a static pricing distribution. Both players are best responding to one another in the space of all pricing algorithms (without any restriction) and so neither player is failing to optimize – but neither are either of the players deploying threats. We suggest that this might require a reconsideration of what constitutes algorithmic collusion.

References

1 Ibrahim Abada and Xavier Lambin. Artificial intelligence: Can seemingly collusive outcomes be avoided? *Energy Engineering (Energy) eJournal*, 2020. URL: `https://api.semanticscholar.org/CorpusID:219341239`.

2 Ibrahim Abada, Xavier Lambin, and Nikolay Tchakarov. Collusion by mistake: Does algorithmic sophistication drive supra-competitive profits? *European Journal of Operational Research*, 318(3):927–953, 2024. `doi:10.1016/j.ejor.2024.06.006`.

3 Eshwar Ram Arunachaleswaran, Natalie Collina, Sampath Kannan, Aaron Roth, and Juba Ziani. Algorithmic collusion without threats, 2024. `doi:10.48550/arXiv.2409.03956`.

4 Stephanie Assad, Robert Clark, Daniel Ershov, and Lei Xu. Algorithmic pricing and competition: Empirical evidence from the german retail gasoline market. *Journal of Political Economy*, 132(3):723–771, 2024. doi:10.1086/726906.

5 Martino Banchio and Giacomo Mantegazza. Artificial intelligence and spontaneous collusion, 2023. arXiv:2202.05946.

6 Jean-Pierre Benoit, Vijay Krishna, et al. Finitely repeated games, 1984.

7 Mark Braverman, Jieming Mao, Jon Schneider, and Matt Weinberg. Selling to a no-regret buyer. In *Proceedings of the 2018 ACM Conference on Economics and Computation*, pages 523–538, 2018.

8 Emilio Calvano, Giacomo Calzolari, Vincenzo Denicolò, and Sergio Pastorello. Artificial intelligence, algorithmic pricing, and collusion. *American Economic Review*, 110(10):3267–3297, 2020. doi:10.1257/aer.20190623.

9 Nicolo Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games*. Cambridge university press, 2006.

10 Sylvain Chassang and Juan Ortner. Regulating collusion. *Annual Review of Economics*, 15(Volume 15, 2023):177–204, 2023. doi:10.1146/annurev-economics-051520-021936.

11 Natalie Collina, Eshwar Ram Arunachaleswaran, and Michael Kearns. Efficient stackelberg strategies for finitely repeated games. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, pages 643–651, 2023. doi:10.5555/3545946.3598695.

12 Vincent Conitzer and Tuomas Sandholm. Computing the optimal strategy to commit to. In *Proceedings of the 7th ACM conference on Electronic commerce*, pages 82–90, 2006. doi:10.1145/1134707.1134717.

13 Yuan Deng, Jon Schneider, and Balasubramanian Sivan. Strategizing against no-regret learners. *Advances in neural information processing systems*, 32, 2019.

14 Sara Fish, Yannai A. Gonczarowski, and Ran I. Shorrer. Algorithmic collusion by large language models, 2024. doi:10.48550/arXiv.2404.00806.

15 Karsten T Hansen, Kanishka Misra, and Mallesh M Pai. Frontiers: Algorithmic collusion: Supra-competitive prices via independent algorithms. *Marketing Science*, 40(1):1–12, 2021. doi:10.1287/MKSC.2020.1276.

16 Joseph E Harrington. Developing competition law for collusion by autonomous artificial agents. *Journal of Competition Law & Economics*, 14(3):331–363, 2018.

17 Jason D. Hartline, Sheng Long, and Chenhao Zhang. Regulation of algorithmic collusion. In *Proceedings of the Symposium on Computer Science and Law*, CSLAW '24, pages 98–108, New York, NY, USA, 2024. Association for Computing Machinery. doi:10.1145/3614407.3643706.

18 Timo Klein. Autonomous algorithmic collusion: Q-learning under sequential pricing. *The RAND Journal of Economics*, 52(3):538–558, 2021. doi:10.1111/1756-2171.12383.

19 Yoav Kolumbus and Noam Nisan. Auctions between regret-minimizing agents. In *Proceedings of the ACM Web Conference 2022*, WWW '22, pages 100–111. ACM, April 2022. doi:10.1145/3485447.3512055.

20 Rohit Lamba and Sergey Zhuk. Pricing with algorithms, 2022. arXiv:2205.04661.

21 Ryan Y. Lin, Siddhartha Ojha, Kevin Cai, and Maxwell F. Chen. Strategic collusion of llm agents: Market division in multi-commodity competitions, 2024. doi:10.48550/arXiv.2410.00031.

22 Michael L Littman and Peter Stone. A polynomial-time nash equilibrium algorithm for repeated games. In *Proceedings of the 4th ACM Conference on Electronic Commerce*, pages 48–54, 2003. doi:10.1145/779928.779935.

23 Uri Nadav and Georgios Piliouras. No regret learning in oligopolies: Cournot vs. bertrand. In *International Symposium on Algorithmic Game Theory*, pages 300–311. Springer, 2010. doi:10.1007/978-3-642-16170-4_26.

24 Denis Nekipelov, Vasilis Syrgkanis, and Eva Tardos. Econometrics for learning agents. In *Proceedings of the sixteenth acm conference on economics and computation*, pages 1–18, 2015.