The Impact of AI Design on Pricing[*]

John Asker
UCLA & NBER

Chaim Fershtman
Tel Aviv & CEPR

Ariel Pakes
Harvard & NBER

September 25, 2022

Abstract

The behavior of artificial intelligence (AI) algorithms is shaped by how they learn about their environment. We compare the prices generated by AIs that use different learning protocols when there is market interaction. Asynchronous learning occurs when the AI only learns about the return from the action it took. Synchronous learning occurs when the AI conducts counterfactuals to learn about the returns it would have earned had it taken an alternative action. The two lead to markedly different market prices. When future profits are not given positive weight by the AI, (perfect) synchronous updating leads to competitive pricing, while asynchronous can lead to pricing close to monopoly levels. We investigate how this result varies when either counterfactuals can only be calculated imperfectly and/or when the AI places a weight on future profits.

Key words: *Artificial Intelligence, Reinforcement Learning, Experience Based Equilibrium, Bertrand, Repeated Games, Learning in Games, Collusion, Antitrust.*

JEL Code: *D43, D82, D83, C72, C73, L13, L41, K21, O33*

[*]We would like to thank participants in various seminars and conferences for their comments. Excellent research assistance was provided by Lorenzo Cattivelli, Masha Melkonyan and Kylie Toh. Contact information: `johnasker@econ.ucla.edu`, `fersht@post.tau.ac.il` and `apakes@fas.harvard.edu`

1 Introduction

Firms increasingly delegate their pricing to algorithms that exploit detailed data on customers' preferences and, in some instances, use a learning process to develop strategies to play an oligopolistic pricing game in the presence of competitors.[1] Artificial intelligence algorithms (AIs) provide a prominent approach to implementation. AIs operate by learning about the returns to taking feasible actions, and then performing the action that they have learned works best. The embedded learning process is at the heart of how an AI performs. This paper shows that when pricing decisions are delegated to AIs the way in which AIs learn can have an economically significant impact on the pricing outcomes realized in the market.

This paper focuses on reinforcement learning. Reinforcement learning is a common way to implement an AI.[2] In reinforcement learning each action at each state is given a value. Learning occurs by updating these values from the information gathered by the AI during the course of play. The optimal action at each state is the one with the highest value.[3]

For clarity, our initial focus is on two learning protocols that lie at opposite extremes in terms of the information that they leverage. Consider a market interaction in which Firm A charges \$7 and its rival, B, charges \$8. The first learning approach we investigate, asynchronous learning, allows the AI to learn only from actions that are actually taken. So, in the two firm market above, all the AI can learn from is the profit that it realizes from charging \$7.[4] The second learning approach we investigate, perfect synchronous learning, allows the AI to conduct counterfactuals to assist learning. When an AI uses perfect synchronous learning, it observes its own profit and that the rival charged \$8, and it can construct the counterfactual profit that would have arisen had it chosen \$6, or any other feasible price.

Synchronous learning (whether perfect or otherwise) requires some under-

[1]See, for instance, the discussions in Chen (2016), Competition and Markets Authority (2018), Derakhshan et al. (2016), Brown and MacKay (2020), Assad et al. (2020) and Calvano et al. (2020).

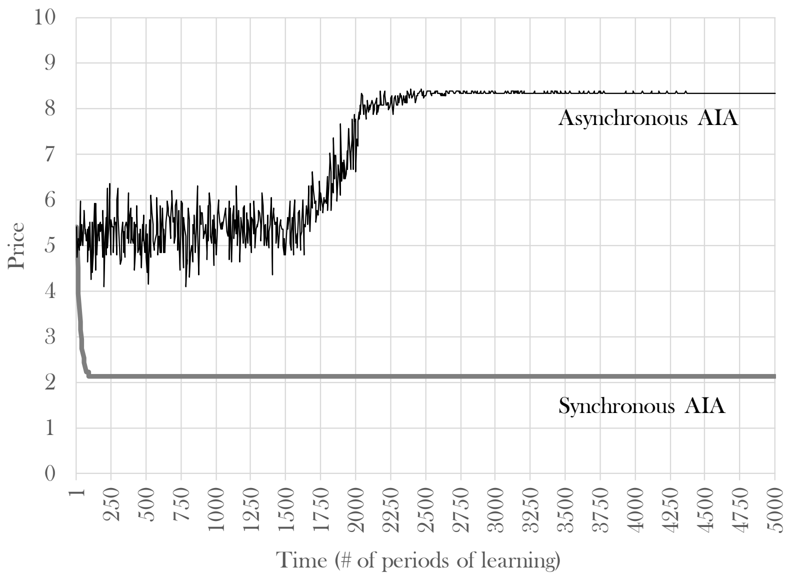
[2]See, for instance, Sutton and Barto (2018).

[3]As discussed later, there are a wide range of ways to implement this broad-brush approach. An early, seminal discussion can be found in Watkins and Dayan (1992). Sutton and Barto (2018) provide a contemporary treatment.

[4]Quite literally, the information that is leveraged is the action (charging \$7) and the realized profit ($\pi(\$7)$). No other information is used in updating the values attached to actions (i.e., learning).

standing of the underlying economic environment.[5] In the implementation discussed above, constructing accurate counterfactuals requires an understanding of demand conditions, competitors' prices, and the market clearing rule. Asynchronous learning requires no understanding of any of this. An AI using asynchronous learning needs no information other than the action it took and the profit it realized.

Figure 1: Price paths with different algorithm designs



Notes: The median price (vertical axis) across 100 simulated runs by period (horizontal axis) is shown for firm 1, in a static Bertrand market in which two firms are selling homogeneous goods. The model is parametrized as follows. Market demand is $Q = 1$ if $P \leq 10$, zero otherwise. Marginal cost = 2. There are 100 feasible prices equally spaced between 0.1 and 10 inclusive. Firms put a zero weight on future profits. The model is parametrized as per figure 2. See notes therein for further details.

The influence of these different learning protocols on pricing outcomes can be seen in figure 1. Figure 1 shows prices in a market in which two AIs

[5]We call an updating protocol *synchronous* if it allows the AI to update the return from playing prices other than the price actually paid. *Perfect synchronous* updating assumes that the AI can compute these counterfactuals perfectly (as in the example in the preceding paragraph). Other implementations, like the *synchronous using downward demand* version discussed in this paper, may use approximations or bounds to guide updating.

sell products that are perfect substitutes in Bertrand competition. The AIs do not care about future profits. Marginal costs are constant and equal to 2. Nash equilibrium prices are just above marginal costs (just above due to the discreteness in the set of feasible prices). The monopoly price is 10. When both AIs employ perfect synchronous learning, they converge to Nash pricing quickly. By contrast, when both AIs employ asynchronous learning they converge to prices that are substantially above marginal costs.

This paper endeavors to explain why this difference in outcomes occurs and evaluates the extent to which it is a general feature that is robust across a variety of environments. In doing so it seeks to contribute to the growing discussion in the economics, legal, and policy literatures on the impact of the adoption of AI pricing algorithms on the competitiveness of markets.[6]

We focus on the two extreme cases of asynchronous and perfect synchronous learning in order to make the pricing implications of learning protocols from AI pricing algorithms transparent. Much of the academic literature on AI pricing algorithms uses asynchronous updating.[7] This seems at odds with how we would expect an AI pricing algorithm to operate in a real market setting as it ignores information which likely would be viewed as helpful.

Even a simple shopkeeper who sees the demand generated at a given price realizes that this has implications for the demand that would be realized at alternative prices. The process of inferring what demand would have been at alternative prices is a synchronous updating process. The precision of the shopkeeper's estimates of demand at alternative prices would depend on the information available on the underlying demand system and on whether competitors' prices are observed. However, even an understanding that the demand function is downward sloping informs perceptions about the values of taking different actions and, as we confirm in section 5.1, can have dramatic

[6]The policy and legal debates are moving faster than the economic literature. On the policy side see Competition and Markets Authority (2018), OECD (2017), Sims (2017), and Federal Trade Commission (2018). All these references discuss algorithms, and AI in particular, as potentially facilitating collusive outcomes. Margrethe Vestager, the EU Commissioner for Competition, commented in 2018 that "The challenges that automated systems create are very real ... If they help companies to fix prices, they really could make our economy work less well for everyone else" (quoted in Hirst (2018)). For commentary see Harrington (2018), Schwalbe 2018, Assad et al (2021), and Veljanovski (2022).

[7]This is often referred to as "Q-learning." However in the machine learning (or AI) literature, Q-Learning tends to have a broader meaning that subsumes both asynchronous and synchronous leaning. See, for instance, Watkins and Dayan (1992).

effects on realized prices. We expect that a sophisticated AI pricing algorithm would 'understand' that demand slopes downward as well. As a result, the view we take in this paper is that actual AI pricing algorithms live on a spectrum with perfect synchronous and asynchronous algorithms at either end, and that the position of any algorithm on the spectrum is determined by the willingness of the firm to invest in algorithmic design, the access the algorithm has to data, and the ability the algorithm provides managers to chose alternative algorithmic settings.

The basic model, a simple Bertrand setting, with two identical firms selling identical goods, is briefly described in section 2. A more formal description of the mechanics of the reinforcement learning algorithm that we investigate makes up the rest of that section. A core result, expanding on figure 1 above, is contained in section 3. There the full set of computational results for this base case is provided. These results mirror those in figure 1 - when the future has no value (the discount factor, $\beta = 0$); i.e., asynchronous updating leads to supra-competitive pricing while perfect synchronous updating does not. The computational results delve deeper into why this is the case.

The asynchronous result is a function of how the algorithm updates the value of the action taken. In a market with competing AI pricing algorithms, both will choose to play the same action in a period if that action is perceived to have the highest value by both algorithms. When subsequent updating does not decrease the value attached to this action, or increase the value attached to other actions, the current action will become a "rest point" and the AIs will have converged to a set of stable prices. Computational results show that this process leads to supra-competitive pricing.

We then provide a proposition with two parts, one of which formalizes the asynchronous computational results, and the other shows that perfect synchronous updating, absent any value being placed on the future (i.e., $\beta = 0$) leads to competitive (Nash) pricing. The proposition requires only mild regularity conditions; in particular it allows for history-dependent pricing, different demand systems and so on, provided the algorithm is programed to maximize current profits. So we view the computational results as robust. Indeed they provide a way of computing Nash equilibrium prices anytime they are needed.

We then delve deeper into the behavior underlying the rest points reached by perfect synchronous and asynchronous updating by showing that each satisfies one of the two sets of equilibrium conditions used in the Experienced Based Equilibrium (EBE) paper of Fershtman and Pakes (2012). Asynchronous updating finds rest points that are EBE, while perfect synchronous updating finds rest-points that satisfy the Restricted Experienced Based Equilibrium (REBE) concept. This result does not require $\beta = 0$ or any restriction on the profit function or the state space (so it allows for history-dependent strategies). It therefore provides a basis for understanding the rest points generated by the broader range of algorithms we turn to next.

The base case is extended in a variety of ways in section 5. Section (5.1) extends the $\beta = 0$ case to allow for increases in the number of firms, experimentation, and intermediate cases lying between the extremes of perfect synchronous and asynchronous updating[8]. Section (5.2) first uses the equilibrium concepts introduced earlier to explain why the results for the $\beta > 0$ case can be different, and then considers the extensions listed above for that case.

As would be expected, having more firms in a market mitigates the price inflation induced by asynchronous updating. In the static setting ($\beta = 0$), a market with five firms displays little propensity to settle on a price above competitive levels.

Experimentation means that the AI periodically chooses a sub-optimal action to explore its payoff.[9] At first glance this experimentation may seem like a substitute for the informationally demanding process of perfect synchronous updating. Since, we cannot discuss rest points if experimentation continues indefinitely, we provide a series of examples in which experimentation is allowed to occur for twice the length of time that the base case asynchronous algorithm takes to converge. Experimentation is then stopped and the algorithm is continued until it reaches a rest point.

What we find is that experimentation does not fully mitigate the price-inflating propensity of asynchronous updating. In fact, experimentation has only limited success at reducing prices, and it comes at a significant cost in

[8]Appendix A.4, conducting the same exercise on quantity-setting (Cournot) games, replicates known results in Waltman and Kaymak (2008) and shows the our distinction between perfect synchronous and asynchronous learning is also relevant in that setting.

[9]Q-learning algorithms that employ experimentation learn about the values of choosing non-optimal actions by using an ε-greedy algorithm. The algorithm chooses with probability 1-ε the optimal action, and with probability ε it experiments and chooses randomly from all feasible actions. Once it choose an action it may update the value of taking that action. See Sutton and Barto (2018). Calvano et al. (2020) provide an application in the economics literature.

computational burden[10]. The limited success and additional computational burden imposed by experimentation, coupled with the potential financial cost of implementing random prices, leads us to be skeptical that naive experimentation is a practical solution to mitigating the price elevation introduced by asynchronous algorithms.

Given the ineffectiveness of experimentation in mitigating price elevation, we consider augmenting the basic asynchronous updating procedure with a simple insight from economics: that demand curves slope downward (creating a low-information form of synchronous updating). The knowledge that this is the case allows the algorithm to use an easily computable bound to update the value the algorithm assigns to counterfactual prices is inconsistent with the observed result and the assumption that demand decreases in price. Leveraging this additional structure is far more effective at mitigating the supra-competitive pricing of asynchronous updating than experimentation. It is also less costly and easier to compute.

In section 5.2 we consider settings in which the future is given positive weight by the algorithm (what we call the 'repeated' setting, with $\beta > 0$). This is done in models in which the state space includes the prices chosen in the prior period; so policies are history dependent. As in prior work (notably, Calvano et al. (2020)), asynchronous updating leads to supra-competitive pricing (indeed, often replicating monopoly). As we explain, when $\beta > 0$ perfect synchronous updating can also lead to supra-competitive pricing. We find that it does, albeit to significantly lower prices than in the asynchronous case.

This extends the results in Calvano et al. (2020) to cases where synchronous updating is used. We show analytically that one source of price elevation in the synchronous case is analogous to the incentives appearing in trigger price equilibrium; i.e., the 'threat' of lower cashflows in the future in the event of 'defection.' However, a comparison to models with no history-dependent pricing, i.e., models which do not allow the algorithm to mimic deviation cum punishment schemes to support collusion, makes it clear that a significant amount of the difference between the prices at the rest point and Nash equilibrium prices has nothing to do with such punishment schemes.

[10]There may also be a cost in terms of the revenue generated, as when the algorithm experiments it does not use what it currently views as the optimal policy.

Related Literature. Our paper is related to several of strands of literature. A number of papers investigate how AIs that employ reinforcement learning could lead to pricing outcomes that are collusive in nature (see, for instance, Calvano et al. (2020, 2021), Klein (2019), der Boer et al (2022), Epivent and Lambin (2022) and Eschenbaum et al (2022)).[11] Calvano et al consider a differentiated product pricing game with logit demands. We extend this work by examining how variation in the structure of the AI varies the degree to which prices deviate from the static Nash behavior. [12]

A related line of research considers settings in which the adoption of algorithms in the pricing process allows firms to better infer demand and how this may impact the propensity to collude. Miklos-Thal and Tucker (2019) construct a theoretical model in which algorithms enhance the ability of firms to forecast demand. They argue that this can lead firms to deviate from collusive conduct more frequently, leading to lower prices and higher consumer surplus (for a similar argument see also O'Connor and Wilson (2019) and Martin and Rasch (2022)). Hansen, Misra and Pai (2020) consider a setting in which price experimentation may make demand appear more inelastic than it actually is which leads to supra-competitive pricing. Brown and McKay (2021) present compelling empirical evidence that the adoption of pricing algorithms materially impacts pricing patterns and complement this with a theoretical exploration of how delegation to algorithms may allow firms to commit to strategies leading to higher prices than would otherwise be

[11] Johnson, Rhodes and Wildenbeest (2020) engage in a related computational study of platform competition, showing that platforms can design their marketplaces in a way to limit the prospects of collusion by merchants, and that this can be successful even when Q-learning algorithms are used by merchants. They show that platform itself can benefit from such design policies, in addition to raising consumer surplus.

[12] Waltman and Kaymak (2008) conduct a related computational experiment in a static Cournot setting. Our paper differs in investigating the impact of the sophistication of the learning process, and comparing environments with static and repeated interactions. A computer science literature also considers the role of AIs using reinforcement learning in shaping market outcomes (see, for instance, Tesauro and Kephart (2002) and Sandholm and Crites (1995)). Ongoing work in this computer science literature has focused on designing algorithms that can sustain cooperation in repeated prisoner dilemmas (see, for instance, Xue et al. (2018)). To our knowledge, this computer science literature has not explored the sensitivity of market outcomes to the basic informational requirements of the AIs learning protocol, or how that interacts with other features of a richer pricing game environment. Mnih et al (2015) compares the broader state of the AI research in computer science to human learning, and considers an extension to the AI literature.

the case.[13] Salcedo (2015) considers a somewhat similar theoretical setting, arguing that demand fluctuations can allow firms to infer the algorithms used by competitors and tailor their algorithms to better realize supra-competitive pricing. Leisten (2021) considers the competition when humans are able to limit, or override, the decisions of algorithms, and provides conditions under which algorithmic competition, even with active human oversight, can still lead to supra-competitive outcomes.

Our companion paper, Asker, Fershtman and Pakes (2022), provides a five-page introduction to some of the results discussed in depth in this paper. Its sole focus is on a subset of the the two-player static results. It omits all discussions of equilibrium, formal proofs relating to convergence, experimentation, cases with more than two players, and anything related to dynamic models ($\beta > 0$) or policy. It is best regarded as a terse summary of a subset of the results that this paper discusses in depth.

Empirical evidence as to the impact of the adoption of AI algorithms is lagging. A notable paper in this regard is Assad et al. (2020), which provides an empirical example in which the inferred adoption of algorithmic approaches to pricing by German gasoline retailers appears to have coincided with an increase in margins of up to 38%.

An energetic legal and policy debate over the collusive potential of AIs underlies much of the interest in the academic literature. Harrington (2018) provides an excellent entry-level overview, while also expressing skepticism about the ability of current cartel law to address any tendency for AIs to induce supra-competitive pricing.[14]

This paper also connects with a range of other literature in economics. Reinforcement learning, and its relationship to Nash equilibrium, has been investigated in both theoretical (see Fudenberg and Levine, 2016, for a survey) and experimental (see Erev and Haruvy, 2016, for a survey) literatures. On the theoretical side, that the AIs we investigate do not always converge to Nash pricing is not surprising given Hart and Mas-Colell (2003)'s results.

Lastly, reinforcement learning has a long history as a tool in the compu-

[13]Aparicio, Metzman and Rigobon (2021) provide related evidence in the context of supermarkets. Similarly, Normann and Sternberg (2021) provide related experimental evidence.

[14]Examples of contributions directed at this policy debate include Mehra (2015), Ezrachi and Stucke (2017), Kuhn and Tadelis (2017), Schwalbe (2018), de Coniere and Taylor (2020), Assad et al (2021), and Veljanovski (2022). Goldfarb, Gans, and Agrawal (2019) provide a broader overview of the likely impact of AI on the economy at large.

tation of dynamic games (see Pakes and McGuire (2001) for an early implementation in a rich oligopoly model). Feshtman and Pakes (2012) propose the experience-based equilibrium concept which we come back to below and is extended in Asker, Fershtman, Jeon, and Pakes (2020).

2 Model

This section introduces AI pricing algorithms highlighting a choice that must be made in implementation. To do so in as transparent a context as possible, we focus on results for a simple model of a market – the homogenous good Bertrand duopoly. We begin by describing this market environment, and then explain the different ingredients of the AI algorithm that determine its pricing policies. The next section considers generalizations and a more formal result.

2.1 The Bertrand pricing game

Consider two firms $i \in \{1, 2\}$ with equal marginal costs: $c_1 = c_2 = c$. Firms are Bertrand competitors selling homogenous goods. We discretize the action space and take the set of possible prices to be $\mathcal{P} = \{p^1, ..., p^M\}$, with $p^{m+1} - p^m = \xi$, so the increment from one price to the next is constant. The prices of firm i are denoted $p_i \in \mathcal{P}$. We let $D(p)$ be the market demand function. We assume that both firms have the capacity to serve all the demand they face. We assume that consumers buy from the firm with the lowest price. In case of a tie, firms split demand equally. Thus the demand faced by firm i is

$$d_i(p_i, p_j) = \begin{cases} D(p_i) & \text{if } p_i < p_j \\ \frac{D(p_i)}{2} & \text{if } p_j = p_i \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

For simplicity we assume that there is a price $V > 0$ above which the demand is zero and we assume that $p^M = V$. We denote firm i's profits as $\pi_i(p_i, p_j)$. In the computational example, we make the assumption that the same quantity is demanded $\forall p < V$. This simplifies implementation.[15] In this

[15]We assume such a structure purely for computational convenience. Any function where residual demand is decreasing in own price would suffice for our major results.

case, if $c \notin \mathcal{P}$ then there are two Nash equilibria for this (static) Bertrand game.[16] The first one is $p_i = p_j = p^c$, which is the lowest regular price greater than or equal to marginal cost, c. The second one is $p_i = p_j = p^{c+1}$. Where $p^{c+1} = p^c + \xi$. If $c \in \mathcal{P}$ then there are three equilibria, all symmetric, corresponding to both firms pricing at either c, $c + \xi$ or $c + 2\xi$. [17]

We allow this stage game to be repeated many times. This repetition allows the AIs, which determine firms' pricing policies, to learn to play. Ultimately we are interested in the pricing outcomes that the AIs converge on. Given this, we are agnostic as to whether the learning occurs off-line, through simulated play, or on-line, through learning from actual interactions with other AIs (or any combination of the two). We now turn to a more detailed description of the AIs.

2.2 The design of the AI Algorithm

The AI algorithm we use to play the Bertrand pricing game is a reinforcement learning algorithm.[18] In a reinforcement learning algorithm the coder defines a goal for the algorithm (e.g., maximizing profits), sets a procedure for determining initial values for all feasible actions, and formulates a rule which is used to update those values. Whenever it is called on to act, the algorithm chooses the highest valued action (the "greedy" action) with some probability (which may be one) and experiments with other actions otherwise. Having acted, it observes the outcome and updates the values it attaches to actions with an updating rule that the algorithm specifies ex ante. It is this updating that allows experience to inform policies; i.e., the updating rule determines how the algorithm learns. Reinforcement learning algorithms are used for a wide range of problems in industry (see, for examples, Sutton and Barto (2018)), and have been used in the economics literature to study the

[16]For the moment, we ignore the possibility that repetition may enable history-dependent strategies.

[17]This equilibrium structure is purely a consequence of the set of feasible prices being discrete. If prices were continuous, this would reduce to the familiar undergraduate example in which $p = c$ is the equilibrium.

[18]We use a variant of Q-learning which was introduced by Watkins (1989). For a survey of the different methods of implementing reinforced learning algorithms, see Sutton and Barto (2018).

interactions of firms in a market for some time.[19]

We will take the goal of the algorithm to be to maximize the expected value of the discounted flow of profits the firm earns. For simplicity our baseline will set the discount factor (β) to zero, resulting in algorithms seeking to maximize present profits, but we will also consider cases where $\beta > 0$, so the discussion below applies to the latter case as well.

A reinforcement learning algorithm has the following ingredients.

(i) Each firm has a set $\mathcal{S}_i$ which is a set of states for firm i (possibly a singleton). The elements of $\mathcal{S}_i$, i.e., the $s_i \in \mathcal{S}_i$, are the components of the firm's information set that the firm conditions on when it determines which action to take. We assume that $\mathcal{S}_i$ has a finite number of elements.[20]

(ii) A set of numbers (or values) for every firm which can be interpreted as the firm's perception of the expected discounted values of net cash-flows conditional on its state and action. That is, for every firm i

$$\mathcal{W}_i = \{W_i(p|s_i)\}_{p\in\mathcal{P}, s_i\in\mathcal{S}}.$$

(iii) A method for choosing an action (in our case, price) in every iteration conditional on $\mathcal{W}_i$.

(iv) An updating rule: The updating rule uses the information the firm observed during that period to update the $\mathcal{W}_i$'s. That is, it is a function that maps $\mathcal{W}_i^k$ into $\mathcal{W}_i^{k+1}$ based on the information observed at iteration (or period) k.

In our context a "firm" in the above description is an algorithm. We will be primarily concerned with how the properties of algorithms that converge

[19]Pakes and Mcguire (2002) show how to use reinforcement learning to compute Markov Perfect solutions to dynamic games, Fershtman and Pakes (2012) and Asker et al. (2020) provide their relationship to EBE (a notion of equilibrium used below), and Calvano, Calzolari, Denicolo, and Pastorello (2020) use them to study pricing in markets where prices are set by computer algorithms. Igami (2020) discusses the many links between AIs and applied econometric practice, particularly in the estimation of dynamic models.

[20]We make this assumption to ensure that at least some subset of the state space is visited repeatedly (i.e., "infinitely often"). This will allow us to formally examine limits of the price process.

to a rest point differ with the updating rule built into the algorithm, and, to the extent that it helps understanding, the relationship of those rest points to notions of equilibrium. To be clear about what we mean by this statement, we introduce the following definitions.

Definition 1 ***(Convergence)****: We say that the AI pricing algorithm has converged at iteration k^* if policies at all $k \geq k^*$ are the same as policies at k^* for all $s \in S$.*

Definition 2 ***(Rest Point)****: We will say that the AI pricing algorithm reaches a rest point by iteration $k = k^*$ if for all $k \geq k^*$ prices are constant, i.e., $\{(p_i^k, p_j^k) = (p_i, p_j)\}_{k \geq k^*}$.*

An AI algorithm can converge to a set of strategies which do not satisfy the rest point condition. For example the converged process might put positive probability on more than one price vector, as in the Edgeworth cycle example studied in Maskin and Tirole (1987a). On the other hand, if the pricing algorithm has reached a rest point the Markov process underlying it has converged.

When designing a reinforcement learning algorithm, certain choices need to be made. As will be shown in the rest of the paper, these choices can have a material impact on the observed price outcomes. As a result, some comments on each of the components of the algorithm will prove useful.

The state space $\mathcal{S}_i$: In a reinforcement learning algorithm the state space defines what the algorithm knows about its environment, and hence what it can condition its actions on. If the state space is a singleton, then from the point of view of the algorithm every period is the same as every other. If the state space includes some information about the history of past play, then it becomes possible for the algorithm to condition actions on that history (although the extent to which it does so will depend on details of the algorithm). Similarly, if the environment is changing (say, demand shifts) and this change is registered in the state space, then the algorithm can condition actions on that information.[21]

[21]The influence of the state space on the set of feasible policies mirrors the influence of the state space on the types of strategies that can be played in a dynamic game. It is common to restrict state spaces in dynamic games to only include payoff relevant and informationally relevant variables (see, Maskin and Tirole (2001), Fershtman and Pakes (2012) and Asker et al. (2020)). For the purposes of this paper, the only necessary restriction is that the state space be finite.

Firms in a given market may condition on different state spaces. If they do, then $S_i \neq S_j$, and if these are the only two firms then the market's states are given by $S = (S_i, S_j)$. For example, firms may condition their prices on different functions of what they observed in the past[22], and not all variables observed by one firm need be observed by other firms.

The values $\mathcal{W}_i^k = \{W_i^k(p|s_i)\}$: Each possible action at each state has a value attached to it in each period (or iteration), our k. The designer of the algorithm determines the procedure for choosing the initial values assigned to each element of $\mathcal{W}_i$, say $\mathcal{W}_i^0 = \{W_i^0(p|s_i)\}_{p\in\mathcal{P},s_i\in\mathcal{S}}$.

The choice of action: In the absence of any experimentation by the reinforcement learning algorithm, the algorithm chooses the action with the highest $W_i^k(p|s_i)$ at the state (the s_i) it finds itself in. That is it pursues a 'greedy' policy which we denote by $p^{k,*}$ where $p^{k,*} \in \arg\max\{W_i^k(p|s_i)\}$. When experimentation is possible we consider variants on '$\epsilon-$greedy' policies. That is, the greedy policy is pursued with probability $1 - \epsilon^k$, and with probability ϵ^k a policy is selected randomly. The k indexes the ϵ^k to allow it to vary with the iteration of the algorithm.

Note that there is a sense that experimentation is costly, as the player does not use the action it perceives to be optimal at the time of play. This loss must be set against the increment in future profits generated by exploring what the returns might be from other actions: the familiar exploitation versus experimentation tradeoff which we return to below.

The updating rule. *After* every period k the algorithm updates one or more of the values $W_i^k(p|s_i)$ that are in memory. That is, for at least some combinations of p and s_i, $W_i^k(p|s_i)$ gets updated and transitions to $W_i^{k+1}(p|s_i)$. If a $W_i^k(p|s_i)$ is not updated, $W_i^{k+1}(p|s_i) = W_i^k(p|s_i)$. There are different possible updating rules, and the rules that are feasible are limited by both the information the algorithm can access on competitor's play and by what market-specific knowledge the algorithm has built into it.

It is useful to begin by distinguishing between two extremes of possible updating rules.

1. *Asynchronous Updating:* If the state in the current period is s_i, then

[22]For an analysis of an AI pricing algorithm that takes this explicitly into account see Brown and McKay (2020).

only $W_i^k(p_i^k|s_i)$ is updated, where p_i^k is the action (price) chosen by firm i in period k.[23]

2. *Synchronous Updating:* If the state in the current period is s_i, then $W_i^k(p|s_i)$ is update for all $p \in P$ at that s_i.

We now detail the updating rules we use in this paper. Let s_i^k be the state of firm i in period k and p_i^k (p_j^k) be the price chosen by firm i (j). To make our points in a transparent way, most of the text assumes that the transition from the current state (s_i^k) to the state in the next period (s_i^{k+1}) is deterministic (later we return to the complications that arise if experimentation is built into the algorithm). If $W_i^k(p|s_i)$ is changed during the update

$$W_i^{k+1}(p|s_i^k) = \tag{2}$$

$$\alpha(k)\left[\pi_i(s_i^k, p) + \beta\left(\max_{y\in\mathcal{P}}\{W_i^k(y|s_i^{k+1})\}\right)\right] + (1-\alpha(k))W_i^k(p|s_i^k).$$

where β is the discount rate. Otherwise $W_i^{k+1}(p|s_i^k) = W_i^k(p|s_i^k)$.

Equation (2) implies that the change in perceived value of play ($W^{k+1}(\cdot) - W^k(\cdot)$) is α_k times the difference between the observed value of the k^{th} period's outcome from choosing p, computed as profits plus the discounted expected continuation value, and the initial (or k^{th} period) perceived value of the action. So $\alpha(k) \in [0, 1)$ determines the impact of current observations on perceived values.

When *asynchronous* updating is used, $W_i^k(p|s_i)$ is updated only for $p = p_i^k$ at $s_i = s_i^k$. In this case $\pi_i(s_i^k, p)$ is the realized profits of firm i in iteration k, and this is the only object the algorithm needs to know to do the update. But when *synchronous* updating is used, $W_i^k(p|s_i)$ can be updated for all $p \in P$ at $s_i = s_i^k$. In our extreme case we assume the algorithm can calculate the $\pi_i(p, p_j^k)$ exactly. We call this *perfect synchronous* updating. This would be possible if p_j^k is observed by the algorithm, and it knows the market demand function and its own costs, which is the case in the initial perfect synchronous algorithm we compute. This coding of the *perfect synchronous* and the *asynchronous*, that is endowing the perfect synchronous algorithm

[23]The terms synchronous and asynchronous are slight abuses of language in the context of the broader reinforcement learning literature. In that literature 'asynchronous' is used to refer to any algorithm in which only a subset of the $W_i(p|s)$'s in the system are updated at each iteration, and that subset can vary with s (see Sutton and Barto (2018)).

with the ability to compute the profits that would have been earned had it taken a different action but assuming the asynchronous algorithm can only learn from the profits generated by the action taken, makes it easy for us to illustrate the issues we focus on.

For both the perfect synchronous and asynchronous algorithms, we need an initial set of perceptions, a $\mathcal{W}_i^0 = \{W_i^0(p|s_i)\}_{p \in \mathcal{P}, s_i \in \mathcal{S}}$. For all the cases we consider in this paper, we take this to be a random draw from the same distribution (specified below).

Baseline Parameterization. In our base case, $\mathcal{S}_i$ is a singleton, $\alpha(k) = \alpha$, $\beta = 0$, and both firms have the same profit function, resulting in the updating rule (equation 2) reducing to

$$W_i^{k+1}(p) = \alpha\pi(p, p_j^k) + (1 - \alpha)W_i^k(p). \qquad (3)$$

After illustrating what happens in this case, we move to more general cases.

More generally the possible updating processes would depend on (i) what the algorithm observes about competitors' play, and (ii) what it knows about the primitives of the game (in our case the demand and cost functions). A reinforcement learning algorithm with perfect synchronous updating can be quite sophisticated. In every period, it updates the values of all feasible prices at the current state and so necessarily considers the counterfactual profits it would have received had it chosen an alternate price. To correctly calculate these counterfactuals it would have to have a model of demand and costs, which is the case in the perfect synchronous updating model that we focus on. By contrast, an algorithm with asynchronous updating is rather simple; all the updating process does is use its current price and profits to compute an update on the values of the profits it expects to receive from that price.

Between the perfect synchronous and asynchronous updating there is room for protocols that incorporate more or less information and approximations that are understood to be noisy or biased. In the example we illustrate below, the firm knows its own costs and quantity, does not know the price of its competitors or the form of the demand function, but does know that its residual demand is downward sloping in its own price. It updates the $W_i^k(p|\cdot)$ at all prices less than the price played whose value can be rationalized only by quantities less than the quantity it received, and it updates all $W_i^k(p|\cdot)$ at higher prices which require quantities greater than the quantity it received.

A comparison between possible updating rules is really a comparison of the sophistication of the AI algorithms the firms employ. This is likely to depend on the complexity of their economic environment and the willingness to devote resources to supporting the AI program underlying the algorithm.

3 Rest Points of Different AI Algorithms

We begin with computational results from the Bertrand pricing game introduced in section 2.1. Firms have a discount rate of zero, so they do not care about future payoffs, and the state space is a singleton, so history-dependent strategies are not feasible. Note that either a discount rate of zero or an inability to formulate history-dependent strategies makes repeated game interactions that could support prices above a static Nash-in-price equilibrium, like trigger price strategies, unattainable.

3.1 Computational results

This section provides computational results for two homogenous product static Bertrand games; one game uses an AI algorithm that updates synchronously and the other asynchronously. Both algorithms use the updating rule in equation 3 with $\alpha = 0.1$ when updating, and take "greedy" actions (i.e., they always chose the price which their perceptions indicate has the highest profits). Both algorithms also use the same procedure to obtain their initial perceptions of the value associated with all feasible actions.

Our parameterization has demand equal to zero for any price above ten (which is also the monopoly price), and equal to one for any price below or equal to ten. Marginal cost is equal to two. Feasible prices are given by a grid with 100 elements, equally spaced between 0.01 and 10, inclusive. There are two Nash equilibria to this game – one in which both firms play 2.03 and the other in which both firms play 2.13. For initial perceptions of the values associated with each price, we draw from a uniform distribution with endpoints ten and twenty (or $U[10, 20]$). The figures we present contain quantiles of the distribution of a hundred price paths we generate in this way.

Note that the initial perception of profits from playing any given price is always above any possible profit outcome (though the extent to which they are above is random). As is well known in the reinforcement learning literature, the fact that each sample path starts with initial values that are higher than any possible outcome induces exploration[24]. We deal with different ini-

[24]That is there are two ways to induce an algorithm to explore. One is through relatively

tial conditions distributions when we consider formal properties of the two updating algorithms below.

Figure 2 summarizes pricing outcomes for the baseline parametrization described in section 2.[25] Panel (a) of this figure provides quantiles of the price paths from an algorithm when both firms employ perfect synchronous updating; panel (b) shows price paths when both employ asynchronous updating. There are five lines in each panel of figure 2. From bottom to top these are, by period, the min, 25th percentile, median, 75th percentile, and max price across the 100 simulations that are run.

Start with panel (b) of figure 2. Convergence is relatively slow, but after 4600 periods none of the paths simulated from the AI algorithm with asynchronous updating change their preferred actions and all of the percentiles stay constant.[26] Most notably there is a distribution of rest point prices, but *all* are significantly higher than the Nash equilibrium prices. The median price is 8.34 and the minimum is 5.06. Note also that all quantiles tend to increase over the course of the learning process; i.e., they move *away* from the Nash equilibria.

By contrast, when the AI learns via perfect synchronous updating prices converge quickly to Nash pricing levels. In this instance all prices, across all 100 simulations settled on 2.13 after approximately 105 iterations. Thus learning was much faster, price outcomes corresponded to a static Nash outcome, and initial conditions had no influence on the limiting prices (there is no variance in the final prices over the 100 simulations).

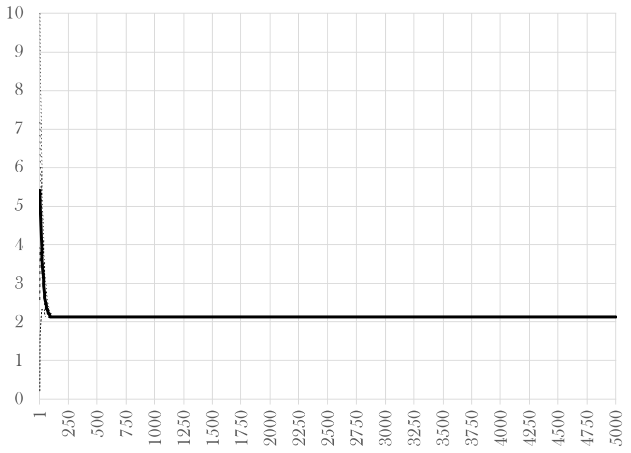
Figure 2 illustrates the importance of the learning (or updating) process in understanding the implications of price competition between AI algorithms on the outcomes that may emerge. Asynchronous updating leads to supra-competitive ('high') prices. perfect synchronous updating leads to pricing that is in line with what economists would think of as a competitive outcome (the static Nash equilibrium in prices). Importantly, this happens in a setting in which firms do not care about the future and are unable to play history-dependent strategies. Thus though standard collusive equilibria of the sort familiar from the repeated game literature are not feasible, the asynchronous

large initial perceptions of different prices; a high $\{W^0(p|s)\}_{p\in P, s\in S}$. Another is through experimentation. Below we augment the algorithm to allow for experimentation, and hence both kinds of exploration.

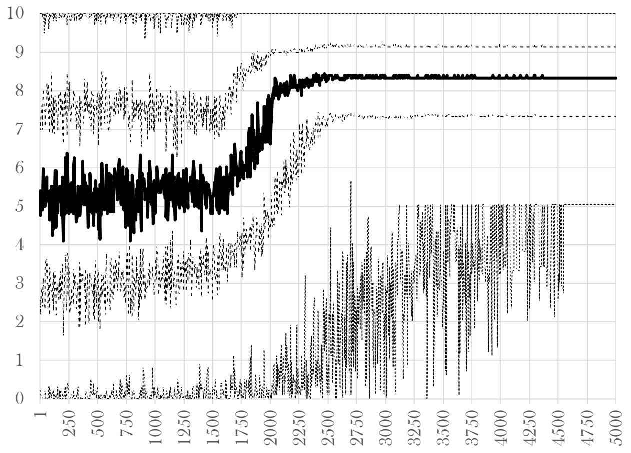
[25]The full details of the parametrization are found in the notes at the bottom of the figure.

[26]Most paths settle on a single price by iteration 2500.

Figure 2: Price outcomes with different algorithm designs



(a) perfect synchronous Updating



(b) Asynchronous Updating

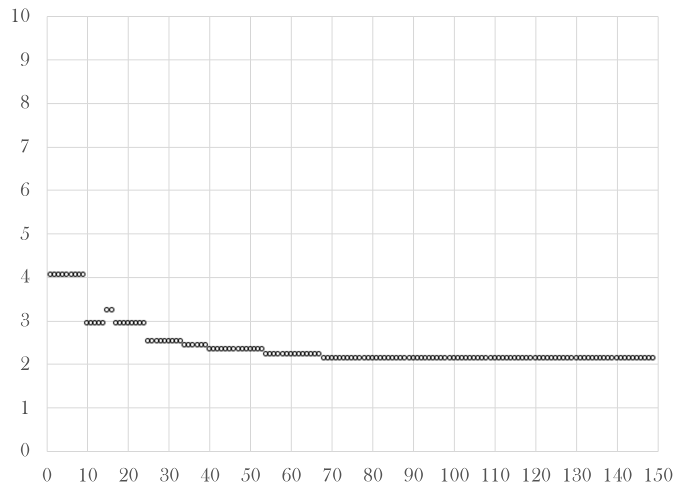
Notes: Prices (vertical axis) by period (horizontal axis) from 100 simulations are shown. The lines, from bottom to top, represent the min, 25th percentile, median, 75th percentile, and max of the distribution of prices in each period. Results are for a static Bertrand market with two firms selling homogeneous goods.. Results are shown for firm 1. The model is parametrized as follows. Demand, $Q = 1$ if $P \leq 10$, zero otherwise. Marginal cost = 2. Feasible prices exist on a grid with 100 elements equally spaced between 0.1 and 10 inclusive. Firms put a zero weight on future profits (the future is discounted to zero). The state space is a singleton. The weight on current returns in updating is given by $\alpha = 0.1$. Initial conditions are i.i.d. draws from $U[10, 20]$, for each $W(p)$ for each firm. For the core code see appendix A.1.

algorithm always generates rest point prices that are significantly above the Nash equilibrium values.

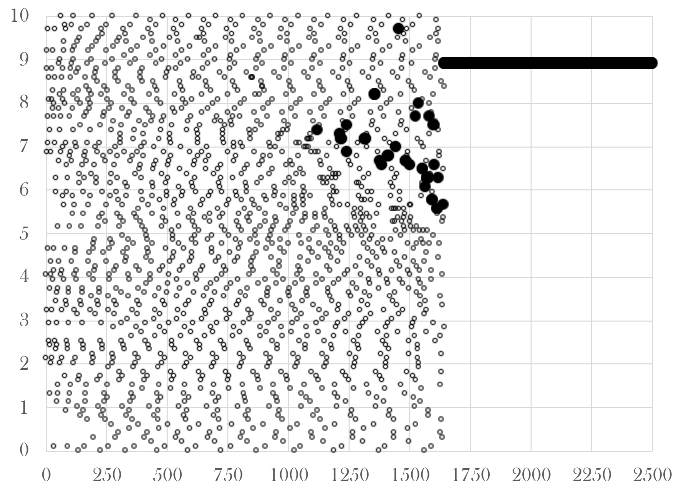
Figure 3 provides more detail on the learning process. Each panel in figure 3 shows the convergence process for a set of initial conditions drawn from U[10,20]. As in figure 2, panel (a) corresponds to the perfect synchronous updating case and panel (b) corresponds to the asynchronous updating case. Each panel shows the prices chosen by firm 1's AI with circles. The hollow circles indicate chosen prices for which the $W^k(p)$ is not updated upward; solid circles indicate chosen prices for which the $W^k(p)$ is updated upward.

In panel (a) of figure 3, the perfect synchronous case, convergence is

Figure 3: Convergence with different algorithm designs



(a) perfect synchronous Updating



(b) Asynchronous Updating

Notes: Prices (vertical axis) by period (horizontal axis) are shown for a firm 1, in a static Bertrand market with two firms selling homogeneous goods. Hollow circles indicate (chosen) prices for which the $W(p)$ is updated downward. Solid circles indicate (chosen) prices for which the $W(p)$ is updated upward. The model is parametrized as per figure 2

fast. The Nash pricing outcome of 2.13 is reached by period 70. Initially, all perceptions are higher than the profits that result from the price chosen. This leads to the updating process (3) in the perfect synchronous case adjusting perceptions of the profits from *all* prices downward. For example, in iteration 1, initial conditions on $W(p)$ lead firm 1 to choose $p = 4.05$, and its the rival chooses 0.92. Firm 1 does not earn any profits from this price combination. In iteration 3, after $W(p)$'s have been updated twice, the rival chooses 3.14. Next, in round 10, firm 1 changes its choice to 2.94, just undercutting 3.14. This, however, coincides with the rival raising its chosen price to 3.84 (which more profitably undercuts firm 1's original choice of 4.05). This combination of chosen prices continues until iteration 15, when firm 1 raises its chosen price to 3.24. After one iteration of updates to the rival's $W(p)$, 3.14 becomes the rival's best choice. Then, in iteration 17, the cumulative updates of firm one's $W(p)$'s are such that 2.94 becomes the best price. From that point on, a pattern of intermittent undercutting develops until the prices converge. This convergence pattern is common, with the influence of initial conditions in early periods leading to a small amount of non-monotonicity, as seen here.

In panel (b) of figure 3 (the asynchronous updating case) convergence is non-monotonic and slow. The prices converge to 8.89 in period 1,645. As in the perfect synchronous case, the fact that the initial condition draws are higher than monopoly profits, implies that in early periods, the evaluations fall. That is, when a price p^k is chosen, it generates a lower profit than the $W^k(p)$ associated with p^k, and so $W^k(p^k)$, and in this asynchronous algorithm only $W^k(p^k)$, is adjusted downward. Eventually the initial valuation of some other price is higher than the profit earned from the given price, so some other price will be chosen. Since only the price that is chosen has its $W^k(p)$ updated, and the second price also starts from a perception which is higher than monopoly profits, it takes quite a few iterations before the chosen price can generate a profit which is higher than the iteration's perception of its value.

The first time the chosen price generates a profit that is higher than its initial valuation, i.e., $W^k(p^k)$, is in iteration $k = 1,119$. This leads to an upward adjustment and a solid black circle in the bottom panel of figure 3. As long as the price of the rival is not reduced to match or undercut the price, firm i does not change its price. Since at k = 1,119 the rival's price is greater than $p^{1,119}$, it obtains no profits and its valuation of its chosen price falls. Eventually it chooses a different price which, if equal to or lower than $p_i^1, 119$, generates positive profits. If the rival's new price is lower, firm i's profits go to zero and its evaluation of $p_i^{1,119}$ falls. However if the rival happens to chose $p_i^{1,119}$ both firms earn positive profits. If, in addition, at that k, $\pi(p^k, p^k) \geq \max[W_i^k(p), W_j^k(p)]$, the algorithm will have converged to p^k.

In this example, the two firms matched on price 22 times before the second requirement for convergence. The asynchronous updating and equation (3) insures that for $k > 1,119$ the absolute value of both $[W_i^k(p*) - \pi(p*, p*)]$ and $[W_j^k(p*) - \pi(p*, p*)]$ converge monotonically to zero, and if $p \neq p*$, $W^k(p)$ remains forever at $W^{k=1,119}(p)$.

3.2 Theoretical results

It should be clear that the price the asynchronous algorithm rests at depends on initial conditions, hence the distribution of converged values illustrated by the quantiles of the sample paths in 2. Indeed, provided the distribution of initial conditions is sufficiently rich, the limiting outcome from the asynchronous algorithm has a positive probability of any $p \in \mathcal{P}$. By contrast, the perfect synchronous algorithm can only converge to a Nash equilibrium. The following provides a formal justification of these points, and establishes that the computational results reflect intrinsic differences in how asynchronous and perfect synchronous updating function.

Proposition 1 *Consider the duopoly pricing game between AI algorithms described above with initial conditions drawn randomly from a distribution which puts positive probability on $\hat{\mathcal{W}}^0 \subset \mathcal{W}^0$ such that $W_i^0 \in \hat{\mathcal{W}}^0 \geq max_{(p_i,p_j)}\pi(p_i,p_j)$.*

1. *If the AI algorithms use perfect synchronous updating*
   - (a) *and the two algorithm choose a pair of prices which is a Nash equilibrium to the static pricing game, they will continue to charge these prices in every future period, and, in addition,*
   - (b) *any rest point of the algorithm must be a Nash pricing equilibrium of the duopoly game.*
2. *If the AI algorithm uses asynchronous updating, there is positive probability that the algorithm converges to any $p \in \mathcal{P}$.*

Proof: (1a). Assume that at period k, which is the initial condition for our inductive argument, the two algorithms charge a pair of prices (p_1^N, p_2^N) which is a Nash equilibrium of the duopoly pricing game. This implies that $\pi_1(p_1^N, p_2^N) \geq \pi_1(p, p_2^N)$ for every $p \in \mathcal{P}$. The updating of the values of the algorithm of firm 1 at period k is done for every $p \in P$ by the updating rule $W_1^{k+1}(p) = \alpha\pi_1(p, p_2^N) + (1-\alpha)W_1^k(p)$ (and analogously for firm 2). Given that $\pi_1(p_1^N, p_2^N) \geq \pi_1(p, p_2^N)$ and that $W_1^k(p_1^N) = \max_{p\in P}\{W_1^k(p)\}$, $W_1^{k+1}(p_1^N) = \max_{p\in P}\{W_1^{k+1}(p)\}$ and therefore the algorithm will choose p_1^N at iteration $k+1$. The same holds for the second firm, and therefore the prices at iteration $k+1$ are (p_1^N, p_2^N), which proves the inductive step of the argument. □

Proof: (1b). Assume to the contrary that there is a k^* such that $\forall\ k > k^*$, $W^k(p_1') = \max_p W_i^k(p)$ with the analogous condition holds for firm j, and let $b_1(p_2')$ be the best response of player 1 in the duopoly pricing game. [27] For the contrary assertion to be true $b(p_2') \neq p_1'$ and $\pi_1(b_1(p_2'), p_2') > \pi_1(p, p_2')$ for every $p \in P$. With synchronized updating $W_1^{k+1}(p) = \alpha\pi_1(p, p_2') + (1 - \alpha)W_1^k(p)$ for every $p \in P$. Consequently, $\lim_{k\to\infty} W_i^k(p_1') = \pi(p_1', p_2') < \lim_{k\to\infty} W_i^k(b(p_1')) = \pi(b(p_2'), p_2')$, a contradiction. $\square$

Proof: (2). Sufficient conditions for p^* to be a rest point to the game when updates are asynchronous are: (i) $W_i(p^*) \geq W_i(p)$ for every $p \neq p^*$ and $i = 1, 2$, and (ii) $\pi_i(p^*, p^*) \geq W_i(p)$ for every $p \neq p^*$ for $i = 1, 2$. Under condition (i) the firms indeed choose the price p^* and condition (ii) guarantees that condition (i) will continue to holds in all future periods. For any p^* there is a positive probability that the random draw on the initial conditions satisfies both (i) and (ii). $\square$

Remark None of the three parts of this proposition depend on the exact structure of the profit function, the updating function provided $\alpha \in (0, 1)$, or the initial condition distribution provided that it satisfies our "sufficiently rich" support condition. As a result, though we have computed results for the same two algorithms from the logit differentiated product case, for quantity-setting games and for different initial conditions distributions we omit most of them from the main text. We discuss quantity-setting (Cournot) in appendix A.4 and logit results in appendix A.2.

4 Equilibria for Algorithmic Pricing Games

The rest points of the asynchronous and perfect synchronous algorithms satisfy different equilibrium conditions, and an understanding of those conditions clarifies why and when different learning rules imply different rest points for AI pricing games. Unless otherwise stated, the results in this section apply to any pricing algorithm that abides by the general description of the algorithms given in section 2.2. The relevant equilibrium conditions are provided in Fershtman and Pakes (2012), and Asker et al. (forthcoming) and the reader interested in a more formal analysis of their implications should

[27] We assume for convenience that there is always a single best response. A best response will always exist since the set of prices is finite.

consult those papers.

In the general case, the algorithm's choice of actions, the choice of $p \in \mathcal{P}$, can differ with $s \in S$ (for e.g., as a function of past prices). If the algorithm has converged, those policies are constant thereafter for each $s \in S$. Any set of such policies generates a Markov chain on S; that is, the state at iteration k together with the prices chosen at that iteration generate a transition to the state at iteration $k+1$. Any finite state Markov chain will eventually wander into a recurrent subset of the points in S, say $\mathcal{R} \subset S$, and stay within it forever. So all rest points of AI pricing games are contained in the recurrent class of the Markov process generated by the policies that the AI converged to. Since S_i can include functions of past prices, this includes policies with history-dependent strategies.

The equilibrium concepts we consider are all refinements of experience based equilibrium (henceforth EBE). An EBE has three components:

- a subset of the state apace, denoted by $\mathcal{R} \subset S$.
- strategies for each i, say $\{p_i^*(s)\}$ for every $s \in S$, and
- a set of values for each i, our $\{W_i(p|s)\}_{p,s}$, that have the interpretation of beliefs about the expected discounted value of profits were the agent to play price p_i at state s_i.

For these objects to satisfy the conditions of an EBE it must be the case that

1. the strategies are optimal given the beliefs embodied in the $\{W_i(p|s)\}$, i.e., $p_i^*(s) = \arg\max_{p\in\mathcal{P}}\{W_i(p|s)\}$ for all i and all $s \in S$,
2. $\mathcal{R}$ is a recurrent class of the Markov process generated by these strategies, and
3. at all $s \in \mathcal{R}$, $W_i(p^*(s)|s)$ does in fact equal the expected discounted value of profits if the policies are followed.

The rest point of an AI pricing game that plays greedy policies and uses asynchronous updating will satisfy the conditions of an EBE. The fact that the policies are greedy insures (1), and if we are at a rest point that point is a recurrent class (insuring 2).[28] The third condition states that if equilibrium

[28]For simplicity, here and below we are assuming that the (s_i, s_j) associated with the rest point (p_i^*, p_j^*) is unique.

policies are played indefinitely the average of the realized discounted value at each $s \in \mathcal{R}$ converges to $W_i(p_i^*(s)|s)$. Given the updating equation (equation 2), the fact that we stay at the rest point indefinitely (so $s = \mathcal{R}$), and the fact that $(p_i^*(s), p_j^*(s))$ is played repeatedly, insures that $W_i(p_i^*(s)|s)$ converges to $[1-\beta]^{-1}\pi_i(p_i^*(s), p_j^*(s))$, the discounted value of returns that would be earned if $(p_i^*(s), p_j^*(s))$ were played indefinitely. An analogous condition holds for firm j.

Notice that EBE does not put as stringent a condition on the perception of returns for feasible policies not played: for policies "off the equilibrium path." It requires only that the perceptions of these returns are less than those of the optimal policy; i.e., that $W_i(p|s) \leq W_i(p_i^*(s)|s)$. The implication is that if randomness in the initial conditions happens to generate high perceived values for both agents at a particular couple of choices, and those values generate profits plus discounted continuation values which are higher than the perceived values at other prices, the values at the other points will never be updated. Then the algorithm will stick with the values that generated the initial choice forever. If the initial iteration's values did not satisfy these conditions, a subsequent iteration could, and we would reach high prices at a later iteration, as in panel (b) of figure 1.

Perfect synchronous updating leads to rest points that satisfy stronger conditions than those of an EBE. In particular, the fact that all values are updated at each iteration also restricts the perceptions of returns for the feasible policies that are not played at the rest point. The implications of the additional conditions are different when the model has $\beta = 0$ and when $\beta > 0$.

When $\beta = 0$, the update for perceptions at the rest point for firm i is just current returns. With perfect synchronous learning, this implies that the update for every $p \in \mathcal{P}$ for firm i is $\pi_i(p, p_j^*)$ and for agent j it is $\pi_j(p, p_i^*)$. Since the rest point is visited repeatedly, the perceived value of play at every p converges to the discounted value of these profits for the two firms. As a result, regardless of initial conditions, the nature of S, or the form of the profit function, the optimal strategy at the rest point must eventually satisfy $\pi(p_i^*, p_j^*) = \max_{p \in \mathcal{P}} \pi_i(p, p_j^*)$ with an analogous condition for firm j. So when $\beta = 0$ the rest point must be a Nash equilibrium, as in panel (a) of Figure 1.

When $\beta \neq 0$, the rest point from a perfect synchronous updating algorithm need not satisfy the condition that the perception of the value of all feasible policies is equal to the expected discounted value obtained from playing the action repeatedly. When $\beta \neq 0$, the rest point satisfies the second set of equilibrium conditions introduced in Fershtman and Pakes (2012); those of a restricted EBE (or REBE). REBE abides by the first two conditions of an EBE, but strengthens the consistency requirement (the third condition) by requiring that the consistency condition hold for feasible actions that are not optimal (for $p(s_i) \neq p^*(s_i)$), but only when the feasible actions result in outcomes that are in $\mathcal{R}$ (i.e. are visited repeatedly).

The restriction of the consistency condition to outcomes that are in $\mathcal{R}$ is also the rest point condition that emanates from perfect synchronous AI pricing algorithms[29]. This is because the discounted value of future returns for points outside of $\mathcal{R}$ are not visited repeatedly, so we cannot use a law of large numbers to prove that their perceived values converge to any particular value. When $\beta \neq 0$ and a perfect synchronous learning model is used, the update for a feasible but non greedy policy consists of current profits plus the discounted perceived value of future play were the non-optimal action played. However since the non-greedy policy is not played at the rest point, though the perfect synchronous algorithm updates current profits with realized profits at the rest point, the discounted perceived values of future play (the "continuation values"), are not updated with realizations from what would have occurred were those actions taken. So if greedy policies are played throughout, the perceived values for feasible actions at the rest point will be determined by the random draw on initial conditions, while if there is some experimentation, the draws on the experiments will also help determine the rest point.

Asker et al. (2020) consider a condition which restricts the perceptions of returns from off the equilibrium path play further and use it in their analysis of dynamic procurement auctions. However their extra condition, labelled "boundary consistency," is not used in any AI pricing game we are aware of, and will not be satisfied "naturally" at a rest point of the algorithm (to check it a separate subroutine must be built into the algorithm). Consequently we focus on the implications of the simpler REBE condition in the extensions investigated in the next section.

[29]Formally, points in $\mathcal{R}$ that can only transit to other points in $\mathcal{R}$ no matter which feasible policy is played are referred to as "interior points," while points in $\mathcal{R}$ for which there is a feasible policy which would transit to points outside of $\mathcal{R}$ are referred to as "boundary points."

5 Extensions to more complex environments

This section considers the implications of learning models when our basic AI pricing algorithm is extended in various ways. Given the results above, we do this first for the case where $\beta = 0$ and then for the case where $\beta > 0$. The extensions include allowing for more than two competitors, experimentation, and cases where the algorithm only has the ability to update the value of counterfactual policies imperfectly.

The latter case is relevant when either competitor's prices are not observed or the demand system is not known. Then the algorithm might still use basic economic reasoning, in our example the assumption that the residual demand curve slopes downward, to update the value of counterfactual policies. This example is extreme in that it assumes that neither the competitor's prices nor the demand system can be even imperfectly inferred from observed behavior. However, it will suffice to make the point that adding a little bit of information to the algorithm, in addition to the profits the algorithm "observes," makes notable changes to the rest point.

When considering models with $\beta > 0$, we allow for history-dependent strategies. This enables the algorithm to generate equilibria that mimic collusive equilibria supported by familiar punishment strategies. Two points should be kept in mind here. First, the strategies generated by the algorithm are partly a function of random draws, in contrast to purposeful behavior. Second, models with larger state spaces are harder to compute. The complexity of an algorithm which keeps past prices (or any other competitor-specific state) in memory increases, often dramatically, with the number of competitors, and this limits their usefulness. [30]

5.1 $\beta = 0$: Algorithms that play static games

Increasing N, the number of firms. Table 1 extends the computational results reported in figure 2 by varying N, the number of firms. Both the perfect synchronous and asynchronous algorithms are proficient at finding the optimal monopoly price when N=1. As N increases to 2 and higher, the

[30] If we simply bin prices and do not impose further restrictions, the memory requirements will increase exponentially in the number of firms, though if we assume, as is often done in applied work, that the policies are exchangeable in the states of competitors, the rate of growth in the number of competitors decreases, from exponential to geometric, see Pakes and McGuire, 1994.

perfect synchronous algorithm converges quickly to Nash pricing outcomes[31]. By contrast, the asynchronous algorithm converges to a price greater than Nash in 100 percent of instances when N=2, and 71.5 percent of instances when N=3. This decreases as N increases further, with this percentage approaching zero by the time $N = 10$. Further, the number of iterations until convergence for the asynchronous algorithm is typically two orders of magnitude higher than for the perfect synchronous algorithm.[32]

The results in figure 2 indicate that having multiple competitors can significantly mitigate any adverse price effects that arise from the use of an asynchronous algorithm. That said, even in only moderately concentrated markets (those with 4 to 6 competitors), pricing above static Nash levels can occur relatively frequently.[33]

Experimentation. Thus far we have investigated algorithms that do not experiment as they learn. If an algorithm is programed to experiment, it deviates from choosing the greedy policy, i.e., the policy associated with the maximum of the W's in memory, and instead chooses a random action and obtains information on what the payoff of that action is. We consider how experimentation might affect the asynchronous results in figure 2.[34]

Deciding how an asynchronous algorithm may experiment involves making a range of decisions. We consider experimentation which occurs with probability $[k^{\frac{1}{\theta}}]^{-1}$ in each round, where k indexes the iteration of the simulation, and θ is a parameter we vary. So the frequency of experimentation declines as k increases and the algorithm has more observations to learn from. For a given k, the frequency of experimentation increases with the θ parameter. When the algorithm does experiment it chooses each feasible action with equal probability.

[31]For $N \in \{2, 3, 4\}$ the highest static Nash price is 2.13 (2.03 is also a Nash outcome). For $N \geq 5$ the only static Nash equilibrium price is 2.03.

[32]These results are qualitatively similar in an environment with a Logit demand system in Appendix A.2. Interestingly, in those results the asynchronous case is shown to reach results higher than monopoly, and also below Nash, in some instances.

[33]For $N = 6$, the percentage greater than static Nash is 2.1. In the case of the Logit model reported in Appendix A.2, the percentages for $N = \{4, 5, 6\}$ are 71.9, 66.6 and 60.6, respectively, suggesting that the results reported in table 1 may understate these probabilities for, at least some, alternative environments.

[34]The addition of experimentation to the perfect synchronous algorithm (panel (a) of figure 2) would be redundant as the perfect synchronous algorithm updates all possible actions anyway.

Table 1: Pricing outcomes as N varies

Algorithm	N	Min	25th	Median	75th	Max	% > Static Nash	Time to converge
perfect synchronous	1	10.00	10.00	10.00	10.00	10.00	-	50
	2	2.03	2.13	2.13	2.13	2.13	0	115
	3	2.03	2.13	2.13	2.13	2.13	0	70
	4	2.03	2.13	2.13	2.13	2.13	0	75
	5	2.03	2.03	2.03	2.03	2.03	0	85
	10	2.03	2.03	2.03	2.03	2.03	0	90
Asynchronous	1	10.00	10.00	10.00	10.00	10.00	-	1,235
	2	4.65	7.28	8.39	9.29	10.00	100	4,550
	3	2.13	2.13	2.84	3.95	10.00	71.5	8,530
	4	2.03	2.13	2.13	2.13	7.38	19.9	10,395
	5	2.03	2.03	2.03	2.03	3.34	12.1	10,395
	10	2.03	2.03	2.03	2.03	2.23	0.1	10,025

Notes: For each N, 1,000 simulation runs were conducted. The minimum, 25th percentile, median, 75th percentile, and maximum of the prices for firm 1, across the 1,000 simulations, once all simulations have reached a rest point, is reported. '% > Static Nash' reports the per cent of simulations that reach a rest point that is higher than the highest static Nash price. For $N \in \{2, 3, 4\}$ the highest static Nash price is 2.13 (2.03 is also a Nash outcome). For $N \geq 5$ the only static Nash equilibrium price is 2.03. 'Time to converge' reports the number of iterations until the reported price moments cease to change (each simulation was eventually stopped after 40,000 iterations). The model is as per figure 2, but for the reported changes in N.

Table 2: Pricing rest points as θ varies

θ	Min # of times an action is played	Pr(experimentation) at iteration 10,000	Min	25th	Median	75th	Max
1	14	0.0001	5.36	7.07	8.03	9.24	10.00
2	17	0.0100	3.04	6.42	7.68	8.89	10.00
3	20	0.0464	3.04	5.46	7.02	8.54	10.00
4	24	0.1000	2.73	4.25	6.27	7.98	10.00
5	29	0.1585	2.53	3.84	5.56	7.68	10.00
6	33	0.2154	2.53	3.54	5.01	7.28	10.00
7	37	0.2683	2.13	3.44	4.65	6.87	10.00
8	42	0.3162	2.13	3.34	4.40	6.57	10.00
9	45	0.3594	2.13	3.34	4.20	6.37	10.00
10	48	0.3981	2.13	3.34	4.15	6.06	10.00

Notes: For each θ, 100 simulation runs were conducted. Experimentation occurs with probability $\frac{1}{k^{\frac{1}{\theta}}}$ in each round, where k indexes the iteration of the simulation. If experimentation occurs, an action is selected at random (with each possible action having equal probability). Experimentation stops after 10,000 iterations. The algorithm continues after that point with no experimentation until a rest point is reached. The minimum, 25th percentile, median, 75th percentile, and maximum of the prices for firm 1, across the 100 simulations, once all simulations have reached a rest point, is reported. $N = 2$. The model is as per figure 2, but for the addition of experimentation.

There will not be a rest point after a finite number of iterations without an end to experimentation. Since the asynchronous algorithm, absent experimentation, reaches a rest point within 5,000 iterations (see figure 2), we allow experimentation to occur for 10,000 iterations. That is, we allow experimentation to double the computational time required by the algorithm to reach a rest point when there is no experimentation. After 10,000 iterations, we stop experimentation and let the algorithm run until it converges.

Table 2 reports the resulting price distributions as θ, the intensity of experimentation, varies. For each value of θ we also report the minimum of the number of times any action is attempted during the course of learning, labeled 'Min # of times an action is played'. This provides a way to judge the extent to which experimentation reaches all feasible actions.

The results in Table 2 indicate that experimentation is not a simple fix for the propensity for the asynchronous algorithm to elevate prices. Apparently a reasonable amount of experimentation does mitigate this propensity, but does not remove it. What an optimal experimental process might look like is unclear as it would have to face the familiar problem of trading off "exploration" against "exploitation" in an environment with competition.[35] It must surely penalize excessive computational burden, and occur at some frequency which is both frequent enough to be informative, but no so frequent as to cause an expected discounted profits decline. What does seem clear is that algorithms that build in experimentation are not likely to completely fix the propensity of asynchronous algorithms to elevate prices.

Imperfect counterfactual updating. As previously noted, the asynchronous and perfect synchronous algorithms require access to very different amounts of information; the asynchronous algorithm needs to know only the profits it received from the price it played, whereas the perfect synchronous algorithm requires an ability to run a complete set of counterfactuals conditional on the state. To obtain accurate counterfactuals the algorithm needs to observe competitors' prices and then be able to map them, together with different selections of its own price, into profits (in our example this would require knowledge of the residual demand curve). We have endowed our perfect synchronous algorithm with this ability, but in many actual markets accurate counterfactual updating may be impossible.

We now explore the behavior of algorithms that have access to less information than our perfect synchronous but more information than our asynchronous algorithms. In particular we leverage a relatively uncontroversial additional bit of information: the knowledge that residual demand curves slope (weakly) downward. That information allows us to employ a less accurate form of synchronous updating. Hence, we call this *synchrnous updating using downward demand.*

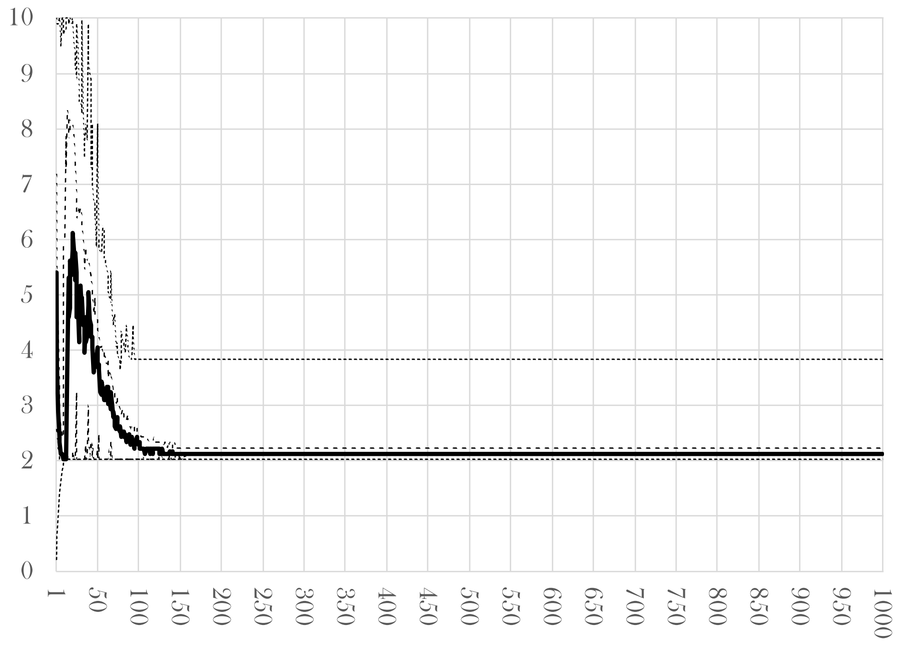
Recall that $d_i(p^*, p_j^k)$ is our notation for the demand received by the firm at the price it chose in iteration k. Our counterfactual updating rules are modified as follows. If $p > p^*$ *and* the current iteration's perception of the value at p is higher than what could be rationalized by assuming demand was given by current demand, then we update the value of playing the counterfactual price assuming that $d_i(p, p_j^k) = d_i(p^*, p_j^k)$, so that $W_i^k(p)$ is adjusted downward. Similarly, if $p < p^*$ *and* $W_i^k(p)$ is lower than could be justified by

[35]See Sutton and Barto (2018).

assuming that $d_i(p, p_j^k) = d_i(p^*, p_j^k)$, then $W_i^k(p)$ is adjusted upward by assuming that $d_i(p, p_j^k) = d_i(p^*, p_j^k)$). Formally the updating rules for different p are[36]

$$\begin{aligned} p = p^*,\ W_i^{k+1}(p^*) &= \alpha(p-c)d_i(p^*, p_j^k) + (1-\alpha)W_i^k(p), \qquad (4) \\ p > p^*,\ W_i^{k+1}(p) &= \alpha \min\left\{(p-c)d_i(p^*, p_j^k), W_i^k(p)\right\} + (1-\alpha)W_i^k(p), \\ p < p^*,\ W_i^{k+1}(p) &= \alpha \max\left\{(p-c)d_i(p^*, p_j^k), W_i^k(p)\right\} + (1-\alpha)W_i^k(p). \end{aligned}$$

Figure 4: Price outcomes with synchronous updating using downward demand



Notes: Prices (vertical axis) by iteration (horizontal axis) from 100 simulations are shown. The lines, from bottom to top, represent the min, 25th percentile, median, 75th percentile, and max of the distribution of prices in each period. The model mirrors that in figure 2, panel (b), but for the addition of the ability of the algorithm to update all $W(p)$'s using the assumption that demand slopes (weakly) downward.

Figure 4 shows the impact of taking the model in panel (b) of figure 2 (static Bertrand with asynchronous updating) and allowing updating to occur

[36]These rules are for $p \geq c$. For $p < c$ these are reversed.

in instances when the underlying values violate the assumption of downward sloping demand. The minimum rest point is 2.03 and the maximum is 3.84. The reason that the outcomes are markedly different to those in figure 2 is that, by imposing the additional demand structure, W's attached to high prices get updated downward frequently in early iterations (i.e. given that W's are initialized at high values, in early iterations this happens whenever a lower price is chosen). This means that the actions actually chosen, by both firms, through the course of the learning process are much more likely to be relatively low. The outcomes reported in figure 4 suggests that leveraging minimal assumptions about the economic environment can have a significant mitigating impact on any tendency for an asynchronous algorithm to generate supra-competitive prices.[37] Leveraging the downward sloping demand assumption also dramatically increases the speed with which the algorithm reaches a rest point. The speed increase is of at least an order of magnitude.

The example shown here, in which the assumption of downward demand is explored, indicates that having a complete model of the economic environment is not necessary to significantly mitigate the propensity for an asynchronous algorithm to generate high prices. Depending on the setting, other justifiable assumptions may be able to be leveraged, whether in addition to downward demand or as a substitute for it.[38]

5.2 $\beta > 0$: Algorithms that consider future returns

When $\beta = 0$, pricing algorithms that use asynchronous learning are expected to generate rest points with supra-competitive pricing, but those that used perfect synchronous learning will not. These results are independent of the details of the algorithm, including the structure of the profit function and the states that strategies can condition on. However the result for perfect synchronous learning does not generalize to models with $\beta > 0$.

Recall that when an algorithm converges to a rest point, that point is

[37]Appendix A.2 shows the impact of this augmented procedure in an environment with Logit demand. Mitigation still occurs, but is somewhat muted relative to the homogenous Bertrand example.

[38]Examples might include: using the realizations of price and quantity to approximate demand, using one or more mis-specified demand models, or using local regression approaches to approximate W's that are close to those that can be updated precisely (see Farias et al. (2012) for a related approach in the context of computing approximate MPE in dynamic oligopoly games).

the only point in the recurrent class, and the conditions that the policies and perceived values a perfect synchronous learning algorithm must satisfy at the rest point are those of a REBE. If $\beta > 0$, the feasible but non optimal policies at the rest point are updated using a correct measure of current profit but a discounted continuation value which is not updated because the policies that are not optimal are never actually played. To explore the implications of this fact in a way that makes their relationship to the literature on repeated games transparent, we consider models with history-dependent strategies.

To see the possible implications of not updating continuation values, it suffices to consider a model where there are two possible actions and a state space that consists of last period's prices, that is

$$\mathcal{P} = (p^1, p^2), \text{ and } \mathcal{S} = \{(p^1, p^1), (p^1, p^2), (p^2, p^1), (p^2, p^2)\}.$$

We consider the conditions that need to be satisfied for (p_i^1, p_j^1) to be a rest point of this algorithm.

Since policies have converged, equation (2) guarantees that perceived values will also. Since states other than (p_i^1, p_j^1) are not recurrent, the value associated with play at those points need not equal the discounted value that would result were those states being visited repeatedly. The REBE conditions for the rest point imply only that in the limit

$$W_i(p^1|p_i^1, p_j^1) = \pi_i(p_i^1, p_j^1) + \beta W_i(p^1|p_i^1, p_j^1), \Rightarrow W_i(p^1|p_i^1, p_j^1) = \pi_i(p^1, p^1)[1-\beta]^{-1},$$

$$W_i(p^2|p_i^1, p_j^1) = \pi_i(p_i^2, p_j^1) + \beta max_{y\in\mathcal{P}} W_i(y|p_i^2, p_j^1) < W_i(p^1|p_i^1, p_j^1),$$

and the analogous conditions for firm j.

So for (p_1, p_1) to be a rest point, all we require is

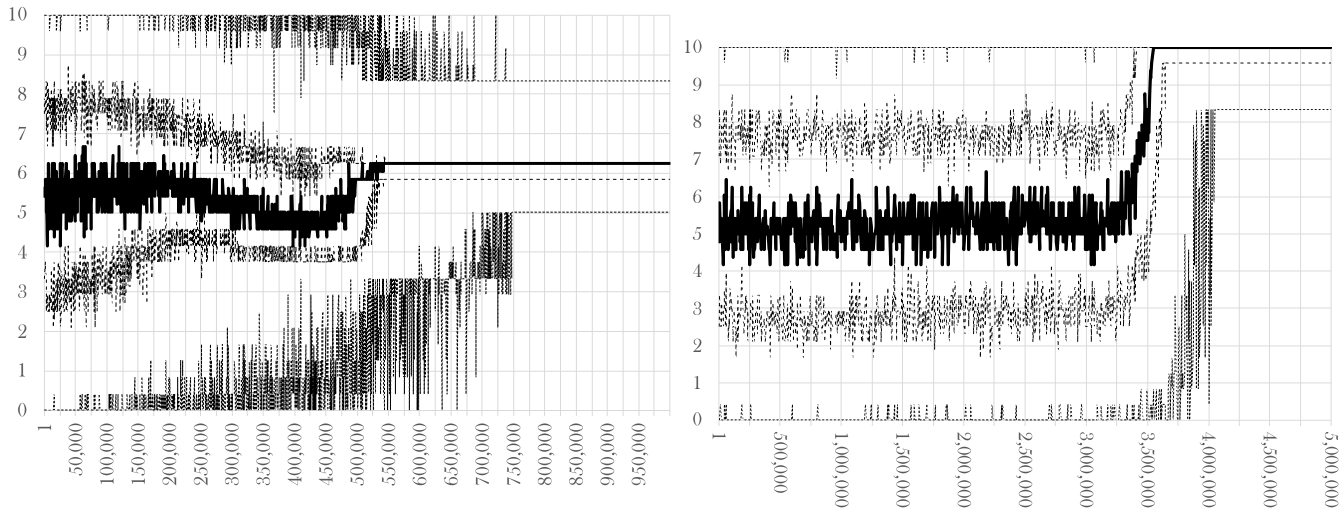
$$[\pi_i(p_i^1, p_j^1) - \pi_i(p_i^2, p_j^1)] > \beta\Big(\max_{y\in\mathcal{P}} W_i(y|p_i^2, p_j^1) - [1-\beta]^{-1}\pi_i(p_i^1, p_j^1)\Big),$$

and the analogous condition for firm j. $(W_i(p_1|p_i^2, p_j^1), W_i(p_2|p_i^2, p_j^1))$ are not constrained to be consistent with the returns that would actually be earned were those policies actually played. Rather they are determined by the random draws – either just those on initial conditions or, if there is experimentation, in conjunction with the draws from experimenting.

These equilibrium conditions also determine what would happen were we to force the algorithm to engage in "off the equilibrium" path behavior, for example if we used the policies to simulate a path in which firm i chose price

p_2. If $\max_{y \in \mathcal{P}} W_j(y|p_i^2, p_j^1) = W_j(p^2|p_i^2, p_j^1)$ firm j would respond in the next period with p_2. If this occurred and $\max_{y \in \mathcal{P}} W_i(y|p_i^2, p_j^2) = W_i(p^2|p_i^2, p_j^2)$, then firm i would chose p^2 in the next period, and if the analogous condition held for firm j both firms would choose p^2 in all subsequent periods. That is (p_1, p_1) would look like it was supported by a "trigger price" strategy. So trigger price strategies are consistent with a REBE, and hence with a rest point to the perfect synchronous game.

Figure 5: Price paths with richer state spaces and $\beta = 0.95$.



(a) Perfect synchronous updating

(b) Asynchronous updating

Notes: Prices (vertical axis) by iteration (horizontal axis) from 100 simulations are shown. The lines, from bottom to top, represent the min, 25th percentile, median, 75th percentile, and max of the distribution of prices in each period. The discount factor, β, is set equal to 0.95. Results are shown for firm 1. The model is parametrized as follows. Feasible prices exist on a grid with 25 elements equally spaced between 0.1 and 10 inclusive. The state space contains the prices of both firms in the prior period. The weight on current returns in updating is given by $\alpha = 0.1$. Initial conditions are i.i.d. draws from $U[200, 210]$, for each $W(p)$ for each firm. In all other respects the model mirrors that in figure 2.

Computational results when algorithms care about the future. Figure 5 provides price paths from AI algorithms that use perfect synchronous and asynchronous updating when $\beta = .95$, the state space is expanded to contain the prices each firm charged in the prior period, and demand is given by equation (1).

As in figure 2, which provided the comparison when $\beta = 0$, the asynchronous algorithm generates substantially higher price outcomes. Indeed setting $\beta = .95$ using a richer state space generates a distribution of pricing rest points from the asynchronous algorithm that stochastically dominates its analogue in the $\beta = 0$ case. However, the perfect synchronous algorithm now also converges to a price that is higher than the static Nash outcome. The median of the perfect synchronous price distribution is 6.25. This is substantially lower than the median price with asynchronous updating (which is 10), but substantially higher than the highest Nash price of 2.52. The rate of convergence for the perfect synchronous algorithm is still substantially faster than that of the asynchronous algorithm[39].

These results extend the conclusions of Calvano et al. (2020) to games played by AI algorithms that use perfect synchronous updating, but only if $\beta > 0$. Moreover the theoretical discussion shows that when $\beta > 0$ the asynchronous algorithm can generate policy patterns that mimic those obtained from trigger-price strategies. However, a comparison among asynchronous algorithms, that is among the algorithm that use the learning rules used in Calvano et. al. (2020), which we investigate further in figure 6, throws a slightly different light on the problem.

This figure displays the frequency of realized prices generated by alternative asynchronous algorithms. Light grey bars are the static model (singleton state space and $\beta = 0$), medium grey bars are for an enriched state space (containing the price of each firm in the prior period) but with $\beta = 0$, and dark bars have the enriched state space and $\beta = 0.95$. As can be seen, there is little economically meaningful difference between the distribution of price outcomes generated by the static model (light grey) and that generated by the model augmented with an enriched state space (medium grey).[40] So a meaningful portion of any adverse price effect does not depend on leveraging history-dependent strategies.

More visually noticeable are the differences when we set $\beta = .95$ and we allow for history-dependent strategies; then the mean is 9.83 and the median is 10. The relative increase in the mass concentrated on a price of 10 is notable.

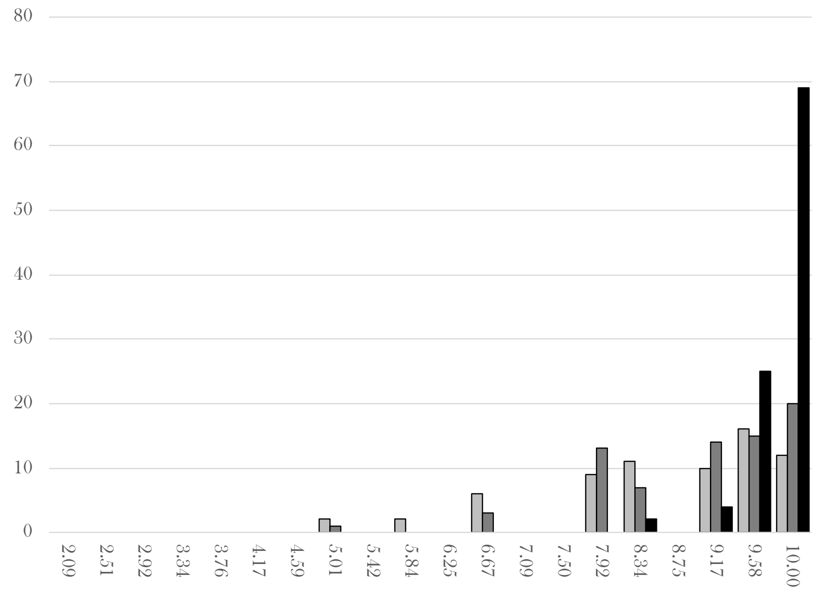
[39]The perfect synchronous price distribution stabilizes after about 750,000 iterations, while asynchronous price distribution does not stabilize until around 4,000,000 iterations.

[40]The mean (and median) prices in the 'static' (light grey) and '$\beta = 0$' (medium grey) cases are 8.33(8.34) and 8.73(8.75) respectively.

Keep in mind, however, that when $\beta = 0$ and we allow for history-dependent strategies, the median price outcome is 8.75. So the asynchronous algorithm generates a substantial increase in price even when it does not consider the impact of its play on the possibility of punishing in the future. That is, at least in our example, much of the difference between rest point prices and Nash equilibrium prices generated by an asynchronous learning algorithm would exist even if we did not allow the algorithm to generate policies that mimic a punishment scheme.[41]

[41]Appendix A.3 illustrates two examples of the path of play following an optimal deviation from a rest point reached by the asynchronous algorithm.

Figure 6: Price outcomes for models with asynchronous updating.



Notes: The frequency of realized prices (horizontal axis), after the asynchronous algorithm has reached a resting point, is displayed. The model is the same as in figure 5 panel (b), with the following adjustments. Light grey bars are the static model (singleton state space and $\beta = 0$). Medium grey bars are for an enriched state space (containing the price of each firm in the prior period) but with $\beta = 0$. Dark bars have the enriched state space and $\beta = 0.95$ as in figure 5 panel (b). For each version of the model, results are constructed from 100 simulations.

Extensions when $\beta > 0$. Table 3 reports the results of extensions to the basic specifications for perfect synchronous and asynchronous learning presented in figure 5. For both it begins by reporting results for the simulations underlying figure 5. As noted when $\beta = 0.95$ and the state space includes the previous period's prices, both learning algorithms generate supracompetitive pricing outcomes with the distribution of prices generated by the asynchronous algorithm stochastically dominating that generated by the perfect synchronous algorithm.

Table 3: Pricing outcomes in extensions, when $\beta = 0.95$

Algorithm	Treatment	Min	25th	Median	75th	Max
Perfect synchronous	$N = 2$ (as in Figure 5)	5.01	5.84	6.25	6.25	8.34
	$N = 2$, Experimentation ($\theta = 4$)	4.17	4.59	4.59	5.01	6.67
	$N = 3$	2.51	3.34	3.34	3.34	5.84
Asynchronous	$N = 2$ (as in Figure 5)	8.34	9.58	10.00	10.00	10.00
	$N = 2$, Experimentation ($\theta = 4$)	6.25	9.17	9.58	10.00	10.00
	$N = 3$	5.42	8.75	9.17	9.58	10.00
Synchronous using downward demand	$N = 2$	3.75	6.67	7.50	7.92	10.00

Notes: For each treatment, 100 simulation runs were conducted. The minimum, 25th percentile, median, 75th percentile, and maximum of the prices for firm 1, across the 100 simulations, once all simulations have reached a rest point, is reported. The monopoly price is 10. Prices of 2.09 and 2.51 are both supportable as outcomes of a static Nash equilibrium. Experimentation occurs for the first 10 million iterations. The model is as per figure 5, but for the noted changes in specification.

The introduction of experimentation in this setting mirrors the qualitative results in table 2. Experimentation appears to have a mild mitigating impact on price elevation in both cases. However the effect is not as large as the effect of introducing a third firm in either case, with the difference being particularly noticeable for the perfect synchronous case. Perhaps most surprising is the impact on the asynchronous algorithm of imposing that the residual demand curve is decreasing in price (see section 5.1). This has a greater downward impact on the price distribution, at all reported moments other than the minimum, than adding an extra firm[42]. With rich state spaces and $\beta > 0$, use of algorithms that abide by at least some information about the economic environment may be at least as important for pricing as increasing the number of firms in the market.

[42]Leveraging this tenet would have no additional benefit for the perfect synchronous algorithm.

6 Conclusion

The results in this paper relate to contemporary policy discussions about appropriate competition policy in a world in which decisions are delegated to algorithms. We illustrate the impact of the design on equilibrium prices. Much of this policy discussion has centered on traditional antitrust tools, particularly laws prohibiting collusion.[43] The results here present significant challenges for the application of those traditional tools. We show the potential for algorithms to lead to supra-competitive pricing in settings where, as in Harrington (2018), it cannot match any economic definition of collusive behavior.

We have not explicitly discussed the choice of algorithms. Of course, even if a firm understands that choosing a naive (e.g., asynchronous) algorithm will lead to supra-competitive pricing and, as a consequence, delegates its pricing to such an algorithm, this is distinct from doing so in concert with other firms as part of some commitment or agreement. Figure 7 shows the pattern of average (rest point) payoffs arising from alternative choices of algorithmic designs given the simulations reported in this paper for the static Bertrand model. It is a Nash equilibrium for both firms to choose to implement asynchronous algorithms. Indeed, no matter what a rival firm implements, each firm is weakly better off with the asynchronous algorithm. That is, employing an asynchronous algorithm is a weakly dominant strategy.[44]

		Firm Y	
		Asynch.	*Sync.*
Firm X	*Async.*	$(8.12, 8.12)$	$(2.13, 2.13)$
	Sync.	$(2.13, 2.13)$	$(2.13, 2.13)$

Figure 7: The algorithm coordination game

Notes: The payoffs reflect the pattern of expected payoffs reported in figure 2 and Appendix A.5.

That the choice of algorithms leading to supra-competitive pricing could be an equilibrium outcome in a static game, seems at odds with the requirement that illegal cartels have some form of agreement at their core. This

[43]See, for instance, the discussion in Harrington (2018).

[44]In the Logit version of the model discussed in the Appendix, employing the asynchronous algorithm is a strictly dominant strategy.

conclusion resonates with ongoing expressions of discomfort, on the part of antitrust scholars, with the heavy emphasis that cartel enforcement puts on finding an agreement (see for instance, Turner (1962), Posner (1976), Whinston (2006), Kaplow (2013), and, in the context of algorithms specifically, Harrington (2018)).

Whether through regulation, or through antitrust oversight, the question of how to identify a "pro-competitive" implementation of an algorithm seems to be central to any evaluation exercise that a regulatory agency might conduct. Most likely this requires access to the underlying code.[45] At least the results in this paper point toward pro-competitive implementations being those that are informed by an understanding of the wider economic environment and that incorporate counterfactual alternatives to the play actually engaged in when learning. Thus, indicia of the possible existence of a "pro-competitive" implementation may include the incorporation of a demand model, likely informed by statistical or econometric studies (such as A-B testing) to generate returns to alternative actions. Training the algorithm using results from sub-populations on whom price experiments have been run may also be a useful indicator of an "pro-competitive" algorithm.[46] Additionally, the greater the number of firms in a market, the more muted any supra-competitive pricing is likely to be.[47]

Developing an empirical understanding of how algorithm adoption impacts pricing, and the choices the algorithms that are actually used gives management, seems invaluable. Empirical studies of the nature and the impact of AI pricing algorithms that have been adopted are one way of evaluating the relative merits of human and algorithmic control of pricing in markets. Assad et al. (2020) provides a valuable first step in this direction. Additional research in this vein, guided by the growing theoretical and computational work on algorithms, strikes us as particularly valuable.

[45]Alternatively, simulations using the program may allow inferences along these lines to be mode. This may be necessary if the program can be used to generate pricing recommendations, but access to underlying code is restricted.

[46]Here the sub-populations should be small enough to have no meaningful impact on the pricing of any competing firm, and exist merely to inform the computation of counterfactual returns. If experimentation occurs on sub-populations that are large, then the possibility may arise that experimentation distorts algorithmic learning.

[47]Additionally, conditional on the design of the algorithm, the more algorithms are trained to optimize current rather than future profits, the better for competition. Similarly, the less information about competitors' actions that resides in the state space, the better.

References

Agrawal, Ajay, Joshua Gans, and Avi Goldfarb. 2019. "The Economics of Artificial Intelligence: An Agenda." University of Chicago Press.

Aparicio, Diego, Zachary Metzman and Roberto Rigobon. 2021. "The Pricing Strategies of Online Grocery Retailers." NBER WP 28639.

Asker, John, Chaim Fershtman, Jihye Jeon, and Ariel Pakes. 2020. "A Computational Framework for Analyzing Dynamic Auctions: The Market Impact of Information Sharing." *RAND Journal of Economics* 51(3): 805-839.

Asker, John, Chaim Fershtman, and Ariel Pakes. 2022. "Artificial Intelligence, Algorithm Design and Pricing." *AEA Papers and Proceedings* , 112: 452-56.

Assad, Stephanie, Robert Clark, Daniel Ershov, and Lei Xu. 2020. "Algorithmic Pricing and Competition: Empirical Evidence from the German Retail Gasoline Market." Working paper.

Assad, Stephanie, Emilio Calvano, Giacomo Calzolari, Robert Clark, Vincenzo Denicolo, Dan Ershov, Justin Johnson, Sergio Pastorello, Andre Rhodes, Lei Xu and Matthijs Wildenbeest. 2021. "Autonomous Algorithmic Collusion: Economic Research and Policy Implications." *Oxford Review of Economic Policy* 37: 459-478.

den Boer, Arnoud, Janusz Meylahn, Maarten Schinkel. 2022. "Artificial Collusion: Examining Supracompetitive Pricing by Q-learning Algorithms". mimeo.

Brown, Zach and Alexander MacKay. 2021. "Competition in Pricing Algorithms." *American Economic Journals: Microeconomics*, forthcoming.

Calvano, Emilio, Giacomo Calzolari, Vincenzo Denicolò, and Sergio Pastorello. 2020. "Artificial intelligence, algorithmic pricing and collusion." *American Economic Review*, 110(10) 3267-97.

Calvano, Emilio, Giacomo Calzolari, Vincenzo Denicolò, and Sergio Pastorello. 2021. "Algorithmic collusion with imperfect monitoring." *International Journal of Industrial Organization*, 79: 102712.

Chen, L., Alan Mislove, and Christo Wilson. 2016. "An Empirical Analysis of Algorithmic Pricing on Amazon Marketplace." *Proceedings of the 25th International Conference on World Wide Web* 1339-1349.

Competition and Markets Authority. 2018. "Pricing algorithms: Economic working paper on the use of algorithms to facilitate collusion and personalised pricing." CMA Working Paper CMA94.

de Corniere, Alexandre, and Greg Taylor. 2020. "Data and Competition: a General Framework with Applications to Mergers, Market Structure, and Privacy Policy." TSE Working Paper n. 20-1076.

Derakhshan, Alireza, Frodi Hammer, and Yves Demazeau. 2016. "PriceCast Fuel: Agent Based Fuel Pricing." Advances in Practical Applications of Scalable Multi-agent Systems. The PAAMS Collection. Lecture Notes in Computer Science 9662: 247-250.

Epivent, Andrea and Xavier Lambin. 2022. On Algorithmic collusion and reward-punishment schemes. Mimeo.

Erev, Ido. and Ernan Haruvy. 2016. "Learning and the economics of small decisions." *The Handbook of Experimental Economics* 2(10): 638-716.

Eschenbaum, Nicolas, Filip Mellgren, and Philipp Zahn. 2022. "Robust Algorithmic Collusion." mimeo.

European Commission. 2020. "Inception Impact Assessment: New Competition Tool."

Ezrachi, Ariel, and Maurice E. Stucke. 2017. "Artificial intelligence & collusion: When computers inhibit competition." *University of Illinois Law Review* 1775-1810.

Farias, Vivek, Denis Saure, and Gabriel Y. Weintraub. 2012. "An approximate dynamic programming approach to solving dynamic oligopoly models." *Rand Journal of Economics* 43(2): 253-282.

Federal Trade Commission. 2018. “FTC Hearing #7: The Competition and Consumer Protection Issues of Algorithms, Artificial Intelligence, and Predictive Analytics.”

Federal Trade Commission. 2018. “FTC Hearing 7: Nov. 14 Welcome Remarks and Session 1 Algorithmic Collusion.”

Fershtman, Chaim and Ariel Pakes. 2012. “Dynamic Games with Asymmetric Information: A Framework for Empirical Work.” *Quarterly Journal of Economics* 127(4): 1611-1661.

Fudenberg, Drew and Levine, David K. 2016. “Whither game theory? Towards a theory of learning in games.” *Journal of Economic Perspectives* 30(4): 70–151.

Goldfarb, A., Gans, J., and Agrawal, A. (2019). The Economics of Artificial Intelligence: An Agenda. University of Chicago Press.

Hansen, Karsten, Kanishka Misra, and Mallesh Pai. 2020.“Algorithmic Collusion: Supra-competitive Prices via Independent Algorithms.” CEPR Discussion Paper, no. DP14372.

Harrington, Joseph. 2018. “Developing Competition Law for Collusion by Autonomous Artificial Agents.” *Journal of Competition Law and Economics* 14:331-363.

Hart, Sergiu and Andreu Mas-Colell. 2003. “Uncoupled Dynamics Do Not Lead to Nash Equilibrium.” *American Economic Review* 93(5): 1830–1836.

Hirst, Nicholas. 2018. “When Margrethe Vestager takes antitrust battle to robots.” *Politico*

Igami, Mitsuru. 2020. “Artificial intelligence as structural estimation: Deep Blue, Bonanza, and AlphaGo.” *Econometrics Journal*, 23(3): S1-S24.

Johnson, Justin, Andrew Rhodes, and Matthijs Wildenbeest 2020. “Platform Design When Sellers Use Pricing Algorithms.” Working Paper.

Kaplow, Louis. 2013. “Competition Policy and Price Fixing.” Princeton University Press, Princeton.

Klein, Timo. 2019. "Autonomous algorithmic collusion: Q-learning under sequential pricing." Amsterdam Law School Research Paper no. 2018–15.

Kühn, Kaiuwe, and Steve Tadelis. 2017. "The Economics of Algorithmic Pricing: Is Collusion Really Inevitable." Unpublished Manuscript.

Jehiel, Philippe. 2003. "Bounded Rationality and Imperfect Learning: Game Theory vs AI." *Greek Economic Review.*

Liang, Annie. 2020. "Games of Incomplete Information Played By Statisticians." Mimeo.

Leisten, Matthew. 2021. "Algorithmic Competition, with Humans." Mimeo.

Martin, Simon and Alexander Rasch. 2022. Collusion by algorithm: The role of unobserved actions. mimeo.

Maskin, Eric and Jean Tirole. 2001. "Markov perfect equilibrium: I. Observable actions." *Journal of Economic Theory* 100(2): 191–219.

Mehra, Salil. 2015. "Antitrust and the robo-seller: Competition in the time of algorithm." *Minnesota Law Review* 1323-1375.

Miklós-Thal, Jeanine and Catherine Tucker. 2019. "Collusion by algorithm: Does better demand prediction facilitate coordination between sellers?" *Management Science* 65(4) : 1552–1561.

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei Rusu, Joel Veness, Marc Bellemare, Alex Graves, Martin Riedmiller, Andres Fidjeland, Georg Ostrovski, et al. (2015). "Human-level control through deep reinforcement learning.?? *Nature* 518: 529-533.

Normann, Hans-Theo, and Martin Sternberg. 2021. "Hybrid Collusion: Algorithmic Pricing in Human-Computer Laboratory Markets." Mimeo.

O'Connor, Jason, and Nathan Wilson. 2019. "Reduced Demand Uncertainty and the Sustainability of Collusion: How AI Could Affect Competition." Working Paper.

OECD. 2017. "Algorithms and Collusion: Competition Policy in the Digital Age."

Pakes, Ariel and Paul McGuire. 1994. '"Computing Markov-Perfect Nash Equilibria: Numerical Implications of a Dynamic Differentiated Product Model." *The RAND Journal of Economics* 25(4):555-589.

Pakes, Ariel and Paul McGuire. 2001. "Stochastic algorithms, symmetric Markov perfect equilibrium, and the 'curse' of dimensionality." *Econometrica* 69(5): 1261–1281.

Posner, Richard. 1976. "Antitrust Law: An Economic Perspective." University of Chicago Press, Chicago.

Salcedo, Bruno. 2015. "Pricing Algorithms and Tacit Collusion." Working Paper.

Sandholm, Tuomas, and Robert Crites. 1995. "On multiagent Q-learning in a semi-competitive domain." *International Joint Conference on Artificial Intelligence* 191-205.

Schwalbe, Ulrich. 2018. "Algorithms, Machine Learning, and Collusion." *Journal of Competition Law and Economics* 14(4): 568-607.

Sims, Rod. 2017. "The ACCC's approach to colluding robots."

Sutton, Richard S., and Andrew G. Barto. 2018. "Reinforcement Learning: An Introduction." Cambridge, MA: MIT Press.

Tesauro, Gerald, and Jeffrey O. Kephart. 2002. "Pricing in agent economies using multi-agent Q-learning." *Autonomous Agents and Multi-Agent Systems* 5(3): 289-304.

Turner, Donald. 1962. "The definition of agreement under the Sherman Act: Conscious parallelism and refusals to deal." *Harvard Law Review* 75: 655-706.

Veljanovski, Cento. 2022. "Algorithmic Antitrust: A Critical Overview." In *Economic Analysis of Law in European Legal Scholarship.* , ed. Aurelien Portuese, 39-64. Cham:Springer.

Waltman, Ludo, and Uzay Kaymak. 2008. "Q-learning agents in a Cournot oligopoly model." *Journal of Economic Dynamics and Control* 32(10): 3275-3293.

Watkins, Christopher. 1989. “Learning from delayed rewards.” PhD Thesis, University of Cambridge, England.

Watkins, Christopher and Peter Dayan. 1992. “Q-learning.” *Machine Learning* 8: 279-292.

Whinston, Michael. 2006. *Lectures on Antitrust Economics* MIT Press, Cambridge.

Xue, Lei, Changying Sun, Donald Wunsch, Yingjiang Zhou, and Fang Yu (2018). “An adaptive strategy via reinforcement learning for the prisoner’s dilemma game.” *Journal of Automatica Sinica* 5(1): 301-310.

A Appendix

A.1 Code for the baseline

The code below, comprises the core content of the m-file required to run the baseline model in Matlab. It is included here to assist the reader in understanding implementation. The code was run in Matlab version R2015a.

```
%------------Set parameters ----------------------------%
theta=10; D=1; nfirms=2; c=2; alpha=.1;  beta=0;
pL=0.01;  pH=theta; dim=100;  p=(linspace(pL,pH,dim))';
%------------Set updating protocol-----------------------------%
synchronous = 0;
%------------Initialization-----------------------------%
rng(2); W = []; W(:,1,1) = 10+10*rand(dim,1); W(:,1,2) = 10+10*rand(dim,1);
itermax=10000;  h=[]; pstar=[]; k=1;
%------------Loop through learning periods-----------------------------%
while k<=itermax
   %------------determine price chosen-----------------------------%
   for i=1:nfirms
         [piestar2, m] = max(W(:,k,i)); pstar(i,k)=p(m,:);
   end
   %------------Updating-----------------------------%
   for i=1:nfirms
         pstaro(i,k)= pstar(i,k); pstar(i,k)=pH+1;
         prival=min(pstar(:,k)); nshares=sum(pstar(:,k)==prival) + 1;
         less= find(p<prival);
         equal= find(p==prival);
         above= find(p>prival);
         W_all(less,k+1,i)= D*(p(less)-c);
         W_all(equal,k+1,i)= (1/nshares)*D*(p(equal)-c);
         W_all(above,k+1,i)= 0*(p(above)-c);
         W(:,k+1,i) = W_all(:,k+1,i);
         if synchronous == 0
                W(:,k+1,i) = W(:,k,i);
                m = find(pstaro(i,k) == p);
                W(m,k+1,i) = W_all(m,k+1,i);
         end
         W(:,k+1,i)=W(:,k+1,i)*alpha + W(:,k,i)*(1-alpha);
         pstar(i,k)= pstaro(i,k);
    end
    k=k+1;
 end
```

This code is used as the base for producing all the figures. For instance, figure 2 uses this base code, and loops over 100 simulation iterations, with a new set of initial condition draws in each iteration.

A.2 Results for a Logit demand system

Table 4: Pricing outcomes with Logit demand

Algorithm	N	Minimum	25th Percentile	Median	75th Percentile	Max
Perfect synchronous	2	2.53	2.53	2.53	2.53	2.53
	3	2.33	2.43	2.43	2.43	2.43
	5	2.33	2.33	2.33	2.33	2.33
	10	2.23	2.23	2.33	2.33	2.33
Asynchronous	2	6.17	8.13	8.59	9.19	10.00
	3	2.63	4.95	6.37	7.93	9.80
	5	2.23	2.43	2.43	2.63	9.90
	10	2.13	2.33	2.43	2.43	2.73
Asynchronous w. downward demand	2	2.33	2.63	2.89	3.59	9.60

Notes: For each N, 100 simulation runs were conducted. The distribution of prices for firm 1, once all simulations have reached a rest point, is reported. The model is as per figure 2, but for the substitution of the Logit demand system described in equation 5 and the reported changes in N. 'downward demand' specifications mirror those discussed in section 5.1.

This subsection reports results reproducing the core static computational analysis in the main text, but substituting the perfect substitutes demand system for a logit style demand system in which each firm's product is differentiated. As in the model considered in the main text, firms play a Bertrand pricing game. The form of the demand system is

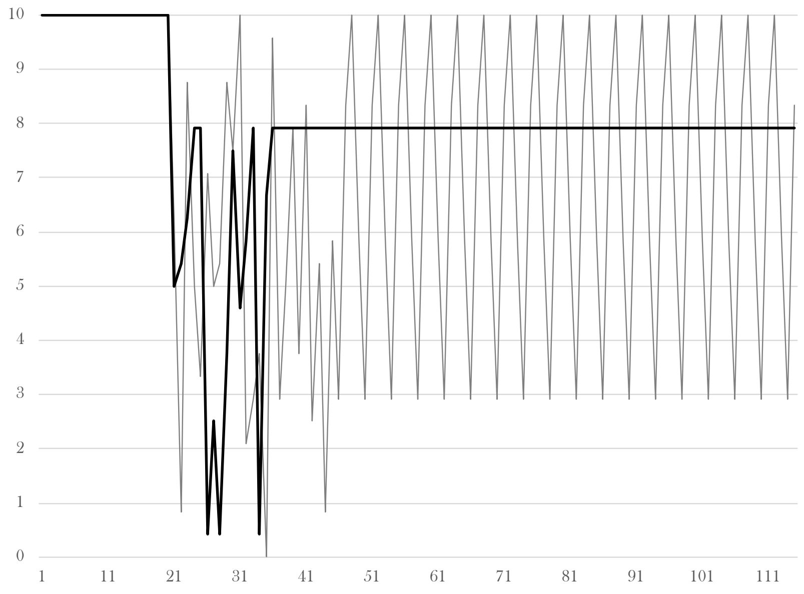
$$Q_i = \frac{e^{a-bp_i}}{1 + e^{a-bp_i} + \sum_{j \neq i} e^{a-bp_j}} \tag{5}$$

where Q_i is the quantity demanded of firm i's product. The parameters are set such that $a = 40$ and $b = 4$. In all other respects, the model and parametrization (including the selection of initial values) are unchanged from that underlying figure 2.

Table 4 shows the distribution of rest points reached in this new pricing game as the number of firms varies. It shows that the core results are qualitatively unchanged in the alternative Logit demand environment.[48]

A.3 Policies in the repeated game with asynchronous learning

Figure 7: Price paths following an optimal deviation



Notes: The underlying model is that in figure 5, panel (B).

Figure 7 provides examples of the path of play, for two simulations, with different initial conditions, in the repeated game with asynchronous learning. The path of play is at the rest point in the first 20 periods ($p = 10$). At period

[48]Differences in time taken to reach a rest point are also qualitatively similar. For $N = 2$, it takes 2,910 iterations to reach the reported distribution in the asynchronous case, and 100 iterations in the perfect synchronous case.

21, firm 1 drops the price by an increment (this perturbation is imposed exogenously). Firm 2 responds according to the policies in memory (the AI does not do any learning in this sequence of play). Figure 7 shows the path of play of firm 2, in each of the two simulations selected as examples. As can be seen, both paths bounce around a lot for the 20-30 periods following the deviation and then settle down. One settles down at a constant price, while the other settles into a regular oscillating pattern.

A.4 Results for a Cournot game

This subsection reports results reproducing the basic core static computational analysis in the main text, but replacing the Bertrand game with a homogenous good Cournot game. The form of the demand system used in this new game form is

$$Q = a - P \tag{6}$$

where $Q = \sum_i q_i$, q_i is the chosen output of firm i and P is the market price. The model is parameterized such that the demand intercept, a, is equal to 10. Quantities (q_i's) can take on 150 values evenly spaced between 1.51 and 3, inclusive. Initial conditions are set such that each initial $W(q)$ is an independent draw from $U[25, 35]$. In all other respects the model and parametrization are unchanged from that underlying figure 2.

Table 5: Pricing outcomes in Cournot

Algorithm	N	Minimum	25th Percentile	Median	75th Percentile	Max
Perfect synchronous	2	4.67	4.67	4.67	4.67	4.67
Asynchronous	2	5.24	5.65	5.84	6.00	6.42

Notes: For each N, 100 simulation runs were conducted. The distribution of prices once all simulations have reached a rest point is reported.

Table 5 shows the distribution of rest points reached in this new pricing game for $N = 2$. It shows that the core results are qualitatively unchanged in the Cournot environment, illustrating that the patterns are not specific to games of strategic complements. The same patterns can be seen in games of strategic substitutes.[49]

A.5 Results when one firm learns with an asynchronous algorithm and one with a perfect synchronous algorithm

This subsection reports results for the model underlying figure 2, with the adjustment that one firm's algorithm uses perfect synchronous updating, while the other uses asynchronous updating. Table 6 reports moments of the resulting price distribution when $N = 2$.[50] Results for Logit demand (see section A.2) are also reported.

Table 6: Pricing: Asynchronous versus perfect synchronous

Algorithm	N	Minimum	25th Percentile	Median	75th Percentile	Max
Homogenous Bertrand						
Perfect synchronous (Firm 1)	2	2.13	2.13	2.13	2.13	2.13
Asynchronous (Firm 2)	2	2.13	2.13	2.13	2.13	2.13
Differentiated Bertrand (Logit)						
Perfect synchronous (Firm 1)	2	2.73	2.94	3.69	4.45	7.28
Asynchronous (Firm 2)	2	2.84	3.14	4.10	4.95	7.98

Notes: For each N, 100 simulation runs were conducted. The distribution of prices once all simulations have reached a rest point is reported.

[49]Differences in time taken to reach a rest point are also qualitatively similar. It takes 6,706 iterations to reach the reported distribution in the asynchronous case, and 346 iterations in the perfect synchronous case.

[50]It takes 950 iterations to reach the reported distribution