Artificial Intelligence, Algorithmic Pricing, and Collusion

This is the Accepted Manuscprit of

Calvano, E., Calzolari, G., Denicolò, V., & Pastorello, S. (2020). Artificial Intelligence, Algorithmic Pricing, and Collusion. *American Economic Review*, *110*(10), 3267–3297.

The editorial version is at DOI: https://doi.org/10.1257/aer.20190623

ARTIFICIAL INTELLIGENCE, ALGORITHMIC PRICING AND COLLUSION[1]

Emilio Calvano, Giacomo Calzolari, and Vincenzo Denicolo, and Sergio` Pastorello

May 2020

Increasingly, algorithms are supplanting human decision-makers in pricing goods and services. To analyze the possible consequences, we study experimentally the behavior of algorithms powered by Artificial Intelligence (Q-learning) in a workhorse oligopoly model of repeated price competition. We find that the algorithms consistently learn to charge supra-competitive prices, without communicating with one another. The high prices are sustained by collusive strategies with a finite phase of punishment followed by a gradual return to cooperation. This finding is robust to asymmetries in cost or demand, changes in the number of players, and various forms of uncertainty.

Keywords: Artificial Intelligence, Pricing-Algorithms, Collusion, Reinforcement Learning, Q-Learning.

J.E.L. codes: L41, L13, D43, D83.

Software programs are increasingly being adopted by firms to price their goods and services, and this tendency is likely to continue.[2] In this paper, we ask whether pricing algorithms may "autonomously" learn to collude. The possibility arises because of the recent evolution of the software, from rule-based to reinforcement learning programs.

[1] Calvano: Bologna University, Toulouse School of Economics and CEPR (emilio.calvano@unibo.it). Calzolari (corresponding author): European University Institute, Bologna University, Toulouse School of Economics and CEPR (giacomo.calzolari@eui.ei). Denicol: Bologna University and CEPR (vincenzo.denciolo@unibo.it). Pastorello: Bologna University (sergio.pastorello@unibo.it). We are grateful to the Editor, Jeffrey Ely, and three anonymous referees for many detailed and helpful comments. We also thank, without implicating, Susan Athey, Ariel Ezrachi, Joshua Gans, Joe Harrington, Bruno Jullien, Timo Klein, Kai-Uwe Ku¨hn, Patrick Legros, David Levine, Wally Mullin, Yossi Spiegel, Steve Tadelis, Emanuele Tarantino and participants at numerous conferences and seminars. Financial support from the Digital Chair initiative at the Toulouse School of Economics is gratefully acknowledged.

[2] While revenue management programs have been used for decades in such industries as hotels and airlines, the diffusion of pricing software has boomed with the advent of online marketplaces. For example, in a sample of over 1,600 best-selling items listed on Amazon, Chen, Mislove and Wilson (2016) find that in 2015 more than a third of the vendors had already automated their pricing. But pricing software is increasingly used also in traditional off-line sectors such as gas stations: see e.g. "Why do gas station prices constantly change? Blame the algorithms," *The Wall Street Journal*, May 8, 2017.

The new programs, powered by Artificial Intelligence (AI), are indeed much more autonomous than their precursors. They can develop their pricing strategies from scratch, engaging in active experimentation and adapting to changing environments. In this learning process, they require little or no external guidance.

In the light of these developments, concerns have been voiced, by scholars and policymakers alike, that AI pricing algorithms may raise their prices above the competitive level in a coordinated fashion, even if they have not been specifically instructed to do so and even if they do not communicate with one another.[3] This form of tacit collusion would defy current antitrust policy, which typically targets only explicit agreements among would-be competitors (Harrington, 2018).

But how real is the risk of tacit collusion among algorithms? That is a difficult question to answer, both empirically and theoretically. On the empirical side, collusion is notoriously hard to detect from market outcomes,[4] and firms typically do not disclose details of the pricing software they use. On the theoretical side, the interaction among reinforcementlearning algorithms in pricing games generates stochastic dynamic systems so complex that analytical results seem currently out of reach.[5]

To make some progress, this paper takes an experimental approach. We construct AI pricing agents and let them interact repeatedly in computer-simulated marketplaces. The challenge of this approach is to choose realistic economic environments, and algorithms representative of those employed in practice. We discuss in detail how we address these challenges as we proceed. Any conclusions are necessarily tentative at this stage, but our findings do suggest that algorithmic collusion is more than a remote theoretical possibility.

The results indicate that, indeed, relatively simple pricing algorithms systematically learn to play collusive strategies. The algorithms typically coordinate on prices that are somewhat below the monopoly level but substantially above the static Bertrand

[3] For the scholarly debate see, for instance, Ezrachi and Stucke (2016, 2017), Harrington (2018), Ku¨hn and Tadelis (2018) and Schwalbe (2019). As for policy, the possibility of algorithmic collusion has been extensively discussed, for instance, at the 7th session of the FTC Hearings on competition and consumer protection (November 2018) and has been the subject of white papers independently issued in 2018 by the Canadian Competition Bureau and the British Competition and Market Authority.

[4] With rich enough data, however, the problem may not be insurmountable (Byrne and De Roos (2019)).

[5] One notable theoretical contribution is Salcedo (2015), who argues that optimized algorithms will inevitably reach a collusive outcome. But this claim hinges crucially on the assumption that each algorithm can periodically observe and "decode" the others, which in the meantime stay unchanged. The practical relevance of Salcedo's result thus remains controversial.

equilibrium. The strategies that generate these outcomes crucially involve punishments of defections.

Such punishments are finite in duration, with a gradual return to the pre-deviation prices. The algorithms learn these strategies purely by trial and error. They are not designed or instructed to collude, they do not communicate with one another, and they have no prior knowledge of the environment in which they operate.

Our baseline model is a symmetric duopoly with deterministic demand, but we conduct an extensive robustness analysis. The degree of collusion decreases as the number of competitors rises. However, substantial collusion continues to prevail when the active firms are three or four in number. The algorithms display a stubborn propensity to collude even when they are asymmetric, and when they operate in stochastic environments.

Other papers have simulated reinforcement-learning algorithms in oligopoly, but ours is the first to clearly document the emergence of collusive strategies among autonomous pricing agents. The previous literature in both computer science and economics has focused on outcomes rather than strategies.[6] But the observation of supra-competitive prices is not, per se, genuine proof of collusion. To us economists, collusion is not simply a synonym of high prices but crucially involves "a reward-punishment scheme designed to provide the incentives for firms to consistently price above the competitive level" (Harrington (2018), p. 336). The reward-punishment scheme ensures that the supra-competitive outcomes may be obtained *in equilibrium* and do not result from a failure to optimize.

The difference is important. For example, in their pioneering study of repeated Cournot competition among Q-learning algorithms, computer scientists Waltman and Kaymak (2008) find that the algorithms reduce output, and hence raise prices, with respect to the Nash equilibrium of the one-shot game.[7] They refer to this as collusion. When the algorithms are far-sighted and are able to condition their current choices on past actions, so that defections can be punished, their findings could indeed be consistent with

[6] Moreover, the vast majority of the literature does not use the canonical model of collusion, where firms play an infinitely repeated game, pricing simultaneously in each stage and conditioning their prices on past history. Rather, it uses frameworks similar to Maskin and Tirole (1988) model of staggered pricing. In this model, two firms alternate in moving, commit to a price level for two periods, and condition their pricing only on rival's current price. (See the recent contribution of Klein (2018), which provides also a survey of the earlier literature.) The postulate of price commitment is however controversial, as software algorithms can adjust prices very quickly. And probably the postulate is not innocuous. Commitment may indeed facilitate coordination, as argued theoretically by Maskin and Tirole (1988) and experimentally by Leufkens and Peeters (2011).

[7] Other papers that study reinforcement learning algorithms in a Cournot oligopoly include Kimbrough

collusive behavior according to economists' usage of the term. But Waltman and Kaymak consider also the case where algorithms are myopic and have no memory of past actions – conditions under which collusion is either unfeasible or cannot emerge in equilibrium – and find that in these cases the output reduction is even larger. This raises doubts that what they observe may not be collusion but a failure to learn an optimal strategy.[7]

Verifying whether the high prices are supported by equilibrium strategies is not just a theoretical curiosity. Algorithms that grossly fail to optimize would, in all likelihood, be dismissed quickly and thus could hardly become a matter of antitrust concern. The implications are instead very different if, as we show, the supra-competitive prices are set by optimizing, or quasi-optimizing, programs.

Yet, there is an important caveat to keep in mind. To present a proof-of-concept demonstration of algorithmic collusion, in this paper we concentrate on what the algorithms eventually learn and pay less attention to the speed of learning. Thus, we focus on algorithms that by design learn slowly, in a completely unsupervised fashion, and in our simulations we allow them to explore widely and interact as many times as is needed to stabilize their behavior. As a result, the number of repetitions required for completing the learning is typically high, on the order of hundreds of thousands. In fact, the algorithms start to raise their prices much earlier. However, the time scale remains an open issue; it will be discussed further below.

The rest of the paper is organized as follows. The next section provides a self-contained description of the class of Q-learning algorithms, which we use in our simulations. Section 3 describes the economic environments where the algorithms operate. Section 4 shows that collusive outcomes are common and are generated by optimizing, or quasi-optimizing, behavior. Section 5 then provides a more in-depth analysis of the strategies that lead to these outcomes. Section 6 reports on a number of robustness checks. Section 7 discusses the issue of the speed of learning. Section 8 concludes with a brief discussion of the possible implications for policy.

---

and Murphy (2009) and Siallagan et al (2013).

[7]According to Cooper, Homem-de-Mello and Kleywegt (2015), such "collusion by mistake" may sometimes emerge also among revenue management systems that do not condition their current prices on rivals' past prices. This may happen in particular when the programs disregard competitors altogether in the process of demand estimation, which biases the estimated elasticity downwards.

1. Q-LEARNING

Following Waltman and Kaymak (2008), we concentrate on Q-learning algorithms. Even if reinforcement learning comes in many different varieties,[8] there are several reasons for this choice. First, one would like to experiment with algorithms that are commonly adopted in practice, and although little is known on the specific software that firms actually use, Q-learning is certainly highly popular among computer scientists. Second, Q-learning algorithms are simple and can be fully characterized by just a few parameters, the economic interpretation of which is clear. This makes it possible to keep possibly arbitrary modeling choices to a minimum, and to conduct a comprehensive comparative statics analysis with respect to the characteristics of the algorithms. Third, Q-learning algorithms share the same architecture as the more sophisticated programs that have recently obtained spectacular successes, achieving superhuman performances in such tasks as playing the ancient board game Go (Silver et al., 2016), the Atari video-games (Mnih et al., 2015), and, more recently, chess (Silver et al., 2018).[9] The downside of Q-learning is that the learning process is slow, for reasons that will become clear in a moment.

In the rest of this section, we provide a brief introduction to Q-learning. Readers familiar with this model may proceed directly to section 3.

1.1. *Single agent problems*

Like all reinforcement-learning algorithms, Q-learning programs adapt their behavior to past experience, taking actions that have proven successful more frequently and unsuccessful ones less frequently. In this way, they may learn an optimal policy, or a policy that approximates the optimum, with no prior knowledge of the particular problem at hand.[9]

Originally, Q-learning was proposed by Watkins (1989) to tackle Markov decision processes. In a stationary Markov decision process, in each period $t = 0,1,2,...$ an agent observes a state variable $s_t \in S$ and then chooses an action $a_t \in A(s_t)$. For any s_t and a_t, the

[8] For a thorough treatment of reinforcement learning in computer science, see Sutton and Barto (2018).
[9]These more sophisticated programs might appear themselves to be a natural alternative to Q-learning. However, they require many modeling choices that are somewhat arbitrary from an economic viewpoint. We shall come back to this issue in Section 7.

[9] Reinforcement learning was introduced in economics by Arthur (1991) and later popularized by Roth and Erev (1995), Erev and Roth (1998) and Ho, Camerer and Chong (2007), among others.

agent obtains a reward π_t, and the system moves on to the next state s_{t+1}, according to a time-invariant (and possibly degenerate) probability distribution $F(\pi_t,s_{t+1}|s_t,a_t)$. Q-learning deals with the version of this model where S and A are finite, and A is not state-dependent.

The decision maker's problem is to maximize the expected present value of the reward stream:

$$E\left[\sum_{t=0}^{\infty}\delta^t\pi_t\right], \tag{1}$$

where $\delta < 1$ represents the discount factor. This dynamic programming problem is usually attacked by means of Bellman's value function

$$V(s) = \max_{a\in A}\{E[\pi|s,a] + \delta E[V(s^0)|s,a]\}, \tag{2}$$

where s^0 is a shorthand for s_{t+1}. For our purposes it is convenient to consider instead a precursor of the value function, namely the Q-function representing the discounted payoff of taking action a in state s.[10] It is implicitly defined as:

$$Q(s,a) = E(\pi|s,a) + \delta E[\max_{a^0\in A} Q(s^0,a^0)|s,a], \tag{3}$$

where the first term on the right-hand side is the period payoff and the second term is the continuation value.[11] The Q-function is related to the value function by the simple identity $V(s) \equiv \max_{a\in A} Q(s,a)$. Since S and A are finite, the Q-function can in fact be represented as an $|S| \times |A|$ matrix.

1.1.1. *Learning*

If the agent knew the Q-matrix, he could then easily calculate the optimal action for any given state. Q-learning is essentially a method for estimating the Q-matrix without knowing the underlying model, i.e. the distribution function $F(\pi,s^0|s,a)$.

[10] The term Q-function derives from the fact that the Q-value can be thought of as an index of the "Quality" of action a in state s.

[11] This is uniquely defined even if the maximization problem does not have a unique solution.

Q-learning algorithms estimate the Q-matrix by an iterative procedure. Starting from an arbitrary initial matrix $\mathbf{Q}_0$, after choosing action a_t in state s_t, the algorithm observes π_t and s_{t+1} and updates the corresponding cell of the matrix $Q_t(s,a)$ for $s = s_t$, $a = a_t$, according to the learning equation:

$$Q_{t+1}(s,a) = (1-\alpha)Q_t(s,a) + \alpha\left[\pi_t + \delta \max_{a\in A} Q_t(s',a)\right]. \tag{4}$$

Equation (4) tells us that for the cell visited, the new value $Q_{t+1}(s,a)$ is a convex combination of the previous value and the current reward plus the discounted value of the state that is reached next. For all other cells $s \neq s_t$ and $a \neq a_t$, the Q-value does not change:

$Q_{t+1}(s,a) = Q_t(s,a)$. The weight $\alpha \in [0,1]$ is called the learning rate.

1.1.2. *Experimentation*

To have a chance to approximate the true matrix starting from an arbitrary $\mathbf{Q}_0$, all actions must be tried in all states. This means that the algorithm has to be instructed to experiment, i.e. to gather new information by selecting actions that may appear suboptimal in the light of the knowledge acquired in the past. Plainly, such exploration is costly and thus entails a trade-off between continuing to learn and exploiting the stock of knowledge already acquired. Finding the optimal resolution to this trade-off may be problematic, but Q-learning algorithms do not even try to optimize in this respect: the mode and intensity of the exploration are specified exogenously.

The simplest possible exploration policy – sometimes called the ε-greedy model of exploration – is to choose the currently optimal action (i.e., the one with the highest Q-value in the relevant state, also known as the "greedy" action) with a fixed probability $1 - \varepsilon$ and to randomize uniformly across all actions with probability ε. Thus, $1 - \varepsilon$ is the fraction of times the algorithm is in *exploitation mode*, while ε is the fraction of times it is in *exploration mode*. Even if more sophisticated exploration policies can be designed,[13] in our analysis we shall mostly focus on the ε-greedy specification.

Under certain conditions, Q-learning algorithms converge to the optimal policy (Watkins and Dayan, 1992).[14] However, completing the learning process may take quite a long time. Q-learning is slow because it updates only one cell of the Q-matrix at a time, and approximating the true matrix generally requires that each cell be visited many times.
The larger the state or action space, the more iterations will be needed.

[13]For example, one may let the probability with which sub-optimal actions are tried depend on their respective Q-values, as in the so-called Boltzmann experimentation model. In this model, actions are chosen with probabilities

$$\Pr(a_t = a) = \frac{e^{Q_t(s_t,a)/T}}{\sum_{a' \in A} e^{Q_t(s_t,a')/T}}$$

where the parameter T is often called the system's "temperature." As long as $T > 0$, all actions are chosen with positive probability. When $T = 0$, however, the algorithm chooses the action with the highest Q-value with probability 1.

[14]A sufficient condition is that the algorithm's exploration policy belong to a class known as *Greedy in the Limit with Infinite Exploration* (GLIE). Loosely speaking, this requires that exploration decreases over time; that if a state is visited infinitely often, the probability of choosing any feasible action in that state be always positive (albeit arbitrarily small); and that the probability of choosing the greedy action go to one as $t \to \infty$.

1.2. *Repeated games*

Although Q-learning was originally designed to deal with stationary Markov decision processes, it can also be applied to repeated games. The simplest approach is to let the algorithms continue to update their Q-matrices according to (4), treating rivals' actions just like any other possibly relevant state variable.[12]

But in repeated games stationarity is inevitably lost, even if the stage game does not change from one period to the next. One source of non-stationarity is that if the state s_t included players' actions in all previous periods, the set of states S would increase with time. But this problem can be avoided by bounding players' memory. With bounded recall, a state s will include only the actions chosen in the last k stages, implying that the state space may be finite and time-invariant.

A more serious problem is that in repeated games the per-period payoff and the transition to the next state generally depend on the actions of all the players. If a player's rivals change their actions over time – because they are experimenting or learning, or both – the player's optimization problem becomes inherently non-stationary.

Such non-stationarity is at the root of the lack of general convergence results for Q-learning in games.[13] There is no ex ante guarantee that several Q-learning agents

[12] In the computer science literature, this approach is called *independent learning*. An alternative approach, i.e. *joint learning* , tries to predict other players' actions by means of some sort of equilibrium notion. However, the joint learning approach is still largely unsettled (Nowe et al. (2012)).

[13] Non-stationarity considerably complicates the theoretical analysis of the stochastic dynamic systems describing Q-learning agents' play of repeated games. A common approach uses stochastic approximation

interacting repeatedly will settle on a stable outcome, nor that they will learn an optimal policy (i.e., collectively, a Nash equilibrium of the repeated game with bounded memory). Nevertheless, convergence and equilibrium play may hold in practice. This can be verified only ex-post, however, as we shall do in what follows.

2. EXPERIMENT DESIGN

We have constructed Q-learning algorithms and let them interact in a repeated Bertrand oligopoly setting. For each set of parameters, an "experiment" consists of 1,000 sessions. In each session, agents play against the same opponents until convergence as defined below.

Here we describe the economic environment in which the algorithms operate, the exploration strategy they follow, and other aspects of the numerical simulations.

2.1. *Economic environment*

We use the canonical model of collusion, i.e. an infinitely repeated pricing game in which all firms act simultaneously and condition their actions on history. We depart from the canonical model only in assuming a bounded memory, for the reasons explained in the previous section.

We take as our stage game a simple model of price competition with logit demand and constant marginal costs. This model has been applied extensively in empirical work, demonstrating that it is flexible enough to fit many different industries.

There are n differentiated products and an outside good. In each period t, the demand for product $i = 1,2,...,n$ is:

techniques (Benveniste, Metivier and Priouret, 1990), with which one can turn stochastic dynamic systems into deterministic ones. This approach has made some progress in the analysis of memoryless systems. The resulting deterministic system is typically a combination of the replicator dynamics of evolutionary games and a mutation term that captures the algorithms' exploration. See e.g. Borgers and Sarin (1997) for the reinforcement learning model of Cross (1973), Hopkins (2002) and Beggs (2005) for that of Erev and Roth (1998), and Bloembergen et al. (2015) for memoryless Q-learning. The application of stochastic approximation techniques to AI agents with memory is more subtle and is currently at the frontier of research, both in computer science and in statistical physics (Barfuss, Donges and Kurths, 2019). To the best of our knowledge, there are no results yet available for ε-greedy Q-learning. But what we know for simpler algorithms suggests that, eventually, the dynamic systems that emerge from the stochastic approximation would have to be integrated numerically. If this is so, however, there is little to gain compared with simulating the exact stochastic system a large number of times so as to smooth out uncertainty, as we do in what follows.

$$q_{i,t} = \frac{e^{\frac{a_i - p_{i,t}}{\mu}}}{\sum_{j=1}^{n} e^{\frac{a_j - p_{j,t}}{\mu}} + e^{\frac{a_0}{\mu}}}. \quad (5)$$

The parameters a_i are product quality indexes that capture vertical differentiation. Product 0 is the outside good, so a_0 is an inverse index of aggregate demand. Parameter μ is an index of horizontal differentiation; the case of perfect substitutes is obtained in the limit as $\mu \to 0$.

Each product is supplied by a different firm, so n is also the number of firms. The perperiod reward accruing to firm i is then $\pi_{i,t} = (p_{i,t} - c_i)q_{i,t}$, where c_i is the marginal cost.

As usual, fixed costs are irrelevant as long as firms stay active.

2.2. *Action space*

Since Q-learning requires a finite action space, we discretize the model as follows. For each value of the parameters, we compute both the Bertrand-Nash equilibrium of the one-shot game and the monopoly prices (i.e., those that maximize aggregate profits). These are denoted by $\mathbf{p}^N$ and $\mathbf{p}^M$, respectively. Then, we take the set A of the feasible prices to be given by m equally spaced points in the interval $[\mathbf{p}^N - \xi(\mathbf{p}^M - \mathbf{p}^N), \mathbf{p}^M + \xi(\mathbf{p}^M - \mathbf{p}^N)]$, where $\xi > 0$ is a parameter. So prices range from below Bertrand to above monopoly.

This discretization of the action space implies that the exact Bertrand and monopoly prices may not be feasible, however, so there may be mixed-strategy equilibria both in the stage and in the repeated game. Since by design our algorithms play pure strategies (as a tie-breaking rule, they are instructed to choose the lowest price), they might then oscillate around a target that is not feasible.

2.3. *Memory*

To ensure that the state space is finite, we posit a bounded memory. Thus, the state is the set of all past prices in the last k periods:

(6) $s_t = \{\mathbf{p}_{t-1},...,\mathbf{p}_{t-k}\}$, where k is the length of the memory.[14]

Our assumptions imply that for each player i we have $|A| = m$ and $|S| = m^{nk}$.

2.4. *Exploration*

We use the ε-greedy model with a time-declining exploration rate. Specifically, we set

(7) $\varepsilon_t = e^{-\beta t}$,

where $\beta > 0$ is a parameter. This means that initially the algorithms choose in purely random fashion, but as time passes, they make the greedy choice more and more frequently.
The greater β, the faster the exploration diminishes.

2.5. *Baseline parametrization and initialization*

Initially, we focus on a baseline economic environment that consists of a symmetric duopoly ($n = 2$) with $c_i = 1,\ a_i - c_i = 1,\ a_0 = 0,\ \mu = \frac{1}{4},\ \delta = 0.95,\ m = 15,\ \xi = 0.1$ and a one-period memory ($k = 1$).[15] For this specification, the price-cost margin is ≈ 47% in the static Bertrand equilibrium, and about twice as large under perfect collusion.

As for the initial matrix $\mathbf{Q}_0$, our baseline choice is to set the Q-values at $t = 0$ at the discounted payoff that would accrue to player i if opponents randomized uniformly:

(8) $$Q_{i,0}(s, a_i) = \frac{\sum_{a_{-i} \in A^{n-1}} \pi_i(a_i, a_{-i})}{(1-\delta)\,|A|^{n-1}}.$$

This is in keeping with the assumption that at first the choices are purely random. In a similar spirit, the initial state s_0 is drawn randomly at the beginning of each session.

[14] The assumption here is perfect monitoring, which is reasonable for many online marketplaces. For example, Amazon's APIs allow sellers to recover current and past prices of any product with a simple query.

[15] It is worth noting that while the assumption of a one-period memory is restrictive, it might have a limited impact on the sustainability of collusion, because the richness of the state space may substitute

Starting from this baseline set up, we have performed extensive robustness analyses, the results of which are reported in Section 6 and the supplementary material file.

3. OUTCOMES

In this section, we focus on our baseline environment and explore the entire grid of the 100 × 100 points that are obtained by varying the learning and experimentation parameters α and β as described presently.[19] The aim of this exercise is to show (*i*) that non-competitive outcomes are common, not obtained at just a few selected points, and (*ii*) that these outcomes are generated by optimizing, or quasi-optimizing, behavior. Once these conclusions are established, in the next section we shall focus on one point of the grid to provide a deeper analysis of the mechanism of collusion.

3.1. *Parameter grid*

The learning parameter α may in principle range from 0 to 1, but it is well known that high values of α may disrupt learning when experimentation is extensive, as the algorithm would forget too rapidly what it has learned in the past. To be effective, learning must be persistent, which requires that α be relatively small. In the computer science literature, a value of 0.1 is often used. Accordingly, our initial grid comprises 100 equally spaced points in the interval [0.025,0.25].

As for the experimentation parameter β, the trade-off is as follows. On the one hand, the algorithms need to explore extensively, as the only way to learn is multiple visits to every state-action cell (of which there are 3,375 in our baseline experiments with $n = 2$, $m = 15$ and $k = 1$, and many more in more complex environments). On the other hand, exploration is costly. One can abstract from the short-run cost by considering long-run outcomes. But exploration entails another cost as well, in that if one algorithm experiments more extensively, this creates noise in the environment, which makes it harder for the other to learn. This externality means that in principle experimentation may be excessive even discounting the short-term cost.

---

for the length of the memory. Indeed, folk theorems with bounded memory have been proved by Barlo, Carmona and Sabourian (2009) for the case of infinite action space, and by Barlo, Carmona and Sabourian (2016) for the case where the action space is finite.

[19] In this paper, we regard α and β as exogenous parameters. It might be interesting, however, to consider a game of delegation where α and β (and possibly also the initial matrix $\mathbf{Q}_0$) are chosen strategically by firms.

To get a sense of what values of β might be reasonable, it may be useful to map β into the expected number of times a cell would be visited purely by random exploration (rather than by greedy choice), over an infinite time horizon. This number is finite as exploration eventually fades away and is denoted by ν.[16] We take as a lower bound $\nu = 4$, which seems barely sufficient to guarantee decent learning. For example, with $\alpha = 0.25$ the initial Qvalue of a cell would still carry a weight of more than 30% after 4 updates, and the weight would be even greater for lower values of α. (In fact, later we shall mostly focus on larger values of ν.)

When $n = 2$ and $m = 15$, the lower bound of 4 on ν implies an upper bound for β of (approximately) $\bar{\beta} = 2 \times 10^{-5}$. As we did for α, we then take 100 equally spaced points in the interval from 0 to $\bar{\beta}$. The lowest value of β we consider proceeding in this way corresponds to $\nu \approx 450$.

3.2. *Convergence*

As mentioned, for strategic games played by Q-learning algorithms there are no general convergence results: we do not know whether the algorithms converge at all; or, if they do, whether they converge to a Nash equilibrium. But while they are not guaranteed, convergence and optimization are not ruled out either, and they can be verified ex post.

To verify convergence, we use the following practical criterion: convergence is deemed to be achieved if for each player the optimal strategy does not change for 100,000 consecutive periods. That is, if for each player i and each state s the action $a_{i,t}(s) = \text{argmax}[Q_{i,t}(a,s)]$ stays constant for 100,000 repetitions, we assume that the algorithms have completed the learning process and attained stable behavior. We stop the session when this occurs, and in any case after one billion repetitions.

Nearly all sessions converged. Typically, a great many repetitions are needed to converge.

The exact number depends on the level of exploration, ranging from about 400,000 when exploration is rather limited to several millions when it is very extensive (details in section A4.1 of the supplementary material file). For example, with $\alpha = 0.125$ and $\beta = 10^{-5}$ (the mid-point of the grid) convergence is achieved on average after 850,000 periods. So

[16] The exact relationship between ν and β is

$$\nu = \frac{(m-1)n}{m^{kn(n+1)}\left[1 - e^{-\beta\ (n+1)}\right]}$$

.

many repetitions are required for the simple reason that with $\beta = 10^{-5}$, the probability of choosing an action randomly after, say, 100,000 periods is still 14%. If the rival is experimenting at this rate, the environment is still too non-stationary for the algorithm to converge. In practice, convergence is achieved only when experimentation is nearly terminated.

It must be noted that only in some of the sessions both algorithms eventually charge a constant price period after period. A non-negligible fraction of the sessions displays price cycles (details in section A4.2). As shown in Table I below, the vast majority of these cycles have a period of two. We shall discuss the cycles more extensively later.

3.3. *Profits*

Having verified convergence, we focus on the limit behavior of our algorithms. We find, first of all, that the algorithms consistently learn to charge supra-competitive prices, obtaining a sizable extra-profit compared to the static Nash equilibrium. To quantify this extra-profit, we use the following normalized measure:

$$\Delta \equiv \frac{\overline{\pi} - \pi^N}{\pi^M - \pi^N}, \tag{9}$$

where $\overline{\pi}$ is the average per-firm profit upon convergence, π^N is the profit in the BertrandNash static equilibrium, and π^M is the profit under full collusion (monopoly). Thus, $\Delta = 0$ corresponds to the competitive outcome and $\Delta = 1$ to the perfectly collusive outcome. Taking π^M as a reference point makes sense when δ is sufficiently high that perfect collusion is attainable in a sub-game perfect equilibrium, as is the case in our baseline specification.[17] We shall refer to Δ as the average profit gain.

The average profit gain achieved upon convergence is represented in Figure 1 as a function of α and β. Over our grid, Δ ranges from 70% to 90%. The corresponding prices are almost always higher than in the one-shot Bertrand-Nash equilibrium but rarely as high as under monopoly (details in section A4.2).

[17] In fact, the largest attainable Δ is slightly lower than 1 as $\mathbf{p}^M$ can at best be approximated. However, the difference is immaterial as the profit function is flat at $\mathbf{p}^M$. On the other hand, Δ can be negative as the action set includes prices lower than $\mathbf{p}^N$. In particular, we have $\Delta \approx -2\%$ if the Nash price is approximated by defect, whereas $\Delta \approx 12\%$ if it is approximated by excess.

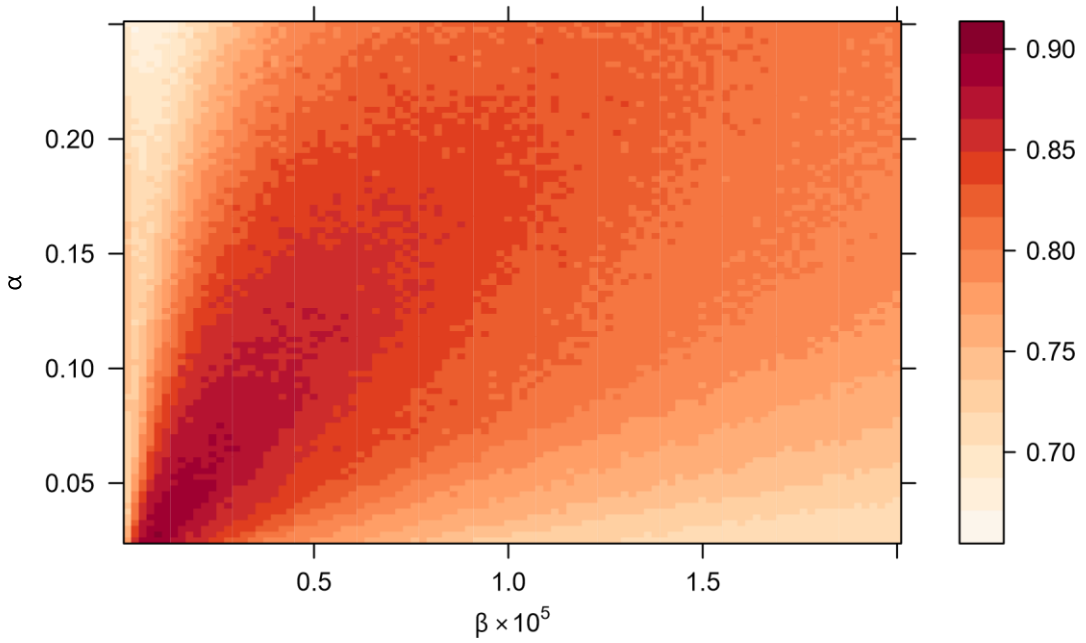


Figure 1: Average profit gain Δ for a grid of values of α and β.

The profit gain does not seem to be particularly sensitive to changes in the learning and experimentation parameters. It tends to be largest when α and β are both low, i.e., exploration is extensive and learning is persistent, but reducing either α or β too much eventually backfires.

3.4. *Equilibrium play*

Even if the algorithms almost always converge to a limit strategy, this may not be an optimal response to that of the rival. Optimality is guaranteed theoretically for singleagent decision making but not when different algorithms are involved.

But again, this property can be verified ex post. We proceed as follows. In each session, for each algorithm we calculate the theoretical Q-matrix under the assumption that the rival uses his limit strategy. This assumption serves to pin down the last term in equation (3), producing a system of linear equations that can be solved for the "true" Q-matrix. With these Q-matrices at hand, we then determine the algorithms' optimal strategies, i.e., the best responses to the rival's limit strategy, and compare them to their own limit strategies. The comparison may be limited to the states that are actually reached *on path* (verifying whether a Nash equilibrium is played), or extended to all states (verifying subgame

perfection). When an algorithm is not playing a best response, we can also compute the forfeited payoff. We express this in percentage terms and refer to it as the "Q-loss".

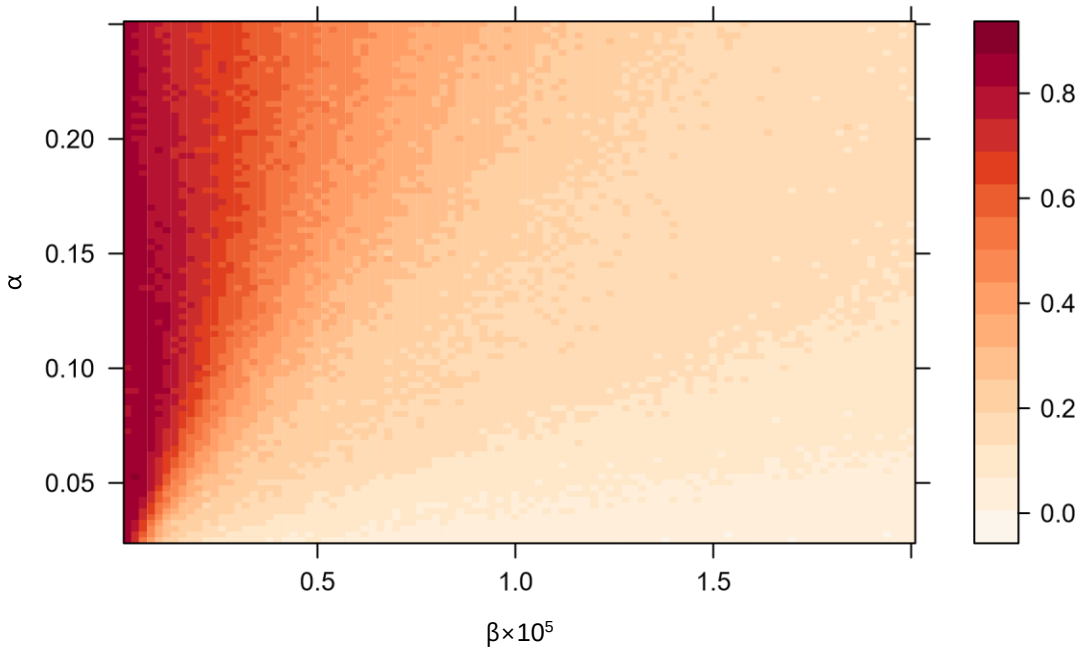


Figure 2: Fraction of sessions converging to a Nash equilibrium, for a grid of values of α and β.

Figure 2 plots the frequency of equilibrium play, i.e., the fraction of sessions where both algorithms play a best response to the rival's limit strategy, on path. Lack of equilibrium is quite common when β is large (that is, exploration is limited). This should not come as a surprise. As noted, when β is close to the upper bound of the grid, exploration is too limited to allow good learning. Nevertheless, even when the algorithms do not play a best response, they are not far from it. Most often, the Q-loss is below 0.5%, and in no point of the grid does it exceed 1.2% (details in section A4.3).

When experimentation is more extensive (i.e., towards the left side of the grid), equilibrium play becomes much more prevalent. For example, when $\alpha = 0.15$ and $\beta = 0.4 \times 10^{-5}$ (meaning that each cell is visited on average 20 times just by random exploration), about half the sessions produce equilibrium play on path, and the Q-loss is a mere 0.2% on average (see Table I below). In many cases, the reason why the algorithms are not exactly optimizing is that they approximate the price, which would be the best response in a continuous action space, by excess rather than by defect, or vice versa. A key

implication is that once the learning process is completed, there is very little scope for exploiting the algorithms, no matter how smart the opponent is.[18]

Off path, things are somewhat different. Very rarely do the algorithms play a subgame perfect equilibrium. Again, this is not surprising, given that the algorithms learn purely by trial and error, and sub-game perfection is a very demanding requirement when the state space is large.[19] Nevertheless, with enough experimentation we observe clear patterns of behavior even off path, as we shall see in the next section.

Summarizing, we have seen that once they are trained, our algorithms consistently raise their prices above the competitive level. These supra-competitive prices do not hinge on sub-optimal behavior: prices are high even if both algorithms play an optimal strategy, or come quite close to it. In fact, a comparison of Figures 1 and 2 suggests a positive, albeit modest, correlation between profit gain and equilibrium play: to be precise, Pearson's coefficient of correlation is 0.12.[20]

4. ANATOMY OF COLLUSION

In this section, we analyze the strategies that generate the anti-competitive outcomes described above. A natural question that arises when prices exceed the Nash-Bertrand level is why firms do not cut their prices. Is it because they are missing an opportunity to increase their payoff? Or is it because they realize that cutting the price would not be profitable given the rival's response in subsequent periods? And in this latter case, what would that response look like? These are the questions addressed in what follows.

To ease the exposition, we shall often focus on one point of the grid, namely $\alpha = 0.15$ and $\beta = 4 \times 10^{-6}$ but the results are robust to changes in these parameters. With these parameter values, for each cell we have on average about 20 updates just by random exploration (i.e., $\nu \approx 20$), so even for cells that are visited purely by chance, the initial Q-value counts for just 3% of their final value.

Table I reports various descriptive statistics for the experiment chosen, both jointly for all sessions and separately for those that converged to a symmetric price, to asymmetric

[18] In the computer science literature, the Q-loss is indeed called "exploitation." Whether Q-learning algorithms can be exploited during the learning phase is an interesting question for future study.

[19] However, Table I below shows that the algorithms are not far from optimizing even off path, with an average Q-loss of less than 2% for the chosen experiment (details in section A4.3).

[20] The correlation is even higher, i.e. 0.24, if equilibrium play is measured by the fraction of cases in which at least one algorithm is playing a best response to the rival's limit strategy.

prices (but still constant over time), or to cycles of differing length. The last column focuses instead on those sessions in which the algorithms have learned to play a Nash equilibrium. Two remarks are in order. First, while in almost all sessions the algorithms manage to coordinate, the exact form of the coordination varies. For example, even if the algorithms are fully symmetric ex ante, only in little more than a fourth of the sessions do they end up charging exactly the same price period after period. All the other sessions display either asymmetries or cycles, or both. Second, the cycles are associated with less equilibrium play and lower profit gain. This is true to a lesser extent for cycles of period 2, which could be interpreted as orbits around a target that is not feasible because of our discretization.[21] However, for cycles of period 3 or longer the effects are quite significant. These cycles, which might reflect the difficulty of achieving coordination purely by trial and error, are not very frequent, however: they materialize in about a tenth of the sessions.

TABLE I

	Sessions by cycle length						Nash equilibria
	1-Sym.	1-Asym.	1	2	≥3	All	
Frequency	0.277	0.366	0.643	0.238	0.119	1	0.505
Avg. Profit Gain	0.866	0.855	0.860	0.846	0.793	0.849	0.854
S.D. Profit Gain	0.115	0.114	0.114	0.104	0.097	0.112	0.108
Freq. of Nash Equilibria	0.686	0.661	0.672	0.294	0.025	0.505	1.000
Avg. Q-Loss (on path)	0.001	0.001	0.001	0.002	0.004	0.002	0.000
S.D. Q-Loss (on path)	0.002	0.004	0.003	0.003	0.006	0.004	0.000
Avg. Q-Loss (all states)	0.018	0.018	0.018	0.018	0.018	0.018	0.018

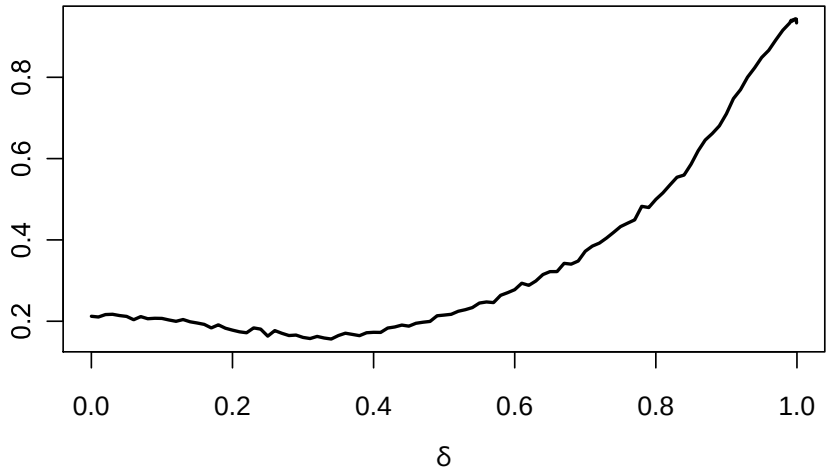
[21] For period-2 cycles, the fall in the profit gain is indeed small. As for equilibrium play, the decrease is more substantial but in part it may be due to the mechanical effect of doubling the number of states that are reached on path.

S.D. Q-Loss (all states)	0.006	0.007	0.006	0.006	0.006	0.006	0.006

4.1. *Competitive environments*

Before inquiring into how cooperation is sustained, we show that the algorithms learn to price competitively, at least approximately, when this is the only rational strategy. In particular, collusion is not feasible when $k = 0$ (the algorithms have no memory and thus cannot punish deviations), and it cannot be an equilibrium phenomenon when $\delta = 0$ (the algorithms are short-sighted and thus the immediate gain from defection cannot be outweighed by any loss due to future punishments).

Consider first what happens when the algorithms are short-sighted. Figure 3 shows how the average profit gain varies with δ. The theoretical postulate that lower discount factors impede collusion is largely confirmed by our simulations. The profit gain indeed decreases smoothly as the discount factor falls, and when $\delta = 0.35$ it has already dropped from over 80% to a modest 16%.[22] (To appreciate this value, remember that with our discretization of the price space, the average profit gain would already be close to 12% if the NashBertrand prices were just approximated by excess rather than by defect.)



[22] The fall in Δ actually starts well before δ gets so low that the monopoly outcome is no longer attainable in a subgame perfect equilibrium. With grim-trigger strategies, for instance, the critical threshold of δ is about 40% for our baseline specification.

Figure 3: The average profit gain Δ as a function of the discount factor δ in our representative experiment.

At this point, however, something perhaps surprising happens: the average profit gain turns back up as δ decreases further. Although the increase is small, it runs counter to theoretical expectations. We believe that this "paradox" arises because changing δ affects not only the relative value of future versus present profits, but also the effective rate of learning. This can be seen from equation (4), which implies that the relative weight of new and old information depends on both α and δ.[23] In particular, a decrease in δ tends to increase the effective speed of the updating, which as noted may impede learning when exploration is extensive.[24] At any rate, the profit gain remains small.

For the case of memoryless algorithms, we again find modest profit gains, only slightly higher than what is entailed by the discretization of the action space (details in section A5.1). All of this means that our algorithms learn to play, approximately, the one-shot equilibrium when this is the only equilibrium of the repeated game. If they do not price so competitively when other equilibria exist, it must be because they have learned other, more sophisticated strategies.

4.2. *Deviations and punishments*

Providing a complete description of these strategies is not straightforward. The problem is not that they must somehow be inferred from observed behavior, as is typically the case in experiments with humans. Here, at any stage of the simulations we know exactly not only what the algorithms do but also what they would do in any possible circumstances. The difficulty lies instead in the description of the strategies. For one thing, strategies are complicated objects (in our baseline experiment, they are mappings from a set of 225 elements to a set of 15 elements). For another, the limit strategies display considerable variation from session to session, and averaging masks relevant information.

---

[23] Loosely speaking, new information is the current reward π_t, and old information is whatever information is already included in the previous Q-matrix, $\mathbf{Q}_{t-1}$. The relative weight of new information in a steady state where $Q = \frac{\pi}{1-\delta}$ then is $\alpha(1 - \delta)$.

[24] A similar problem emerges when δ is very close to 1. In this case, we observe that the average profit gain eventually starts decreasing with δ. This reflects a failure of Q-learning for $\delta \approx 1$, which is well known in the computer science literature.

We therefore start by asking, specifically, whether unilateral price cuts are profitable or not in view of the rival's reaction. To this end, we focus once again on the algorithms' limit strategies. As discussed above, these generally entail supra-competitive prices. Starting, in period $\tau = 0$, from the prices the algorithms have converged to, we step in and exogenously force one algorithm to defect in period $\tau = 1$. The other algorithm instead continues to play according to his learned strategy. We then examine the behavior of the algorithms in the subsequent periods, when the forced cheater reverts to his learned strategy as well.

Figure 4 shows the average of the impulse-response functions derived from this exercise for all 1,000 sessions of our representative experiment.[25] It shows the prices chosen by the two agents after the deviation. In particular, Figure 4 depicts the evolution of prices following a one-period deviation to the static best-response to the rival's pre-deviation price.[26]

Clearly, the deviation gets punished. As Table III below shows, in more than 95% of the cases the punishment makes the deviation unprofitable; that is, "incentive compatibility" is verified.

The dynamic structure of the punishment is very interesting. After an initial price war, the algorithms gradually return to their pre-deviation behavior. This pattern looks very different from the one that would be implied, for instance, by grim-trigger strategies.[27]

These latter strategies, which are the workhorse of many theoretical analyses of collusion, are never observed in our experiments. The reason for this is simple: with experimentation, one algorithm would sooner or later defect, and when this happened both would be trapped

[25] When the algorithms converge to a price cycle, we consider deviations starting from every point of the cycle and take the average of all of them.

[26] We have also considered the case of an exogenous deviation that lasts for 5 periods. The dynamic pattern is similar to that of one-period deviations (details in section A5.2).

[27] Strictly speaking, grim-trigger strategies require unbounded memory, but it is easy to define their one-period memory counterpart.

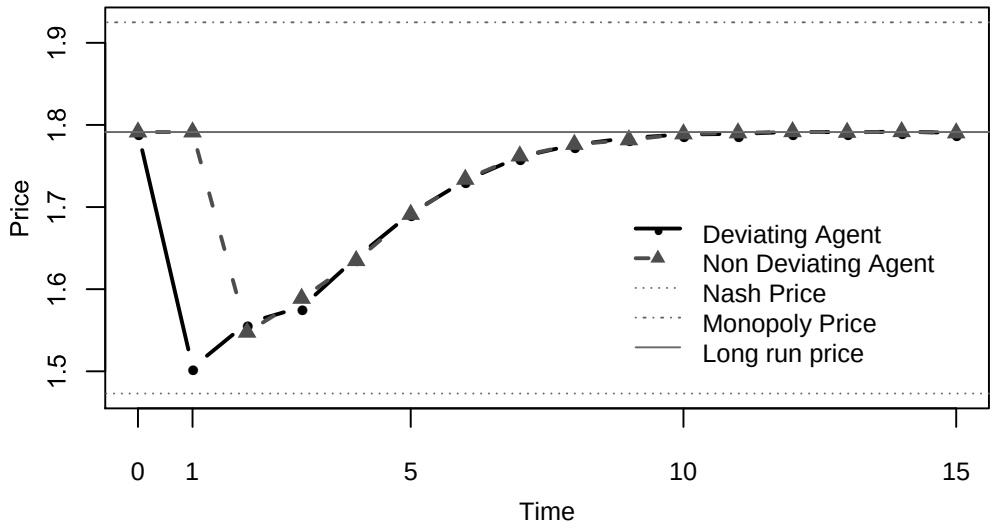


Figure 4: Prices charged by the two algorithms in period τ after an exogenous price cut by one of them in period $\tau = 1$. The forced cheater deviates to the static best response, and the deviation lasts for one period only. The figure plots the average prices across the 1,000 sessions. For sessions leading to a price cycle, we consider deviations starting from every point of the cycle and take the average of all of them. This counts as one observation in the calculation of the overall average.

in a protracted punishment phase that would last until further (joint) experimentation drove the firms out of the trap. Our algorithms, by contrast, consistently learn to re-start cooperation after a deviation. This seems necessary in an environment characterized by extensive experimentation, where coordination would inevitably be disrupted if it were not robust to idiosyncratic shocks.[28]

The pattern of punishment we observe is more similar to the "stick-and-carrot" strategies of Abreu (1984). However, there are differences with Abreu's strategies as well: the initial punishment is not as harsh as it could be (prices remain well above the static Bertrand-Nash equilibrium), and the return to the pre-deviation prices is gradual rather than abrupt.

To show that the pattern depicted in Figure 4 is not an artifact of the averaging, Figure 5 reports more information on the distribution of the impulse responses.[29] While there is

[28] This is not a foregone conclusion, however, as the algorithms may start to cooperate only after experimentation had already faded away. That cooperation begins earlier is confirmed by the analysis in Section 7.

[29] Here we restrict attention to sessions that converge to constant prices to avoid spurious effects that may arise because of the averaging across different initial conditions.

considerable variation across sessions, especially in the first periods after the deviation, the pattern is robust. (See also the fan chart in section A5.2.)

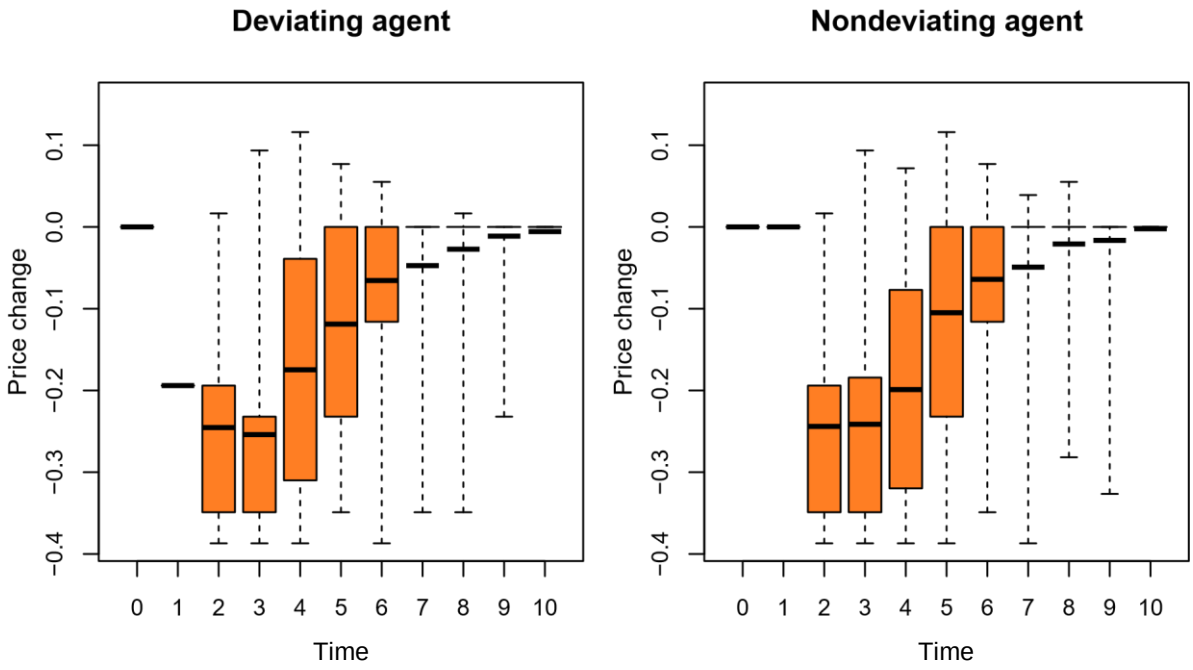


Figure 5: For each period τ, the figure shows the mean (black line), the 25th and 75th percentiles (shaded rectangles), and the ranges (dashed intervals) of the prices charged after an exogenous price cut in period $\tau = 1$. To be precise, the variable on the vertical axis is the difference between the current and the long-run price.

Figures 4 and 5 focus on deviations that maximize the short-run gain from defection. However, we have performed the same type of exercise for all possible price cuts. Table II reports the prices charged by the two algorithms immediately after the defection (i.e., in period $\tau = 2$). The initial punishment tends to be slightly harsher for bigger price reductions, but the effect is small and non-systematic. What is systematic is the return to the initial prices; in most of the cases, the punishment ends after 5-7 periods. (See table A2 in section A5.2.) Table III shows that these deviations, too, are almost always unprofitable.

For example, consider a pre-shock price of 1.78 (this is the 10-th price of the grid, and it accounts for almost 20% of the cases where both algorithms converged to the same price). Table II shows that, irrespective of the size of the rival's exogenous deviation in period τ

= 1, the non-deviating algorithm would cut the price in period $\tau = 2$ by approximately the same percentage amount (i.e. 13%, leading to a price of approximately 1.54). The deviating algorithm, in contrast, raises his price if the deviation was big and

TableII

Panela:Relativepricechangebythenondeviatingagentinperiod τ =2														
Pre-shock price	Freq.	Deviationprice 1.43	1.47	1.51	1.54	1.58	1.62	1.66	1.70	1.74	1.78	1.82	1.85	1.89	1.93	1.97
1.62	0.01	-0.04	-0.08	-0.07	-0.04	-0.08	0									
1.66	0.06	-0.08	-0.09	-0.09	-0.09	-0.08	-0.08	0								
1.70	0.11	-0.10	-0.09	-0.10	-0.10	-0.10	-0.10	-0.10	0							
1.74	0.16	-0.11	-0.11	-0.12	-0.11	-0.12	-0.11	-0.11	-0.11	0						
1.78	0.19	-0.13	-0.13	-0.13	-0.13	-0.13	-0.13	-0.13	-0.12	-0.13	0					
1.82	0.18	-0.15	-0.15	-0.14	-0.15	-0.14	-0.14	-0.14	-0.14	-0.14	-0.14	0				
1.85	0.11	-0.16	-0.16	-0.17	-0.17	-0.16	-0.16	-0.15	-0.15	-0.15	-0.16	-0.15	0			
1.89	0.09	-0.18	-0.18	-0.17	-0.18	-0.16	-0.17	-0.16	-0.16	-0.17	-0.16	-0.17	-0.16	0		
1.93	0.05	-0.19	-0.20	-0.19	-0.17	-0.19	-0.17	-0.18	-0.17	-0.18	-0.18	-0.18	-0.18	-0.16	0	
1.97	0.03	-0.19	-0.20	-0.21	-0.21	-0.21	-0.21	-0.18	-0.17	-0.17	-0.19	-0.18	-0.18	-0.17	-0.18	0

Panelb:Relativepricechangebythedeviatingagentinperiod τ withrespectto τ =1														
Pre-shock price	Freq.	Deviationprice 1.43	1.47	1.51	1.54	1.58	1.62	1.66	1.70	1.74	1.78	1.82	1.85	1.89	1.93	1.97
1.62	0.01	0.06	0.04	0.04	-0.01	-0.05	0									
1.66	0.06	0.07	0.06	0.01	-0.01	-0.02	-0.05	0								
1.70	0.11	0.08	0.06	0.04	0.01	-0.02	-0.03	-0.07	0							
1.74	0.16	0.09	0.07	0.03	0.02	-0.01	-0.04	-0.05	-0.08	0						
1.78	0.19	0.09	0.06	0.03	0	-0.02	-0.04	-0.05	-0.08	-0.11	0					
1.82	0.18	0.09	0.07	0.04	0.01	-0.03	-0.05	-0.07	-0.09	-0.12	-0.12	0				
1.85	0.11	0.09	0.08	0.03	0.01	-0.01	-0.02	-0.04	-0.07	-0.09	-0.11	-0.12	0			
1.89	0.09	0.10	0.08	0.03	0.01	-0.03	-0.04	-0.06	-0.08	-0.11	-0.12	-0.14	0			
1.93	0.05	0.10	0.07	0.05	0.01	0.01	-0.02	-0.04	-0.07	-0.09	-0.09	-0.11	-0.15	-0.16	0	
1.97	0.03	0.13	0.10	0.07	0.02	0	-0.03	-0.02	-0.04	-0.06	-0.10	-0.11	-0.12	-0.13	-0.18	0

TableIII

Panela:Averagepercentagegainfromthedeviationintermsofdiscountedprofits		
Pre-shock price	Freq.	Deviationprice 1.431.471.511.541.581.621.661.701.741.781.821.851.891.931.97
1.62	0.01	-0.03-0.02-0.02-0.01-0.020.00
1.66	0.06	-0.02-0.02-0.02-0.02-0.02-0.020.00
1.70	0.11	-0.03-0.03-0.03-0.03-0.03-0.03-0.030.00
1.74	0.16	-0.03-0.03-0.03-0.03-0.03-0.03-0.03-0.030.00
1.78	0.19	-0.04-0.03-0.03-0.03-0.03-0.03-0.03-0.03-0.030.00
1.82	0.18	-0.04-0.04-0.04-0.03-0.03-0.03-0.03-0.03-0.04-0.040.00
1.85	0.11	-0.04-0.04-0.04-0.04-0.03-0.03-0.03-0.04-0.03-0.04-0.040.00
1.89	0.09	-0.04-0.04-0.04-0.03-0.03-0.03-0.03-0.03-0.03-0.03-0.04-0.040.00
1.93	0.05	-0.04-0.04-0.04-0.04-0.03-0.03-0.03-0.03-0.03-0.03-0.03-0.04-0.040.00
1.97	0.03	-0.04-0.04-0.03-0.03-0.03-0.03-0.03-0.03-0.03-0.03-0.03-0.03-0.03-0.03-0.040.00
Panelb:Frequencyofunprofitabledeviations.		
		1.431.471.511.541.581.621.661.701.741.781.821.851.891.931.97
1.62	0.01	1.000.910.910.820.910.00
1.66	0.06	1.000.960.950.950.950.970.00
1.70	0.11	0.990.980.990.980.980.990.980.00
1.74	0.16	0.990.990.970.970.950.950.950.970.00
1.78	0.19	0.990.990.980.970.960.970.980.970.980.00
1.82	0.18	0.991.000.981.000.990.990.970.970.980.990.00
1.85	0.11	0.990.980.970.970.970.980.990.990.960.980.970.00
1.89	0.09	0.980.980.970.970.950.940.970.940.960.950.970.980.00
1.93	0.05	0.990.971.000.970.930.970.970.990.990.940.971.000.960
1.97	0.03	1.001.001.000.970.970.940.880.940.910.910.940.970.9410

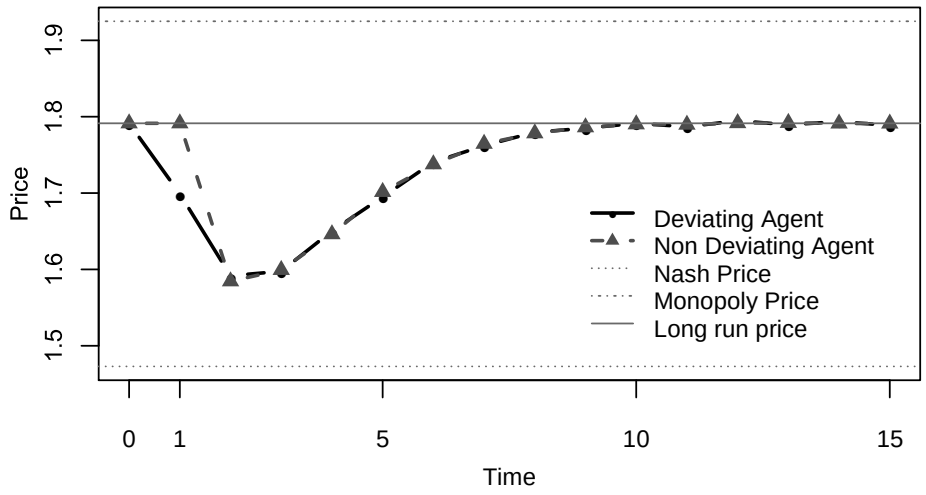


Figure 6: This figure is similar to Figure 4, except that the exogenous price cut is smaller. As a result, prices fall further down in period $\tau = 2$. In other words, the impulse-response function exhibits "overshooting."

further lowers the price if the deviation was small, pricing on average just above its rival.[30]Table III shows that even if the algorithms manage to re-start cooperation pretty soon, the deviation reduces the forced cheater's discounted profits by 3-4% on average. Only in a tiny fraction of the cases the deviation is profitable.

For small price cuts, the pattern just described represents a form of "overshooting:" that is, both algorithms cut their prices further in period $\tau = 2$, below the exogenous initial reduction of period $\tau = 1$. This is illustrated in Figure 6, which shows the average impulseresponse corresponding to one of these smaller deviations. The overshooting would be difficult to rationalize if what we had here was simply a stable dynamic system that mechanically returns to its rest point after being perturbed. But it makes perfect sense as part of a punishment.

As mentioned, these results do not depend on the specific values chosen for α and β: we observe punishment of deviations over the entire grid considered in the previous section. To illustrate, Figure 7 plots an index of the intensity of the punishment (i.e., the average percentage price cut of the non-deviating agent in period $\tau = 2$) as a function of α and β. The figure confirms that punishment is ubiquitous. The harshness of the punishment is

[30] It is tempting to say that the deviating algorithm is actively participating in its own punishment. At the very least, the deviating algorithm is anticipating the punishment – otherwise it would have no reason to reduce its price as soon as it regains control, i.e. in period $\tau = 2$, given that the rival's price was still high in period $\tau = 1$.

strongly correlated with the profit gain: the coefficient of correlation is 76.2%. This is one more sign that the supra-competitive prices are the result of genuine tacit collusion.

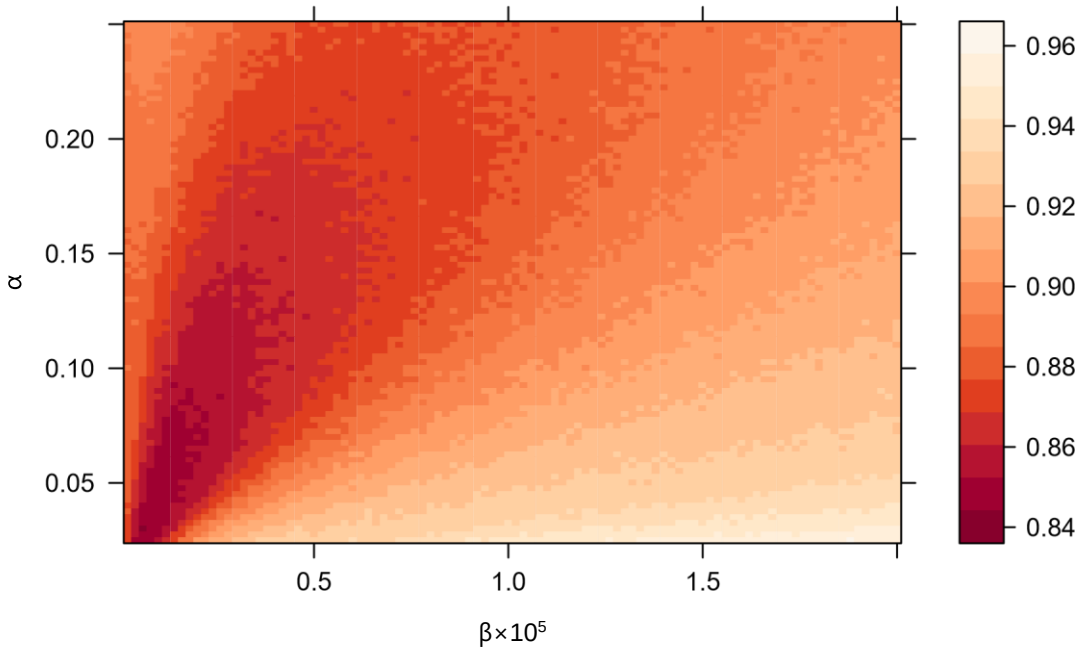


Figure 7: Average percentage price reduction by the non deviating agent in period $\tau = 2$, for a grid of values of α and β.

4.3. *The graph of strategies*

Let us now face the problem of describing the limit strategies more fully. Generally speaking, with a one-period memory strategies are mappings from the past prices $(p_{1,t-1},p_{2,t-1})$ to the current price $p_{i,t}$: $p_{i,t} = F_i(p_{1,t-1},p_{2,t-1})$. In our experiments, the algorithms systematically coordinate on one pair of prices (or a cycle) and punish any move away from the agreed upon prices. However, these prices vary from session to session, and the intensity of the punishment is variable as well, depending rather capriciously on the distance from the long-run prices. For this reason, one cannot derive a representative strategy by simply averaging across different functions F_i (details in appendix A5.3).[31]

One obvious way to work around this problem would be to average only across those sessions where the algorithms converge to the same pair of supra-competitive prices. In this case, the average function F must obviously exhibit a spike at that point. Elsewhere

[31] The average function would be almost flat, ranging over prices that are fairly competitive.

prices must be much lower, reflecting the punishment of deviations. But apart from these obvious properties, even such conditional averages display no recognizable pattern.

Evidently, there is considerable variation not only in the prices on which the algorithms converge to but also in their limit behavior off path. In other words, the exact way the algorithms achieve coordination depends on the specific history of their interactions. One

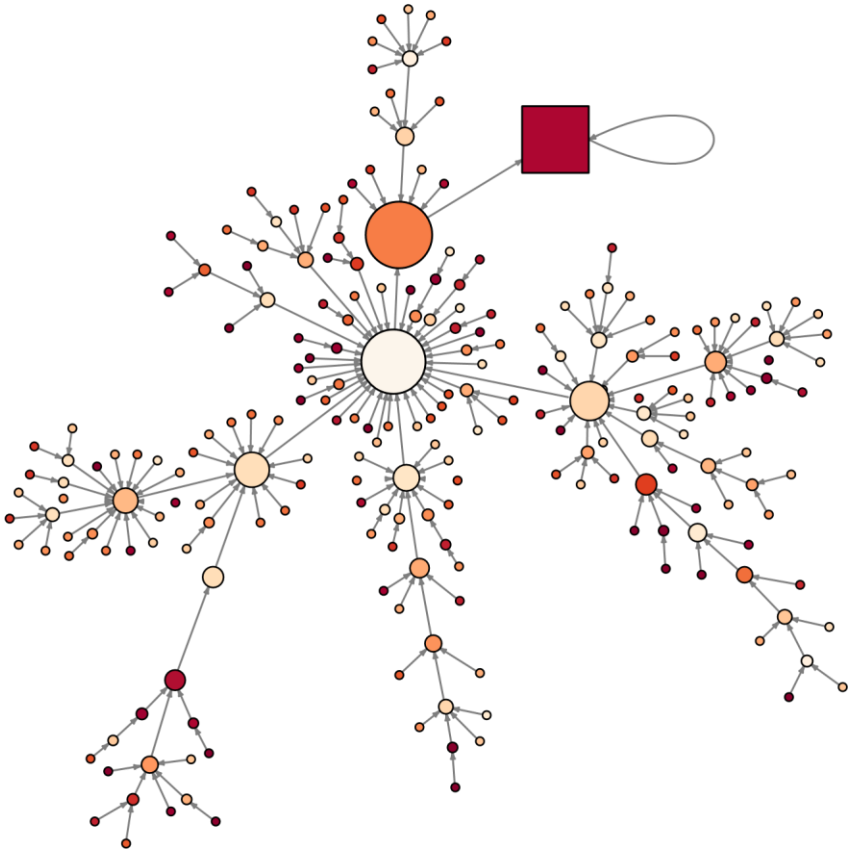


Figure 8: The directed graph of the limiting strategies in one session of the representative experiment. The absorbing node (corresponding to the long-run prices) is represented by the square, all other nodes by circles. The brightness of the nodes represents the profit gain (the darker the node, the higher the profit gain), while the size represents the node's centrality (as measured by betweenness centrality).

could not, perhaps, expect anything else from agents that learn purely by trial and error.

This suggests that limit strategies may be better studied in pairs, looking at the combined behavior of those algorithms that interacted with one another. This combined behavior may be described using the directed graph produced by any pair of strategies. For example, Figure 8 depicts the graph of the limit strategies obtained in one session of our

representative experiment. In any graph like this, the node corresponding to the long-run prices (which is marked as a square in the figure) is absorbing.[32]

The graph is quite complex but exhibits a few remarkable properties. First and foremost, all the nodes eventually lead to the absorbing node. This means that the algorithms

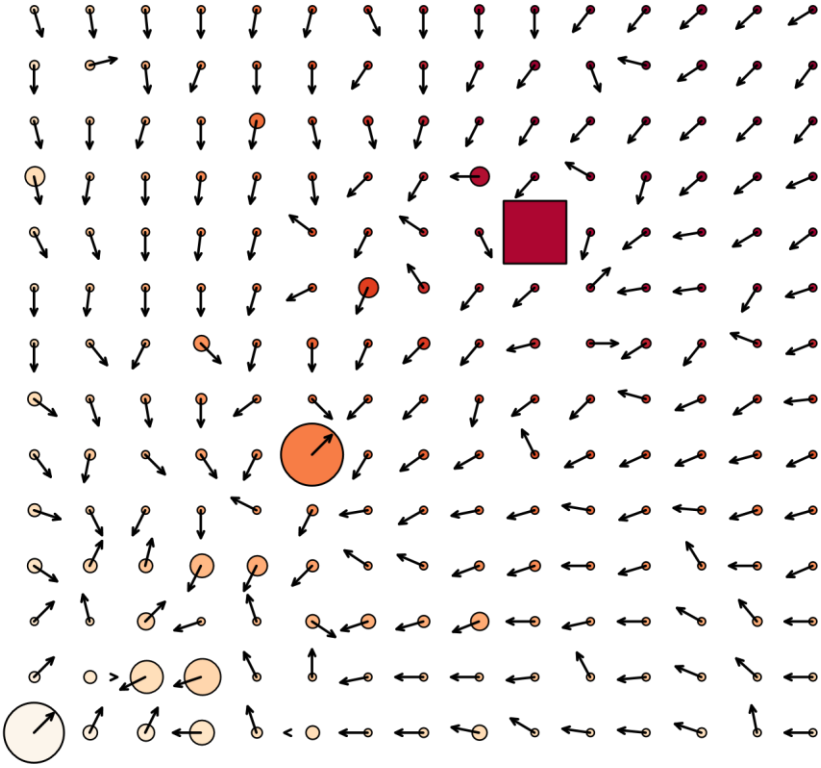


Figure 9: Phase portrait of the limiting strategies. The Bertrand-Nash price is best approximated by the third lowest price, the monopoly price by the third highest. Form, size and brightness of the nodes are as in Figure 8.

systematically re-start cooperation not only after unilateral but also after bilateral deviations. Second, there are a few key nodes that act as gateways, either directly or indirectly, to the absorbing node. Third, the paths to the absorbing node are generally rather short: the average length of the path is 6, and the maximum length is 18. The supplementary material file (section A5.3) shows that the properties exhibited by this example are in fact much more general. For example, in 92% of the sessions the system converges to the long-run prices starting from any possible node; and in 98% of the

[32] For sessions that converge to a price cycle, the system would cycle around two or more nodes.

sessions there are fewer than 3 nodes, out of 225, that do not eventually lead to the long-run prices.

Figure 9 represents, for the same example, the limit strategies in a way that facilitates the economic interpretation of the nodes. Nodes are ordered according to the level of the prices charged by algorithm 1 (horizontal axis) and 2 (vertical axis). The arrows starting from each node indicate the direction of the price change, but to make the figure easier to read they do not extend as far as the next node that is reached. The figure shows that starting from any node other than the absorbing one, the system initially moves towards the low part of the main diagonal and then climbs up to the long-run prices. This suggests that cooperation does not re-start immediately but only after a punishment phase, and that bilateral deviations are punished in pretty much the same way as unilateral deviations.

5. ROBUSTNESS

How robust are our baseline results to changes in the economic environment? In this section, we consider a number of factors that may affect firms' ability to sustain a tacit collusive agreement. Throughout, we continue to focus on our chosen values for the learning and experimentation parameters, $\alpha = 0.15$ and $\beta = 4 \times 10^{-6}$. The supplementary material file provides more details and presents several other robustness exercises.

5.1. *Number of players*

Theory predicts that collusion is harder to sustain when the market is more fragmented. We find that, indeed, the average profit gain Δ decreases from 85% to 64% in simulations with three firms. With four agents, the profit gain is still a substantial 56%. The decrease in the profit gain seems slower than in experiments with human subjects.[33]

[33] The early experimental literature indeed found that in the lab, tacit collusion is "frequently observed with two sellers, rarely in markets with three sellers, and almost never in markets with four or more sellers" (Potters and Suetens (2013) p. 17). More recently analyses paint a more nuanced picture, though. In some experiments, three or four human subjects manage to achieve levels of coordination comparable to our algorithms: see Horstmann (2018) and Friedman et al (2015).

These results are all the more remarkable because the enlargement of the state space interferes with learning. Indeed, moving from $n = 2$ to $n = 3$ or $n = 4$ enlarges the Q-matrix dramatically, from 3,375 to around 50,000 or over 750,000 entries. Since the parameter β is held constant, the increase in the size of the matrix makes the effective amount of exploration much lower. If we reduce β so as to compensate for the enlargement of the matrix, at least partially, the profit gain increases. For example, with three firms we find values of Δ close to 75%.[34]

The impulse-response functions remain qualitatively similar to the case of duopoly. We still have punishments, which however tend to be more prolonged and generally harsher than in the two-firms case.

TABLE IV

Cost asymmetry ($c_1 = 1$).

c_2	1.000	0.875	0.750	0.625	0.500	0.250
2's Nash market share	0.500	0.545	0.588	0.627	0.662	0.722
Δ	0.849	0.841	0.812	0.781	0.759	0.713
$\frac{\pi_1/\pi_1^N}{\pi_2/\pi_2^N}$	0.997	1.050	1.121	1.193	1.265	1.442

5.2. *Asymmetric firms*

The conventional wisdom has it that asymmetry impedes collusion. Firms contemplating a tacit collusive agreement must solve a two-fold problem of coordination: they must choose both the average price level, which determines the aggregate profit, and the relative prices, which determine how the total profit is split among the firms. Achieving coordination on both issues without explicit communication is often regarded as a daunting task.

To see how Q-learning algorithms cope with these problems, we considered both cost and demand asymmetries of different degrees. Table IV reports the results for the case of cost asymmetry (the case of demand asymmetry is similar).

[34] In order to make the learning process more effective, the increase in the amount of experimentation is matched by a decrease in the learning rate. The increase in the profit gain goes hand in hand with the increase in the frequency of equilibrium play.

As the table shows, asymmetry does reduce the average profit gain, but only to a limited extent. In part the decrease is simply a consequence of the absence of side payments. To see why this is so, consider how the two algorithms divide the aggregate profit. As the last row of the table shows, the gain from collusion is split disproportionately in favor of the less efficient firm.

This division clearly has an impact on the joint profit level. The maximization of joint profit indeed requires that the more efficient firm expand and the less efficient one contract relative to the Bertrand-Nash equilibrium.[35] However, this would produce a division strongly biased in favor of the more efficient firm. Conversely, a proportional division of the gain, or one that favors the less efficient firm, entails a cost in terms of the total profit.

This by itself explains why the average profit gain decreases as the degree of asymmetry increases. In other words, it seems that asymmetry doesn't actually make the coordination problem tougher for the algorithms but simply leads them to coordinate on a solution that does not maximize total profit.

5.3. *Stochastic demand*

While the baseline model is deterministic, in principle each of the model parameters could be subject to random shocks. In particular, here we investigate the case where the level of demand (a_0) is stochastic, and the case of stochastic entry and exit.

Consider first the case where the aggregate demand parameter a_0 varies stochastically. Specifically, a_0, which in the benchmark is nil, is now assumed to be an i.i.d. random variable that may take on three values, i.e. $a_0^L = -a_0^H$, 0 and a_0^H, with the same probability, thus generating both negative and positive demand shocks. The algorithms do not observe the value of a_0 before making their choices. The shocks are purely idiosyncratic and have no persistency – a challenging situation for the algorithms.

When $a_0^H = 0.15$, the average profit gain under uncertainty decreases slightly, from 85% to 80%; and even when $a_0^H = 0.25$ the average profit gain is still 70%. Apparently, then, demand variability does hinder collusion among firms, as one would have expected, but it does not eliminate it.

[35] This effect may be so pronounced that the less efficient firm may actually earn less under joint profit maximization than in the Bertrand-Nash equilibrium.

5.4. *Variable market structure*

Next, we analyze the impact of a variable market structure. In particular, we repeat the simulations with one firm (the "outsider") entering and exiting the market in random fashion. This exercise is performed both for the case of two players (the market thus alternating between monopoly and duopoly) and of three players (duopoly and triopoly).

We take entry and exit to be serially correlated. Formally, let I_t be an indicator function equal to 1 if the outsider is in the market in period t and to 0 otherwise. We set

$$\text{prob}\{I_t = 1 | I_{t-1} = 0\} = \text{prob}\{I_t = 0 | I_{t-1} = 1\} = \rho. \tag{10}$$

This implies that the unconditional probability of the outsider's being in at some random time is 50%. Equivalently, the market is a duopoly half the time on average. The probability of entry and exit ρ is set at 0.1% or at 0.01%, so that when the outsider enters, it stays in the market for an average of 1,000 (resp., 10,000) periods. Since in marketplaces where algorithmic pricing is commonly adopted periods can be very short, these levels of persistency are actually rather low.

The state s now includes the prices of the previous period if all firms were active, or the prices of the active firms and the fact that the outsider was not active.

This turns out to be the extension where collusion is most seriously hindered. The average profit gain decreases to less than 60%, equilibrium play is observed on path in less than 10% of the sessions, and punishments are rather mild, making deviations profitable in a sizeable fraction of cases. All of this is the combined effect of the increase in the size of the matrix, which as noted impedes learning, and uncertainty. Still, we remain far from the competitive benchmark.

5.5. *Product substitutability*

In the logit model, a decrease in μ means that the demand for each particular variety becomes more price-sensitive. That is, the reduction in μ captures an increase in product substitutability. In principle, the impact of changes in substitutability on the likelihood of collusion is ambiguous: on the one hand, when products are more substitutable the gain from deviation increases, but at the same time punishment can be harsher. This ambiguity is confirmed by the theoretical literature (see e.g. Tyagi, 1999).

In our setting, we test the consequences of changing parameter μ from 0.25 (baseline) up to 0.5 and down to 0, where products are perfect substitutes. The average profit gain decreases slightly when μ decreases, but when the products are perfect substitutes (μ = 0) it is still greater than 77%.

5.6. *Initialization*

Our baseline choice was to initialize the Q-matrix in accordance with the fact that the algorithms start by randomizing uniformly across all possible actions. As a robustness check, we also study other initializations, such as setting $\mathbf{Q}_0$ to the value corresponding to the rival always playing the Nash-Bertrand price,[36] or a grim-trigger strategy, or else setting $\mathbf{Q}_0$ at constant, large values. In this last case, the value of any cell that is visited inevitably decreases at first, so different actions are tried next. Thus, the updating of the matrix in itself induces the algorithms to explore systematically, in addition to the random experimentation entailed by the ε-greedy model. That is, one could set ε = 0 and still have "experimentation" and learning.

The average profit gain is somewhat sensitive to the initialization but always remains well above 70%. The average profit gain is lowest when the Q-matrix is initialized at Nash, or at grim-trigger strategies. When instead the matrix is initialized at a large, constant value, and exploration is shut down, the algorithms learn to collude almost perfectly.

5.7. *Action set*

We have explored the consequences of enlarging the price grid by increasing ξ, enlarging the grid only downwards so that the lowest feasible price is just below the marginal cost, and making the grid finer (raising the number of feasible prices m from 15 to 50 or 100).

The greater flexibility in price setting - below Bertrand or above monopoly - turns out to have a limited impact. This is not surprising, given that the players never converge on these very low or very high prices. Enlarging the grid only in the downwards direction decreases the profit gain, confirming that the way in which coordination is achieved is history dependent. However, the profit gain remains above 60%.

The increase in the number of actions, in principle, could engender misunderstandings in the absence of explicit communication and thus could prevent cooperation. Indeed, the

[36] In fact, this may produce two different initializations depending on whether the Bertrand price, which is not available on our price grid, is approximated by excess or by defect. We have chosen the closest approximation.

average profit gain decreases with m, but with m = 100 it is still a substantial 70%. In interpreting this result, one should also keep in mind that with m = 100 the Q-matrix is much larger than in the baseline model, but β is held constant. To achieve the same level of learning, instead, more experimentation would be required.

The supplementary material file reports the results of more robustness checks, including the case of longer memory, linear demand, Boltzmann experimentation, and asymmetric algorithms.

6. TIME SCALE

So far we have focused on limit outcomes and strategies; that is, on what the algorithms do once they have attained stable behavior. But convergence requires a very large number of periods, on the order of hundreds of thousands. Even if a "period" lasted just a few minutes, this would correspond to several years or more. In this section, we discuss the extent to which this limits the practical implications of our results.

6.1. *Transition*

To begin with, note that the algorithms start to collude long before convergence is achieved. This is illustrated in Figure 10, which shows the evolution of the average profit gain in our representative experiment. The profit gain starts from a fairly large value, but this is simply because the algorithms initially randomize uniformly across prices that, on average, exceed the competitive level. This effect disappears as experimentation draws towards a close. One can abstract from this effect by taking as a competitive benchmark not $\Delta = 0$ but the profit gain that would result if the algorithms set the Bertrand price

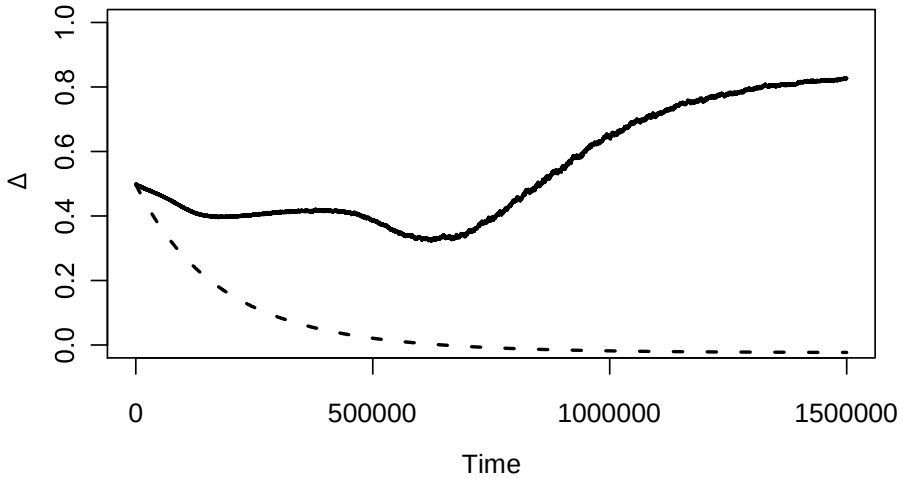


Figure 10: The average profit gain as a function of the number of repetitions (moving average over the last 100 repetitions). The dashed line is the profit gain that results from exogenous exploration, on the assumption that when they do not explore, the algorithms set the BartrandNash price (approximated by defect).

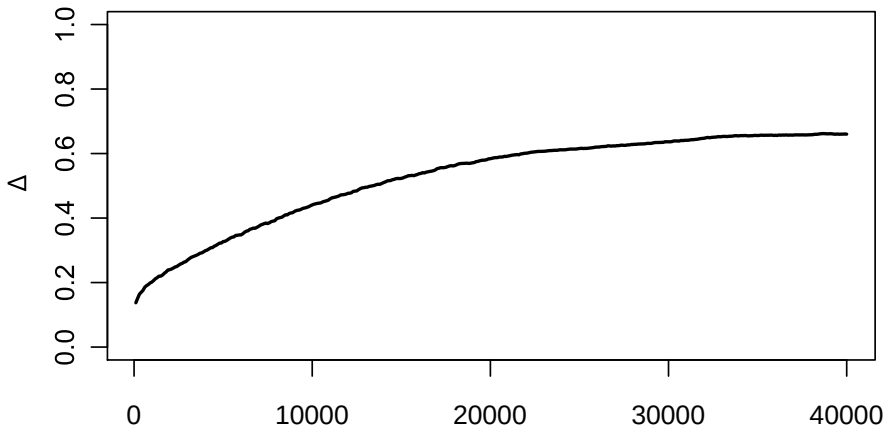
whenever they do not explore. This is represented by the smoothly declining curve in Figure 10.

Even against this benchmark, our algorithms begin to increase their profits very soon. The gain is modest initially but gradually increases. Thus, a non-negligible degree of collusion may emerge well before the algorithms have completed their learning.

6.2. *Off-line training*

Typically, algorithms are trained in artificial environments before being put to work in the real world. For example, AlphaGo was trained for several weeks in self-play mode before facing professional human players.[37] Likewise, firms presumably train their pricing algorithms off-line before deploying them in real marketplaces. If much of the learning process can be completed off-line, the algorithms might start to collude the moment they engage in real action.

However, there is an important difference between zero-sum board games and games of pricing. For the former, almost everything that has been learned off-line can be directly applied in real contests (the only problem being that human opponents may adopt a



[37] By way of comparison, the "training" of our algorithms takes just a few seconds of CPU time in any session.

Time

Figure 11: The average profit gain as a function of the number of repetitions for pairs of algorithms re-matched as described in the text (moving average over the last 100 repetitions).

different style of play). But games of pricing involve coordination in an essential way, and different sets of players may learn to coordinate in different ways. Moreover, the training environment may not exactly reflect the reality of the markets in which the algorithms will be deployed. This implies that what an algorithm has learned off-line may be of little help in colluding in real life.

To see how far the knowledge gained in playing against one opponent can be transferred to interacting with another, we re-match the algorithms once they have converged and let them start to play again. In the newly formed pairs, we shut exploration down by setting $\varepsilon = 0$. Nevertheless, faced with the "unexpected" choices made by the new competitor, the algorithms change their strategies. In an initial phase, they keep trying actions that performed well in the past but are no longer good in the new environment. After this learning phase, however, they once again stabilize their behavior.

Figure 11 shows the evolution of the average profit gain for such re-matched pairs. At first the average profit gain falls from 85% to about 20%, confirming that coordination is almost completely pair-specific. As the algorithms adapt to the new environment, however, the profit gain rises quite rapidly. Learning ends in less than one tenth of the time it took in the original interactions, even though the eventual profit gain is somewhat lower. (The original levels of collusion can be re-produced by re-activating exploration.) This suggests that even in games of pricing, off-line learning may not be completely useless after all.

6.3. *Financial markets*

In financial markets, both price adjustments and transactions occur much more frequently than in goods markets. In other words, a "period" is much shorter. As a result, millions of interactions could easily take place in days, or even just hours.

Naturally, however, our analysis cannot be applied to financial markets as it stands. Demand and supply need to be modelled in a different way, and market power is typically

more limited in financial than in goods markets. On the other hand, even a modest price effect could result in large extra-profits and thus become a matter of antitrust concern.

6.4. *More advanced algorithms*

As noted, Q-learning algorithms learn slowly by design, as they update only one cell of the Q-matrix at a time. This is clearly inefficient when the matrix is in fact the discrete approximation of a smooth function, as in our model, because it totally neglects the topological structure of the function.

There exist more efficient algorithms, capable of taking advantage of that structure. For example, value-function-approximation algorithms estimate the Q-function by iterative updating methods similar to (4) and then derive the Q-matrix by discrete approximation. In this case, at each period the algorithm would update not only the most recently visited cell of the matrix but also a number of neighboring cells, thus possibly speeding up the learning process. The downside of these faster algorithms is that they require modeling choices that are somewhat arbitrary from an economic viewpoint, in this respect resembling black boxes.[38] This is, in our opinion, a good reason to start the analysis of algorithmic collusion from Q-learning, as we have done here. But extending the analysis to algorithms that learn more quickly is clearly an important objective for future research. In particular, it is crucial to address the issue of the time scale of collusion.

7. CONCLUSIONS

We have shown that Q-learning pricing algorithms systematically learn to collude. Collusion is typically partial and is enforced by punishment in case of deviation. The punishment is of finite duration, with a gradual return to pre-deviation prices. The algorithms learn to play these strategies by trial and error, requiring no prior knowledge of the operating environment. They leave no trace whatever of concerted action: they do not communicate with one another, nor have they been designed or instructed to collude.

From the standpoint of competition policy, these findings should probably ring an alarm bell. Today, the prevalent approach to tacit collusion is relatively lenient, in part because tacit collusion among human decision-makers is regarded as extremely difficult to

[38] To begin with, one must specify a functional form for the Q-function. Further, these methods are often implemented by means of neural networks organized on several layers (*deep learning*). In a model of deep learning one must also specify the number of estimation layers and the structure of the neural network in each layer. The arbitrariness of these modeling choices may make it hard to interpret the results.

achieve.[39] While we have no direct comparative evidence for algorithms relative to humans, our results suggest that algorithmic collusion might not be that improbable. If this is so, then the advent of algorithmic pricing could well heighten the risk that tolerant antitrust policy will produce too many false negatives.

On the other hand, algorithmic pricing may open the way to new forms of antitrust intervention. When they suspect collusive conduct, agencies and the courts can subpoena and test pricing algorithms in environments that closely replicate the particular industry under investigation. With humans this was not possible, so the risk of aggressive antitrust enforcement producing too many false positives may be reduced. Therefore, the advent of AI pricing could alter the balance between the two types of error, possibly calling for policy adjustment.

More research is needed, however, to confirm the robustness and external validity of our findings. Several issues stand out. First, the realism of the economic environment: we have considered a good many extensions of the baseline model, but all separately, so the model remains quite highly stylized. In particular, we have not yet considered persistent, firm-specific demand or cost shocks. In the presence of such shocks, it is not clear how a rival firm ought to respond to a price cut. In principle, this depends on whether the price cut is driven by exogenous shocks or represents a deviation from the implicit agreement. But when a firm's shocks are part of its state but not of the rival's one, the rival faces a non trivial inference problem. The difficulty of "interpreting" price cuts might then pose a challenge to the sustainability of collusion.

Another important issue is the diversity of the competing algorithms. There are many different forms of reinforcement learning, and Q-learning algorithms themselves come in different varieties. Since tacit collusion is, essentially, a problem of coordination, one may wonder that the problem is easier when the programs belong to the same class. It would seem therefore necessary to extend the analysis to the case of player heterogeneity.

A third issue is the speed of learning. As discussed above, further inquiry into this problem must use algorithms that learn faster. It would also be interesting to move away from algorithms that adopt a purely model-free approach to learning, considering algorithms that incorporate some economic structure.

On a more general note, we need a better understanding of the dynamics of the learning process. This is important not only conceptually but also practically, as it could help identify factors that may destabilize collusion.

[39] Another reason is the difficulty of devising proper remedies (Harrington (2018)).

All of these challenging but important tasks are left for future research.

REFERENCES

Arthur W B. (1991), Designing Economic Agents that Act like Human Agents: A Behavioral Approach to Bounded Rationality, *The American economic review*, 81(2), 353-359.

Barfuss W., Donges J F. and Kurths J. (2019), Deterministic limit of temporal difference reinforcement learning for stochastic games, *Physical Review E*, 99(4), 043305.

Barlo M., Carmona G. and Sabourian H. (2009), Repeated games with one-memory, *Journal of economic theory*, 144, 312-336.

Barlo M., Carmona G. and Sabourian H. (2016), Bounded memory Folk theorem, *Journal of economic theory*, 163, 728-774.

Beggs A W. (2005), On the convergence of reinforcement learning, *Journal of economic theory*, 122(1), 1-36.

Benveniste A., Metivier M. and Priouret P. (1990), *Adaptive Algorithms and Stochastic Approximations*, Springer.

Byrne, D. P., and De Roos, N. (2019). Learning to coordinate: A study in retail gasoline. *American Economic Review*, 109(2), 591-619.

Bloembergen D., Tuyls K., Hennes D. and Kaisers M. (2015), Evolutionary Dynamics of Multi-Agent Learning: A Survey, *Journal of Artificial Intelligence Research*, 53, 659-697.

Borgers T. and Sarin R. (1997), Learning Through Reinforcement and Replicator Dynamics, *Journal of economic theory*, 77(1), 1-14.

Chen L., Mislove A. and Wilson C. (2016), An Empirical Analysis of Algorithmic Pricing on Amazon Marketplace, in *Proceedings of the 25th International Conference on World Wide Web*, WWW'16, 1339-1349, International World Wide Web Conferences Steering Committee.

Cooper, W.L., Homem-de-Mello, T. and Kleywegt, A.J., 2015. Learning and pricing with models that do not explicitly incorporate competition. *Operations research*, 63(1), pp.86-103.

Cross J G. 1973, A Stochastic Learning Model of Economic Behavior, *The Quarterly Journal of Economics*, 87(2), 239-266.

Erev I. and Roth A E. (1998), Predicting How People Play Games: Reinforcement Learning in Experimental Games with Unique, Mixed Strategy Equilibria, *The American economic review*, 88(4), 848-881.

Ezrachi A. and Stucke M E. (2016), Virtual Competition, *Journal of European Competition Law & Practice*, 7(9), 585-586.

Ezrachi A. and Stucke M E. (2017), Artificial intelligence & collusion: When computers inhibit competition, *University of Illinois law review*, 1775.

Friedman D., Huck S., Oprea R., and Weidenholzer S., From Imitation to Collusion: Long-run Learning in a Low-Information Environment, *Journal of Economic Theory*, 155 (2015), 185-205.

Harrington J E. (2018), Developing Competition Law for Collusion by Autonomous Artificial Agents, *Journal of Competition Law & Economics*, 14(3), 331-363.

Ho T H., Camerer C F. and Chong, J-K., Self-tuning experience weighted attraction learning in games, *Journal of economic theory*, 133(1), 177-198.

Hopkins E. (2002), Two competing models of how people learn in games, *Econometrica*, 70(6), 2141-2166.

Horstmann, N., Krmer, J. and Schnurr, D., 2018. Number effects and tacit collusion in experimental oligopolies. *The Journal of Industrial Economics*, 66(3), pp.650-700.

Kimbrough, S. O., and Murphy, F. H. (2009). Learning to collude tacitly on production levels by oligopolistic agents. Computational Economics, 33(1), 47.

Klein T. (2018), Assessing Autonomous Algorithmic Collusion: Q-Learning Under Short-Run Price Commitments, doi 10.2139/ssrn.3195812, mimeo.

Ku¨hn K-U. and Tadelis S. (2018), The Economics of Algorithmic Pricing: Is collusion really inevitable?, mimeo.

Leufkens K. and Peeters R. (2011), Price dynamics and collusion under short-run price commitments, *International Journal of Industrial Organization*, 29(1), 134-153.

Maskin E. and Tirole J. (1998), A Theory of Dynamic Oligopoly, II: Price Competition, Kinked Demand Curves, and Edgeworth Cycles, *Econometrica*, 56(3), 571-599.

Mnih V., Kavukcuoglu K., Silver D., Rusu A A., Veness J., Bellemare M., Graves A., Riedmiller M., Fidjeland A K., Ostrovski G., Petersen S., Beattie C., Sadik A., Antonoglou I., King H., Kumaran D., Wierstra D., Legg S. and Hassabis D. (2015), Human-level control through deep reinforcement learning, *Nature*, 518(7540), 529-533.

Nowe, A., Vrancx, P., and Hauwere, Y.-M. D. (2012). Game theory and multi-agent reinforcement learning. In Wiering, M. and van Otterlo, M., editors, Reinforcement Learning: State-of-the-Art, pages 441467. Springer-Verlag, Berlin.

Potters, J. and Suetens, S., 2013. Oligopoly experiments in the current millennium. *Journal of Economic Surveys*, 27(3), pp.439-460.

Roth A E. and Erev I. (1995), Learning in extensive-form games: Experimental data and simple dynamic models in the intermediate term, *Games and economic behavior*, 8(1), 164-212.

Salcedo B. (2015), Pricing Algorithms and Tacit Collusion, mimeo.

Schwalbe, U. (2019). Algorithms, machine learning, and collusion. Journal of Competition Law & Economics, 14(4), 568-607.

Silver D., Huang A., Maddison C., Guez A., Sifre L., van den Driessche G., Schrittwieser J., Antonoglou I., Panneershelvam V., Lanctot M., Dieleman S., Grewe D., Nham J., Kalchbrenner N., Sutskever I., Lillicrap T., Leach M., Kavukcuoglu K., Graepel T. and Hassabis D. (2016). Mastering the game of Go with deep neural networks and tree search, *Nature*, 529(7587), 484-489.

Silver D., Hubert T., Schrittwieser J., Antonoglou I., Lai M., Guez A., Lanctot M., Sifre L., Kumaran D., Graepel T., Lillicrap T., Simonyan K. and Hassabis, Demis (2018), A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play, *Science*, 362(6419), 1140-1144.

Siallagan, M., Deguchi, H., and Ichikawa, M. (2013). Aspiration-Based Learning in a Cournot Duopoly Model. Evolutionary and Institutional Economics Review, 10(2), 295-314.

Sutton R . and Barto A G. (2018), *Reinforcement learning: An introduction*, MIT Press.

Tyagi R K. (1999), On the relationship between product substitutability and tacit collusion, *Managerial and Decision Economics*, 20(6), 293-298.

Waltman L. and Kaymak U. (2008), Q-learning agents in a Cournot oligopoly model, *Journal of economic dynamics & control*, 32(10), 3275-3293.

Watkins C J. (1989), *Learning from delayed rewards*, Ph.D. Thesis, King's College, Cambridge UK.

Watkins C J. and Dayan P. (1992), Q-learning, *Machine learning*, 8(3), 279-292. plain