ARTIFICIAL INTELLIGENCE, ALGORITHMIC PRICING AND COLLUSION[†]

EMILIO CALVANO[*‡], GIACOMO CALZOLARI[&‡§],
VINCENZO DENICOLÒ[*§], AND SERGIO PASTORELLO[*]

APRIL 2019

Increasingly, pricing algorithms are supplanting human decision makers in on-line markets. 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 classical 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.

Click here for the latest version

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

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

[†]We thank without implicating Susan Athey, Areil Ezrachi, Joshua Gans, Joe Harrington, Kai-Uwe Kuhn, Patrick Legros, David Levine, Yossi Spiegel, Steve Tadelis, Emanuele Tarantino and participants at the 2018 NBER Economics of Artificial Intelligence Conference, Cambridge University, Universidad Carlos III, U. Napoli Federico II, Universidad Autonoma de Barcelona, Berlin IO workshop 2019, Paris School of Economics, EAGP workshop at the European Commission (DG Competition), the 2018 Toulouse biannual postal conference, the Mannheim MaCCi workshop on Big Data, 12th Toulouse Digital Conference, 11th Paris Digital Conference, 2nd Bergamo IO workshop, EIEF (Rome) for useful comments. Financial support from the Digital Chair initiative at the Toulouse School of Economics is gratefully acknowledged.

Corresponding author: Giacomo Calzolari, giacomo.calzolari@eui.eu

[*]Bologna University; [‡]Toulouse School of Economics; [&]European University Institute; [§]CEPR

1. INTRODUCTION

Firms are increasingly adopting software algorithms to price their goods and services. In a sample of over 1,600 best-selling items listed on Amazon, Chen et al. (2016) find that in 2015 more than a third of the vendors had already automated their pricing. A repricing-software industry has since arisen, supplying turnkey pricing systems to smaller vendors and customizing software for the larger ones.[1] As this industry is developing steadily, algorithmic pricing is likely to become even more prevalent in the future.

In fact, algorithmic pricing is not new,[2] but the software has recently evolved from rule-based to reinforcement learning programs. These latter programs are much more "autonomous" than their precursors. Powered by Artificial Intelligence (AI), they develop their pricing strategies from scratch, engaging in active experimentation and adapting to the evolving environment. In this learning process, they require little or no external guidance.

The diffusion and evolution of pricing algorithms raise various issues for competition policy (Harrington, 2017). In particular, legal scholars and policy-makers alike have voiced concern that reinforcement learning algorithms may learn to collude tacitly, i.e., without communicating with one another and without having been specifically instructed to cooperate instead of competing.[3] While so far no one has brought an antitrust case against autonomously colluding algorithms,[4] antitrust agencies are discussing the problem seriously.[5]

[1] https://www.techemergence.com/ai-for-pricing-comparing-5-current-applications/ last accessed October 4, 2018. Algorithmic pricing is now so widely diffused that Amazon advertises its MarketPlace WebService API stressing that the API facilitates pricing automation. See https://developer.amazonservices.com/, last accessed October 4, 2018.

[2] Airline companies, for instance, have been using revenue management software for decades now. But this software would set prices mechanically following the instructions of the programmer, who thus remains effectively in charge of the strategic choices. Any price-fixing scheme involving these algorithms could therefore be directly traced back to human beings, raising no really new issues for antitrust policy.

[3] See, for instance, Ezrachi and Stucke (2016, 2017), the remarks of M. Vestager, European Commissioner, at the Bundeskarellamt 18th Conference on Competition, Berlin, March 16, 2017 ("Algorithms and Competition"), and the speech of M. Ohlhausen, Acting Chairman of the FTC , at the Concurrences Antitrust in the Financial Sector Conference, New York, May 23, 2017 ("Should We Fear the Things That Go Beep in the Night? Some Initial Thoughts on the Intersection of Antitrust Law and Algorithmic Pricing").

[4] A few cases that involve rule-based pricing software are discussed in Gata (2019).

[5] Recently, for instance, the possibility of algorithmic collusion was extensively discussed at the 7th session of the FTC Hearings on competition and consumer protection (November 13-14, 2018). In fact, agencies have been actively engaged on the issue for a couple of years now. The OECD sponsored a Roundtable on *Algorithms and Collusion* already in June 2017; in September 2017, the Canadian Competition Bureau released a white paper discussing the ability of algorithms to collude (*Big data and Innovation: Implications for Competition Policy in Canada*); and in October 2018 the British CMA published a white paper on *Pricing Algorithms*.

But how real is the risk of tacit collusion among AI pricing 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,[6] and firms typically do not disclose details of the pricing software they use. On the theoretical side, the analysis of reinforcement-learning agents in strategic settings has been carried out only for algorithms so naive that, by construction, they cannot learn to collude. For AI agents that in principle can, an analytical approach seems currently out of reach, as we shall discuss at greater length below.[7]

To make some progress, this paper therefore 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, and more work is certainly required to confirm the external validity of our research. However, our findings do suggest that algorithmic collusion is more than a remote theoretical possibility.

The results indicate that, indeed, even relatively simple pricing algorithms systematically learn to play collusive strategies. The strategies 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.

To our knowledge, this is the first paper that clearly documents the emergence of collusive strategies among autonomous pricing algorithms. The previous literature in both computer science and economics has focused on outcomes rather than strategies. But the observation of supra-competitive prices is not, per se, genuine proof of collusion, as high prices could be due to the algorithms' failure to learn the competitive equilibrium.[8] If

[6]See Andreoli-Versbach and Franck (2015) for a recent discussion of the issue. To date, the most promising approach appears to be the estimation of structural models. For example, Decarolis and Rovigatti (2018) study on-line ad auctions in this way and find a great deal of anticompetitive bidding. They conjecture that this may be due to algorithmic bidding, especially when several bidders delegate the task to a common AI agent.

[7]One theoretical contribution is Salcedo (2015), who argues that optimized algorithms will inevitably reach a collusive outcome. But this claim crucially hinges on the (strong) assumption that each algorithm can periodically observe and "decode" the other algorithms, which in the meantime have not changed. The practical relevance of Salcedo's result remains therefore controversial.

[8]For example, Waltman and Kaymak (2008) study repeated Cournot competition among Q-algorithms both where the algorithms cannot condition current choices on past actions (where tacit collusion is by definition impossible) and where they can (where collusion becomes a possibility). They find that the

this were so, there would be little ground for concern, in that the problem would presumably fade away as AI developed further. If instead pricing algorithms even now can learn to collude, then with the spread of "smarter" programs the problem may actually grow worse.

Our analysis not only shows that pricing algorithms do learn to collude but further suggests they may be better than humans at colluding tacitly. The experimental literature has consistently found that human subjects are hardly able to coordinate without explicit communication. In a laboratory setting with no communication possible, two agents sometimes manage to converge to slightly supra-competitive prices, but three agents typically set prices close to the static Nash equilibrium, and four or more actually tend to be more aggressive than Nash – a "rivalry" effect that is often attributed to experimental subjects' tendency to behave as if they were involved in a contest.[9] Our algorithms, by contrast, display a stubborn propensity to collude. Even though, in keeping with theory, the degree of collusion decreases as the number of competitors rises, substantial collusion continues to prevail when the active firms are three or four in number, when they are asymmetric, and when they operate in stochastic environments.

These results raise the issue of whether current policy on tacit collusion is still adequate in the age of AI. In most countries (including the U.S. and the European Union), tacit collusion is not now regarded as illegal.[10] The rationale is twofold. First, tacit collusion is held to be a chimera: illusory and practically impossible to achieve. And second, the position is that even if tacit collusion nevertheless occurred, it would be hard to detect. These assumptions imply that aggressive antitrust enforcement risks producing many false positives, while tolerant policy would result in relatively few cases going undetected. Our findings, however, suggest that the advent of AI pricing could well alter the balance between the two types of errors. More research is certainly needed before policy conclusions can be drawn, but there is good reason to hold that the issue deserves closer scrutiny.

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 reviews the related economics and computer science literature. Section 4 then gives a detailed description of the economic environments where the simulations are performed. Section 5 reports the main results, and Section 6 reports on a number of robustness checks.

---

extent to which firms manage to reduce output with respect to the Cournot-Nash static equilibrium "seems to be somewhat lower for firms with a memory [...] than for firms without a memory." (p. 3283) This suggests, however, that what they observe is not collusion but a failure to learn Nash.

[9]See for instance Huck et al. (2004) and the literature cited therein.

[10]In fact, the treatment of tacit collusion has long been a controversial issues in competition policy, and although the prevalent approach is tolerant, some jurisdictions take a more aggressive stance.

Section 7 concludes with a discussion of some major open issues.

2. Q-LEARNING

All reinforcement learning algorithms share a common structure, which naturally appeals to economists. That is, they adapt their behavior to past experience, taking actions that have proven successful more frequently and unsuccessful ones less frequently. In this way, the algorithms may learn an optimal policy, or a policy that approximates the optimum, with little or no prior knowledge of the particular problem at hand.

Beyond this basic structure, reinforcement learning comes in many different varieties.[11] Ideally, one would like to experiment with those algorithms that are more prevalent in practice but, as mentioned above, little is known on the specific software firms actually use. For our purposes, however, the choice must be restricted to algorithms that have a memory and thus can keep track of rivals' past actions, as memoryless agents are unable to punish rivals' past defections and thus cannot genuinely collude. This rules out from the outset many reinforcement learning models.

Within the set of models that can be endowed with a memory, we have chosen to focus on Q-learning algorithms. There are several reasons for this choice. First, Q-learning algorithms are designed expressly to maximize the present value of a flow of rewards in problems of repeated choice. Second, they are highly popular among computer scientists. Third, they are simple and can be fully characterized by few parameters, the economic interpretation of which is clear. This allows us to conduct a comprehensive comparative statics analysis with respect to the characteristics of the algorithms. Last but not least, 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. 2017), the Atari video-games (Mnih et al. 2015), and, more recently, chess (Silver et al. 2018).

These more sophisticated programs might appear themselves to be a natural alternative to Q-learning. However, they require modeling choices that are somewhat arbitrary from an economic viewpoint, and in this respect they resemble "black boxes." Moreover, the frontier of AI is moving fast, and it is difficult to tell what the state of the art will be in the near future. From this perspective, our focus on simple textbook algorithms is conservative. It is predicated on the conjecture that the risk of tacit collusion will, if anything, increase as the programs get smarter. Since colluding tacitly is a complex

[11] For a comprehensive treatment of reinforcement learning in computer science, see Sutton and Barto's textbook (2018).

endeavor that involves subtle strategic reasoning, the conjecture seems reasonable. But of course it needs to be verified, and this is a high-priority task in the agenda for future work.

We now provide a self-contained introduction to Q-learning algorithms.

2.1. *Single-agent problems*

Q-learning was originally proposed by Watkins (1989) to tackle Markov decision processes.[12] In a 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 agent obtains a reward π_t, and the system moves on to the next state s_{t+1}, according to a (time invariant) probability distribution $F(\pi_t, s_{t+1}|s_t, a_t)$. In the simplest settings, the distribution may be degenerate, so π_t and s_{t+1} become deterministic functions of s_t and a_t.

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

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

where s' 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 cumulative discounted payoff of taking action a in state s.[13] It is implicitly defined as:

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

The Q-function is related to the value function by the simple identity $V(s) \equiv \max_{a \in A} Q(s, a)$. If the agent knew the Q-function, he could then easily calculate the optimal policy, i.e. the optimal action for any given state.

[12] The reader is referred to Sutton and Barto (2018) for an in-depth discussion.

[13] 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.

2.1.1. *Estimating the Q-matrix*

Q-learning is essentially a method for estimating the Q-function without knowing the underlying model, i.e. the distribution function $F(\pi, s'|s, a)$. This task is performed under the assumption that S and A are finite, and that A is not state-dependent. On these assumptions, the Q-function becomes an $|S| \times |A|$ matrix.

Q-learning algorithms estimate this 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:

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

For all other cells $s \neq s_t$ and $a \neq a_t$, the Q-value does not change: $Q_{t+1}(s, a) = Q_t(s, a)$. 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. The weight $\alpha \in [0, 1]$ is called the learning rate.

If the iterative process converges to the true Q-matrix, the algorithm will learn the optimal policy $a(s) = \arg\max[Q(s, a)]$ with absolutely no prior knowledge of the function $F(\pi, s'|s, a)$. If there is to be a chance of really approximating the true matrix, however, 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 experimentation is costly and thus entails a trade-off. The benefit of exploration is better learning, which in turn improves future decisions. The downside is that by picking an action randomly instead of choosing the one with the largest current Q-value, the algorithm does not fully exploit its stock of acquired knowledge.

Finding the optimal resolution to this trade-off is problematic. In the economic theory of optimal experimentation, progress has been made only in simplified settings. For frameworks as general as those considered here, we do not yet know what an optimal exploration policy might look like. At any rate, 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 other 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*. More sophisticated exploration policies can also be designed: for example, one may let the probability ε vary over time, as we do in what follows, or let the probability with which sub-optimal actions are tried depend on their respective Q-values.[14]

2.2. *Convergence and off-the-job training*

Under quite general conditions, Q-learning algorithms converge to the optimal policy (Dayan and Watkins, 1992). This result requires that the algorithm's exploration policy belong to a class known as *Greedy in the Limit with Infinite Exploration* (GLIE). Loosely speaking, these policies satisfy three intuitive requirements: that exploration decrease 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 $t \to \infty$.

While convergence is guaranteed under these conditions, 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.

In practice, the issue of the slowness of learning is addressed by training the algorithms off the job, that is, before putting them to work. For example, AlphaZero was trained in self-play mode before facing the best human players of Go (Silver et al. 2016, 2017). Even though AlphaZero learns much faster than our Q-learners, its training lasted for several weeks. For autonomous driving software, to make another example, the training phase is typically even longer. As discussed at greater length below, we too allow the algorithms enough time to learn before assessing their behavior.

[14]For example, in the so-called Boltzman experimentation model actions are chosen with probabilities

$$\Pr(a_t = a) = \frac{e^{Q_t(s_t,a)/\tau}}{\sum_{a' \in A} e^{Q_t(s_t,a')/\tau}}$$

where the parameter τ is often called the system's "temperature." As long as $\tau > 0$, all actions are chosen with positive probability. When $\tau = 0$, however, the algorithm chooses the action with the highest Q-value with probability 1, and thus ceases to experiment.

2.3. *Q-learning in repeated games*

Although Q-learning was originally designed to deal with Markov decision processes, it can also be applied to repeated games. In this case, however, 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 in repeated games with perfect information, where strategies consist of mappings from the history of past play to the set of actions, the state s_t must be defined so as to include players' actions in previous periods. But if the actions of all previous periods are included, the set of states increases exponentially with time.[15]

This problem can be fixed 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 player i's rivals change their actions over time – because they are experimenting or learning, or both – player i's optimization problem becomes inherently non-stationary.

As we shall see in a moment, such non-stationarity has implications for convergence. Before discussing this issue, however, it may be worth noting that even if the algorithms are playing a game, they may continue to learn in a non-strategic way. That is, algorithm i regards its rivals as part of the environment and continues to update its Q-matrix according to (4). It does not even have to be aware that it is interacting strategically with other players: from its standpoint, the past actions of rivals are treated just like any other possibly relevant state variable. In the computer science literature, this approach to Q-learning in repeated games is called *independent learning*. This is the approach we use in our experiments.

2.4. *Convergence in theory*

In repeated games, there are no general convergence results for Q-learning algorithms. That is, there is no guarantee that several Q-learning agents interacting repeatedly will settle to a stable outcome, nor that they will learn an optimal policy (that is, collectively, a Nash equilibrium of the repeated game with bounded memory). The ultimate reason for this lack of convergence results is the non-stationarity discussed above. Such non-stationarity considerably complicates the analysis of the stochastic dynamical systems describing Q-learning agents' play of repeated games.

[15]Furthermore, there would be no hope of visiting the same state twice.

The analysis can be made somewhat more tractable by using stochastic approximation techniques (Benveniste et al. 1990), which allow to turn stochastic dynamical systems into deterministic ones. In economics, this approach was pioneered by Borgers and Sarin (1997) for the memoryless reinforcement learning model of Cross (1973), and it was later applied by Hopkins (2002) and Beggs (2005) to the memoryless agents of Erev and Roth (1998). These papers show that the resulting dynamics is analogous to the replicator dynamics of evolutionary games. More recently, stochastic approximation techniques have been applied also to memoryless Q-learning: see Bloembergen et al. (2015) for a survey. In this case, the outcome is a combination of replicator dynamics and a mutation term that captures the algorithms' exploration.

The stochastic approximation approach is certainly useful but it has also several drawbacks. First, many authors have noted that the asymptotic properties of the deterministic systems are not exactly the same as those of the stochastic ones they are derived from. Second, for the ε-greedy model of exploration used in this paper, the deterministic dynamics is complicated by the jumps that occur when the Q-values of the best and second-best actions cross, which give raise to hybrid dynamical systems (Gomes and Kowalczyk, 2009; Wunder et al. 2010). Third, while the dynamics of simple games can be characterized fully (see e.g. Kianercy and Galstyan 2012 for 2×2 games), the available qualitative results are much less informative when the set of strategies and equilibria gets larger.

This is especially true for the AI agents with memory that we are interested in. Applying stochastic approximation techniques to such agents is currently the frontier of the research, both in computer science and in statistical physics (Barfuss et al., 2018). The resulting deterministic systems appear to be more complicated than their memoryless counterparts, and the relationship with replicator dynamics looser. In particular, to the best of our knowledge there are no results yet available for ε-greedy Q-learning with memory. But what we know for simpler algorithms suggests that, for games as complex as those we are interested in, in any case one would have to resort to numerical integration of the dynamical systems that may emerge from the stochastic approximation.

If this is so, however, there is little to gain compared with numerically simulating the exact stochastic system, perhaps a large number of times so as to smooth out uncertainty. This is precisely what we do in what follows. Convergence and equilibrium play, which are not guaranteed ex ante, can be verified ex-post.

2.5. *Beyond Q-learning*

The simple Q-learning algorithms, which we use in our simulations, are adopted widely in computer science and indeed form the basis of the most recent developments in the field. Such algorithms have two significant drawbacks, however. First, they are unable to deal with continuous state and action spaces. The spaces must be discretized, and the finer the discretization, the slower the learning. Second, the algorithms do not realize that they are playing a game, so their learning is non-strategic. The recent computer science literature has tried to address these problems.

2.5.1. *Deep learning*

With continuous state and action spaces, the Q-matrix becomes a smooth function. Value-function approximation algorithms estimate this function by means of iterative updating methods similar to (4). Often, these methods are implemented by means of neural networks organized on several layers (deep learning). When the state and action spaces are finite but large, a smooth approximation of the Q-matrix may also serve to speed up the learning process. In this case, at each period a deep learning algorithm would update not only the last cell of the matrix that has been visited but also a number of neighboring cells.

Deep learning has long suffered from failures to achieve convergence, but significant progress on this front has been made recently. For example, in 2015 Google patented a method for updating the Q-function that has proved quite successful (even if theoretical proof of convergence is still lacking). Loosely speaking, the method uncouples the updating of the two Q-functions that appear on the right-hand side of (4). This is the method that has enabled the algorithms to attain superhuman performance in the complex strategic tasks mentioned above.

2.5.2. *Joint learning*

Independent learning implies that the competing algorithms do not realize they are playing a game. In computer science, this issue has been addressed in the so-called *joint learning* literature. A number of different approaches have been proposed. For example, with equilibrium-based joint learning, other players' actions are predicted by means of some sort of equilibrium notion.[16]

Generally speaking, joint learning faces three main challenges: (*i*) the observability of

[16]Different notions of equilibrium may be used: for example, Hu and Wellman (1998) use the Nash equilibrium, Greenwald and Hall (2003) the correlated equilibrium.

rivals' payoffs, necessary in order to anticipate their behavior but much more demanding, as a requirement, than the observability of actions; (*ii*) a curse of dimensionality, which emerges because tracing rivals' optimization increases the amount of information that each algorithm has to store and process; and (*iii*) the treatment of multiple equilibria: when different algorithms focus on different equilibria, problems of mis-coordination can arise.[17]

2.5.3. *State of the art*

The computer science literature has not yet developed a consensus approach to joint learning. And the more sophisticated "reasoning" of which joint-learning algorithms are capable does not seem to offset the advantage of independent learning in terms of lower dimensionality. In fact, the most spectacular successes in complex multi-agent environments have come through independent-learning algorithms (e.g. Silver et al. 2016, 2017, 2018, Tampuu et al. 2017).

The deep learning approach, by contrast, seems both more settled and more promising. While the iterative method and the structure of the neural network are a matter of continued research, considerable progress has unquestionably been made. Deep learning is currently being applied to the most diverse strategic settings, and there is little doubt that it will have a major impact on pricing software as well.

Nevertheless, here we have chosen to focus on simpler, "tabular" Q-learning algorithms. As discussed above, the simplicity of Q-learning helps keep possibly arbitrary modeling choices to a minimum.[18] The downside of such relatively simple AI technology is the slowness of learning. But this is not a problem as long as one focuses on the behavior of the algorithms once they have completed the training process.

This paper takes precisely such off-the-job approach. That is, we conform to the standard practice of training the algorithms before assessing their performance. This is a sensible approach as long as the economic environment is predictable.[19] If the environment may change in an unpredictable way, however, the issue is more delicate.[20] In this case, some

[17]Shoham et al. (2007) provide an excellent survey of these methods and their limits.

[18]More sophisticated algorithms have a more complex architecture requiring further parametrization. For example, with deep learning one must specify not only the learning and experimentation parameters but also a functional form for the Q-function, the number of estimation layers, and the structure of the neural network in each layer.

[19]Predictability is by no means synonymous to simplicity. Autonomous driving algorithms, for instance, are trained in realistic simulations of day-to-day traffic and also learn to face exceptional circumstances such as pedestrian jaywalking or cars passing a red light. Likewise, repricing software developers probably build complex artificial environments to train their algorithms.

[20]On the other hand, it is generally recognized that structural breaks may disrupt price-fixing agree-

learning must take place also on the job; the speed of learning then becomes important. We shall come back to this issue in the concluding section.

3. THE LITERATURE

A few papers have analyzed how Q-learning algorithms perform in oligopolies, competing either against fixed-strategy opponents or against other Q-learners. In the former case, theory predicts that the algorithms will learn to play a best response, and this is what the literature in fact finds. The latter case, on which we focus in what follows, is more interesting.

Before proceeding, it must be said that so far this literature has been dominated by computer scientists, who naturally pursue their own research agenda. As a result, the primary focus has been on algorithms' performance in terms of ability to converge, speed of convergence, and payoff. Relatively few studies have addressed the issue of whether Q-learning algorithms can learn to collude, and none provides clear evidence on this question.

3.1. *Staggered prices*

One group of studies specifically analyzes pricing games. To the best of our knowledge, all these papers consider a model of staggered pricing where the two firms alternate in moving and commit to a price level for two periods, as in Maskin and Tirole (1988). Like Maskin and Tirole, these papers posit Markov behavior, meaning that a firm can condition its price only on rivals' current prices and not on past history (including its own previous prices).

In this setting, several works have found supra-competitive prices, at least to some extent. Sometimes, however, either the models or the findings exhibit anomalies that raise doubts about the interpretation of the results. For example, the demand function posited by Greenwald et al. (1999) and Tesauro and Kephart (2002) for the case of differentiated products is quite special. In addition, both these studies find that the more efficient firm makes lower profits than its less efficient competitor. As another example, Kononen (2006) finds that prices are already substantially supra-competitive when $\delta = 0$ (a case in which firms should effectively be playing a one-shot game) and do not increase significantly with the discount factor δ. One wonders whether this may be evidence not so much of collusion as of a failure to learn the optimal strategy.

ments. Collusion is regarded as more likely in relatively stable environments, so it seems natural to start our inquiry precisely from such environments.

The best paper along these lines is perhaps Klein (2018). He considers Maskin and Tirole's baseline example with homogeneous products and linear demand. In accordance with theory, he finds that two Q-learning algorithms converge either to a constant price or to a pattern that resembles an Edgeworth cycle. In his baseline case with 6 possible price levels, profits are roughly midway between the static Bertrand-Nash level and the collusive level: in the terminology of this paper, the *average profit gain*[21] is around 50%. Enlarging the set of actions to 12 or 100 possible prices increases the average profit gain and makes it much more likely that the algorithms will converge to an Edgeworth cycle rather than a constant price.

These results are interesting and in broad agreement with our own findings. However, the postulate of price commitment is controversial, especially given that software algorithms can adjust prices very quickly. The assumption is probably consequential, as both theory and experimental evidence suggest that commitment may facilitate coordination. In point of theory, Maskin and Tirole (1988) have shown that the set of equilibrium profits that can be supported given commitment is bounded away from the static, one-shot level of profit. By contrast, the folk theorem for repeated games says that there exist subgame perfect equilibria in which profits are lower than in the static equilibrium. As for the experimental evidence, Leufkens and Peeters (2011) have shown that in a model of staggered prices, substantial cooperation is easily attained even among human subjects. In fact, as Klein recognizes, Q-learning algorithms appear to achieve less cooperation than humans in similar settings.

We differ from this literature in using the canonical model of collusion, i.e. an infinitely repeated Bertrand game in which all firms act simultaneously and condition their actions on past history. We depart from the canonical model only in assuming a bounded memory, for the reasons explained in the previous section. Also, we make an explicit analysis of the punishment strategies that sustain the collusive outcomes we observe. It turns out that these strategies hinge on the possibility of conditioning one's current price not only on competitors' but also on one's own past prices.

3.2. *Cournot competition*

Another group of papers, including most notably Waltman and Kaymak (2008), has studied repeated Cournot games. Here all players act simultaneously and have bounded recall, as in our model. They, too, find outcomes that are far from fully competitive: specifically, the average profit gain in their experiment is around 60%.

[21] For a formal definition, see equation (9) below.

In principle, the only difference with respect to our analysis is that the stage game is one of strategic substitutes rather than complements. It is not clear, a priori, whether this impedes or facilitates collusion. But in practice Waltman and Kaymak's findings do not seem to be based on equilibrium behavior. As noted above, they find less competitive outcomes when firms are myopic and have no memory – conditions under which collusion is either unfeasible or cannot emerge in equilibrium – than when they are more far-sighted and can condition current on past prices. This suggests that what Waltman and Kaymak actually find is a failure to learn to play Nash.

3.3. *Repeated prisoner's dilemma*

When the action space is reduced to two feasible actions, both Bertrand and Cournot games collapse to some sort of prisoner's dilemma. There is a substantial literature on reinforcement learning in the repeated prisoner's dilemma, which sometimes finds a significant amount of cooperation. But the prisoner's dilemma is a special type of stage game, in that there is only one way to cooperate. It is often argued that tacit collusion is hard to achieve in practice because there are many supra-competitive prices, and players may find it difficult to coordinate in the absence of explicit communication.[22] Analyses based on the prisoner's dilemma cannot address this concern. Thus, even if Q-learning algorithms managed to cooperate in the repeated prisoner's dilemma, this would not constitute conclusive evidence that they can learn to collude in more realistic settings.

3.4. *Reinforcement learning in economics*

At a broader level, our analysis is also related to the economic literature on market experimentation and reinforcement learning. The main difference from the market experimentation literature (see Bergemann and Välimäki (2006) for an early survey) is that Q-learning algorithms take the exploration policy as exogenously given and do not experiment optimally. Optimizing along this dimension may certainly improve the algorithms' performance, especially in the short run. But since our aim here is positive and not normative, we stick to what most existing algorithms actually do.

Reinforcement learning was introduced in economics by Arthur (1991) and later popularized by Roth and Erev (1995), Erev and Roth (1998) and Ho et al. (2007), among others. Borgers and Sarin (1997) noted the analogy between reinforcement learning and evolutionary game theory. The analogy becomes especially apparent if one applies the

[22]Khun and Tadelis (2018).

stochastic approximation techniques discussed in section 2.5, which show that reinforcement learning typically leads to dynamical systems that comprise some sort of replicator dynamics. Hopkins (2002) extended the analogy to models of fictitious play (see Fudenberg and Levine 1998), which also display a similar approximated dynamics.

Much of the reinforcement learning literature in economics is aimed at explaining the findings of laboratory experiments. The working hypothesis is that a good part of the difference between human behavior in the lab and the *Homo Oeconomicus* model stems from learning (Duffy, 2006). By and large, the literature has focused on rather naive models of learning on the ground that experimental subjects may not be very sophisticated.

This paper, by contrast, is motivated by the concern that the pricing algorithms employed in practice may outperform not only the typical experimental subjects but also professional decision-makers. In particular, the concern is that the algorithms may succeed in colluding tacitly, where humans typically fail. Accordingly, we focus on smarter algorithms than those considered in most of the previous economic literature, and we allow them enough time to learn effectively.

4. EXPERIMENT DESIGN

We construct a number of Q-learning algorithms and let them interact in a repeated Bertrand oligopoly setting. Here we describe the economic environment in which the algorithms operate, the exploration strategy they follow, and other details of the numerical simulations.

4.1. *Economic environment*

Since we do not want to restrict attention to any specific market, we have chosen as our workhorse a most flexible model, that is, a model of price competition with logit demand and constant marginal costs. This model has several parameters that can be modified to analyze factors such as vertical and horizontal product differentiation, or the level of demand. The model has been applied extensively in empirical work, demonstrating that it can indeed provide a good fit to many different markets.

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

$$q_{it} = \frac{e^{\frac{a_i - p_{it}}{\mu}}}{\sum_{j=1}^{n} e^{\frac{a_j - p_{jt}}{\mu}} + e^{\frac{a_0}{\mu}}}. \tag{5}$$

The parameters a_i may be taken as product quality indexes, which as such capture vertical differentiation, whereas μ is an index of horizontal differentiation. Product 0 is the outside good, so a_0 is an inverse index of aggregate demand.

Each product is supplied by a different firm, so n is also the number of firms. The per-period reward accruing to firm i is then $\pi_{it} = (p_{it} - c_i)q_{it}$, where c_i is the marginal cost. As usual, fixed costs are irrelevant as long as firms stay active.

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

One well-known limit of the logit model is that it satisfies the controversial property of *Independence of Irrelevant Alternatives*. As a robustness check, we therefore consider also the case of linear demand functions derived from quadratic preferences *à la* Singh and Vives (1984), which may violate that property.

4.1.1. *Discretization of the action space*

In a Bertrand game, the algorithms choose prices. 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 $[(1-\xi)\mathbf{p}^N, (1+\xi)\mathbf{p}^M]$, where $\xi > 0$ is a parameter. For example, in our baseline specification we set $\xi = 0.1$, so prices range from 10% below Bertrand to 10% above monopoly.

Our way of discretizing the strategy space implies that the exact Bertrand and monopoly prices may not be feasible, so there may be mixed-strategy equilibria both in the stage and in the repeated game. Since by design our algorithms generically end up playing pure strategies, they might then cycle around a target that is not feasible.

4.1.2. *Memory*

To ensure that the state space is finite, we posit a bounded memory. Specifically, in the baseline model the state is defined simply as the set of all past prices in the last k periods:

$$s_t = \{\mathbf{p}_{t-1}, ..., \mathbf{p}_{t-k}\}, \tag{6}$$

where k is the length of the memory. The assumption here is perfect monitoring, which is reasonable for markets where algorithmic pricing is used.[23]

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

4.2. *Exploration*

As is pointed out in Section 2, given the set of states S and the set of actions A, a Q-learning algorithm is fully specified by the learning rate α, the discount factor δ, and an exploration strategy. Here we use one of the simplest exploration policies in the GLIE class, namely, an ε-greedy model with a time-declining exploration rate. More specifically, we set

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

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 exploration fades away.

4.3. *Initialization*

Q-learning algorithms may well start with a clean slate, such as $\mathbf{Q}_0 = \mathbf{0}$.[24] However, to speed up the learning process the initial Q-matrices may be set in accordance with certain hypotheses. For example, our baseline choice is to set $Q_{i,0}(s, a)$ at the discounted payoff that would accrue to player i if opponents randomized uniformly:

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

This is in keeping with our 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.

However, we have run robustness checks with alternative initializations, corresponding to rivals playing, e.g., the static Bertrand-Nash equilibrium. With enough exploration, the results are insensitive to the way the matrix is initialized.

[23] For example, the Amazon.com API allows sellers to recover current and past prices of any product with a simple query.

[24] The "zero" in the name of the celebrated program AlphaZero stands precisely for "zero prior knowledge."

4.4. *Convergence*

As mentioned, for strategic problems like ours 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 convergence can be verified in practice. We use the following 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 $a_{i,t}(s) = \arg\max [Q_{i,t}(a, s)]$ stays constant for 100,000 repetitions, we assume that the algorithms have completed the learning process and attained stable behavior.[25] We stop the simulation when this occurs, and in any case after one billion repetitions.

4.5. *Experiments and outcomes*

For each set of parameters, an experiment consists of 1,000 sessions. In each session, agents play against the same opponents for a large number of periods: up to a billion, or until convergence as defined above.

For sessions that converge, we calculate the outcomes associated with the strategies the algorithms have eventually settled to. As discussed, we focus on long-run outcomes on the assumption that the algorithms are trained off-line before being put to work. In each experiment, we then calculate the mean and standard error for the 1,000 independent sessions.

We can observe all variables of interest (prices, profits, market shares, etc), but we often elect to concentrate on one economic indicator, namely average profit gain Δ, defined as:

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

where π^N is the per-firm profit in the Bertrand-Nash static equilibrium, π^M is the profit under full collusion (monopoly), and $\overline{\pi}$ is the profit upon convergence. Thus, $\Delta = 0$ corresponds to the competitive outcome and $\Delta = 1$ to the perfectly collusive outcome. The main reason for focusing on Δ is that this index can be compared directly across different economic settings.

[25] An alternative standard of convergence would require that the entries of Q-matrices not change by more than some $x\%$ for a certain number of periods. This criterion is different from the one we use because even if $\arg\max [Q_{i,t}(a, s)]$ does not change, the Q-matrices themselves might keep changing.

5. RESULTS

In this section, we focus on a baseline economic environment that consists of a symmetric duopoly ($n = 2$) with $c_i = 1$, $a_i - c_i = 1$, $a_0 = 0$, $\mu = \frac{1}{4}$, $\delta = 0.95$ and a one-period memory ($k = 1$). Holding this environment constant, we vary the learning and experimentation parameters α and β, which for now are taken to be the same for both competing algorithms.

In the next section, we run extensive comparative statics on the economic parameters, but it is worth making two remarks at the outset. First, for the baseline specification of the logit model, the price-cost margin would be around 47% in the static Bertrand equilibrium, and about twice as large under perfect collusion. These values do not seem implausible but it is important to note that our analysis is very robust in this respect. For example, we could as well have taken as our starting point the limiting case with homogeneous products (i.e., $\mu \to 0$), where in the absence of dynamic cooperation firms would price at cost.

Second, the assumption of a one-period memory is less restrictive than it might seem, because, as is noted by Barlo et al. (2016), the richness of the state space may substitute for the length of the memory. Even with $k = 1$, the algorithms can effectively replicate strategies that, strictly speaking, would require a longer memory.

To illustrate, consider the simple case of a stage game with only two feasible actions, of the prisoner's dilemma type. With $m = 2$ and $k = 1$, there are four possible states, namely (with obvious notation) CC, CD, DC and DD. In this setting, cooperating in state CC and defecting in all other states is a strategy whose incentive compatibility constraint is exactly the same as that of a true grim-trigger strategy, which in principle requires an infinitely long memory. Likewise, cooperating in states CC and DD and defecting otherwise has the same incentive compatibility constraint as a "one-period punishment" strategy. Loosely speaking, a unilateral defection is interpreted as a genuine deviation, which as such is punished. In contrast, joint defection is interpreted as punishment for a previous deviation and is accordingly followed by return to cooperation.[26]

When the state and action spaces are richer, it is possible to mimic more complex strategies as well. In fact, we find that our algorithms are able to learn punishment strategies that

[26]Grim trigger strategies turn out not to be an effective way for Q-learning algorithms to cooperate. This is because with experimentation one algorithm would sooner or later defect, and when this happened both would be trapped in a protracted punishment phase that would last until further (joint) experimentation drove the firms out of the trap. In fact, in our preliminary experiments with two feasible prices only (not reported here) we found that cooperation was typically achieved by means of one-period punishment strategies.

have finite but long duration, with a slow return to the pre-deviation prices.

5.1. *Learning and experimentation*

We try a wide range of values of the characteristic parameters of the algorithms, α and β, discarding only those that seem implausible a priori. For example, the learning parameter α may in principle range from 0 to 1. However, it is well known that high values of α may disrupt learning, especially when experimentation is extensive, because the algorithm would forget too rapidly what it has learned in the past. In the computer science literature, a value of 0.1 is often used (Sutton and Barto, 2018). We consider also higher values but take 0.25 as an upper bound. This upper bound is conservative, given that the algorithms are trained off the job and in the process are allowed to experiment extensively, as we shall discuss in a moment. Our initial grid then comprises 100 equally spaced points in the interval $[0.025, 0.25]$.[27]

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 3375 in our baseline experiments with $n = 2$, $m = 15$ and $k = 1$, and many more in more complex environments). On the other hand, in the short run exploration is costly. Here we abstract from this cost to consider only long-run outcomes (i.e., after the convergence of the algorithms). 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. This means that the parameter β should be small enough to allow for enough experimentation, but not too small.

To get a more concrete sense of what values of β might be reasonable, it may be useful to map β into ν, the expected number of times a "sub-optimal" cell (i.e., a cell that corresponds to actions that are perceived as sub-optimal) would be visited.[28] Clearly, if ν were too small the algorithm would not be given a fair chance to learn. We take as a lower bound $\nu = 4$, which seems barely sufficient to guarantee decent learning. For example, the

[27] A positive lower bound is necessary because when $\alpha = 0$ the algorithm does not learn at all. In this respect, a value of 0.025 seem conservative as well. With lower values, the learning becomes very slow and may actually fail altogether.

[28] The exact relationship between ν and β is

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

initial Q-value would still carry a weight of more than 30% after 4 updates with $\alpha = 0.25$, and the weight would be even greater for lower values of α.

When $n = 2$ and $m = 15$, the lower bound on ν implies an upper bound for β of (approximately) $\bar{\beta} = 2 \times 10^{-5}$. Like 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 $\nu \approx 450$ – a value that is perhaps even excessive in view of the learning externality mentioned above.

As said, initially both algorithms share the same learning and exploration parameters. Section 6 explores the consequences of relaxing this assumption.

5.2. *Convergence*

In spite of the absence of theoretical guarantees, in more than 99.9 % of the sessions we have obtained convergence by our definition.

Typically a great many repetitions are needed for convergence.[29] The exact number depends on the level of exploration, ranging from about 400,000 when exploration is rather limited to over 2,500,000 when it is more extensive. 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. That 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.

5.3. *Consistency*

Outcomes are quite stable across sessions. This is not a foregone conclusion, given that learning requires extensive experimentation which, being random, might well create considerable variation. Yet the standard error of the profit gain is typically less than 1 percentage point. Another sign of the consistency of the learning process is that the two algorithms, which in the baseline model are ex-ante symmetric, perform in a very similar manner: the difference between the two firms in profit gain is never statistically significant. This suggests that what the algorithms eventually do is not casual.

Furthermore, prices are quite stable upon convergence. That is, in almost half the sessions

[29] In terms of computing time, by contrast, convergence is actually quite fast, requiring on average less than one minute of CPU time.

both algorithms keep charging the same price period after period. The other sessions display price cycles, but more than three quarters of them have a period of two, and all involve adjacent prices. We interpret these cycles not as Edgeworth cycles but rather as an artifact of our discretization, with actual prices orbiting around constant targets.[30]

5.4. *Equilibrium play*

Even if the algorithms learn to behave consistently, this might not ensure that they eventually play an optimal strategy. Again, this is guaranteed theoretically for single-agent decision making but not when different algorithms are involved. In this case, if both algorithms converged to an optimal strategy in the presence of the rival, they would converge to a Nash equilibrium of the repeated game with bounded recall.

Although not guaranteed ex ante, this property can be verified ex post. To this end, upon convergence we check whether an algorithm's strategy is a best response to the one the rival has converged to. If not, we then calculate the distance from best response. That is, we compare each agent's final Q-matrix with the true Q-matrix corresponding to the rival's final strategy. This check can be performed either on path (verifying whether a Nash equilibrium is played), or both on and off path (verifying subgame perfection).

As shown in Figure 1, the extent to which the algorithms converge to a Nash equilibrium depends on the learning and exploration parameters. Intuitively, learning is more effective when experimentation is extensive (β low). In particular, when β is close to the upper bound of the grid, there is clearly too little exploration (sub-optimal cells are visited only 4 times on average). But already for moderate levels of experimentation the algorithms learn quite successfully. For example, when sub-optimal cells are visited on average just 10 times (which requires $\beta = 8 \times 10^{-6}$), with $\alpha = 0.15$ in about one third of the sessions both algorithms play a best response to the strategy eventually adopted by the rival. For the remaining sessions, the observed Q-values differ from those associated with best response by less than 3% on average.

With still more extensive experimentation, equilibrium play becomes prevalent. For example, when $\beta = 3 \times 10^{-6}$ (meaning that sub-optimal cells are visited on average 25 times) and $\alpha = 0.15$, a Nash equilibrium is attained in 60% of cases, and each algorithm plays an individual best response more than 70% of the times. Moreover, even when the algorithms do not play best responses, they come quite close: the potential profit gain from a true best response is, on average, less than 1%.[31]

[30] In theory the algorithms might also learn to split the market dynamically by taking turns, but apparently they do not.

[31] That is, excluding the best responses for which the distance is nil. To some extent, equilibrium play is

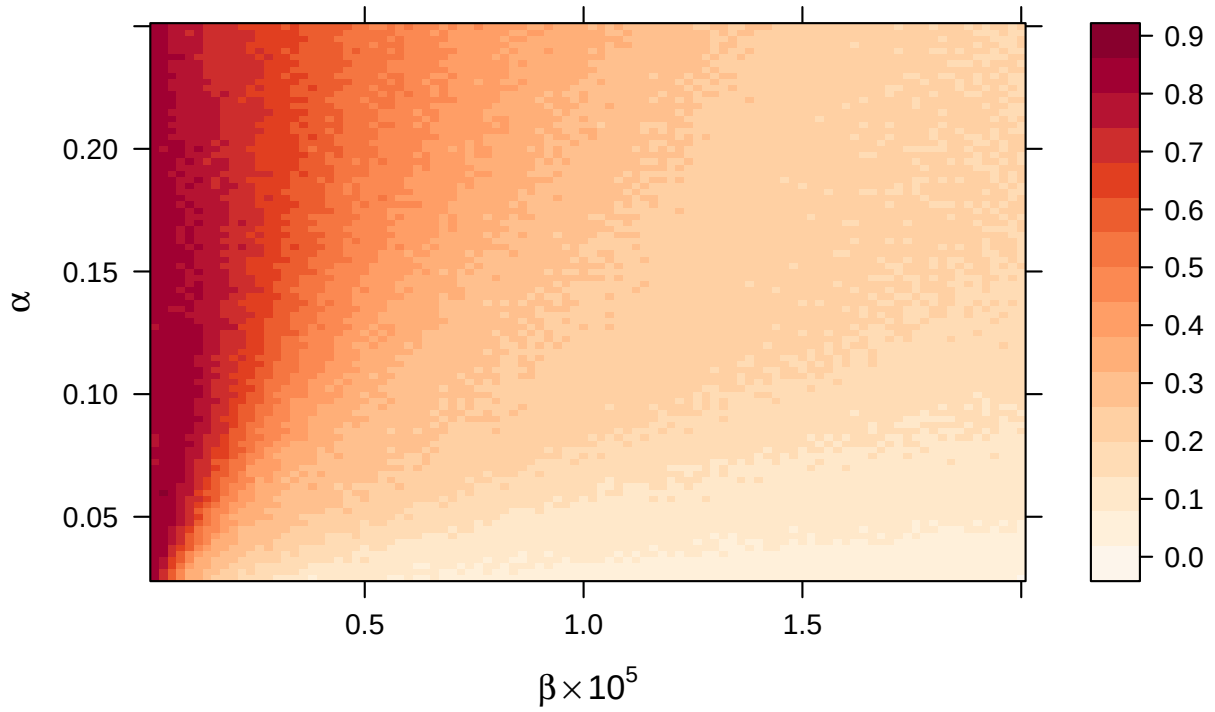

Figure 1: Fraction of strategy profiles converging to NE for a grid of values of α and β. (Note that β has been rescaled by 10^5.)

These numbers suggest that the algorithms do learn to play Nash equilibria, provided that they are given a fair chance to experiment and learn. A key implication is that once the learning process is complete, the algorithms cannot be exploited, no matter how smart the opponent is.[32]

Off the equilibrium path, things are somewhat different. For the values of α and β considered above, fewer than 2% of the sessions produce a subgame perfect Nash equilibrium. This should not come as a surprise, however, as Q-learning algorithms visit sub-optimal cells only a few times. As a result, the algorithms cannot learn perfectly how to respond off path. Since experimentation is independent, this is especially true for states corresponding to past actions far from the optimal. As one gets closer to the equilibrium, however, off-path cells are visited more often. This allows the algorithms to learn quite effective punishment strategies, as we shall see below.

more common with even more extensive experimentation. For example, with $\alpha = 0.15$ and $\beta = 6 \times 10^{-7}$, the algorithms play a Nash equilibrium 80% of the times. Yet increasing experimentation even further eventually backfires, probably because of the exploration externality mentioned above.

[32]Whether Q-learning algorithms could be exploited during the learning phase, if it were conducted on-the-job, is an interesting and difficult question for future study.

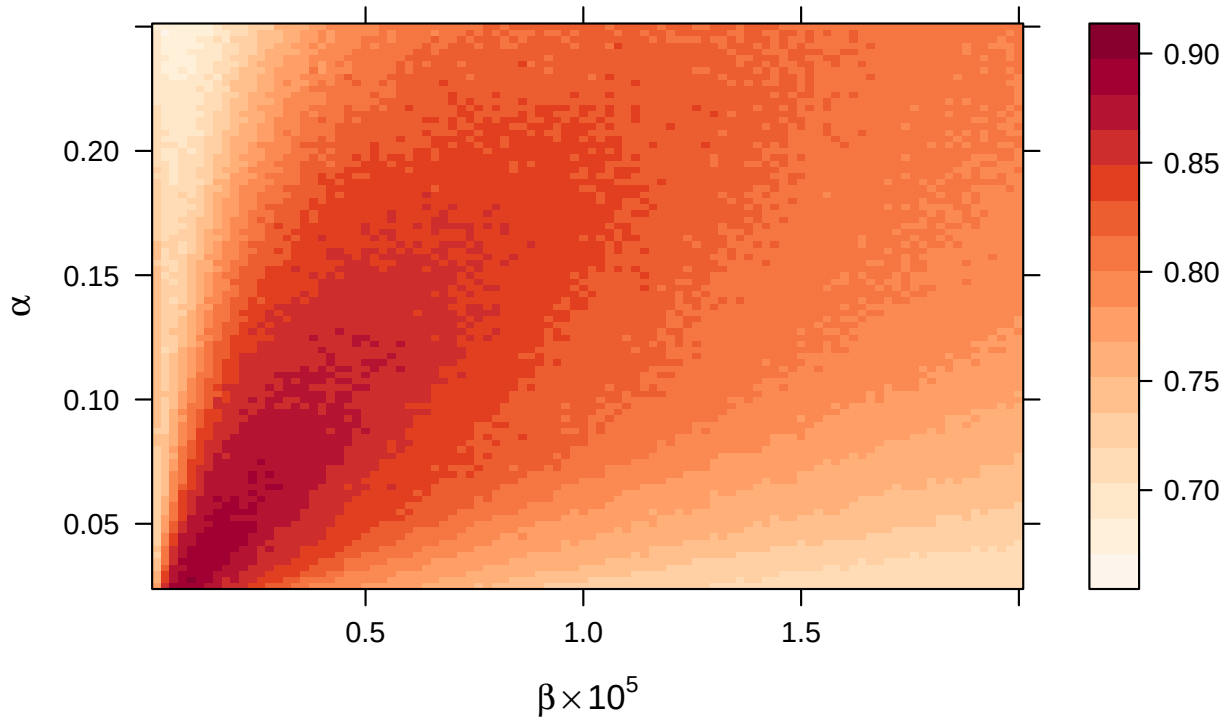

Figure 2: Average profit gain Δ for a grid of values of α and β.

5.5. *Outcomes*

What do the algorithms do once they are trained? In this subsection, we show that they systematically charge supra-competitive prices and earn supra-competitive profits. In the next subsection we inquire into the strategies that underpin these collusive-looking outcomes.

The average profit gain Δ is represented in Figure 2 as a function of α and β. In the interval we have considered, Δ ranges from 70% to 90%. That is, non-competitive outcomes are quite common, not obtained at just a few selected points.[33] Moreover, the profit gain is not particularly sensitive to changes in the learning and experimentation parameters.

Given the consistency across sessions documented above, the observed values of Δ arise not because the algorithms sometimes converge to the monopoly price and sometimes to the competitive one, but rather because they learn systematically to raise prices to some

[33]In a further robustness check, we have also explored the algorithms' performance for higher values of α than those considered in our grid. The profit gain is smallest when exploration is extensive (β small) and α is very high, above 0.8. For this range of parameters, the algorithms explore widely but forget rapidly. This does not look like a particularly effective way of learning, and in fact it leads to relatively limited equilibrium play. Yet not even this stolid approach leads to cutthroat competition, and the average profit gain remains around 40%.

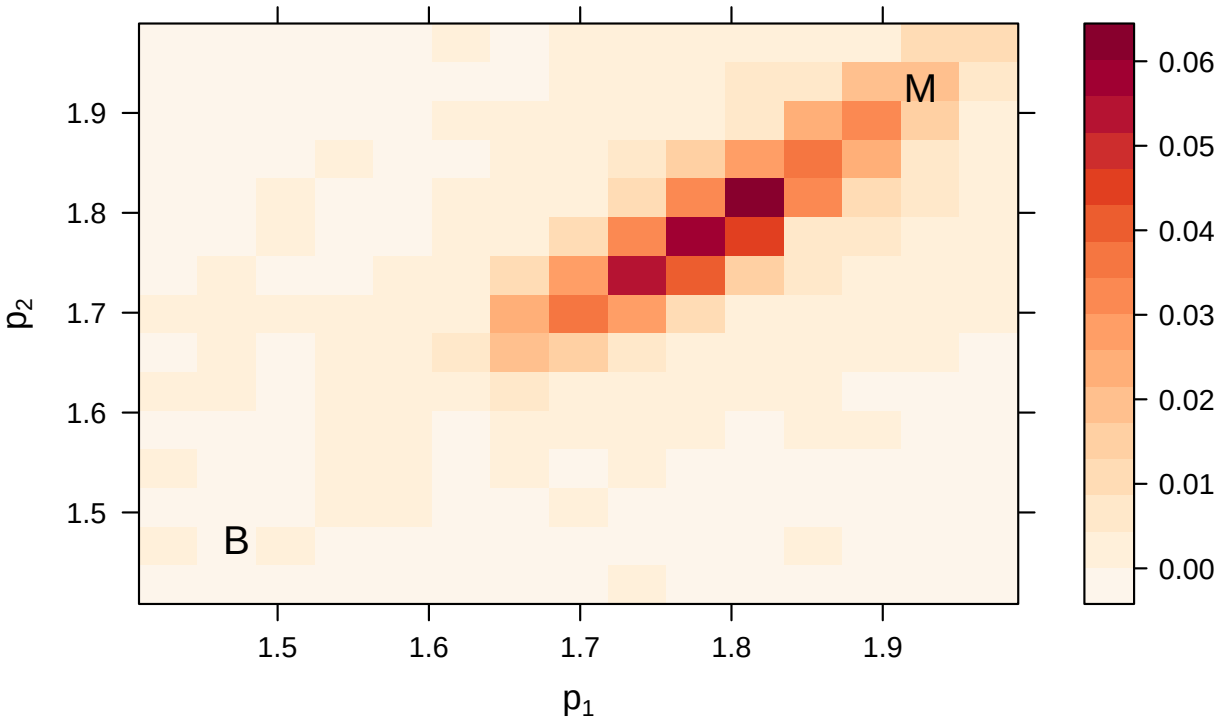

Figure 3: Distribution of states visited in the last 100,000 iterations pooling all data from 1000 sessions with $\alpha = 0.15$, $\beta = 4 \times 10^{-6}$. "M" corresponds to the fully collusive price, "B" to the Bertrand equilibrium price.

intermediate level (partial collusion). This is confirmed by Figure 3, which shows the relative frequency of the different price combinations eventually charged for a representative experiment. Prices are rarely as high as under monopoly but are almost always higher than in the Bertrand-Nash equilibrium. Price dispersion is low, and firms tend to price symmetrically.

In sum, our algorithms consistently and symmetrically charge supra-competitive prices, obtaining a sizable profit gain.

In view of the robustness of the results with respect to the learning and experimentation parameters, to ease exposition we shall henceforth focus on one representative experiment, corresponding to $\alpha = 0.15$ and $\beta = 4 \times 10^{-6}$. (This is the experiment illustrated in Figure 3.) With these parameter values, sub-optimal cells are visited on average about 20 times, and the initial Q-value of such cells counts for just 3% of their final value. A Nash equilibrium is learned 54% of the times, and the average profit gain is 85%. None of these values seems extreme. But at any rate, we have systematically conducted robustness analyses with respect to α and β not only for the baseline case but also for the extensions considered below so as to confirm that our results are not sensitive to these parameters.

5.6. *Strategies*

Let us now turn to the issue of what strategies underpin the documented non-competitive outcomes. The key question is whether the high prices are the result of the algorithms' failure to learn the static Bertrand-Nash equilibrium or of genuine collusion. The policy implications would be radically different: the former means that the algorithms are not smart enough, the latter that they are already, in a sense, "too smart." As AI technology advances, in the former case the problem is likely to fade away; in the latter, to worsen.

At this point, it may be useful to spell out exactly what we mean by collusion. Following Harrington (2017), we define collusion as "a situation in which firms use a reward-punishment scheme to coordinate their behavior for the purpose of producing a supra-competitive outcome." That is, what is crucial is not the level of profits as such but the way the supra-competitive result is achieved. Even extremely high profit levels may be regarded as collusive only insofar as deviations that are profitable in the short run would trigger a punishment.

That Q-learning algorithms actually do learn to collude is suggested by the fact that the profit gain tends to increase with the amount of equilibrium play. But the correlation is far from perfect,[34] so here we set forth two additional, perhaps more compelling arguments. First, in economic environments where collusion cannot arise in equilibrium, we find that the algorithms learn instead to set competitive prices. Second, going back to settings where collusion is possible, we consider exogenous defections and observe how the algorithms react. We find that such defections gets punished, and that the punishment makes the defections unprofitable. This is perhaps the most direct possible evidence of collusive behavior where collusion is tacit.

5.6.1. *Competitive environments*

In certain environments, collusion is impossible by default; in others, it can never arise in equilibrium. If the supra-competitive prices that we find were the result of erratic choices, or of something other than collusion, then they should also be expected in settings where collusion is impossible. In particular, collusion is impossible when $k = 0$ (the algorithms have no memory), and it cannot arise in equilibrium when $\delta = 0$ (the immediate gain from defection cannot be outweighed by the loss due to future punishments). As noted in Section 3, the previous literature has indeed found supra-competitive prices even in these

[34] In particular, increasing exploration initially improves learning and increases profits but eventually backfires; the same is true for increasing α. But while for the profit gain the downside of decreasing β and increasing α is already evident in Figures 1 and 2, for equilibrium play it only appears for values of α and β that lie outside of the interval shown in the figures.

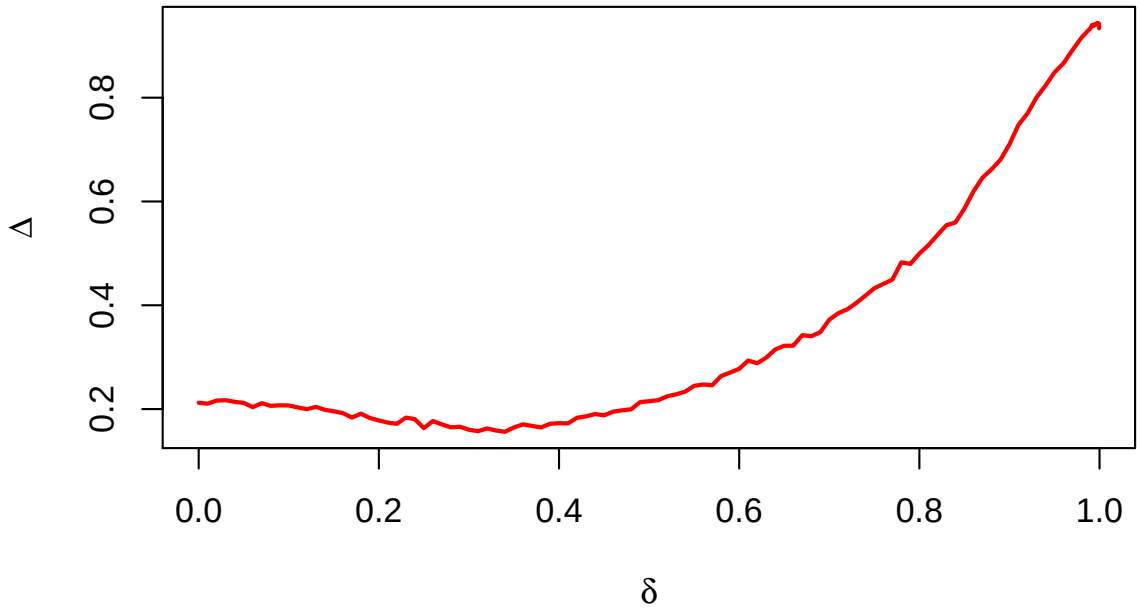

Figure 4: Average Profit Gain Δ for $\alpha = 0.15, \beta = 4 \times 10^{-6}$

settings, which poses the question of how to interpret the findings in economic terms.

In this respect, our results are quite different.[35] Consider first what happens when the algorithms get short-sighted. Figure 4 shows how the average profit gain varies with δ. The theoretical postulate that lower discount factors, i.e. less patient players (or else less frequent interaction), impede collusion, is largely supported 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%.

At this point, however, something perhaps surprising happens: the average profit gain starts increasing as δ decreases further. Although the increase is small, it runs counter to theoretical expectations. 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.[36] 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.[37] Figure 4 suggests that if one could abstract from this spurious effect, collusion

[35]The difference might be due to the fact that we use a different economic model, and our algorithms are allowed to learn more effectively, than in previous studies.

[36]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 $\alpha(1-\delta)$.

[37]A similar problem emerges when δ is very close to 1. In this case, we observe another "paradox," i.e.,

would tend to disappear when agents become short-sighted.

Turning to the case of memoryless algorithms, we find profit gains of less than 5%. These are almost negligible and do not vanish altogether simply because our discretization of the strategy space implies that the one-shot Bertrand equilibrium can at best be approximated.

All of this means that the algorithms do learn to play the one-shot equilibrium when this is the only equilibrium of the repeated game. If they do not play such competitive equilibrium when other equilibria exist, it must be because they have learned more sophisticated strategies.

5.6.2. *Deviations and punishments*

To look into how these strategies are structured, we perturb the system once the learning is completed. That is, upon convergence we step in and manually override one agent's choice, forcing it to defect. We impose not only defections lasting for a single period but also defections lasting several periods; and defections both to the static best-response and to smaller price cuts. For all of these cases, we then examine the reaction of both agents in the subsequent periods. In a word, we derive impulse-response functions.

Figure 5 shows the average of the impulse-response functions derived from this exercise for all 1000 sessions of our representative experiment. It shows the prices chosen by the two agents τ periods after the deviation. In particular it depicts the evolution of prices (top) and profits (bottom) following agent 1's one-period deviation to the static best-response to the pre-deviation price.

Clearly, the exogenous deviation gets punished. The punishment is not as harsh as it could be (e.g., a reversion to the static Bertrand-Nash equilibrium), and it is only temporary: in the subsequent periods, the algorithms gradually return to their pre-deviation behavior.[38] This pattern seems natural for Q-learning algorithms. These algorithms experiment widely, at least initially, so it would be difficult to sustain collusion if any defection triggered a permanent switch to the one-shot Bertrand equilibrium.

But in any case, the punishment is harsh enough to make the deviation unprofitable as illustrated by the evolution of profits after the shock in Figure 5 (bottom). Evidently, agent 2's retaliation wipes out agent 1's profit gain already in the very next period: that is, incentive compatibility is verified.

---

the average profit gain eventually starts decreasing with δ. This failure of Q-learning for $\delta \approx 1$ is well known in the computer science literature.

[38] In fact, prices stabilize on a new level that on average is slightly lower than the pre-deviation one.

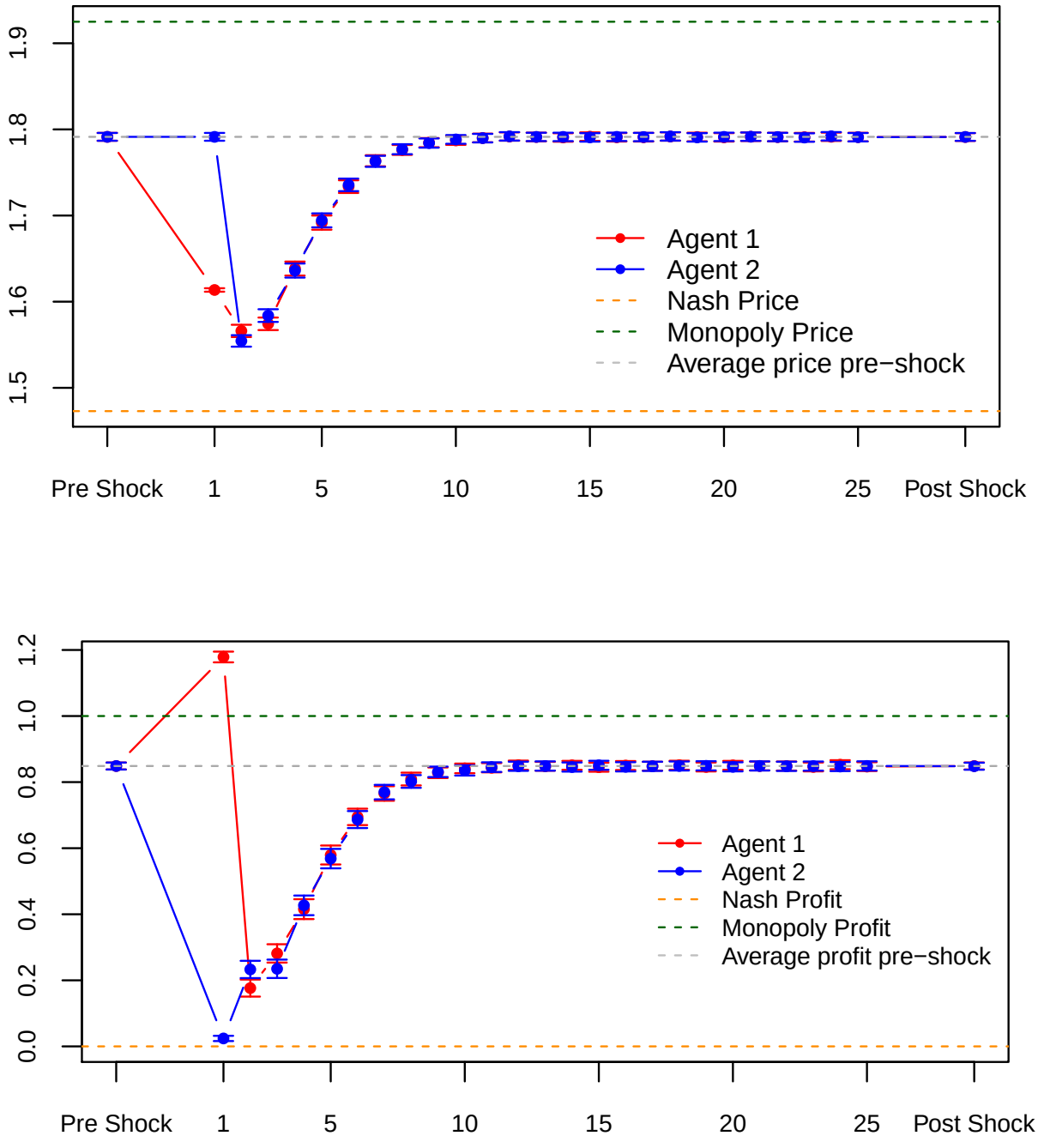

Figure 5: Prices (top) and Profits (bottom) impulse response, $\alpha = 0.15, \beta = 4 \times 10^{-6}, \delta = 0.95$.

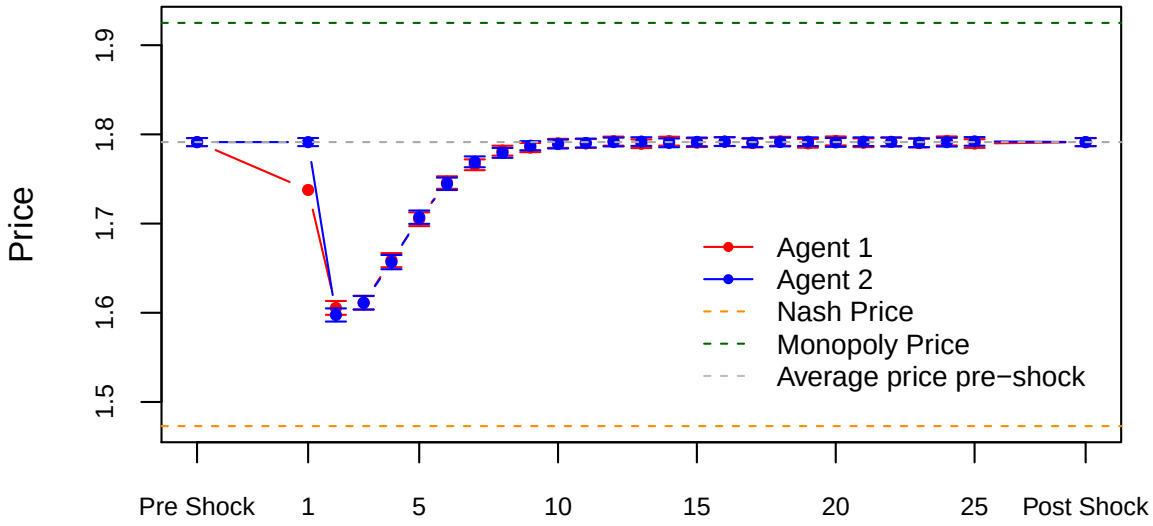

Figure 6: Price Impulse response, 1-period deviation to the 9th price (1.74), $\alpha = 0.15, \beta = 4 \times 10^{-6}, \delta = 0.95$.

This shows, by itself, that what we have here is collusion in the sense of Harrington. At a more speculative level, however, one may wonder whether the algorithms really intend to punish the defection, or whether instead the price cuts serve simply to regain market share. To put it differently, the issue is whether we observe true strategic behavior or just a stable dynamical system that mechanically returns to its rest point after being perturbed. Looking only at Figure 5, in fact, it is hard to tell these alternative explanations apart. But if one considers smaller deviations (Figure 6), the difference becomes perfectly clear. In response to the deviation, both algorithms now cut their prices further down, below the exogenous initial reduction. Such "overshooting" would be difficult to rationalize unless the non-deviating algorithm is really punishing the rival.

In fact, it is tempting to say that the deviating algorithm is actively participating in its own punishment. Such self-reactive behavior is indeed often instrumental in achieving collusion. At the very least, the deviating algorithms is anticipating the punishment – otherwise it would have no reason to reduce its price as soon as it regains control, i.e. in period $\tau = 1$, given that the rival's price was still high in period $\tau = 0$. This means that the algorithm responds not only to its rival's but also to its own past action.

To further confirm that the algorithms behave strategically, we derive the impulse-response function for the case $\delta = 0$, where collusion does not arise.[39] In this case, an exogenous price cut is followed by immediate return to the original prices. From this we conclude that the dynamics that we observe when the algorithms are patient is a clear sign of

[39]For memoryless algorithms, the entire exercise would be pointless.

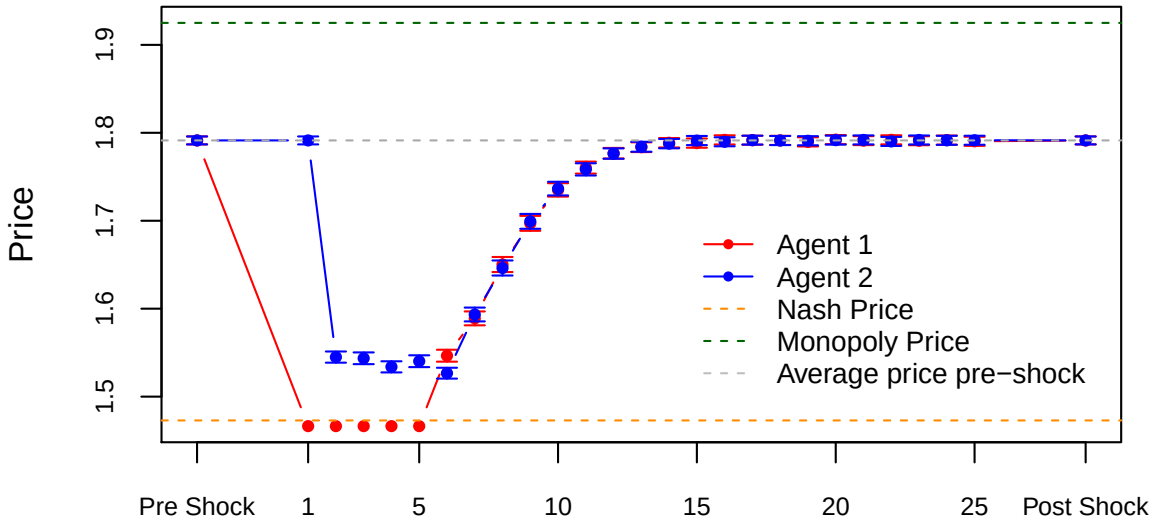

Figure 7: Price Impulse response, 5-period deviation, $\alpha = 0.15, \beta = 4 \times 10^{-6}, \delta = 0.95$.

strategic, genuinely collusive behavior.

Going back to the case of patient agents, Figure 7 illustrates the case of a 5-period deviation. The punishment gets a bit harsher after the second deviation but then stabilizes. In any case, the non-deviating firm stands ready to forgive: as soon as the defection ends, there is a gradual return to the original prices, on a pattern similar to the one-period deviation.

As a final check, we calculate an index of the intensity of the punishment for the entire grid of values of α and β considered in this section. A comparison of Figures 8 and 2 clearly shows the high positive correlation between the profit gain and the harshness of the punishment. This further confirms that the supra-competitive prices are the result of true tacit collusion.

Summarizing, we find clear evidence of collusive behavior. The supra-competitive profits are supported by off-path punishments that have a finite duration, with a slow return to the pre-deviation prices.

6. ROBUSTNESS

How robust is this result to changes in the economic environment? In this section, we consider a number of economic factors that may affect firms' ability to sustain a tacit collusive agreement. We start from factors that in theory might disrupt coordination, and we then move to factors whose impact is not clear a priori. In this comparative statics exercise, we keep the learning and exploration parameters α and β constant at $\alpha = 0.15$

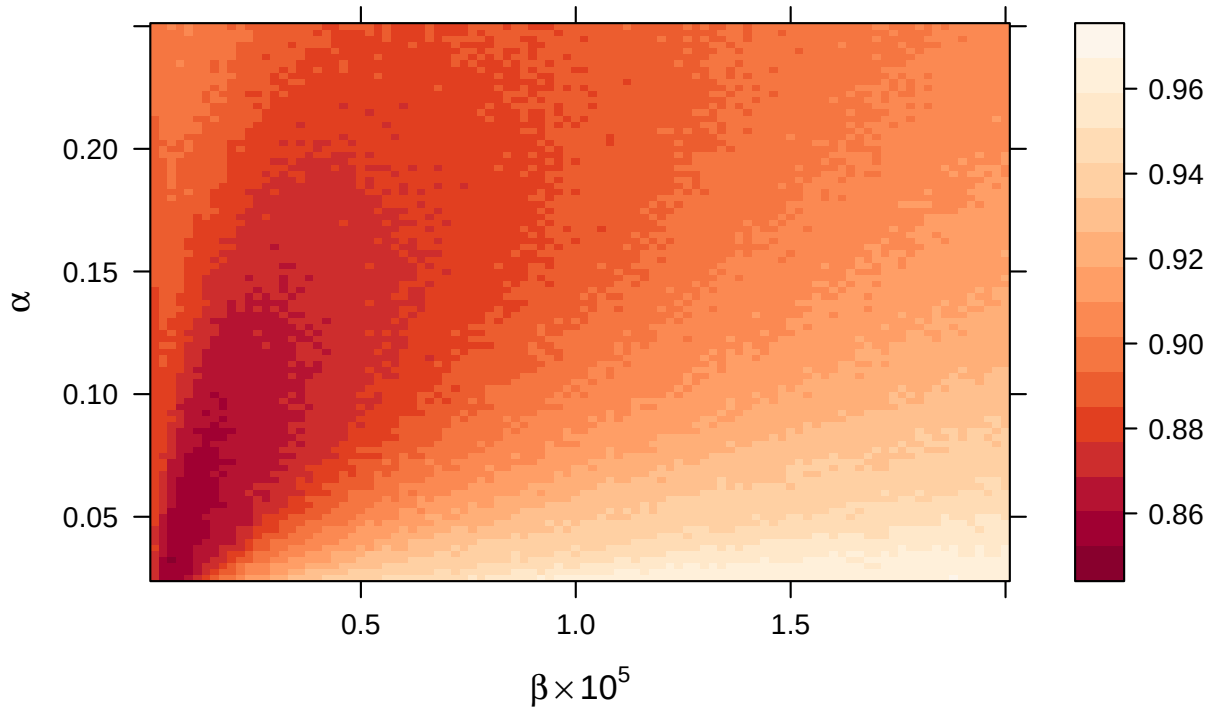

Figure 8: Average % price drop of non deviating agent 2 one period after deviation.

and $\beta = 4 \times 10^{-6}$.

6.1. *Number of players*

Theory predicts that collusion is harder to sustain when the market is more fragmented. This is indeed one reason why antitrust authorities regulate mergers: apart from the unilateral effects, the concern is that more concentrated markets may be conducive to tacit collusion.

The experimental literature provides some support for this thesis, showing that without explicit communication it is very difficult for more than two agents to collude. With three agents, prices are typically pretty close to the static Bertrand-Nash equilibrium, and with four agents or more they may even be lower.[40]

In this respect, Q-learning algorithms would appear to be different. In simulations with three firms, the average profit gain Δ decreases from 85% to 64%. With four agents, the profit gain is still a substantial 56%. The fact that Δ decreases accords with the theoretical predictions; the fact that it decreases so slowly seems to be a peculiar and

[40]See for instance Huck et al. (2004). Cooper and Khun (2014) stress the importance of communication to sustain collusion in the lab.

somewhat worrying property of pricing algorithms.

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 $n = 4$ drastically enlarges the Q-matrix, 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 close to 75% for lower values of β.

With three or four agents, the impulse response functions remain qualitatively similar to the case of duopoly. We still have punishments, and in fact the punishment is harsher than in the two-firms case.

6.2. *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 generally 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. The results are not sensitive to the source of the asymmetry, so Table 1 reports only the case of cost asymmetry.

c_2	1	0.875	0.750	0.625	0.5	0.25
2's Nash market share	50%	55%	59%	63%	66%	72%
Δ	85%	84%	81%	78%	76%	71%
$\frac{\pi_1/\pi_1^N}{\pi_2/\pi_2^N}$	1	1.05	1.12	1.19	1.27	1.446

Table 1: Cost asymmetry ($c_1 = 1$).

As the table shows, asymmetry does reduce the average profit gain, but only to a limited extent. For example, in a variant of the baseline model where the more efficient firm (firm 2) has a 12.5% cost advantage (that is, $c_1 = 1$ and $c_2 = 0.875$), the average profit gain falls from 85% to 84%. With a 25% cost advantage, the average profit gain is 81%, and even with a 50% advantage it is still 76%.

In part the decrease in the average profit gain 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 disproportionally 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.[41] 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.

6.3. *Stochastic demand*

In the baseline model, the economic environment is deterministic. The only source of uncertainty for a player is rivals' experimenting and learning. It is often argued that tacit collusion is harder to sustain when market conditions vary stochastically.

To assess the algorithms' ability to cope with uncertainty, we let the aggregate demand parameter a_0 vary stochastically. Specifically, a_0, which in the benchmark is nil, is now assumed to be an i.i.d. binary random variable that takes two values, $a_0^H = x$ and $a_0^L = -x$ with equal probability. This corresponds to negative and positive demand shocks, respectively. The shocks are purely idiosyncratic and have no persistency – perhaps the most challenging situation for the algorithms.

When $x = 0.15$, the average profit gain under uncertainty falls from 85% to 81%; and even when $x = 0.25$ (so demand is more than 20% higher in good than in bad states) the average profit gain is still 73%. Apparently, then, demand variability does hinder collusion among firms, as one would have expected, but it does not eliminate it.

6.4. *Variable market structure*

In our baseline experiments, all firms are active at all times. To analyze the impact of a variable market structure, we repeat the exercise with one firm (the "outsider") entering

[41] This effect may be so pronounced that the less efficient firm may actually get less profit under joint profit maximization than in the Bertrand-Nash equilibrium. This is why the level of the individual profit gains Δ_i given asymmetry needs to be interpreted with caution.

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 $\mathbb{I}_t$ be an indicator function equal to 1 if the outsider is in the market in period t and to 0 otherwise. We set

$$\text{prob}\{\mathbb{I}_t = 1|\mathbb{I}_{t-1} = 0\} = \text{prob}\{\mathbb{I}_t = 0|\mathbb{I}_{t-1} = 1\} = \rho. \tag{10}$$

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 99.9%, so that when the outsider enters, it stays in the market for an average of 1,000 periods. Since in marketplaces where algorithmic pricing is commonly adopted periods can be very short, this level of persistency is actualy rather low.[42]

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.

We find that the average profit gain decreases to 56%. This substantial fall is the combined effect of the increase in the size of the matrix, which as noted impedes learning, and uncertainty. Still, we are far from the competitive benchmark. With more persistence ($\rho = 99.99\%$) the average profit gain is 62%, close to what we find with three firms and no uncertainty.

6.5. *Memory*

As we have seen, the assumption of one-period memory is less restrictive than it might seem, but at all events we have also run experiments with two-period memory ($k = 2$). One might presume that longer memory allows more flexible strategies and hence facilitates collusion. However, longer memory increases the size of the Q-matrix and thus makes learning harder, exactly like an increase in the number of firms. We find that this latter effect actually prevails, and the average profit gain decreases from 85% to 57%.

6.6. *Product substitutability*

In the logit model, a decrease in μ means that individual demand becomes more price-sensitive. That is, the reduction in μ captures an increase in product substitutability.

[42]If a period were a quarter of an hour, say, then 1,000 periods would be less than two weeks.

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.[43]

In our setting, we test the consequences of changing parameter μ from 0.25 (baseline) up to 0.5 and down to 0.01, where products are almost perfect substitutes. The average profit gain decreases slightly when μ decreases, but when the products are almost perfect substitutes ($\mu = 0.01$) it is still greater than 80%.

6.7. *Linear demand*

We repeated the analysis for the case of duopoly with linear demand functions derived from a quadratic utility function of the Singh and Vives (1984) type, i.e.

$$u = q_1 + q_2 - \frac{1}{2}(q_1^2 + q_2^2) - \gamma q_1 q_2 \tag{11}$$

for various values of the horizontal differentiation parameter γ. The average profit gain is non monotone in γ: it is well above 80% when γ is below 0.4 or above 0.9 (when the products are fairly good substitutes) but reaches a minimum of 65% when γ is around $\frac{3}{4}$. The impulse-response functions are almost identical to those observed with logit demand. Other alternative demands could also be tested, of course, but it would seem that the results are robust to the demand specification.

6.8. *Finer discretization*

As Section 3 notes, one reason why the repeated prisoner's dilemma may be a misleading approach to the analysis of collusion is that coordination is much easier when there are only few possible actions. With two actions only, for instance, there is basically only one way to cooperate. As the number of actions and hence strategies increases, the strategy on which the players should coordinate is no longer self-evident. Failing explicit communication, this could engender misunderstandings and prevent cooperation.

To determine whether a richer strategy space facilitates or impedes cooperation, we run experiments with 50 or 100 feasible prices, not just 15. The average profit gain decreases slightly, but with $m = 100$ it is still a substantial 70%. We conjecture that Δ decreases because a larger number of actions and states necessitates more exploration to achieve

[43]For example, Tyagi (1999) shows that greater substitutability hinders collusion when demand is linear or concave but may either hinder or facilitate it when demand is convex.

the same amount of learning as in baseline model.

We have also considered the case of parameter ξ higher than 10%, but greater flexibility in price setting - below Bertrand-Nash or above monopoly - turns out to be immaterial. This is not surprising, given that the players never converge on these very low or very high prices.

6.9. *Asymmetric learning*

Going back to the baseline environment, we consider the case in which the two algorithms have different learning rates α, or different intensity of experimentation. Collusion appears to be robust to these changes. For example, when α_2 is set at 0.05 or 0.25 with α_1 constant at 0.15, the average profit gain is 82% in both cases. In both cases, the firm with lower α gains more, suggesting that setting $\alpha = 0.15$ still gives too much weight to new information.

We also halved and doubled the value of β for one algorithm only, keeping the other fixed at $\beta = 4 \times 10^{-6}$. In both cases, asymmetry reduces the average profit gain but only marginally, remaining well above 75%. The algorithm that explores more underperforms.

These highly preliminary results for situations in which different algorithms follow different learning and exploration strategies suggest that diversity among the algorithms does not significantly affect the degree of collusion.

7. CONCLUSION

We have shown that in stationary environments Q-learning pricing algorithms systematically learn to collude. Collusion tends to be partial and is sustained by punishment in case of defection. 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 environment in which they operate. 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 clearly ring a bell. They suggest that with the advent of Artificial Intelligence and algorithmic pricing, tacit collusion may become more prevalent, heightening the risk that tolerant antitrust policy may produce too many false negatives and so possibly calling for policy adjustments.

However, more research is needed to confirm the external validity of our findings. Three issues stand out: the realism of the economic environment, the speed of learning, and the

diversity of the competing algorithms. As for the first issue, we have considered a good many extensions of the baseline model, but all separately, the model thus remaining quite highly stylized. To produce a more realistic setting for analysis, one should perhaps posit a model with several firms, longer memory, stochastic demand and possibly also structural breaks.

Such more realistic environments may however defy the learning capacity of our simple, tabular Q-learners. It might therefore be necessary to use algorithms whose learning is more efficient - say, deep learning algorithms. This point is related to the second issue mentioned above, i.e. the speed of learning. Besides managing more complex environments, deep learning can indeed also speed the learning process. This is important, because the training of the algorithms cannot always be conducted entirely off-line, and in the short run experimentation is costly. On-the-job learning seems necessary, in particular, when the economic environment is changeable, or when the training environment does not exactly reflect the reality of the markets where the algorithms are eventually deployed.

One reason why training environments may not be fully realistic is that it is difficult to guess what specific algorithms competitors are using. In this respect, here we have restricted attention mostly to training in self-play mode. In reality, there are many different forms of reinforcement learning, and Q-learning algorithms themselves come in different varieties. It would therefore seem necessary to extend the analysis to the case of heterogeneous algorithms more systematically. Our robustness exercises in this direction have just scratched the surface of the problem.

Addressing these issues is clearly an important task for future work. But whatever may be done or left undone in the abstract, skeptics may always doubt that algorithms actually collude in the real world. Ultimately, the issue can only be decided by antitrust agencies and the courts. Unlike academic scholars, they can subpoena and extensively test the algorithms firms actually use, in environments that closely replicate the specific industry under investigation. What academic research can do is help make a preliminary assessment: that is, whether opening such investigations is a waste of resources, perhaps with the risk of many false positives, or instead may be necessary to fight collusion in the age of Artificial Intelligence. This paper is one contribution in this direction.

REFERENCES

[1] Andreoli-Versbach, P. and U. Franck, J. (2015). Econometric Evidence to Target Tacit Collusion in Oligopolistic Markets. *Journal of Competition Law and Economics*, **11** (2), 463–492.

[2] 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.

[3] 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.

[4] Barlo, M., Carmona, G. and Sabourian, H. (2016). Bounded memory Folk Theorem. *Journal of economic theory*, **163**, 728–774.

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

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

[7] Bergemann, D. and Välimäki, J. (2008). Bandit Problems. In S. N. Durlauf and L. E. Blume (eds.), *The New Palgrave Dictionary of Economics: Volume 1 – 8*, London: Palgrave Macmillan UK, pp. 336–340.

[8] 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.

[9] Börgers, T. and Sarin, R. (1997). Learning Through Reinforcement and Replicator Dynamics. *Journal of economic theory*, **77** (1), 1–14.

[10] 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, Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, pp. 1339–1349.

[11] Cooper, D. J. and Kühn, K.-U. (2014). Communication, Renegotiation, and the Scope for Collusion. *American Economic Journal: Microeconomics*, **6** (2), 247–278.

[12] Cross, J. G. (1973). A Stochastic Learning Model of Economic Behavior. *The quarterly journal of economics*, **87** (2), 239–266.

[13] Decarolis, F. and Rovigatti, G. (2018). From Mad Men to Maths Men: Concentration and Buyer Power in Online Advertising.

[14] Duffy, J. (2006). Agent-based models and human subject experiments. *Handbook of computational economics*.

[15] 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.

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

[17] — and — (2017). Artificial intelligence & collusion: When computers inhibit competition. *University of Illinois law review*, p. 1775.

[18] Fudenberg, D. and Levine, D. K. (2016). Whither Game Theory? Towards a Theory of Learning in Games. *The journal of economic perspectives*, **30** (4), 151–170.

[19] Gata, J. E. (2019). Controlling Algorithmic Collusion: Short Review of the Literature, Undecidability, and Alternative Approaches.

[20] Greenwald, A., Hall, K. and Serrano, R. (2003). Correlated Q-learning. In *ICML*, aaai.org,

vol. 3, pp. 242–249.

[21] Greenwald, A. R., Kephart, J. O. and Tesauro, G. J. (1999). Strategic Pricebot Dynamics. In *Proceedings of the 1st ACM Conference on Electronic Commerce*, EC '99, New York, NY, USA: ACM, pp. 58–67.

[22] Harrington, J. E., Jr (2017). Developing Competition Law for Collusion by Autonomous Price-Setting Agents.

[23] Ho, T. H., Camerer, C. F. and Chong, J.-K. (2007). Self-tuning experience weighted attraction learning in games. *Journal of economic theory*, **133** (1), 177–198.

[24] Hopkins, E. (2002). Two competing models of how people learn in games. *Econometrica*.

[25] Hu, J., Wellman, M. P. and Others (1998). Multiagent reinforcement learning: theoretical framework and an algorithm. In *ICML*, vol. 98, pp. 242–250.

[26] Huck, S., Normann, H.-T. and Oechssler, J. (2004). Two are few and four are many: number effects in experimental oligopolies. *Journal of economic behavior & organization*, **53** (4), 435–446.

[27] Kianercy, A. and Galstyan, A. (2012). Dynamics of Boltzmann Q learning in two-player two-action games. *Physical review. E, Statistical, nonlinear, and soft matter physics*, **85** (4 Pt 1), 041145.

[28] Klein, T. (2018). Assessing Autonomous Algorithmic Collusion: Q-Learning Under Short-Run Price Commitments.

[29] Könönen, V. (2006). Dynamic pricing based on asymmetric multiagent reinforcement learning. *International Journal of Intelligent Systems*, **21** (1), 73–98.

[30] Kühn, K.-U. and Tadelis, S. (2018). The Economics of Algorithmic Pricing: Is collusion really inevitable?

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

[32] Maskin, E. and Tirole, J. (1988). A Theory of Dynamic Oligopoly, II: Price Competition, Kinked Demand Curves, and Edgeworth Cycles. *Econometrica*, **56** (3), 571–599.

[33] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., 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.

[34] Rodrigues Gomes, E. and Kowalczyk, R. (2009). Dynamic Analysis of Multiagent Q-learning with E-greedy Exploration. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, New York, NY, USA: ACM, pp. 369–376.

[35] 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.

[36] Salcedo, B. (2015). Pricing Algorithms and Tacit Collusion.

[37] Shoham, Y., Powers, R., Grenager, T. and Others (2007). If multi-agent learning is the answer, what is the question? *Artificial intelligence*, **171** (7), 365–377.

[38] Silver, D., Huang, A., Maddison, C. J., 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.

[39] —, Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M.,

Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K. and Hassabis, D. (2018). A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, **362** (6419), 1140–1144.

[40] —, Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T. and Hassabis, D. (2017). Mastering the game of Go without human knowledge. *Nature*, **550** (7676), 354–359.

[41] Singh, N. and Vives, X. (1984). Price and quantity competition in a differentiated duopoly. *The Rand journal of economics*, pp. 546–554.

[42] Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

[43] Tampuu, A., Matiisen, T., Kodelja, D., Kuzovkin, I., Korjus, K., Aru, J., Aru, J. and Vicente, R. (2017). Multiagent cooperation and competition with deep reinforcement learning. *PloS one*, **12** (4), e0172395.

[44] Tesauro, G. and Kephart, J. O. (2002). Pricing in Agent Economies Using Multi-Agent Q-Learning. *Autonomous agents and multi-agent systems*, **5** (3), 289–304.

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

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

[47] Watkins, C. J. C. H. (1989). *Learning from delayed rewards*. Ph.D. thesis, King's College, Cambridge.

[48] — and Dayan, P. (1992). Q-learning. *Machine learning*, **8** (3), 279–292.

[49] Wunder, M., Littman, M. L. and Babes, M. (2010). Classes of multiagent q-learning dynamics with epsilon-greedy exploration. *Proceedings of the 27th International Conference on Machine Learning*.