Algorithmic Collusion with Coarse Memory

Massimiliano Furlan

July 2022

Chapter 1

Introduction

Firms competing in digital markets often delegate their pricing decisions to computer software. These algorithms can process large volumes of data in real-time, allowing firms to react quickly to changes in competitors' prices and market conditions. In the past, pricing algorithms used to follow simple rules of thumb. They were instructed, for example, to set prices within close range of those of competitors or to track the price set by a leading firm. With the recent advancements in artificial intelligence, new opportunities opened up for firms seeking to automate their pricing strategies. Self-learning algorithms, in particular, have proved to be extremely successful in a variety of different applications. For instance, Silver et al. (2017) developed an algorithm that achieved superhuman performances at the board game Go after less than a week of self-play training. Jumper et al. (2021) developed an algorithm able to accurately predict the shape of proteins, one of the most complex tasks in biology. More recently, Degrave et al. (2022) have used a self-learning algorithm for controlling nuclear fusion plasma in a tokamak. Compared to those applications, choosing prices to maximize profits seems to be a task self-learning algorithms can easily handle. This, together with the decline in the monetary cost of computing power, make self-learning algorithms at the very least attractive to firms.

Scholars and regulators have raised concerns about the implications of increased automation in online markets. Self-learning algorithms can develop pricing strategies from scratch without human assistance. Worries are that, given their impressive ability to solve complex problems, these algorithms might autonomously learn to collude. As proof of concept, Calvano et al. (2020) shows that even rudimental artificial intelligence algorithms can learn collusive strategies. In their experiment, they let two or more reinforcement learning agents interact in a simulated Bertrand model of competition over an infinite horizon. By only knowing a short history of prices and nothing else about the environment, algorithms are able to systematically coordinate on prices well above the competitive level. To sustain high prices in equilibrium, they learn strategies that entail punishments of unilateral defections. Without any external aid or communication abilities, algorithms are thus able to find textbook examples of cooperative solutions to the repeated pricing game.

In this thesis, I build on Calvano et al. (2020) and study the effect of limiting the algorithms' memory of past prices. With a coarse memory, algorithms cannot perfectly remember past prices and only have partial information on them. For instance, they may be only allowed to remember if the past price of their opponent was high, intermediate, or low. By introducing this limitation, I hinder their ability to condition actions on the history of prices; the main channel which allows them to develop collusive strategies. The memory of past prices is also the only information those algorithms possess about the environment in which they operate. In line with theoretical expectations, I find that without memory of their rival's past price algorithms learn to price competitively. However, partial limitations in their memory do not prevent algorithms from developing collusive strategies. Even with a minimal perception of their rival's past price, algorithms are able to punish downward deviations and sustain collusion. This highlights that reinforcement learning algorithms require very little information to find cooperative solutions to the pricing game.

The thesis is organized as follows. Chapter 2 briefly reviews the recent economic literature on algorithmic collusion. Chapter 3 introduces the class of algorithms I use in the experiments and the simulated economic environment in which they operate. In chapter Chapter 4, I replicate Calvano et al. (2020)'s experiment, first with algorithms having perfect memory and then having coarse memory of their rival's past price. The conclusions are summarized in Chapter 5.

Chapter 2

Related literature

There is a growing experimental literature showing reinforcement learning algorithms are able to solve standard economic games for cooperative equilibria. In one of the first contributions, computer scientists Waltman and Kaymak (2008) show that Q-learning agents playing in an infinitely repeated Cournot game often learn to set the output level below Cournot-Nash, thus pushing the price above competitive levels. Oddly, they find that algorithms do so even when they are myopic or have no memory of past actions; both cases in which collusion is theoretically unattainable in equilibrium. This suggests that what they observe is not real collusion but more likely the results of the algorithms' failure to optimize.

Rather than merely focusing on the simulations' outcomes, more recent studies have focused on the equilibrium behavior of algorithms. Calvano et al. (2020) are the first to analyze the strategies algorithms learn to sustain collusion. They find that, if given enough time, Q-learning algorithms learn to sustain supra-competitive prices using strategies that entail punishments of unilateral defections. They also show this result is robust to changes in the number of players, in the demand function, or to the introduction of cost asymmetries across firms. Klein (2021) shows that under a sequential price competition framework (Maskin and Tirole, 1988), Q-learning algorithms converge to supra-competitive Edgeworth price cycles. Following the same approach of Calvano et al. (2020), the author finds that also in this case algorithms learn to use punishment-reward schemes to deter defections from the collusive equilibria.

An open issue of algorithmic collusion was that Q-learning algorithms take millions of iterations to learn such collusive strategies. Based on this observation, skeptics sometimes argue it simply takes too long for algorithms to actually be able to collude in real markets (Schwalbe, 2018). More recent contributions to the literature, however, have proved that more sophisticated algorithms can learn reward-punishment schemes much quicker than the conventional Q-learning algorithm. Hettich (2021) shows that Deep Q-Network algorithms — an enhancement of the Q-learning algorithm — are able to learn collusive strategies three to four times quicker than Q-learning. Notably, the recent work of Frick (2022) shows actor-critic reinforcement learning algorithms can learn the same strategies even a hundred times faster than Q-learning. Frick (2022) is also the first to show algorithms can collude in a continuous state-action space, where the coordination problem is arguably more complex than in the discretized space imposed by Q-learning.

There is ample evidence of firms using algorithmic pricing software (Chen et al., 2016; Brown and MacKay, 2021), but the empirical evidence of algorithmic collusion in real markets is still limited. In one of the very few examples, Assad et al. (2020) find that the adoption of algorithmic pricing software in the German retail gasoline market had a negative impact on competition among stations. The authors show that when two competing nearby stations adopted algorithmic pricing software, their margins increased above competitive levels. In cases where only one of the stations or none of them delegated pricing decisions to algorithms, they observe no such change in the stations' margin. Notably, pricing algorithms seem to display behaviors consistent with Calvano et al. (2020)'s findings. Competing stations adopting algorithmic pricing are more likely to respond to price cuts from their rival with a sudden decrease in their price. For this reason, these stations were less likely than non-automated stations to undercut their rival, suggesting algorithms had learned it would be unprofitable.

Chapter 3

Experimental setting

3.1 Reinforcement learning

Reinforcement learning algorithms are a class of algorithms that learn from past experience with the goal of maximizing in the long term the expected total reward of a Markov decision process (Sutton and Barto, 2018). In each step $t = 0, 1, 2, 3, \ldots$ of a discrete-time Markov decision process, the agent observes the state of the world $s_t \in S$, where S is the set of possible states of the world, and takes action $a_t \in A(s_t)$, where $A(s)$ is the set of feasible actions in state s. As a consequence of action a_t in state s_t, the decisionmaker collects a reward r_t and the process transitions to a new state s_{t+1}, according to a probability distribution $F_t(r_t, s_{t+1}|s_t, a_t)$.

Formally, the agent wants to maximize

$$\sum_{t=1}^{\infty} \delta^{t-1} E_F[r_t|s_t, a_t], \tag{3.1}$$

where $\delta \in [0, 1)$ is the discount factor. The optimal policy for this problem can be found using Bellman's principle of optimality

$$v(s_t) = \max_{a_t \in A(s_t)} \left\{ E_F[r_t + \delta v\left(s_{t+1}\right)|s_t, a_t] \right\}. \tag{3.2}$$

Reinforcement learning algorithms are methods to learn the value function $v(s)$ when the underlying process $F_t(r_t, s_{t+1}|s_t, a_t)$ is unknown.[1] Following Calvano et al. (2020), I focus on the Q-Learning algorithm in particular. This algorithm applies to finite Markov decision processes where the set of actions A is state invariant. Rather than directly estimating the value function, the Q-learning algorithm seeks to estimate the value of taking action a in state s and following the optimal policy thereafter. Such value is represented by the action-value function, denoted by $q(s, a)$, and defined by the functional equation

$$q(s_t, a_t) = E_F\left[r_t + \delta \max_{a \in A} q(s_{t+1}, a) \;\middle|\; s_t, a_t\right]. \tag{3.3}$$

[1] In the computer science literature, $v(s)$ is known as the optimal value function. This is opposed to generic value functions, which define the value of each state under suboptimal policies. For the sake of simplicity, I abstract from this complexity and defer the reader to Sutton and Barto (2018, Chap. 3) for a thorough analysis on policies and value functions in Markov decision processes.

With no knowledge of $F_t(r_t, s_{t+1}|s_t, a_t)$, the agent learns the action-value function from experience. The estimate it holds for $q(s_t, a_t)$ in t, denoted $Q_t(s_t, a_t)$, is updated in each period according to a convex combination of the previous period estimate and the new information obtained as a consequence of its action. Formally,

$$Q_{t+1}(s_t, a_t) \leftarrow (1-\alpha)Q_t(s_t, a_t) + \alpha\Big[r_t + \delta \max_{a\in A} Q_t(s_{t+1}, a)\Big], \tag{3.4}$$

where $\alpha \in (0, 1]$ is the learning rate, $Q_0(s, a)$ is defined arbitrarily for each state-action pair and s_0 is arbitrarily initialized. Given that S and A are finite, the estimated action-value function $Q_t(s, a)$ can conveniently be represented by an $|S| \times |A|$ matrix, called the Q-matrix. Each of its entries, called Q-values, contain the estimate in t for the action-value function evaluated at a given state-action pair.

To maximize the amount of rewards in the long term, the agent must trade off exploitation with exploration. The agent is said to exploit current knowledge when, in each state, it chooses the best-known action according to its current estimate for the action-value function. Formally, $a^*_t(s) = \arg\max_a Q_t(s, a)$. This policy is the greedy policy according to $Q_t(s, a)$. However, to improve its estimate and thus future action selections, the agent must also engage in exploration. In fact, all pairs (s, a) must be visited multiple times to reliably learn the long-term expected reward of each action in given states. The simplest policy to balance the trade-off between exploration and exploitation is called epsilon greedy policy. Under this policy, the agent randomizes uniformly across its set of actions with probability ε and chooses the greedy action in t with probability $1 - \varepsilon$.

Watkins and Dayan (1992) show that for stationary finite Markov decision processes, Q-learning converges to the action-value function if each state-action pair is visited infinitely often. An epsilon greedy policy satisfies this condition and therefore grants $Q(s, a) \to q(s, a)$ for all $(s, a) \in S\times A$ as $t \to \infty$. However, since it entails a fixed probability of exploration, the epsilon-greedy policy will never converge to an optimal policy. Even when eventually $Q_t(s, a) \approx q(s, a)$, the agent will continue to select actions randomly with a probability of ε. An improvement of this exploration method, known as decayed epsilon greedy, is to let ε decay toward zero in time. This can still guarantee convergence of $Q_t(s, a)$ to the action-value function, and as long as $\varepsilon_t \to 0$ in the limit for $t \to \infty$, decayed epsilon greedy policies converge to an optimal policy too.

3.2 Economic environment

The economic environment replicates the one in Calvano et al. (2020). There are n symmetric firms, each producing a single good at a constant marginal cost c. Goods are differentiated, and the demand for the good produced by firm i is

$$q_i(p_i, \boldsymbol{p}_{-i}) = \frac{\exp[(a-p_i)/\mu]}{\sum_{j=1}^{n} \exp[(a-p_j)/\mu] + \exp(a_0/\mu)}, \tag{3.5}$$

where p_i is the price set by firm i and $\boldsymbol{p}_{-i}$ is the set of prices set by firm i's rivals. The parameter a captures goods' quality and is identical across all n goods produced by firms. Good 0 is an outside good and has quality a_0. Product differentiation is determined by the parameter μ, with goods being perfect substitutes in the limit for $\mu \to \infty$. The profit function of firm i is

$$\pi_i(p_i, \boldsymbol{p}_{-i}) = (p_i - c)\, q_i(p_i, \boldsymbol{p}_{-i}). \tag{3.6}$$

Firms compete à la Bertrand in an infinitely repeated game, and each wants to maximize the discounted value of its expected stream of profits

$$\sum_{t=1}^{\infty} \delta_i^{t-1}\, E\left[\pi_i(p_{it}, \boldsymbol{p}_{-it})\right], \tag{3.7}$$

where $\delta_i \in [0, 1)$ denotes the discount factor for firm i. To do so, firms delegate pricing decisions to independent Q-learning algorithms (or agents). In each period t, the algorithms choose prices simultaneously and non-cooperatively, conditional on their memory of past prices.

3.3 State-action space

To implement Q-learning in a continuous state-action space, it is necessary to discretize the environment, such as to have a finite A and S. I proceed as follows.

Let p^N and p^M be the symmetric Bertrand-Nash and monopoly price, respectively.[2] I define the set of actions A as a set of m equally spaced prices $p \in \mathbb{R}^+$ in the interval $[p^N - \varphi,\, p^M + \varphi]$, where $\varphi = (p^M - p^N)/(m-3)$ and $m \geq 4$. This definition is such that both equilibrium prices are included in the set of actions, being respectively the second-lowest and the second-greatest prices in the set. For convenience, I index the elements of A in monotone increasing order. Then $A = \{p^1, p^2, \ldots, p^{m-1}, p^m\}$, where $p^1 > p^2 > \ldots > p^{m-1} > p^m$, and therefore $p^2 = p^N$ and $p^{m-1} = p^M$.

As in Calvano et al. (2020), the state of the environment in t describes the prices set by agents in the preceding k periods. Formally, $s_t = \{\boldsymbol{p}_{t-1}, \ldots, \boldsymbol{p}_{t-k}\}$, where $\boldsymbol{p}_t = (p_{1t}, \ldots, p_{nt})$ is the set of prices chosen in period t by agents. The set of all states then is obtained as the cartesian product $S = A^{nk}$ and has cardinality $|S| = m^{nk}$. Note that bounding agents' memory to k periods is necessary for S to be finite in size, as an unbounded memory would imply $|S|$ to grow each period. For simplicity, I focus on the case where agents have a one-period long memory, that is $k = 1$ and $s_t = \boldsymbol{p}_{t-1}$.

3.4 Uncertainty and non-stationarity

The probability for agent i to collect profit $\pi_i(\boldsymbol{p}_t)$ and transition to state $s_{t+1} = \boldsymbol{p}_t$, as a consequence of its action p_{it} in state $s_t = \boldsymbol{p}_{t-1}$, is denoted $F_{it}(\pi_i(\boldsymbol{p}_t), s_{t+1} | s_t, p_{it})$. Note that given p_{it}, this probability depends only on the realization of $\boldsymbol{p}_{-it}$, as it uniquely determines $\boldsymbol{p}_t = (p_{it}, \boldsymbol{p}_{-it})$, and therefore both s_{t+1} and $\pi_i(\boldsymbol{p}_t)$ with it. This means that agents must deal with the strategic uncertainty deriving from their opponents' pricing decisions. Each agent, with its own pricing decisions, contributes to generating uncertainty for all others. As an agent learns and updates its policy through exploration and exploitation, it directly affects the learning process of its rivals, which in turn affects its own, and so on ad infinitum. For this

[2]See Section A.1 in the Appendix for their derivation.

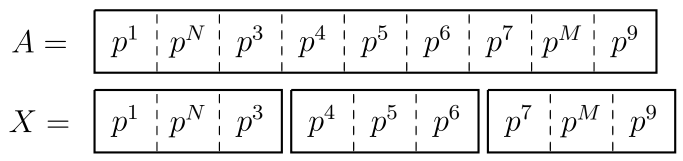


Figure 3.1: An example of admissible partition of A with $m = 9$: X partitions A in three equally sized subsets and the agent can only remember if a given price was low $p \in \{p^1, p^N, p^3\}$, intermediate $p \in \{p^4, p^5, p^6\}$ or high $p \in \{p^7, p^M, p^9\}$.

reason, F_{it} is inherently non-stationary — as it evolves with the policies of agent i's rivals — and there is no theoretical guarantee for Q-learning to converge to a fixed policy (let alone to an optimal one). Nonetheless, convergence can always be assessed ex-post to the experiment, when it has either realized or not realized.

3.5 Coarse memory

I assume algorithms cannot perfectly remember past prices and, for this reason, have partial observability of the state of the environment. Let a memory $X = \{x_1, \dots, x_H\}$ be a partition of the set of prices A. I define a memory function $x : A \to X$ as a function that returns $x \in X$ if and only if $p \in x$. In this way, an agent can only remember $p \in x$ rather than directly p.[3] I restrict my focus on memories X that satisfy $p < p'$ for all $p \in x_h$ and $p' \in x_{h+1}$, with $h \in \{1, \dots, H-1\}$. That is, I only consider partitions that split the set of prices without altering its monotone order — see an example in Figure 3.1. Depending on the cardinality of A, there are in total $2^{|A|-1}$ different admissible memories which satisfy this property.

To determine whether one partition induces a more precise memory of prices $p \in A$ than another, one can pairwise compare their fineness (Marschak and Radner, 1972). Formally, a memory X is finer (or less coarse) than another memory X', if $X \neq X'$ and X is a refinement of X'. Denoting by $x(p)$ and $x'(p)$ respectively their associated memory functions, this implies

$$x(p) \subseteq x'(p) \quad \text{for all} \quad p \in A.$$

The finest memory gives perfect memory of past prices by partitioning the set of prices in singletons. Formally, perfect memory of p is $X = \{\{p\} : p \in A\}$, which implies $x(p) = \{p\}$ for all $p \in A$. In contrast, the coarsest memory gives no memory of prices, formally, $X = \{A\}$ and therefore $x(p) = \{A\}$ for all $p \in A$. Between these two extremes, memories differ in the number of subsets in which they partition A and in how they distribute the precision of memory across prices.

Fineness only provides an incomplete ordering of partitions, as it does not allow for ranking any two memories — see the examples in Figure 3.2. A memory $X = \{x^1, \dots, x^H\}$ has $2^{|A|-H} - 1$ possible refinements and is the refinement of $2^{H-1} - 1$ other partitions. In total, a memory X can be compared (including itself) with $2^{|A|-H} + 2^{H-1} - 1 \leq 2^{|A|-1}$ other memories, with equality holding only for $H \in \{1, |A|\}$. Derivations are shown in Section A.2 in the Appendix.

I assume agents may remember prices with different precision depending on who sets them. For example, an agent might have a perfect memory of its past prices but a coarse memory

[3]The memory function borrows from the concept of information structure developed by Marschak and Radner (1972).

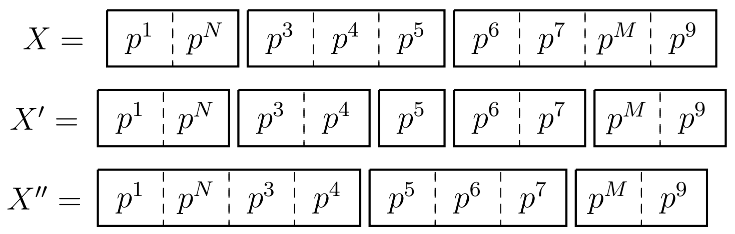

$X =$ p^1 p^N | p^3 p^4 p^5 | p^6 p^7 p^M p^9
$X' =$ p^1 p^N | p^3 p^4 | p^5 | p^6 p^7 | p^M p^9
$X'' =$ p^1 p^N p^3 p^4 | p^5 p^6 p^7 | p^M p^9

Figure 3.2: X' is finer than X and X'', but X'' is not fineness-comparable with X. Both X and X'' partition A in three disjoint subsets, however, X is more precise than X'' on low prices while the opposite is true for high prices.

of its opponents' prices. I denote by X_{ij} the memory that agent i has on the prices set by agent j and call the sequence of its memories $\{X_{ij}\}_{j=1}^n$ memory structure. Formally, a memory structure induces a Euclidean space $\times_{j=1}^n X_{ij}$ that partitions the space of prices A^n. Similarly to the case of a single price, agent i, rather than remembering directly $\boldsymbol{p}_t \in A^n$ can only remember $\boldsymbol{p}_t \in \boldsymbol{x}_{it}$, where $\boldsymbol{x}_{it} \in \times_{j=1}^n X_{ij}$.

An imperfect memory of past prices implies partial observability of the state of the environment. In state $s_t = \boldsymbol{p}_{t-1}$, agent i observes, according to its memory structure, $o_{it} = \boldsymbol{x}_{it-1}$. Because I assume one-period memory, the observation of the state and the observation of the past period's prices coincide. Therefore, the set of possible state-observations is $O_i = \times_{j=1}^n X_{ij}$ and has cardinality $|O_i| = \prod_{j=1}^n |X_{ij}|$. Figure 3.3 shows how a memory structure partitions the state space in a case with two agents and six prices.

The concept of fineness extends also to memory structures. A memory structure $\{X_{ij}\}_{j=1}^n$ is finer than another memory structure $\{X'_{ij}\}_{j=1}^n$ if and only if $\{X_{ij}\}_{j=1}^n \neq \{X'_{ij}\}_{j=1}^n$ and X_{ij} refines X'_{ij} for all $j \in \{1, \dots, n\}$. The finest memory structure is perfect memory of all past prices, formally $X_{ij} = \{\{p\} : p \in A\}$ for all $j \in \{1, \dots, n\}$. With a perfect one-period memory, $o_{it} = s_t = \boldsymbol{p}_{t-1}$ and $O_i = S$. The coarsest memory structure instead is no memory of past prices, that is, $X_{ij} = \{A\}$ for all $j \in \{1, \dots, n\}$. In this case, identically to imposing zero memory, $o_{it} = S = A^n$ for all $s_t \in S$ and $|O_i| = 1$.

Depending on the algorithm's memory structure, the same state of the environment s results in different observations o. In implementing Q-learning, I assume algorithms treat those observations as their subjective representation of the state of the environment. Then, the Q-matrix of agent i has dimension $|O_i| \times |A|$ and the agent's policy, rather than mapping states into actions, maps state-observation into actions.

3.6 Learning and exploration

In each update of $Q(o, a)$, the learning rate α determines the weight recent rewards have relative to past rewards. A high learning rate implies old rewards are quickly forgotten and replaced with more recent information on the (o, a) pair. With a small learning rate instead, past rewards continue to influence the estimate for longer periods. For each experiment, I consider an interval of 50 equally spaced points in the interval $[0.025, 0.25]$. At the lower bound $\alpha = 0.025$, it takes roughly 180 updates for a past reward to weigh less than 1%, while at the upper bound $\alpha = 0.25$ rewards weigh less than 1% after only 16 updates.

Algorithms use an ε-greedy policy with decaying ε. In particular, I let exploration decrease exponentially according to $\varepsilon_t = \beta\, \varepsilon_{t-1}$, with $\beta \in [0, 1)$ and $\varepsilon_0 = 1$. When β is close to one,

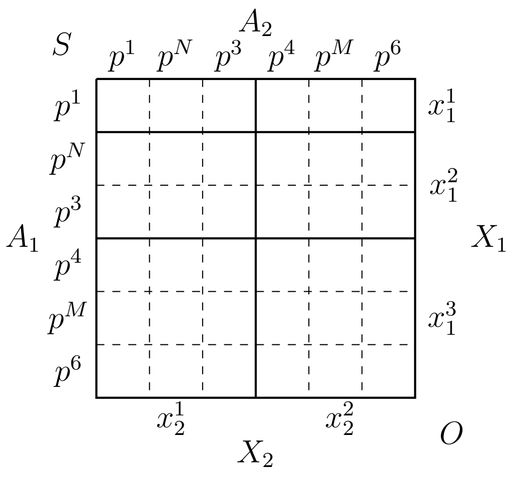


Figure 3.3: The memory structure $\{X_1, X_2\}$ induces a Euclidean space $O = X_1 \times X_2$ (solid lines) that partitions the state space $S = A_1 \times A_2$ (dashed lines). The agent can only remember if the price set by agent 1 was very low $p \in \{p^1\}$, somewhat intermediate $p \in \{p^N, p^3\}$ or high $p \in \{p^4, p^M, p^6\}$ and if the price of agent 2 was low $p \in \{p^1, p^N, p^3\}$ or high $p \in \{p^4, p^M, p^6\}$. For example, rather than observing precisely $s = (p^3, p^M)$ or $s = (p^N, p^6)$, the agent in both cases only observes $o = (x_1^2, x_2^2)$.

exploration decays slowly, and agents explore extensively. To better grasp how the magnitude of β affects exploration, I map its value into the expected number of times each (o, a) pair gets visited thanks to agents' exploration over an infinite horizon. A given pair is reached in period t due only to agents' exploration if $o \in O$ is the result of every agent exploring in period $t-1$, and the agent takes action $a \in A$ by exploring in period t. Over an infinite horizon, a pair (o, a) is visited only due to random exploration a number of times equal to

$$\nu(o) = \frac{|o|}{|S| \cdot |A|} \frac{\beta}{1-\beta^{n+1}}. \tag{3.8}$$

Its derivation is shown in Section A.3 in the Appendix. Note that state-observations encompassing more states are visited more often, namely $\nu(o) > \nu(o')$ if and only if $|o| > |o'|$.

To measure the intensity of exploration, I take the average $\nu(o)$ over all state-observations $o \in O$, that is, $\bar{\nu} = |O|^{-1} \sum_{o \in O} \nu(o)$. Fixing $\bar{\nu}$ rather than β allows to net out differences in outcomes that are only due to variations in the intensity of exploration. In fact, coarser memories induce smaller $|O|$, so that agents need less exploration to maintain the same intensity of exploration $\bar{\nu}$. In each experiment, I consider a set of 50 values for $\bar{\nu}$ in the interval $[3, 66]$. The implied values for β will depend on the particular memory structure of the agents.

The choice for $\bar{\nu}$ and α must ensure outcomes are not driven by the initialization of the Q-matrix. In practice, α and $\bar{\nu}$ need to be large enough to allow for the initial values of the Q-matrix to be reliably replaced with actual estimates. I consider exploration and learning to be sufficient when exploration-only updates imply in expectation the initial value of all pairs (o, a) to weight less than $1/3$ in the limit for $t \to \infty$. Note that this does not mean pairs will retain on average one-third of their initial value at the end of the experiment. In fact, because agents update their Q-matrix in every period regardless of exploration, most pairs will retain only an infinitesimal fraction of their initial value.

3.7 Convergence rule

The interest of the analysis is to study the behavior of algorithms once they have converged to some sort of equilibrium. Similarly to Calvano et al. (2020), I consider convergence to be reached when the greedy policy of every agent does not change for 20×10^4 periods. That is, convergence is reached in period $\bar{t}$ if $\arg\max_a Q_{i\bar{t}}(o, a) = \arg\max_a Q_{i\bar{t}-k}(o, a)$ for all $i \in \{1, \dots, n\}$ and $k \in \{1, \dots, 20 \times 10^4\}$. However, as previously said, there are no guarantees for interacting Q-learning agents to converge to a fixed policy. For this reason, I let each session continue for at most 520×10^4 periods, after which I shut the experiment and deem convergence as not reached. In any case, in all experiments I focus on, convergence is always reached before this threshold. I count the number of periods required to reach convergence as $\bar{t} - 20 \times 10^4$, excluding the period in which policies remained constant.

Algorithms converge to price cycles in which they loop over a finite set of prices. Most cycles have a length of one period — that is, algorithms charge a constant price — or of two periods — where algorithms alternate two prices. Longer price cycles are progressively less frequent.

3.8 Implementation

For any initial set of hyperparameters, I run 500 independent sessions in which separate Q-learning algorithms interact with each other until convergence or the maximum number of periods is reached. Outcomes of converged sessions are then analyzed and eventually aggregated in summary statistics. All experiments and their analysis are carried out using JuliaLang (Bezanson et al., 2017). I release the code under the GNU Affero General Public License v3.0.[4]

I use the same hyperparameters' configuration as in the baseline experiment in Calvano et al. (2020) and consider $n = 2$ symmetric firms producing differentiated products of quality $a = 2$ at a constant marginal cost $c = 1$. The outside good has quality $a_0 = 0$, and the intensity of product differentiation is characterized by the parameter $\mu = 1/4$. Firms' algorithms have a one-period memory and can choose among $m = 15$ prices, with the second-lowest and second-greatest prices being respectively the Bertrand-Nash and Monopoly prices, $p^N \approx 1.473$ and $p^M \approx 1.925$.[5] Firms discount profits each period by a factor $\delta = 0.95$.

The Q-matrix of agent i is initialized with values corresponding to the expected discounted infinite stream of profits firm i would get from each action if its rivals were to randomize uniformly across their set of actions. That is,

$$Q_{i0}(o, a) = \frac{\sum_{\boldsymbol{a}_{-i} \in A^{n-1}} \pi_i(a_i, \boldsymbol{a}_{-i})}{|A|^{n-1}(1-\delta)} \qquad \text{for all } (o, a) \in O_i \times A. \tag{3.9}$$

The initial state $s_0 = \boldsymbol{p}_0$ is randomly drawn from the set S, which translates in each agent i observing a separate state-observation $o_{i0} = \boldsymbol{x}_{i0}$. Note that, given $n = 2$, $k = 1$ and $|A| = 15$, there are in total $|S| = 225$ states. The number of state-osbervations each agent has, instead, depends on its particular memory structure.

[4]https://github.com/massimilianofurlangit/algorithmic_pricing

[5]In the baseline experiment of Calvano et al. (2020) the set of prices is a set of 15 equally spaced points in the interval $[p^N - 0.1(p^M - p^N), p^M + 0.1(p^M - p^N)]$. This is very similar to the discretization of prices I use, as it roughly corresponds to having one price below the Nash-Bertrand price and one price above the monopoly price.

Chapter 4

Experiments

I reproduce the baseline experiment in Calvano et al. (2020), in which agents have perfect one-period memory, and discuss its main results. I use this as a benchmark to understand the effect of coarse memory on pricing algorithms that interact with each other. In studying coarse memory, I focus on cases in which algorithms can perfectly remember their past price but have coarse memory on the one set by their rival.

The degree of collusion is measured using a normalized measure of profits, called profit gain (Calvano et al., 2020),

$$\Delta(\pi) = \frac{\pi - \pi^N}{\pi^M - \pi^N}. \tag{4.1}$$

The average profit gain that firms realize together on the equilibrium path — that is, at the price cycle they have converged to — is simply denoted by Δ and referred to as average profit gain. This measure takes value one when algorithms converge to full collusion and equals zero when firms price competitively in equilibrium.

To assess the algorithms' learning process, I compute in each session the theoretical action-value function upon convergence and compare it with the estimate the agent holds for it. More precisely, I measure the average absolute percentage error of the estimated action-value function with respect to its true value, the relative frequency of suboptimal action selections that are due to this error, and the eventual foregone payoff of not having taken the optimal action. I also keep track of the share of sessions that converge to a Nash equilibrium — where both agents, by always choosing optimal actions, best respond to each other.[1] I consider these metrics only at states-observations on the equilibrium path. Off the path of equilibrium, agents generally hold worse estimates for the action-value function and often fail to best reply to their rival. Despite lacking subgame perfection, Calvano et al. (2020) show that agents' off-path behavior still displays clear and recognizable patterns. This will be discussed at length in the following section.

4.1 Perfect memory

In Calvano et al. (2020) agents have a perfect one-period memory. That is, every agent i has a memory structure $\{X_{ij}\}_{j=1}^{n}$ where $X_{ij} = \{\{p\} : p \in A\}$ for all $j \in \{1, \ldots, n\}$. This memory structure partitions the state space in singletons, implying states and state-observations coin-

[1]Section A.4 in the Appendix provides more details on how these metrics are computed.

			Cycle length				
	Overall	Nash	1	2	3	4	> 4
Frequency		0.497	0.625	0.25	0.062	0.031	0.017
Periods to converge ($\times 10^4$)	164.92	148.05	154.4	167.8	205.4	203.61	231.91
	(46.05)	(34.35)	(37.55)	(45.66)	(52.45)	(60.55)	(50.52)
Profit gain	0.856	0.86	0.874	0.847	0.795	0.809	0.741
	(0.109)	(0.1)	(0.105)	(0.104)	(0.113)	(0.083)	(0.1)
Q error on path (%)	0.019	0.01	0.013	0.021	0.034	0.054	0.054
	(0.017)	(0.01)	(0.011)	(0.013)	(0.018)	(0.022)	(0.017)
Nash equilibria (%)	0.497	–	0.664	0.304	0.097	0.0	0.0
Suboptimal actions on path (%)	0.233	–	0.182	0.26	0.39	0.387	0.569
Foregone payoff on path (%)	0.005	–	0.006	0.004	0.006	0.004	0.007

Table 4.1: Perfect one-period memory. Values are averages across 1000 sessions.

cide. Simply put, with perfect memory algorithms can directly condition their actions on past prices.

4.1.1 Outcomes

To better understand the underlying mechanisms of algorithmic collusion, it is useful to first focus on a single combination of values for α and $\bar{\nu}$. I consider in particular $\alpha = 0.15$ and $\bar{\nu} = 25$, for which I run 1000 independent sessions. Outcomes are summarized in Table 4.1.

On average, it takes 165×10^4 periods for algorithms to stabilize and reach converge.[2] Figure 4.1 shows the whole evolution of the firms' profits in six independent sessions. Profit gains start approximately at 0.5 because agents randomize uniformly among a set of prices skewed above Bertrand-Nash. As exploration decays and strategic decisions become prevalent, agents tend to exploit their rival by undercutting it when it raises its price. During this phase, firms' profits oscillate frequently, and the average profit gain remains on average below 0.4. Eventually, agents stop rivaling and start cooperating, with the two combined obtaining in equilibrium $\Delta = 0.856$ on average across sessions.

On the equilibrium path, agents' estimated action-value function has a mean absolute percentage error of 1.9% across sessions.[3] This leads agents to take suboptimal actions — by not best responding to their rival's strategy — approximately 23% of the time. Nonetheless, these "mistakes" result on average in a negligible 0.5% loss in terms of foregone discounted profits over an infinite horizon. Overall, roughly half of the sessions converge to a Nash equilibrium, where both agents dynamically best respond to each other in all the state-observations they reach in equilibrium.

It is important to note, however, that it is not entirely fair to deem suboptimal action selections as "mistakes". When studying optimal actions at convergence, agents' policies are fixed by definition. However, when an agent learns its policy, it also accounts for the fact that its actions influence the policy of its rival. An agent's action in equilibrium might then figure

[2]As a reference point, consider that in this setting $\bar{\nu} = 25$ implies $\beta \approx 0.99999605$, meaning that it takes approximately 120×10^4 periods for agents' individual probability of exploration to fall below 1%.

[3]Off the path of equilibrium, the mean absolute percentage error is 10%.

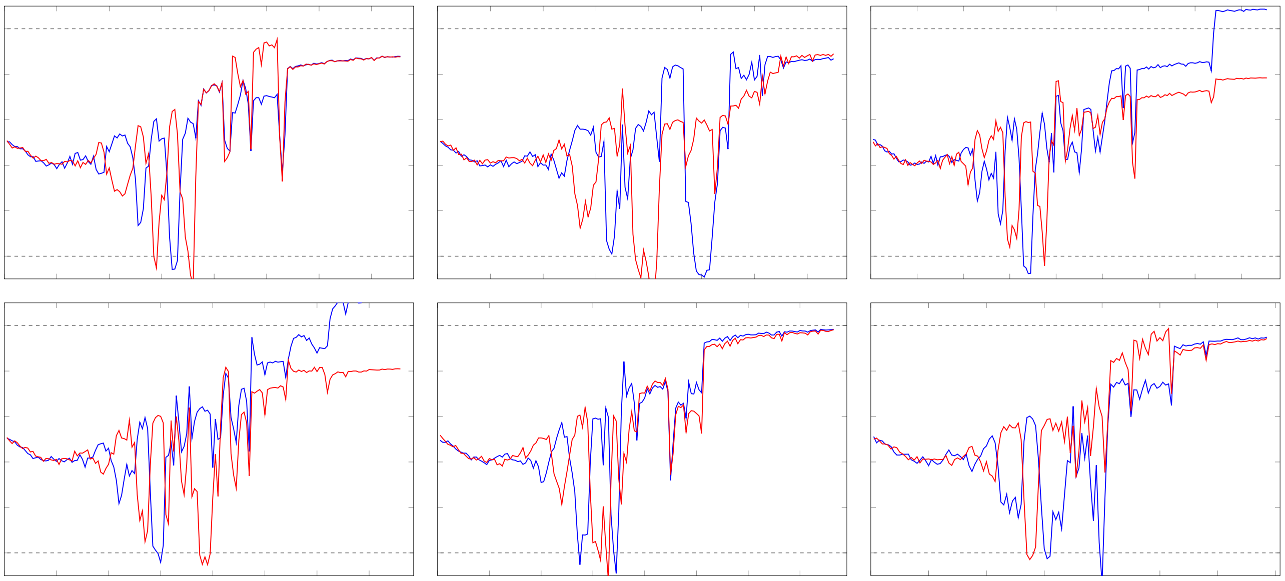


Figure 4.1: Agents' profit in six randomly selected sessions. Starting from the top, the dotted lines on the y-axis correspond to monopoly profit and Bertrand-Nash profit, respectively. Each tick on the x-axis correspond to 20×10^4 periods. Plotted values are averages over 10^4 periods. The plots interrupt when the condition for convergence is met.

as suboptimal only because it learned that, by choosing what looks to be the optimal action, it would ultimately end up adversely affecting its rival's policy. Hence, for this reason, it prefers choosing a different action — possibly the optimal action in the fully dynamic environment. Still, the fraction of suboptimal action selections at convergence provides, in most cases, a good picture of how much sense the behavior of agents makes.

4.1.2 Collusive strategy: punishments

Collusion is not simply the case of firms pricing their goods above competitive levels. Firms pricing above the Bertrand-Nash price have a unilateral incentive to deviate and undercut their rival, appropriating a larger market share and increasing their profit. For collusion to be an equilibrium, it must be sustained by a credible threat of punishment which would make unilateral defections unprofitable in the long run. In case high prices are not sustained by such collusive strategy, they are solely the result of the firms' failure to optimize, rather than representing collusion. The critical question that Calvano et al. (2020) ask is: why do algorithms charge high prices in equilibrium? Is it because they fail to learn the optimal response to their rival's strategy, or do they find it unprofitable to undercut their rival?

Part of the answer to this question can be addressed by noting that an action is suboptimal if there exists a deviation from it that grants the agent higher discounted profits over an infinite horizon. As previously discussed, roughly 23% of the action selections in equilibrium are suboptimal, so that at least one profitable deviation from them exists. However, because these deviations generally imply negligible gains, algorithms may have learned that exploiting them adversely affects the other agent's policy. Also, the mere existence of a single profitable deviation does not imply all deviations are profitable. Indeed, as it will be shown later, most of them are not. Most importantly, in the remaining 77% of action selections in equilibrium, algorithms best respond to their rival, and any possible deviation from the supra-competitive

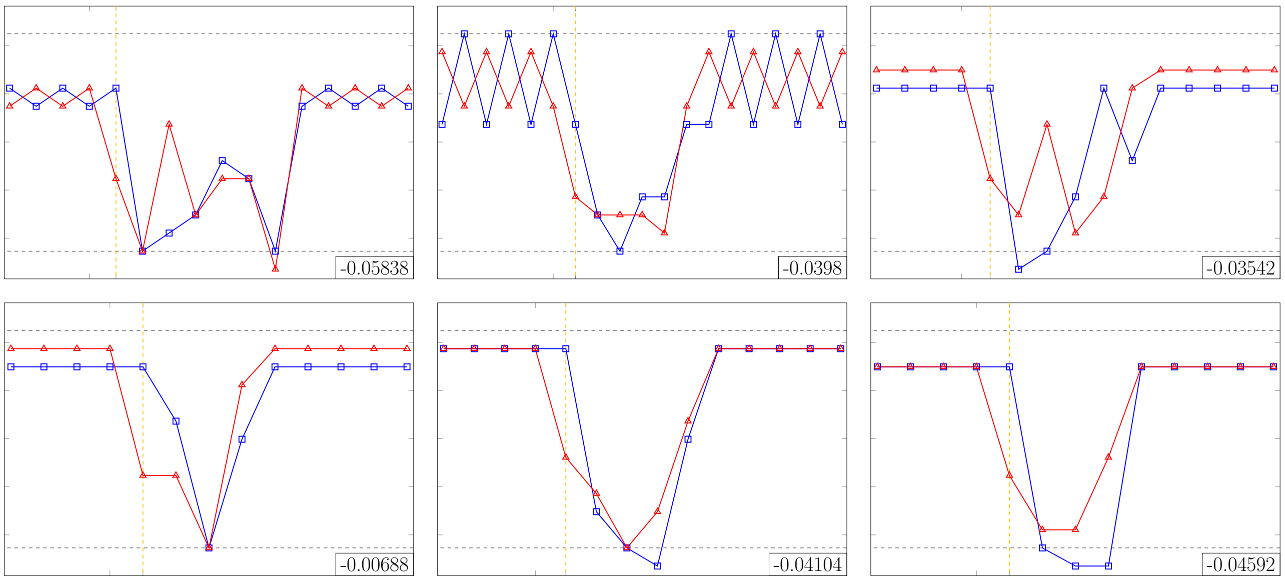
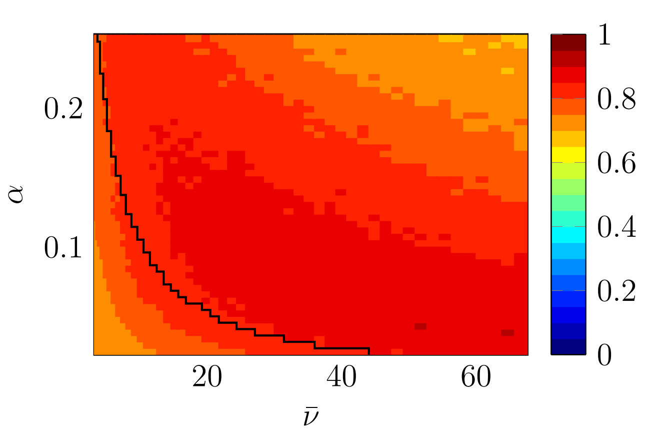


Figure 4.2: Impulse responses in six randomly selected sessions. Starting from the top, the horizontal dashed lines correspond to the monopoly price $p^M \approx 1.925$ and the Bertrand-Nash price $p^N \approx 1.473$, respectively. The deviating agent (in red) deviates to the static best response of its rival's price in correspondence with the vertical orange dashed line. On the bottom-right corner of each plot it is shown the percent gain in terms of discounted profits over an infinite horizon accruing to the deviator. In all six cases, the deviation is unprofitable.

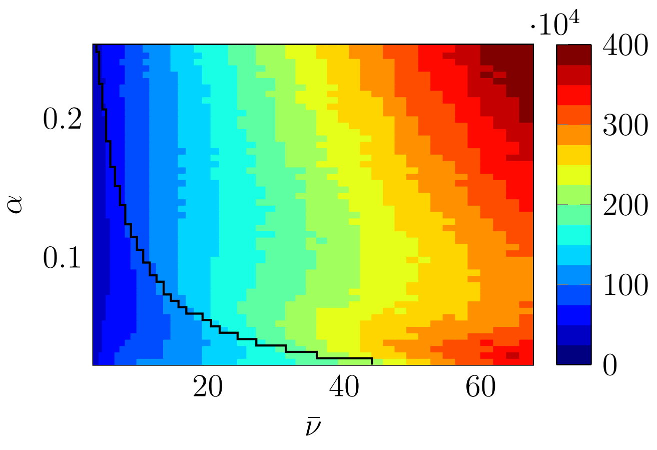
equilibrium is unprofitable. This suggests that the high profit gains observed in equilibrium, rather than representing mistakes in the algorithms' optimization process, result from their ability to sustain high prices via punishments that make unilateral defections unprofitable.

To better grasp how algorithms sustain high prices, I follow in Calvano et al. (2020)'s footsteps and analyze algorithms' response to exogenously introduced price deviations. Once algorithms have converged, I force one of the two to deviate for one period to the static best response of its rival's price — that is, to the price that maximizes the instantaneous gain from defection — and then let them play according to the policies they have learned. I repeat this process in all sessions, for each agent and period of the cycle they converged to. Figure 4.2 shows a single of these impulse responses for each of the same six sessions I also shown in Figure 4.1. Impulse responses are heterogeneous across sessions, but despite their differences, they display a recognizable pattern. An even clearer pattern can be seen by averaging them as is shown in Figure B.1 in the Appendix. Algorithms respond to unilateral defections by lowering their price for a finite number of periods.[4] In 70% of the cases, the punishment entails prices below or equal to the Bertrand-Nash price. Most punishments last between 3 and 7 periods, after which agents almost always return to the price cycle they had previously converged to. Overall, unilateral price deviations to the static best response are unprofitable in 92% of the cases, with the deviating agent losing on average 2.7% in terms of discounted profits over an infinite horizon. Equivalent results are obtained also for deviations other than the static best response as is shown in Table B.1 and Table B.2 in the Appendix. Overall, more than 96% of all possible downward deviation from the equilibrium price are unprofitable, with the deviating

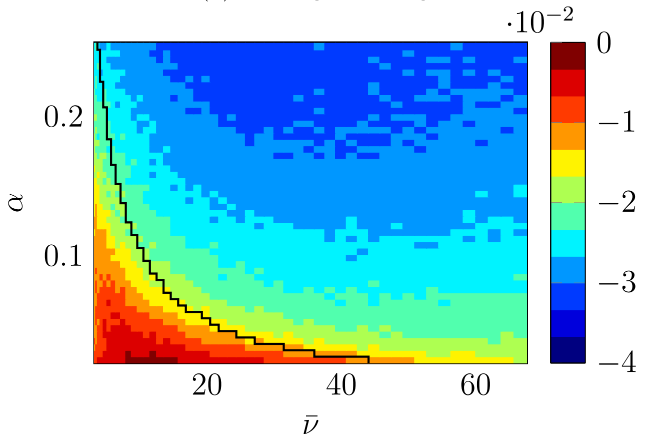
[4]The fact that punishments have finite length is a consequence of the trial and error process by which algorithms learn: if punishments were not forgiving, cooperation would quickly be jeopardized by the agents' random exploration.



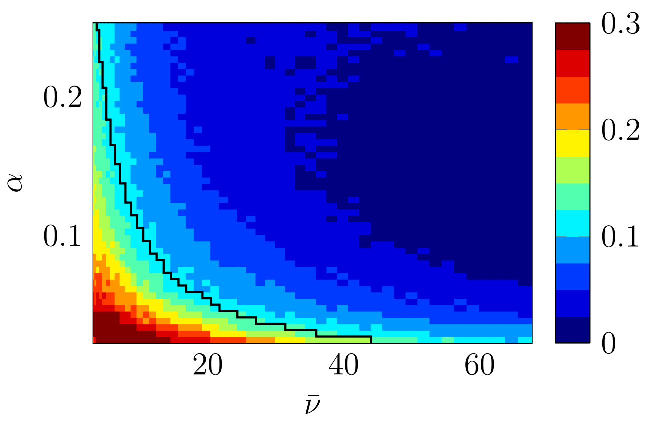
(a) Average profit gain



(b) Average periods to converge



(c) Average percent gain by deviating to the static best response



(d) Average fraction of profitable downward deviations

Figure 4.3: The grids show, for different combinations of α and $\bar{\nu}$, (a) the average profit gain, (b) the average number of periods to reach convergence, (c) the average percentage gain or loss implied by deviating to the static best response and (d) the average fraction of profitable downward deviations from the equilibrium price, across 500 sessions. Outcomes to the left of the black segmented line are heavily influenced by the initialization of the Q-matrix and are therefore unreliable.

agent losing on average 3% in terms of discounted profits.

4.1.3 Hyperparameter grids

I repeat the same experiment for different values of α and $\bar{\nu}$, running 500 sessions for each combination of parameters. The hyperparameter grids in Figure 4.3 summarize (a) the profit gain, (b) the number of periods it takes algorithms to reach convergence, (c) the percentage gain or loss implied by deviating to the static best response, and (d) the fraction of profitable downward deviations from the equilibrium price, each averaged over 500 sessions. The black segmented line delimits to its right the values of α and $\bar{\nu}$ for which exploration-only updates grant in expectation the initial value of all pairs (o, a) to weight less than $1/3$ in the limit for $t \to \infty$. Outcomes to the left of the segmented line are strongly dependent on the initialization of the Q-matrices and are therefore unreliable. For this reason, in describing the results, I will not refer to them when not explicitly mentioned.

Algorithms systematically obtain a high profit gain in equilibrium, with a minimal average Δ of 0.68 up to a maximum of 0.9. Generally, the average profit gain is higher when α is small and $\bar{\nu}$ is high, while it is lower when both α and $\bar{\nu}$ are high. The number of periods required to reach convergence naturally increases with the intensity of exploration $\bar{\nu}$ and has a nontrivial relation with α.

For values of α and $\bar{\nu}$ to the right of the segmented line, algorithms sustain the high profit gains via punishment of unilateral defections. Figure 4.3(d) shows that most downward deviations from the equilibrium price are made unprofitable by the rival's punishments. When exploration is extensive and α is high, more than 98% of all possible downward deviations are unprofitable. In fact, in the vast majority of cases, algorithms converge to prices that best respond to their opponent's strategy and from which there are no profitable deviations at all — see Figure B.2(a) in the Appendix.

Punishments are harsher, inflicting larger losses to the defector, when agents explore more or α is high. This can be seen in Figure 4.3(c) for the case of deviations to the static best response, but similar patterns also apply to other deviations. Figure B.2(b) in the Appendix displays an almost identical layout, but rather than analyzing a single deviation, it shows the average loss of deviating from the equilibrium price to any price below it.

For values of α and $\bar{\nu}$ that are too low to reliably replace the initial value of the agents' Q-matrices, punishments are either absent or ineffective, and thus profitable deviations are more frequent. This empathizes that the positive profit gains algorithms obtain with those parameter configurations are not due to the use of collusive strategies — which they did not have enough time to learn — but are simply a distortion due to the initialization of the agents' Q-matrices.

4.2 Coarse memory of the rival's past price

I consider now the case where algorithms can perfectly remember their own past price but have a coarse memory on those set by their opponent. For simplicity, I assume every agent has the same memory X on its rival's past price. Formally, each agent i has memory structure $\{X_{ij}\}_{j=1}^{n}$ having $X_{ij} = X$ for all $j \in \{1, \ldots, n\}$ with $j \neq i$ and $X_{ii} = \{\{p\} : p \in A\}$. For the rest, the configuration of the experiment remains unchanged.

I use the shorthand notation $(|x^1| - \ldots - |x^H|)$ to uniquely identify a memory $X = \{x^1, \ldots, x^H\}$. For instance, rather than writing $X = \{\{p^1\}, \{p^N, p^3\}, \{p^4, p^M, p^6\}\}$, I simply write $X = (1\text{–}2\text{–}3)$. In this subsection, I use X_{ij} to indicate the memory each agent has on the prices set by its rival, taking for granted that algorithms can perfectly remember their own prices.

With $|A| = 15$ there are in total 2^{14} possible different memories. Given such a large number, I necessarily restrict the focus to a small number of them. I begin by studying memories (3–3–3–3–3) and (5–5–5), which give agents different but homogeneous memory precision on their rival's past price. Then, I analyze the limiting case in which algorithms have no memory of their rival's past price. I choose these memories because their associated memory structures partition the state space into equally sized subsets, so that $\nu(o)$ is constant and equals $\bar{\nu}$ for all $o \in O$. This allows for studying coarse memory while netting out its effect on the distribution of $\nu(o)$. Note, however, that (3–3–3–3–3) and (5–5–5) are not fineness comparable. Eventually, I also consider memories that are either coarser than (3–3–3–3–3) or than (5–5–5). In those cases, coarse memory affects the distribution of $\nu(o)$, which will not be uniform anymore.

			Cycle length				
	Overall	Nash	1	2	3	4	> 4
Frequency		0.261	0.376	0.367	0.124	0.055	0.078
Periods to converge ($\times 10^4$)	64.75	54.39	58.58	66.19	73.64	72.11	68.36
	(19.14)	(17.25)	(18.67)	(18.9)	(17.36)	(17.19)	(17.15)
Profit gain	0.802	0.82	0.838	0.802	0.754	0.764	0.732
	(0.14)	(0.13)	(0.128)	(0.139)	(0.143)	(0.144)	(0.139)
Q error on path (%)	0.031	0.02	0.022	0.03	0.045	0.046	0.051
	(0.024)	(0.02)	(0.02)	(0.02)	(0.027)	(0.021)	(0.032)
Nash equilibria (%)	0.261	–	0.58	0.112	0.016	0.0	0.0
Suboptimal actions on path (%)	0.411	–	0.242	0.456	0.543	0.586	0.676
Foregone payoff on path (%)	0.006	–	0.009	0.005	0.006	0.005	0.009

Table 4.2: Summary statistics. Memory $X_{ij} = (3–3–3–3–3)$ of the rival's past price.

			Cycle length				
	Overall	Nash	1	2	3	4	> 4
Frequency		0.144	0.266	0.353	0.184	0.093	0.104
Periods to converge ($\times 10^4$)	41.01	35.46	38.42	42.43	41.89	40.91	41.32
	(11.87)	(11.72)	(11.66)	(11.99)	(11.97)	(10.9)	(11.79)
Profit gain	0.742	0.79	0.783	0.752	0.7	0.72	0.695
	(0.181)	(0.19)	(0.19)	(0.172)	(0.181)	(0.167)	(0.176)
Q error on path (%)	0.042	0.03	0.03	0.042	0.047	0.053	0.057
	(0.031)	(0.02)	(0.027)	(0.03)	(0.028)	(0.028)	(0.038)
Nash equilibria (%)	0.144	–	0.466	0.045	0.022	0.0	0.0
Suboptimal actions on path (%)	0.562	–	0.342	0.592	0.643	0.69	0.768
Foregone payoff on path (%)	0.008	–	0.012	0.007	0.007	0.007	0.01

Table 4.3: Summary statistics. Memory $X_{ij} = (5–5–5)$ of the rival's past price.

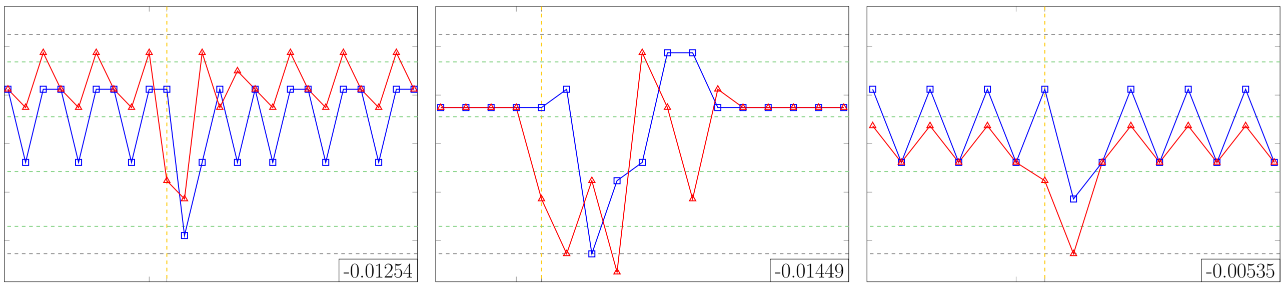
4.2.1 Homogeneous coarse memory

I consider the cases where agents have memory (3–3–3–3–3) and (5–5–5) on their rival's past price. As in the previous section, I first focus on a single hyperparameter configuration having $\alpha = 0.15$ and $\bar{\nu} = 25$, for which I run 1000 independent sessions. The outcomes of the two experiments are are summarized in Table 4.2 and Table 4.3, respectively.

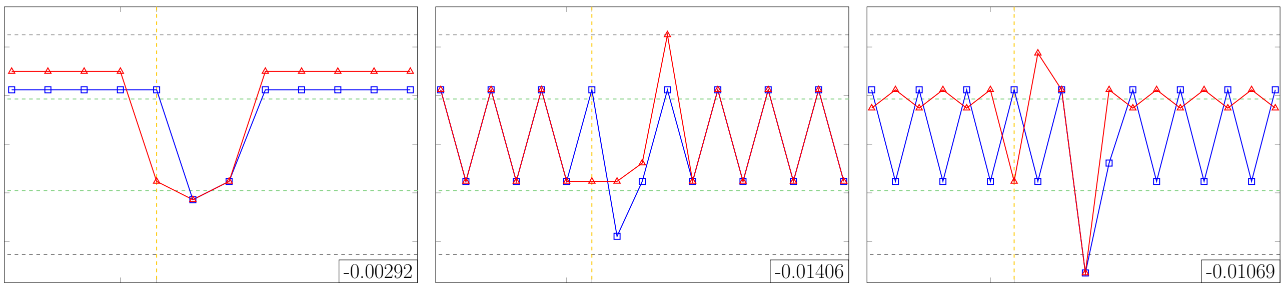
Introducing to algorithms coarse memories (3–3–3–3–3) and (5–5–5) on their rival's past price slightly reduces the average profit gain they obtain in equilibrium. Compared to when they have perfect memory, the average profit gain decreases from $\Delta = 0.856$, to $\Delta = 0.802$ and $\Delta = 0.742$, respectively. The average number of periods algorithms take to reach convergence is considerably lower, decreasing from 165×10^4 periods to 65×10^4 periods with memory (3–3–3–3–3) and to 41×10^4 periods with memory (5–5–5). This is due to the fact that a coarse memory reduces the number of state-observations $|O|$ and, therefore, also the size of the agents' Q-matrix.[5] When $|O|$ is smaller, less exploration is needed to keep $\bar{\nu}$ fixed, implying quicker convergence.

With memories (3–3–3–3–3) and (5–5–5) algorithms converge less often to constant pricing, and price cycles in equilibrium are frequently of length two or longer. The fraction of suboptimal

[5]With memory (3–3–3–3–3) of the rival's past price $|O| = 75$ and with memory (5–5–5) instead $|O| = 45$.



(a) Memory (3–3–3–3–3)



(b) Memory (5–5–5)

Figure 4.4: Impulse responses in three randomly selected sessions for the case of memories (3–3–3–3–3) and (5–5–5), respectively. The horizontal green dashed lines describe the granularity of the agents' memory on their rival's past price. The deviating agent (in red) deviates to the static best response of its rival's price in correspondence with the vertical orange dashed line.

action selections in equilibrium — from which a profitable deviation exists — is higher compared to the case of perfect memory, and the foregone payoff of not taking the optimal action is slightly higher as well. Both get worse when price cycles are longer, but similar conclusions apply even when outcomes are compared across sessions converged to cycles of the same length, indicating that suboptimal action selections are not simply due to differences in cycle length.

A coarse memory makes it more difficult for algorithms to track and punish deviations. Nonetheless, with memory (3–3–3–3–3) and (5–5–5) on their rival's past price, algorithms still manage to punish most of them. When algorithms have memory (3–3–3–3–3), punishments make 88% of unilateral price deviations to the static best response unprofitable, with an average 2.6% loss in terms of discounted profits over an infinite horizon. Most punishments last between 2 and 6 periods and are on average slightly shorter than in the case of perfect memory. With memory (5–5–5), punishments are shorter, lasting in most cases less than 4 periods. Because they are shorter, they are also slightly less effective, with deviations to the static best response being unprofitable 77% of the time and implying an average 2.1% loss. Figure 4.4 shows some impulse responses for unilateral deviations to the static best response in the two cases of memory at hand. While being more chaotic and visibly shorter, punishments can still be recognized. In all six cases, deviations end up being unprofitable.

Note that by having coarse memory of their rival's past prices, algorithms may not be able to detect all deviations. For example, with memory (5–5–5) on rival's prices, a one-time deviation from p^M to p^{12} or from p^{10} to p^6 would go unnoticed since both prices induce the same memory $x \in X$ to the rival agent. However, following the exogenous deviation, the deviant agent — who always remembers its deviation — may not return immediately to cooperation and select prices that its opponent can detect. When this happens, even defections that are not directly

Deviation price	Equilibrium price									
	1.963	1.925	1.887	1.850	1.812	1.774	1.737	1.699	1.661	1.624
1.925	0.618									
1.887	0.569	0.696								
1.850	0.736	0.854	0.909							
1.812	0.785	0.822	0.900	0.625						
1.774	0.743	0.789	0.836	0.554	0.617					
1.737	0.792	0.874	0.893	0.887	0.906	0.934				
1.699	0.812	0.846	0.874	0.881	0.878	0.923	0.619			
1.661	0.785	0.822	0.870	0.854	0.869	0.913	0.494	0.618		
1.624	0.840	0.866	0.925	0.917	0.911	0.924	0.923	0.878	0.902	
1.586	0.826	0.879	0.941	0.911	0.913	0.924	0.923	0.895	0.916	0.695
1.548	0.847	0.891	0.929	0.926	0.917	0.929	0.923	0.881	0.896	0.656
1.511	0.882	0.895	0.945	0.917	0.933	0.951	0.932	0.925	0.914	0.84
1.473	0.889	0.915	0.961	0.940	0.946	0.956	0.946	0.930	0.933	0.855
1.435	0.903	0.960	0.970	0.940	0.955	0.963	0.949	0.953	0.937	0.87
Frequency	0.033	0.057	0.100	0.077	0.126	0.205	0.081	0.098	0.119	0.030

Table 4.4: Frequency of unprofitable deviations for different equilibrium and deviation prices, when agents have memory (3–3–3–3–3) on their rival's past price. Horizontal lines separate values for undetectable deviations, which lie above the horizontal lines, from values where deviations are detectable, which lie below the horizontal lines.

Deviation price	Equilibrium price									
	1.963	1.925	1.887	1.850	1.812	1.774	1.737	1.699	1.661	1.624
1.925	0.599									
1.887	0.437	0.551								
1.850	0.444	0.502	0.561							
1.812	0.317	0.429	0.439	0.564						
1.774	0.754	0.773	0.835	0.869	0.891					
1.737	0.732	0.753	0.816	0.848	0.870	0.583				
1.699	0.697	0.733	0.793	0.832	0.859	0.523	0.543			
1.661	0.754	0.745	0.782	0.841	0.856	0.489	0.524	0.500		
1.624	0.711	0.745	0.777	0.812	0.839	0.447	0.440	0.461	0.555	
1.586	0.817	0.850	0.874	0.901	0.911	0.835	0.849	0.846	0.840	0.849
1.548	0.824	0.842	0.874	0.902	0.914	0.831	0.826	0.846	0.835	0.85
1.511	0.859	0.862	0.899	0.916	0.920	0.831	0.846	0.876	0.864	0.885
1.473	0.887	0.887	0.905	0.932	0.930	0.865	0.888	0.888	0.871	0.889
1.435	0.901	0.870	0.919	0.951	0.946	0.910	0.910	0.899	0.881	0.914
Frequency	0.027	0.047	0.068	0.109	0.195	0.051	0.068	0.083	0.117	0.142

Table 4.5: Frequency of unprofitable deviations for different equilibrium and deviation prices, when agents have memory (5–5–5) on their rival's past price. Horizontal lines separate values for undetectable deviations, which lie above the horizontal lines, from values where deviations are detectable, which lie below the horizontal lines.

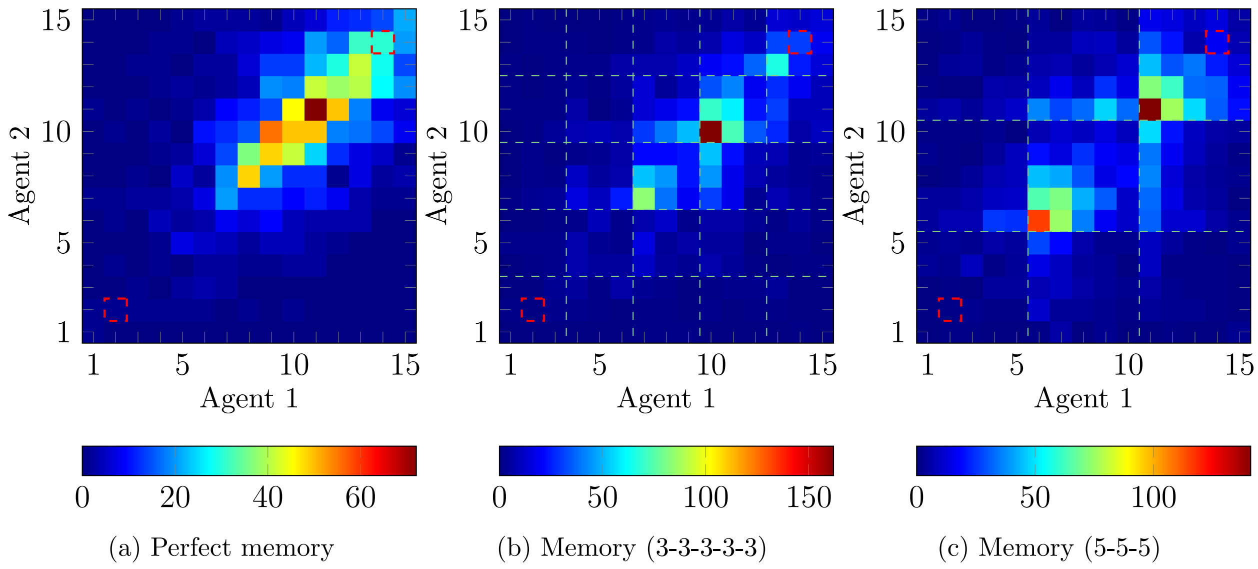


Figure 4.5: Tuple of prices chosen at convergence in 1000 separate experiments. Values on the axes indicate the price position in the set A. The Bertrand-Nash and the monopoly outcomes are highlighted with a red dashed square. The green vertical dashed lines describe agent 1's memory of its rival's prices and the horizontal ones describe the same for agent 2.

detectable may end up being punished. In Table 4.4 and Table 4.5, I highlight deviations that are not directly detectable with horizontal lines: undetectable downward deviations from the equilibrium price lay above the horizontal line of its corresponding column, whereas deviations that can always be detected lay below it. It can be seen that the share of unprofitable deviations is lower — more deviations go unpunished — when they cannot be directly detected. With memory (3–3–3–3–3), only 61% of deviations are unprofitable when not directly observable, compared to 91% of them when they are. Similarly, with memory (5–5–5) of the rival's past price, when deviations are not directly detectable only 47% of them is unprofitable against 87% of them when they are. Table B.3 and Table B.4 in the Appendix show the average deviation gain (loss) by deviating from different equilibrium prices to any possible downward deviation. With both memories, when deviations cannot be directly detected, the percentage loss in terms of discounted profits due to the eventual punishment is not statistically different from zero. This indicates punishments are largely ineffective when algorithms choose prices from which deviations are not directly observable.

Figure 4.5 shows how a coarse memory on the rival's past price affects the distribution of prices chosen by agents in equilibrium. When algorithms have coarse memory of their rival's past price, they tend to select prices from which downward deviations can be better monitored. For instance, with memory (5–5–5), algorithms often set the price equal to p^6 or p^{11} so that any downward deviation can be promptly detected by their rival. Algorithms choose those prices more frequently because, as discussed, they are easier to sustain in equilibrium. On the contrary, it is difficult for agents to sustain high prices from which deviations may eventually go unpunished. Algorithms' strategies, therefore, naturally tend toward prices from which deviations are better monitored. This is supported by a strong correlation between the frequency with which a price is chosen in equilibrium and the average percentage loss implied by deviating downward from that price — see Figure B.3 in the Appendix. Once excluded the Nash price, from which downward deviations are arguably irrational, this correlation amounts

			Cycle length				
	Overall	Nash	1	2	3	4	> 4
Frequency		0.444	0.565	0.238	0.067	0.037	0.093
Periods to converge ($\times 10^4$)	17.3	18.67	18.35	16.82	15.63	14.57	14.46
	(3.33)	(2.26)	(2.61)	(3.4)	(3.39)	(4.11)	(3.68)
Profit gain	0.078	0.0	0.019	0.103	0.157	0.258	0.246
	(0.157)	(0.02)	(0.097)	(0.186)	(0.162)	(0.161)	(0.154)
Q error on path (%)	0.059	0.05	0.059	0.057	0.054	0.072	0.065
	(0.045)	(0.03)	(0.038)	(0.055)	(0.038)	(0.046)	(0.057)
Nash equilibria (%)	0.444	–	0.777	0.021	0.0	0.0	0.0
Suboptimal actions on path (%)	0.345	–	0.128	0.499	0.647	0.807	0.863
Foregone payoff on path (%)	0.006	–	0.007	0.006	0.005	0.007	0.006

Table 4.6: Summary statistics. No memory on the rival's past price.

to 68% in the case of memory (3–3–3–3–3) on the past price of the rival and to 65% when agents have memory (5–5–5). This correlation is even higher in the case of perfect memory, in which case it amounts to 92%.

4.2.2 No memory of the rival's past price

With coarse memories (3–3–3–3–3) and (5–5–5) on the rival's past price, algorithms can still punish most defections and are therefore able to collude. A natural question that arises is what do algorithms do when they are completely precluded from the ability to remember the past price of their opponent? In that case, all deviations are undetectable, and retaliation is impossible. With no ability to condition their actions on their rival's past price, canonical collusion should theoretically be unfeasible.

I run 1000 session in which agents have no memory of the past price set by their rival but have perfect one-period memory on their own price. Formally, the memory structure of agent i' is defined as $\{X_{ij}\}_{j=1}^{n}$ with $X_{ij} = \{A\}$ for $j \neq i$ and $X_{ii} = \{\{p\} : p \in A\}$. With this memory, agents can only condition actions on their own past price: in each period, they decide whether to raise, lower, or keep the same price as the previous periods. They cannot instead in any way respond to their rival's actions. The experiment outcomes are summarized in Table 4.6.

As was expected, with no memory of their rival's past price, agents cannot detect and punish deviations and therefore are unable to sustain collusion. In 42.5% of the session algorithms converge to the static Bertrand-Nash equilibrium, obtaining $\Delta = 0.0$. In 84% of the sessions at least one of the two agents selects the Bertrand-Nash price in equilibrium. On average, the profit gain algorithms obtain in equilibrium is $\Delta = 0.078$. Given the reduced size of the agent's Q-matrices, convergence is reached quickly, taking on average only 17.3×10^4 periods.[6] Nonetheless, as it will be shown in the following subsection, results are robust also to more extensive agents' exploration.

Most sessions converge to price cycles lasting one or two periods, with longer cycles becoming gradually less frequent. Compared to the cases of coarse memory (3–3–3–3–3) and (5–5–5), the

[6]With no memory of the rival's past price and perfect memory of its own price, there are only $|O| = 15$ state-observations, one for each price in A. Therefore, the agents' Q-matrices are 15 times smaller than in the case of perfect memory, having dimension 15×15 rather than 225×15.

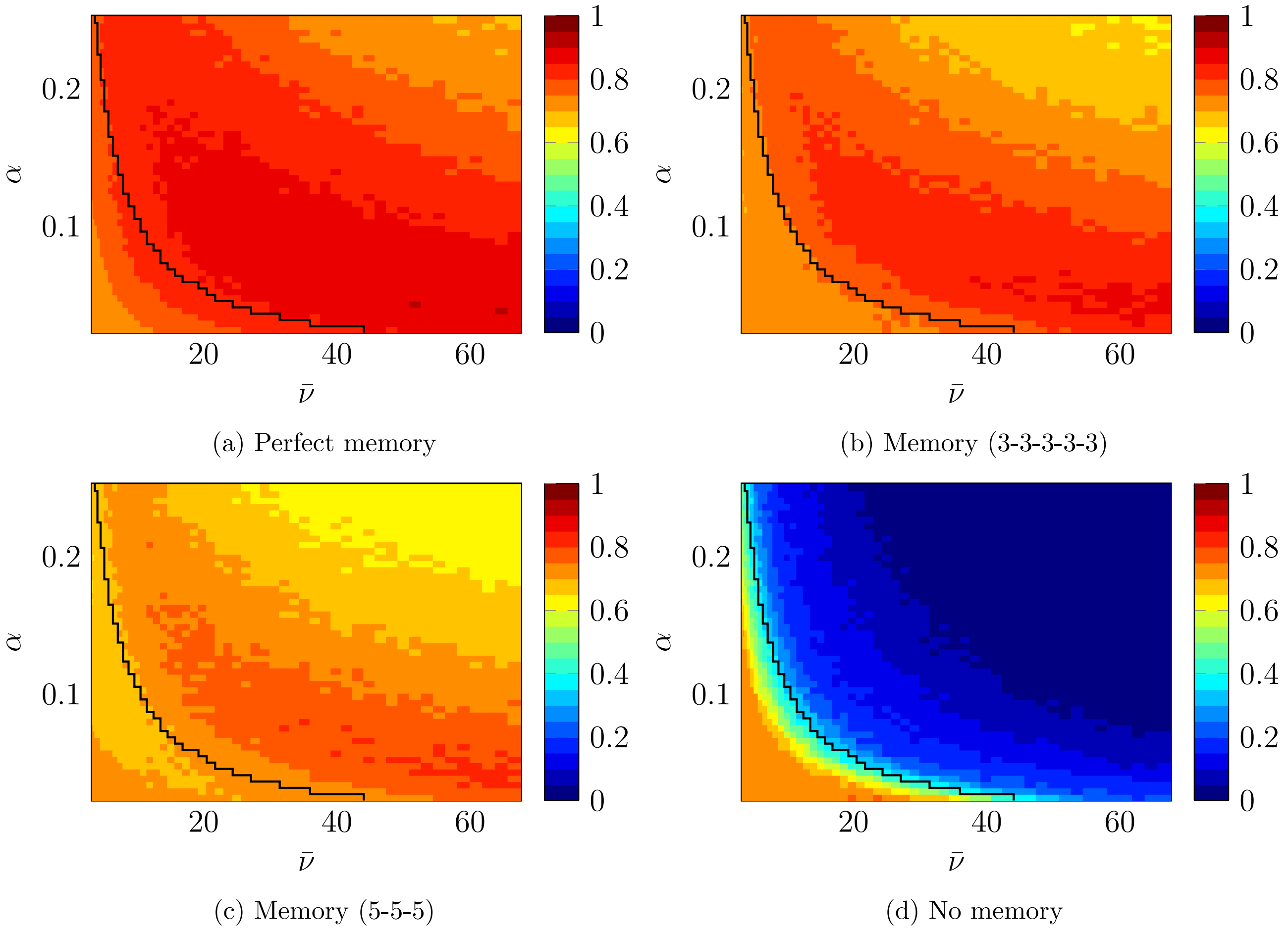


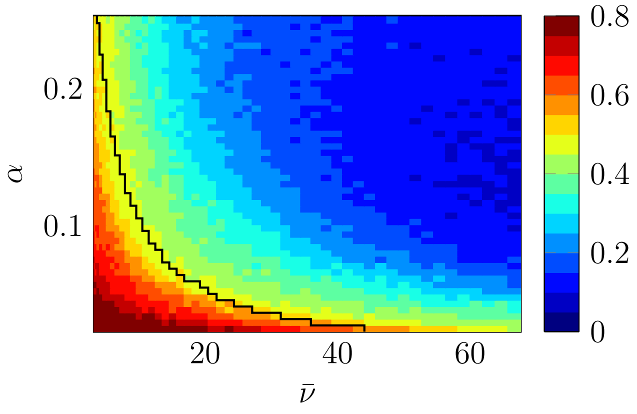
Figure 4.6: Average profit gain.

fraction of suboptimal action selections in equilibrium is lower and Nash equilibria are more frequent. This is particularly evident when algorithms converge to constant pricing. When instead algorithms converge to longer price cycles, suboptimal action selections progressively become more common and Nash equilibria less frequent or absent. Suboptimal action selections are also associated with higher average profit gains, which, however, are clearly not sustained through punishments of defections and are instead the result of the algorithms' failure to properly optimize.

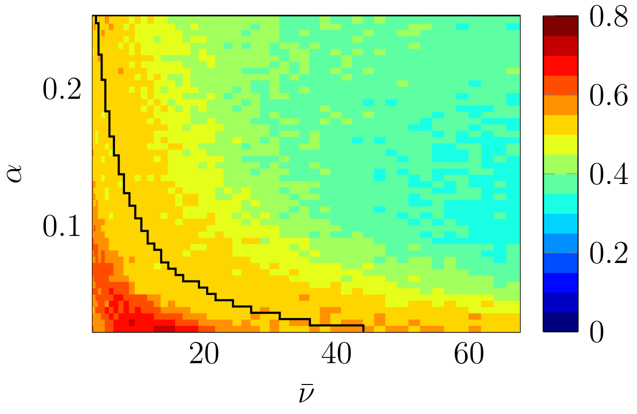
4.2.3 Hyperparameter grids

To ensure results are robust to changes in hyperparameters, I repeat the experiments for different values of α and $\bar{\nu}$, running 500 sessions each. The hyperparameter grids in Figure 4.6 show for different values α and $\bar{\nu}$ the average profit gain algorithms obtain on the case of (a) perfect memory, (b) memory (3–3–3–3–3), (c) memory (5–5–5) and (d) no memory on their rival's past price. In the same way, grids in Figure B.6 shows the average fraction of suboptimal action selections in equilibrium. Figures from B.4 to B.7 in the Appendix show, in order, the average number of periods it takes algorithms to reach convergence, the average length of price cycles in equilibrium, the average fraction of profitable downward deviations in equilibrium, and the average gain (loss) implied by deviating downward from the equilibrium price.

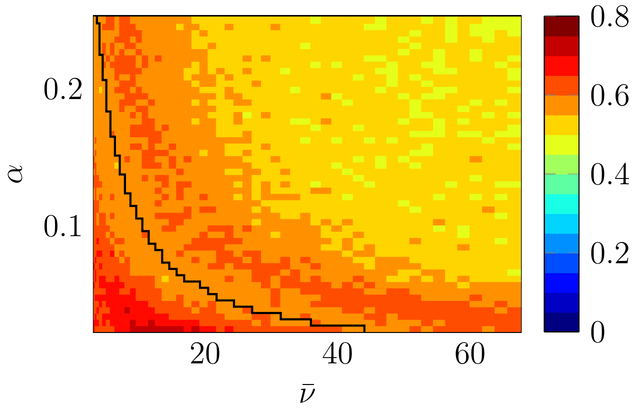
Results appear largely robust to changes in exploration and in the learning rate. When



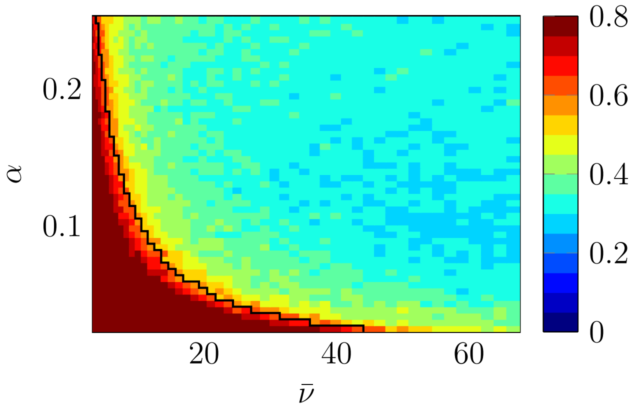
(a) Perfect memory



(b) Memory (3–3–3–3–3)



(c) Memory (5–5–5)



(d) No memory

Figure 4.7: Average fraction of suboptimal action selections in equilibrium.

algorithms have coarse memories (3–3–3–3–3) and (5–5–5), the average profit gain they achieve in equilibrium is lower but still considerable — see Figure 4.6. In the first case, the average profit gain ranges from a minimum of 0.64 to 0.87, while in the second case from 0.6 to 0.82. For all combinations of α and $\bar{\nu}$ to the north-east of the segmented line, the average profit gain is higher with perfect memory than it is with memory (3–3–3–3–3) and is higher with memory (3–3–3–3–3) than it is with memory (5–5–5). Algorithms realize higher profit gains when exploration is extensive and learning slow.

As already mentioned, algorithms with a coarse memory converge more quickly because the agents' Q-matrices are smaller, and less exploration is needed to maintain the intensity of exploration $\bar{\nu}$ fixed. This remains true regardless of the initial choice of parameters — see Figure B.4 in the Appendix. A coarse memory also tends to increase the length of price cycles at convergence — see Figure B.5 in the Appendix. In particular, cycles are longer when algorithms exploration is more limited, suggesting that further exploration might help algorithms reach more stable equilibria.

Confirming previous results, when algorithms have no memory of their rival's past price, they are unable to collude. For values of α and $\bar{\nu}$ to the right of the black segmented line, the average profit gain algorithms achieve in equilibrium is almost always lower than 0.3 and most often below 0.1. When the learning rate is high and exploration extensive, algorithms coverage to the static Bertrand-Nash equilibrium more often. For values of α and $\bar{\nu}$ to the left of the segmented line, the high profit gains are purely due to the initialization of the Q-matrix distorting the final outcome.

The fraction of suboptimal action selections in equilibrium is noticeably higher in the case of memories (3–3–3–3–3) and (5–5–5), indicating that algorithms are more likely not to punish all possible downward deviations. Compared to when algorithms have perfect memory, punishments are also less harsh — see Figure B.7 in the Appendix. Nonetheless, coarse memory hardly prevents algorithms from learning reward-punishment schemes. With both memory (3–3–3–3–3) and (5–5–5), deviations from the equilibrium price remain in most cases unprofitable — see Figure B.6 in the Appendix.

4.2.4 Heterogeneous coarse memory

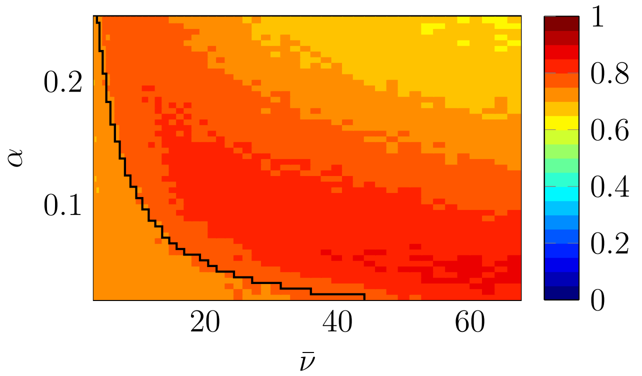
Until this point, I have considered cases where algorithms have homogeneously distributed memory precision of their rival's past price. I now consider memories that are coarser than (3–3–3–3–3) and that distribute the agents' memory precision of their rival's past price asymmetrically.[7] This serves two purposes. First, it will help understand if there exists a monotone relation between the fineness of memory and the degree of collusion algorithms are able to achieve. A coarser memory unequivocally reduces the information that algorithms have access to, and conventional wisdom suggests this should reduce their ability to find cooperative solutions to the pricing game. Secondly, it will provide insights into what information matters most to algorithms for attaining collusive equilibria. If the memory of higher prices and lower prices have different importance in forming collusive strategies, this will eventually be reflected in the outcomes.

I focus on six memories that are coarser than (3–3–3–3–3). The first three, (6–3–3–3), (9–3–3), and (12–3), give algorithms progressively coarser memory on low prices. The other three, (3–3–3–6), (3–3–9), and (3–12), are completely specular, giving algorithms progressively coarser memory on high prices. Figure 4.8 shows the average profit gain algorithms are able to achieve in equilibrium depending on α and $\bar{\nu}$ and on the memory they have on their rival's past price. Figures from B.8 to B.10 in the Appendix show the average fraction of suboptimal action selections in equilibrium, the average fraction of profitable downward deviations from the equilibrium price and the mean percentage gain (loss) by deviating downward from the equilibrium price.

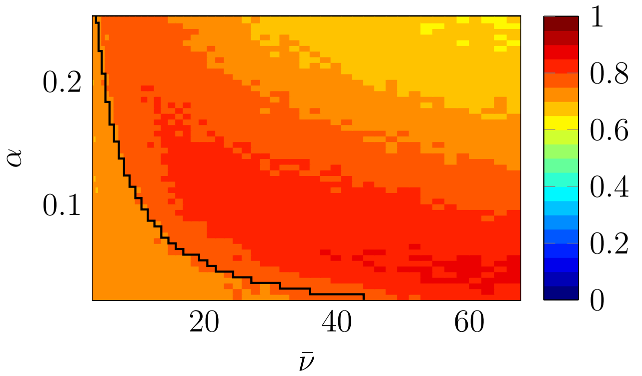
Figure 4.8 confirms that a coarse memory on the rival's past price implies, in most cases, only a small reduction in the extent of collusion algorithms are able to achieve. While the average profit gain is always lower than in the case of perfect memory, algorithms stubbornly maintain high levels of collusion. Only drastic limitations in the algorithms' memory induce meaningful reductions in profit gains — see Figure 4.8(g) and Figure 4.8(h).

Memories giving less precise information on high prices seem to make it more difficult for algorithms to sustain collusion. The coarser their memory on high prices, the less average profit gain algorithms obtain. Almost the opposite seems true when the algorithms' memory reduces the information on low prices instead. In that case, a coarser memory often implies a slightly larger average profit gain. This indicates there is no monotone relationship between the fineness of memory and collusion. In other words, limiting the information algorithms can have access to does not necessarily reduce collusion.

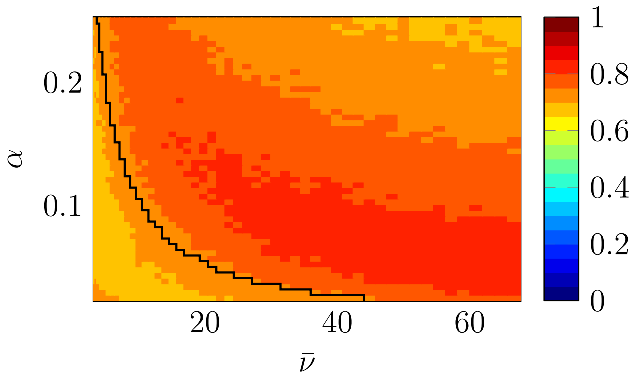
[7] I do not consider memories coarser than (5–5–5) because conclusions are virtually identical.



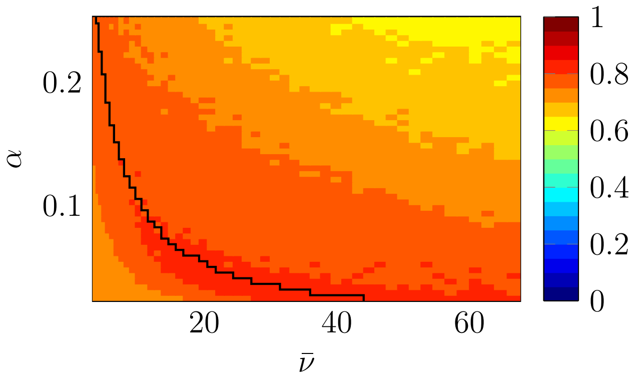
(a) Memory (3–3–3–3–3)



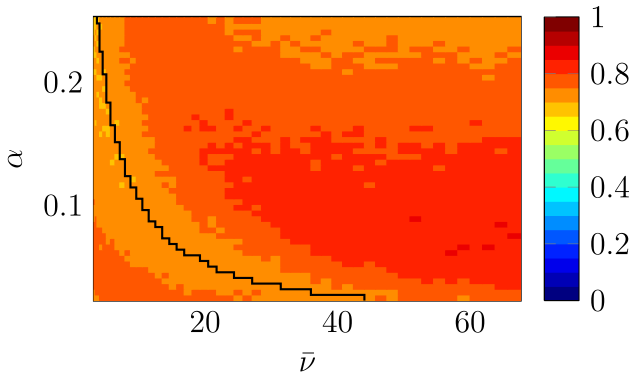
(b) Memory (3–3–3–3–3)



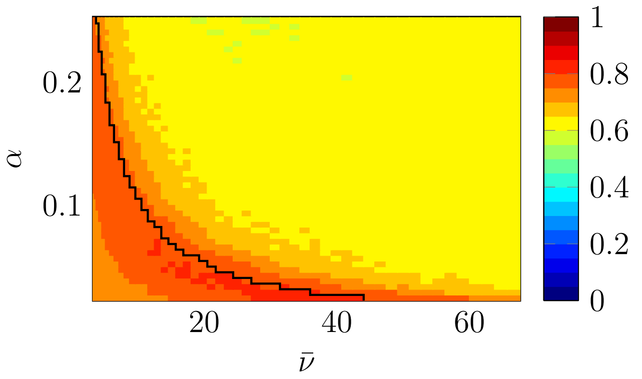
(c) Memory (6–3–3–3)



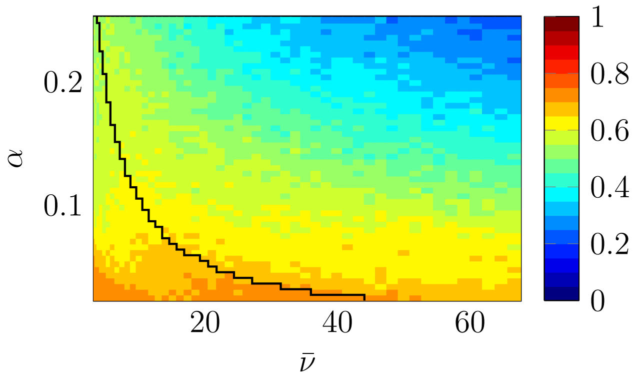
(d) Memory (3–3–3–6)



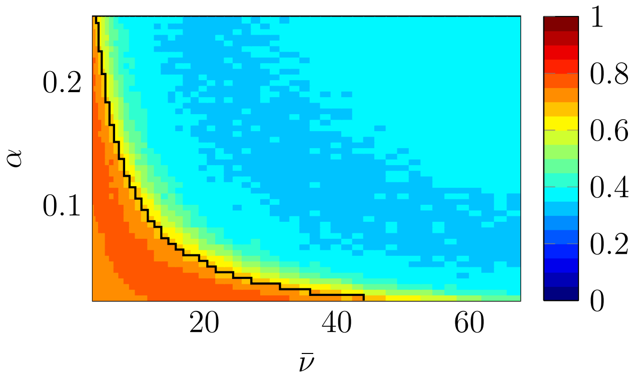
(e) Memory (9–3–3)



(f) Memory (3–3–9)



(g) Memory (12–3)



(h) Memory (3–12)

Figure 4.8: Average profit gain.

An explanation for these phenomena is that, as we have seen, algorithms tend to converge at the highest price they can, from which there are the least undetectable downward deviations. In this way, they maintain the ability to punish downward deviations and sustain collusion even with a coarse memory of their rival's past price. Reducing algorithms' information on high prices naturally makes them less sustainable in equilibrium, as more downward deviations from those prices become undetectable. On the contrary, reducing algorithms memory on low prices does very little in this direction. The very thing algorithms seem to need in order to achieve collusion is the existence of high prices from which deviating downward is detectable. Additional information on prices might help algorithms achieve superior equilibria, but it does not seem critical in their formation of collusive strategies.

Chapter 5

Conclusions

In this thesis, I study the effect of introducing coarse memory to Q-learning agents playing in an infinitely repeated game of Bertrand competition. With a perfect one-period memory of past prices, these algorithms are able to autonomously learn collusive strategies, solving the pricing game for cooperative equilibria. As in textbook examples of collusion, they sustain prices above Bertrand-Nash via the threat of punishments for unilateral defections. Critically, this kind of collusive strategy depends on the possibility for algorithms to condition their current price selections on past prices. A coarse memory impacts the ability of algorithms to reliably track and punish deviations, adding complexity to their coordination problem. In line with theoretical expectations, I find that with no memory of their rival's past price, algorithms often converge to the static Bertrand-Nash equilibrium. Instead, partially limiting algorithms' memory of their rival's past price does not prevent algorithmic collusion. Even with a coarse memory, algorithms find ways to sustain supra-competitive equilibria via punishments of defections. The additional complexity introduced by coarse memory only implies modest reductions in the extent of collusion they achieve.

Collusion is maximal in the case of perfect memory and virtually absent in case of no memory of the opponent's past price. For the rest, I do not find a monotone relationship between the fineness of memory and the degree of collusion algorithms are able to achieve. I find that a coarser memory diminishes firms' aggregate profit in equilibrium only when it reduces the information algorithms have on high prices. Limiting algorithms' information on prices close to the Bertrand-Nash price instead does not affect their ability to learn collusive strategies. Overall, algorithmic collusion seems not to depend on the amount of information algorithms have at their disposal, but rather on the presence of high prices from which downward deviations can be detected.

This research highlights that Q-learning algorithms require very limited information to learn collusive strategies based on punishments for unilateral defections. As long as these algorithms have a minimal perception of their opponent's past price, they are able to find ways to punish defections and sustain collusion.

Appendix A

General appendix

A.1 Equilibrium Prices

The static Nash-Bertrand price for n symmetric firms (Anderson and de Palma, 1992) is a fixed point of the equation

$$p^N = c + \frac{\mu}{1 - (n + \exp[(a_0 - a + p^N)/\mu])^{-1}}. \tag{A.1}$$

The monopoly price maximizes the joint profits of the n firms in the market, solving

$$\boldsymbol{p}^M = \arg\max_{\boldsymbol{p} \in \mathbb{R}^n} \sum_{i=1}^{n} \pi_i(p_1, \ldots, p_n) \tag{A.2}$$

which yields $\boldsymbol{p}^M = (p^M, \ldots, p^M)$.

A.2 Number of admissible refinements

The set of prices A is a discrete set whose elements are indexed in monotone increasing order. A partition X which does not alter this order simply splits A between two adjacent prices p^j and p^{j+1} for some point $j \in \{1, \ldots, |A| - 1\}$. Since X can either split or not split A in each point $j \in \{1, \ldots, |A| - 1\}$, this means there are $2^{|A|-1}$ possible partitions with this property.

Let X partition A into H disjoint subsets satisfying the abovementioned property. Then, X splits A in $H - 1$ points. It follows that one could further split A at most $|A| - H$ times, after which perfect memory is obtained. With the same logic as before, it follows that X has $2^{|A|-H}$ possible refinements (including itself) not altering the monotone order of the elements.

Also, since X splits A in $H-1$ different points, it means one could join at most $H-1$ pairs of adjacent subsets $x^h, x^{h+1} \in X$ for $h \in \{1, \ldots, H - 1\}$, after which the coarsest memory is obtained. This means X is the refinement of 2^{H-1} partitions (including itself) that do not alter the monotone order of the elements.

Finally, this implies X can be compared with $2^{|A|-H}+2^{H-1}-1$ different partitions (including itself), which is strictly less than the total number of admissible partitions $2^{|A|-1}$ for all $H \in \{2, \ldots, |A| - 1\}$.

A.3 Exploration intensity

For a given pair (o, a) to be reached in period t thanks only to agents' exploration it must be true that: (1) the state of the environment $s \in S$ is the result of every agent exploring in period $t-1$; (2) agent i has a memory structure such that, given state s, it observes $o \in O$; and (3) agent i explores in period t resulting in it taking action a.

The state of the environment in period t is completely determined by agents' exploration if all agents explored in period $t-1$, which happens with probability $\beta^{(t-1)n}$. Given a random state $s \in S$, the probability of observing $o \in O$ depends on the proportion of states in S which imply the same observation o, namely $|o|/|S|$. Finally, the probability with which agent i explores in period t and this result in it taking action $a \in A$ is $\beta^t/|A|$. Finally, the probability of reaching a given pair $(o, a) \in O \times A$ in period t is the joint probability of these events,

$$\frac{|o|}{|S| \cdot |A|}(\beta^{t-1})^n \beta^t, \tag{A.3}$$

and the expected number of times (o, a) gets visited due only to agents' exploration over an infinite horizon is simply

$$\nu(o) = \sum_{t=1}^{\infty} \frac{|o|}{|S| \cdot |A|}(\beta^{t-1})^n \beta^t = \frac{|o|}{|S| \cdot |A|} \frac{\beta}{(1-\beta^{n+1})}. \tag{A.4}$$

A.4 Q-learning metrics

The absolute percentage error of the estimated action-value function with respect to its true value is computed as

$$\frac{Q(o, a) - q(o, a)}{q(o, a)}. \tag{A.5}$$

The frequency of suboptimal action selections in equilibrium measures the percentage of suboptimal actions made by agents during the price cycle they have converged to. An agent takes a suboptimal action if it does not best respond to its rival, that is, if

$$\arg\max_a Q(o, a) \neq \arg\max_a q(o, a). \tag{A.6}$$

In that case, taking action $\arg\max_a q(o, a)$ is a profitable deviation (the most profitable one). The foregone payoff of not taking the optimal action represents the percentage loss in terms of discounted payoff over an infinite horizon of taking action $a^* = \arg\max_a Q(o, a)$ rather than $a^{**} = \arg\max_a q(o, a)$, in state-observation o, namely

$$\frac{q(o, a^{**}) - q(o, a^*)}{q(o, a^*)}. \tag{A.7}$$

Finally, a session converges to a Nash equilibrium if both agents choose the optimal action in all state-observations they visit in equilibrium.

Appendix B

Figures and Tables

B.1 Perfect memory

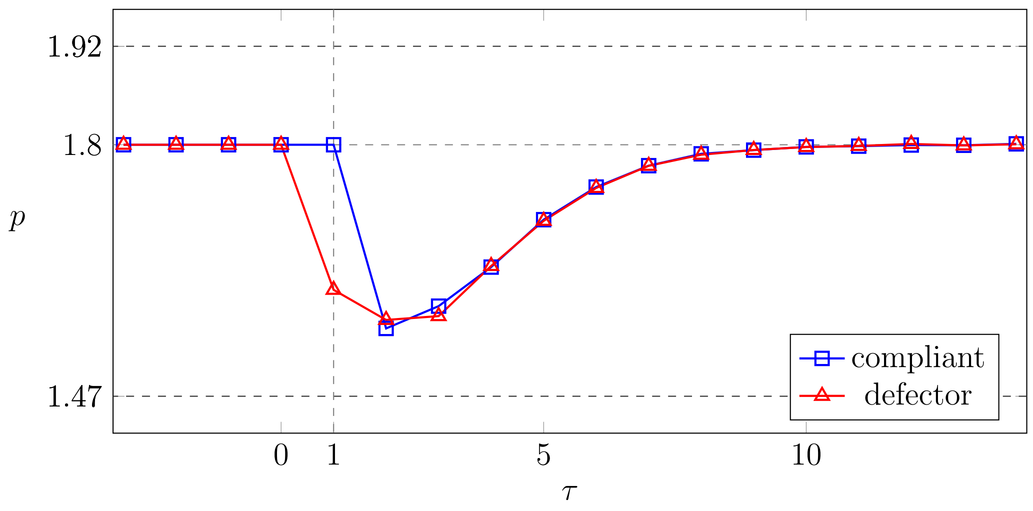
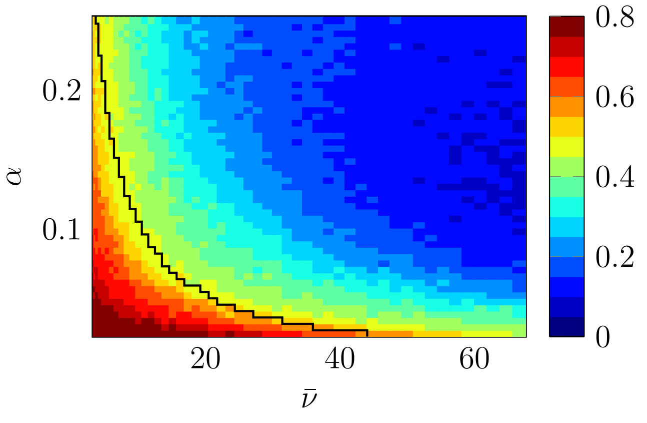
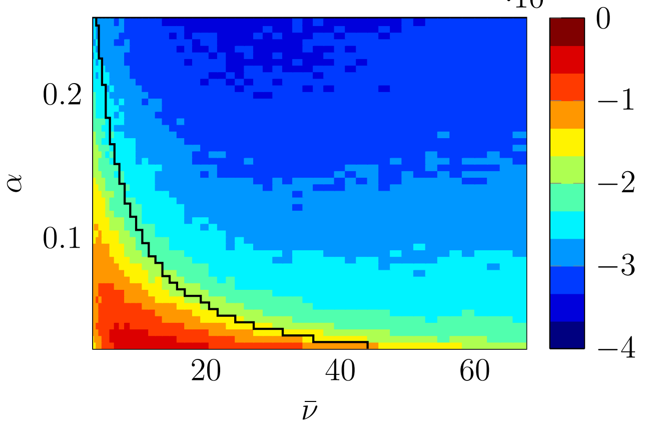


Figure B.1: Average impulse response across all sessions, agents, and possible deviation periods. The defecting agent deviates in $\tau = 1$ to the static best response of its rival's price.



(a) Average fraction of suboptimal action selections in equilibrium.



(b) Average loss by deviating downward from the equilibrium price.

Figure B.2

Deviation price	Equilibrium price									
	1.963	1.925	1.887	1.850	1.812	1.774	1.737	1.699	1.661	1.624
1.925	-0.033									
1.887	-0.031	-0.034								
1.850	-0.030	-0.031	-0.035							
1.812	-0.026	-0.029	-0.030	-0.032						
1.774	-0.024	-0.027	-0.030	-0.029	-0.031					
1.737	-0.024	-0.027	-0.030	-0.030	-0.031	-0.031				
1.699	-0.021	-0.028	-0.029	-0.028	-0.028	-0.030	-0.029			
1.661	-0.023	-0.025	-0.028	-0.027	-0.028	-0.029	-0.028	-0.027		
1.624	-0.022	-0.025	-0.029	-0.028	-0.028	-0.029	-0.029	-0.028	-0.025	
1.586	-0.024	-0.028	-0.029	-0.030	-0.029	-0.029	-0.028	-0.028	-0.027	-0.025
1.548	-0.027	-0.030	-0.030	-0.030	-0.030	-0.031	-0.028	-0.028	-0.026	-0.026
1.511	-0.027	-0.031	-0.030	-0.030	-0.033	-0.032	-0.030	-0.029	-0.028	-0.026
1.473	-0.031	-0.033	-0.034	-0.034	-0.035	-0.033	-0.030	-0.029	-0.030	-0.029
1.435	-0.034	-0.035	-0.035	-0.036	-0.035	-0.033	-0.032	-0.032	-0.030	-0.026
Frequency	0.052	0.084	0.108	0.123	0.155	0.142	0.124	0.087	0.060	0.029

Table B.1: Average percentage gain in terms of discounted profits for different equilibrium and deviation prices.

Deviation price	Equilibrium price									
	1.963	1.925	1.887	1.850	1.812	1.774	1.737	1.699	1.661	1.624
1.925	0.964									
1.887	0.947	0.952								
1.850	0.935	0.960	0.972							
1.812	0.899	0.923	0.904	0.97						
1.774	0.870	0.912	0.937	0.922	0.964					
1.737	0.858	0.908	0.937	0.952	0.942	0.961				
1.699	0.870	0.941	0.943	0.940	0.958	0.961	0.96			
1.661	0.864	0.893	0.940	0.922	0.958	0.952	0.945	0.947		
1.624	0.882	0.890	0.943	0.922	0.942	0.950	0.955	0.950	0.927	
1.586	0.882	0.934	0.932	0.935	0.936	0.965	0.955	0.964	0.917	0.978
1.548	0.947	0.926	0.949	0.955	0.960	0.967	0.960	0.954	0.959	0.946
1.511	0.899	0.938	0.957	0.977	0.972	0.974	0.965	0.968	0.953	0.946
1.473	0.959	0.960	0.966	0.977	0.966	0.965	0.960	0.954	0.964	0.978
1.435	0.964	0.960	0.977	0.985	0.970	0.974	0.985	0.982	0.959	0.978
Frequency	0.052	0.084	0.108	0.123	0.155	0.142	0.124	0.087	0.060	0.029

Table B.2: Frequency of unprofitable deviations for different equilibrium and deviation prices.

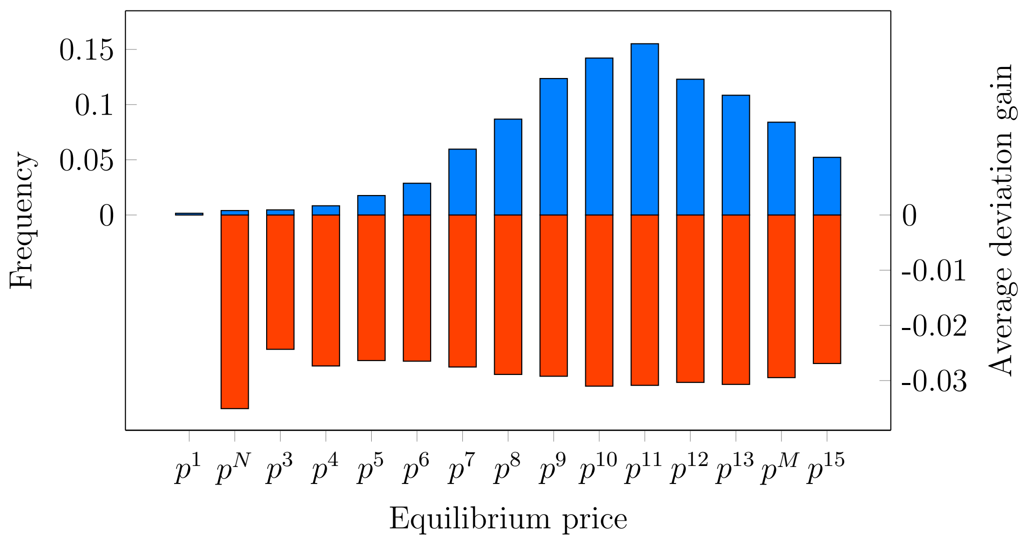
B.2 Coarse memory of rival's prices

Deviation price	Equilibrium price									
	1.963	1.925	1.887	1.850	1.812	1.774	1.737	1.699	1.661	1.624
1.925	-0.015									
1.887	-0.011	-0.019								
1.850	-0.024	-0.031	-0.033							
1.812	-0.025	-0.031	-0.031	-0.019						
1.774	-0.022	-0.026	-0.027	-0.013	-0.015					
1.737	-0.022	-0.029	-0.032	-0.035	-0.032	-0.032				
1.699	-0.023	-0.028	-0.031	-0.003	-0.028	-0.030	-0.015			
1.661	-0.021	-0.028	-0.031	-0.029	-0.029	-0.029	-0.010	-0.011		
1.624	-0.027	-0.031	-0.032	-0.030	-0.030	-0.030	-0.029	-0.026	-0.023	
1.586	-0.028	-0.028	-0.033	-0.033	-0.030	-0.029	-0.027	-0.026	-0.023	-0.012
1.548	-0.025	-0.029	-0.033	-0.032	-0.030	-0.030	-0.028	-0.025	-0.023	-0.010
1.511	-0.029	-0.030	-0.036	-0.034	-0.030	-0.032	-0.029	-0.027	-0.025	-0.020
1.473	-0.029	-0.035	-0.038	-0.036	-0.034	-0.032	-0.030	-0.027	-0.027	-0.021
1.435	-0.031	-0.035	-0.041	-0.036	-0.035	-0.035	-0.032	-0.029	-0.028	-0.024
Frequency	0.033	0.057	0.1	0.077	0.126	0.205	0.081	0.098	0.119	0.030

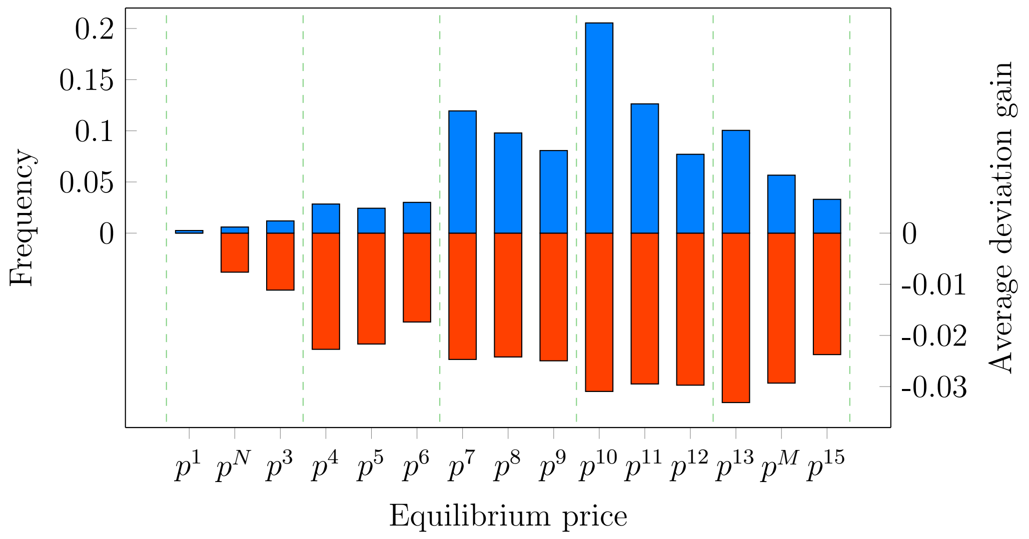
Table B.3: Average percentage gain in terms of discounted profits for different equilibrium and deviation prices, when agents have memory (3–3–3–3–3) on their rival's past price. Values above the horizontal lines are deviations which are not directly detectable.

Deviation price	Equilibrium price									
	1.963	1.925	1.887	1.850	1.812	1.774	1.737	1.699	1.661	1.624
1.925	-0.014									
1.887	-0.008	-0.012								
1.850	-0.010	-0.011	-0.012							
1.812	-0.005	-0.008	-0.007	-0.014						
1.774	-0.026	-0.025	-0.031	-0.035	-0.032					
1.737	-0.028	-0.023	-0.028	-0.033	-0.032	-0.012				
1.699	-0.022	-0.023	-0.027	-0.032	-0.029	-0.009	-0.011			
1.661	-0.024	-0.022	-0.026	-0.032	-0.028	-0.013	-0.011	-0.007		
1.624	-0.022	-0.024	-0.028	-0.030	-0.027	-0.008	-0.009	-0.007	-0.007	
1.586	-0.028	-0.029	-0.031	-0.035	-0.032	-0.025	-0.027	-0.025	-0.022	-0.021
1.548	-0.032	-0.030	-0.033	-0.034	-0.032	-0.026	-0.028	-0.023	-0.020	-0.021
1.511	-0.034	-0.032	-0.036	-0.037	-0.034	-0.030	-0.026	-0.026	-0.020	-0.022
1.473	-0.034	-0.032	-0.034	-0.038	-0.035	-0.027	-0.027	-0.025	-0.021	-0.022
1.435	-0.038	-0.035	-0.039	-0.040	-0.038	-0.027	-0.028	-0.028	-0.023	-0.024
Frequency	0.027	0.047	0.068	0.109	0.195	0.051	0.068	0.083	0.117	0.142

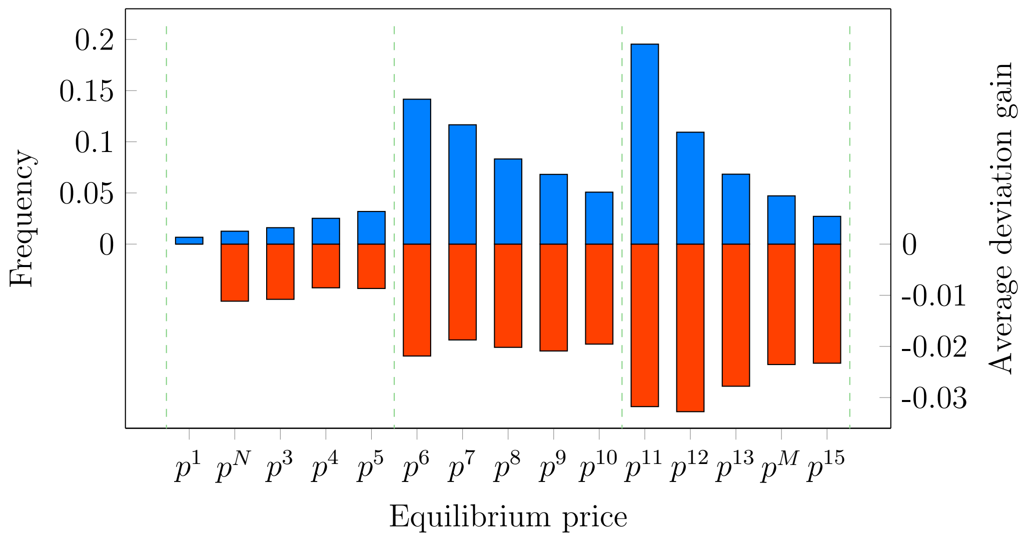
Table B.4: Average percentage gain in terms of discounted profits for different equilibrium and deviation prices, when agents have memory (5–5–5) on their rival's past price. Values above the horizontal lines are for deviations which are not directly detectable.



(a) Perfect memory

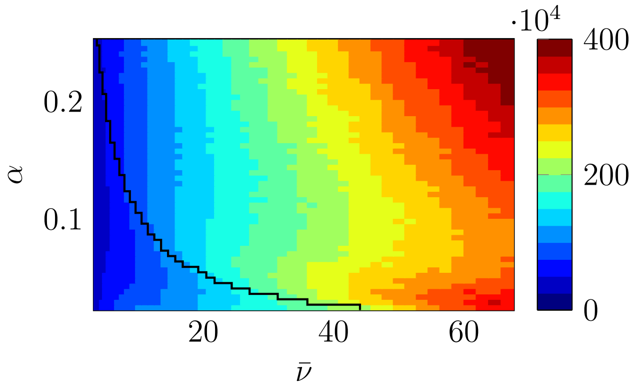


(b) Memory (3–3–3–3–3)

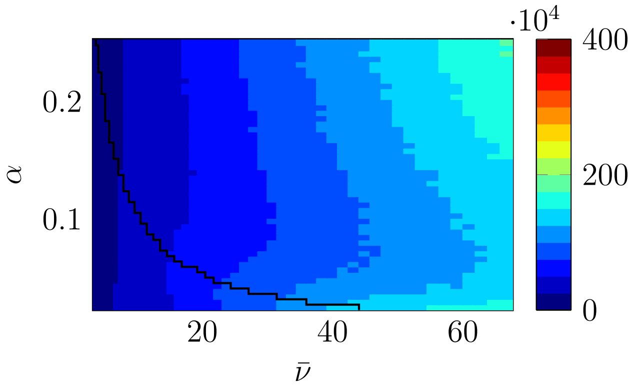


(c) Memory (5–5–5)

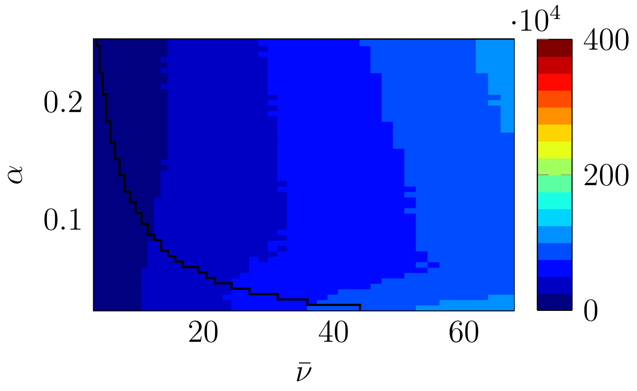
Figure B.3: The blue bars (left scale) represent the frequency of prices in equilibrium. The red bars (right scale) describe the average gain by deviating downward from that price. The green vertical dashed lines describe the granularity of the memory on the rival's past price.



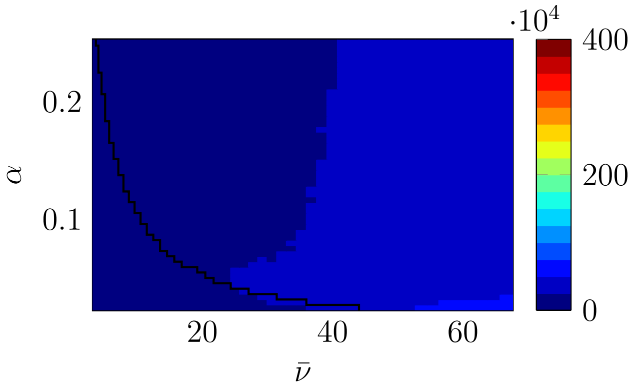
(a) Perfect memory



(b) Memory (3–3–3–3–3)

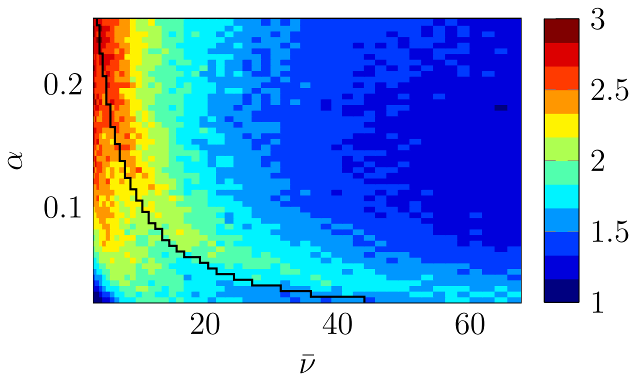


(c) 5-5-5

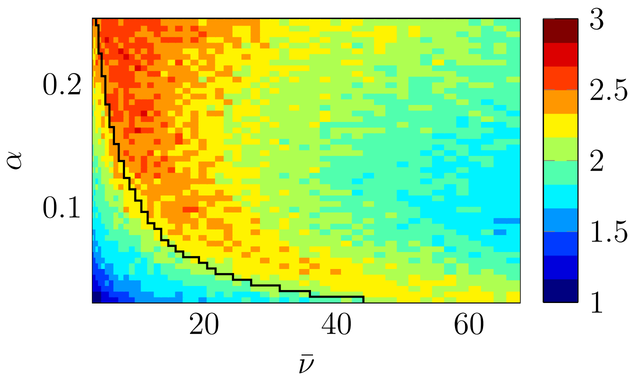


(d) No memory

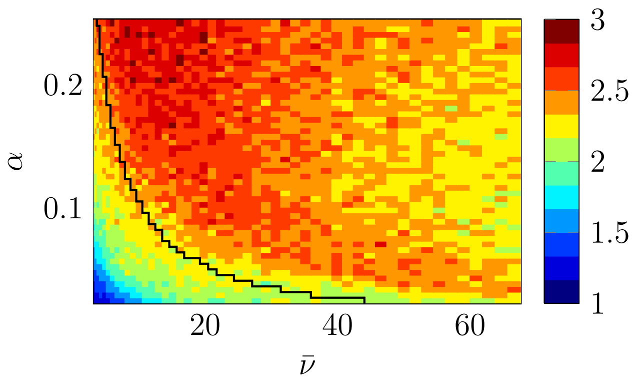
Figure B.4: Average number of periods required to reach convergence.



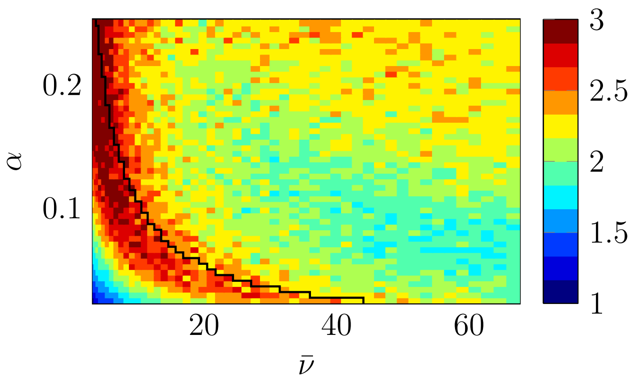
(a) Perfect memory



(b) Memory (3–3–3–3–3)

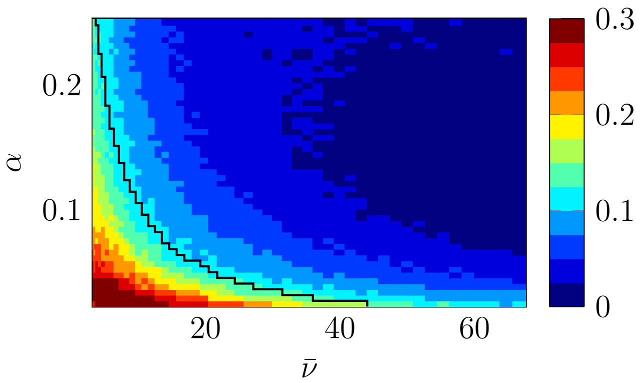


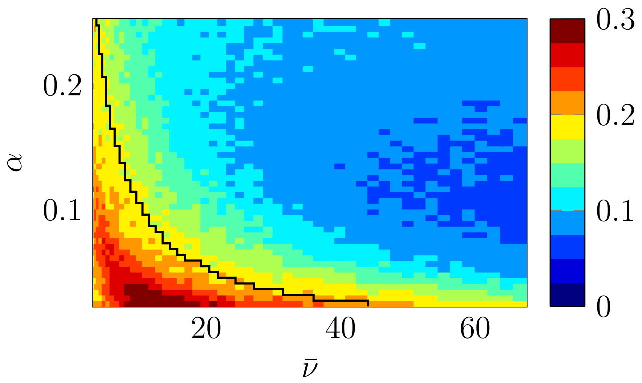
(c) 5-5-5

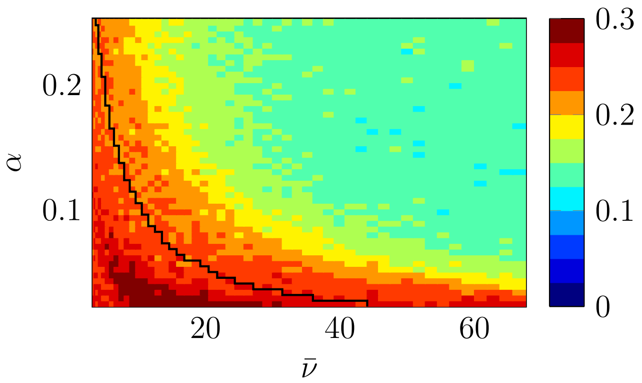


(d) No memory

Figure B.5: Average length of price cycles at convergence.


(a) Perfect memory


(b) Memory (3–3–3–3–3)


(c) 5-5-5

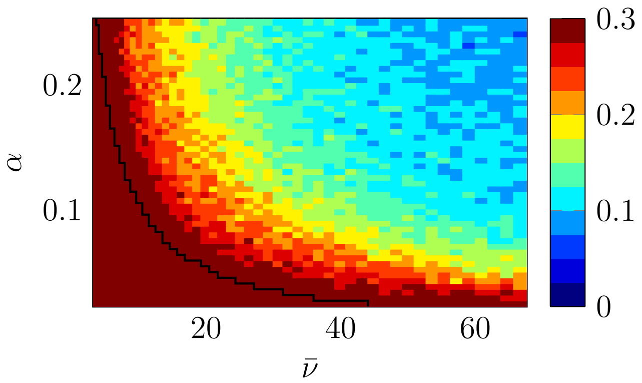
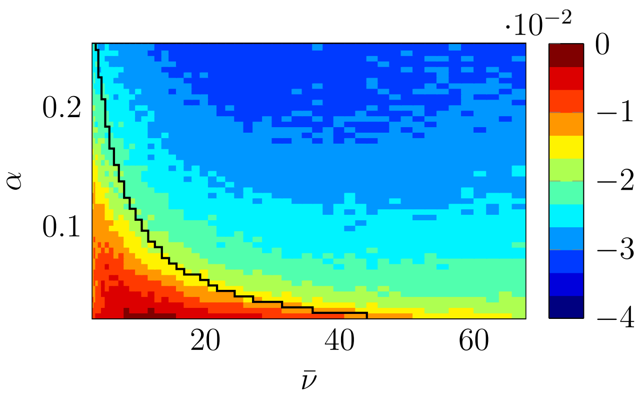
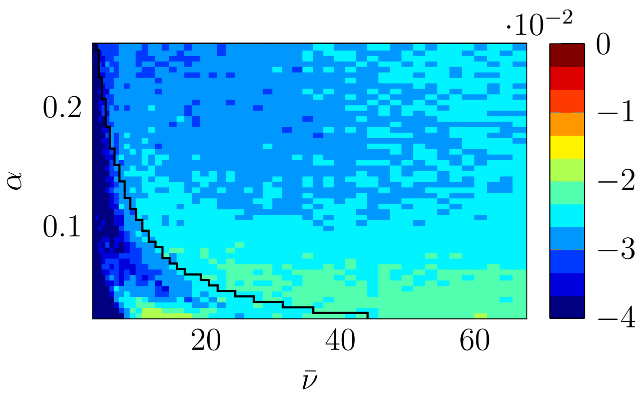
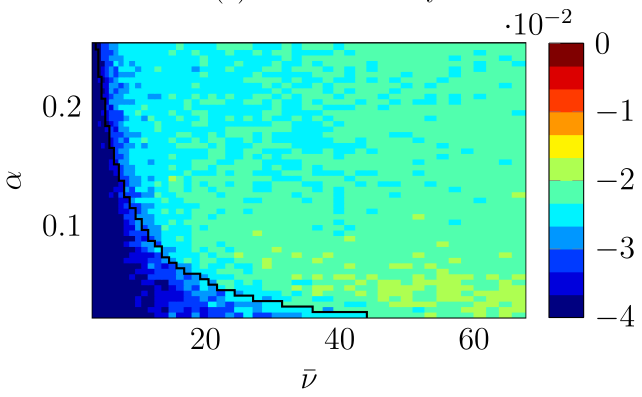

(d) No memory

Figure B.6: Average fraction of profitable downward deviation.


(a) Perfect memory


(b) Memory (3–3–3–3–3)


(c) 5-5-5

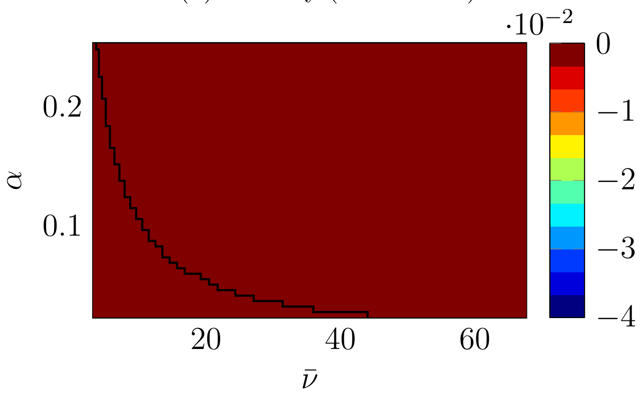
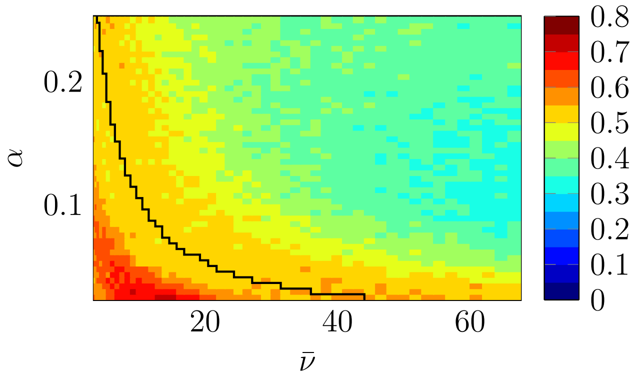
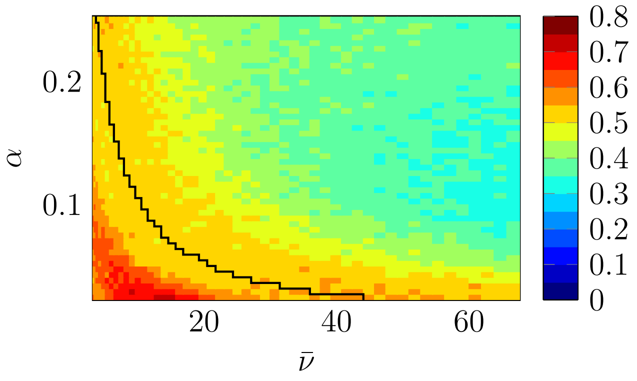

(d) No memory

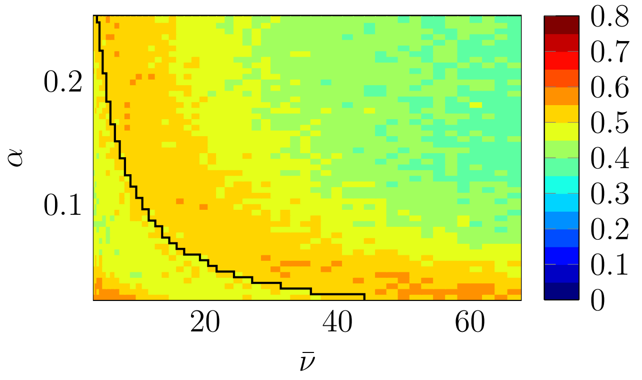
Figure B.7: Average percentage gain (loss) by deviating to the static best response.



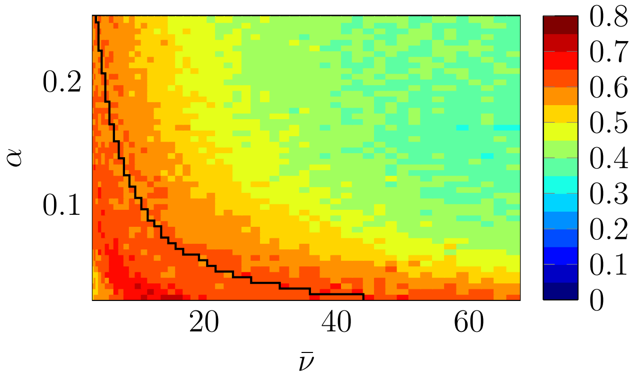
(a) Memory (3–3–3–3–3)



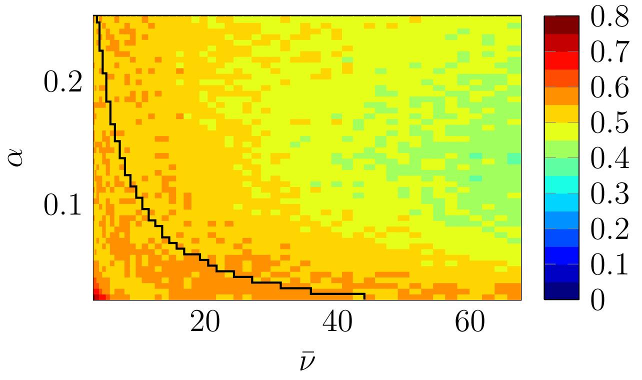
(b) Memory (3–3–3–3–3)



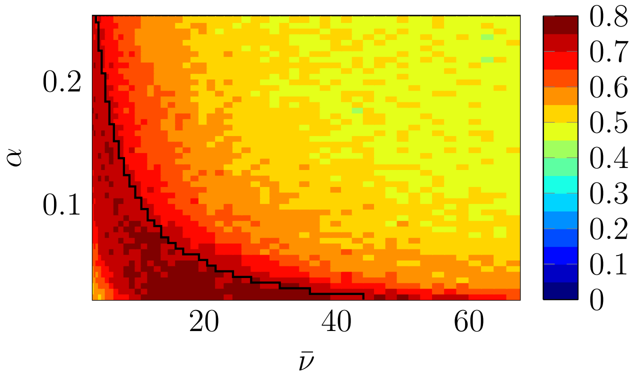
(c) Memory (6–3–3–3)



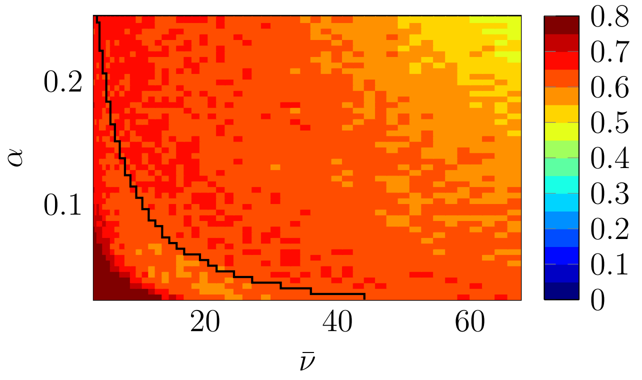
(d) Memory (3–3–3–6)



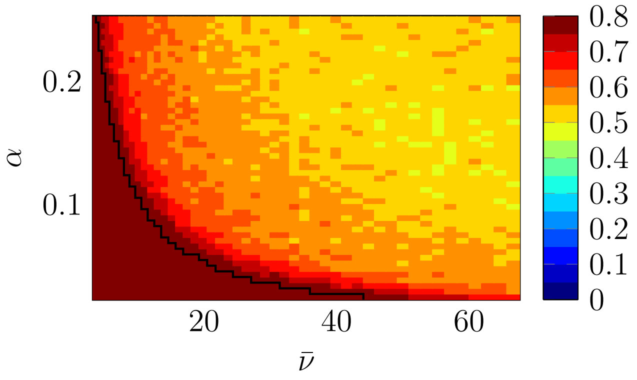
(e) Memory (9–3–3)



(f) Memory (3–3–9)

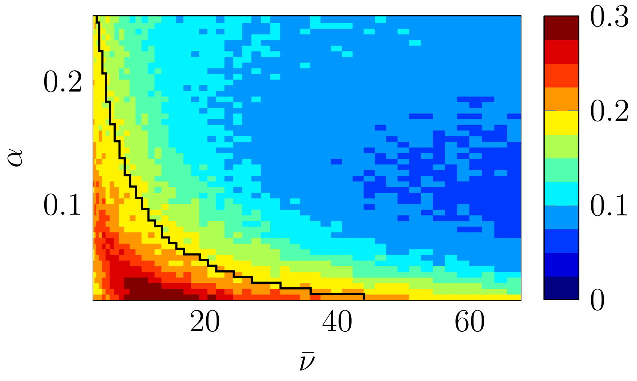


(g) Memory (12–3)

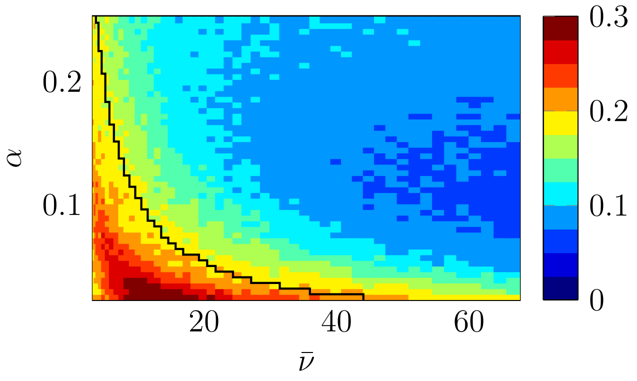


(h) Memory (3–12)

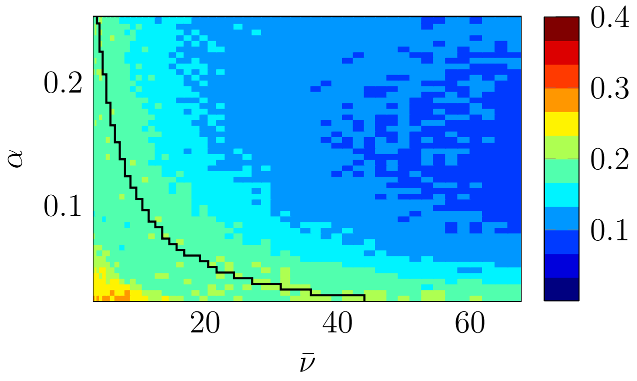
Figure B.8: Average fraction of suboptimal action selections in equilibrium.



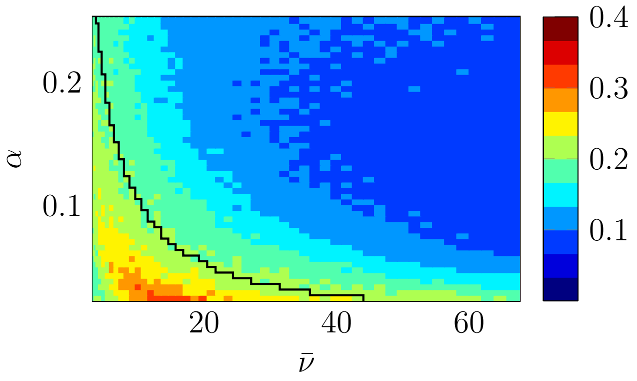
(a) Memory (3–3–3–3–3)



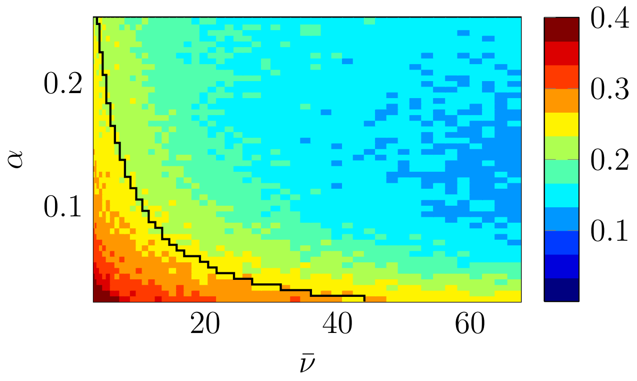
(b) Memory (3–3–3–3–3)



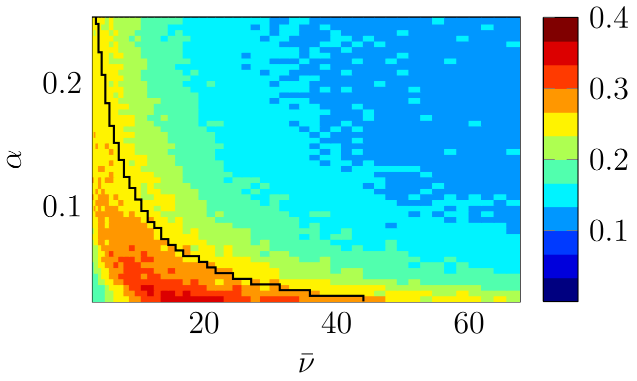
(c) Memory (6–3–3–3)



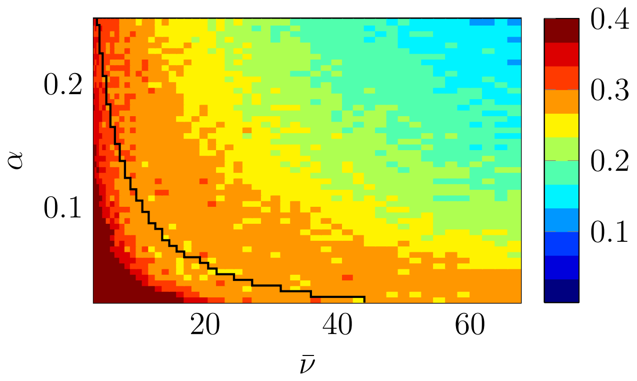
(d) Memory (3–3–3–6)



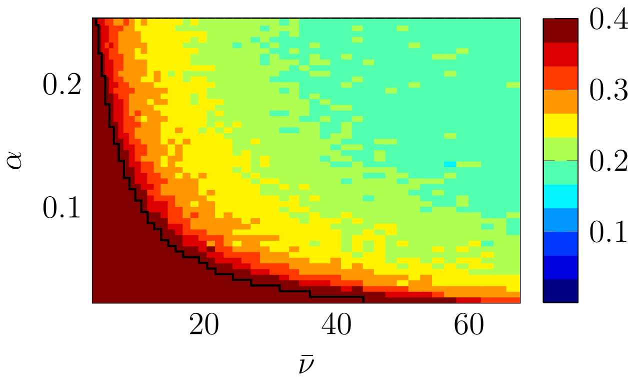
(e) Memory (9–3–3)



(f) Memory (3–3–9)

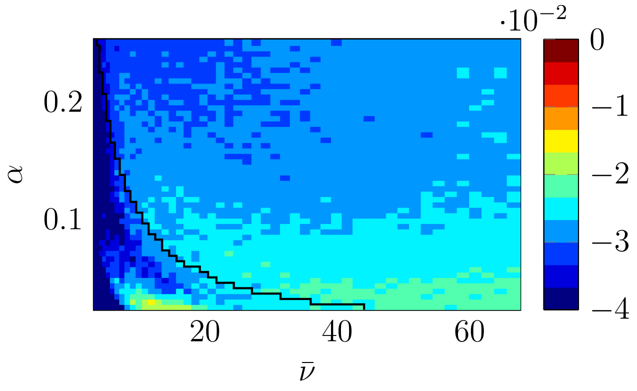


(g) Memory (12–3)

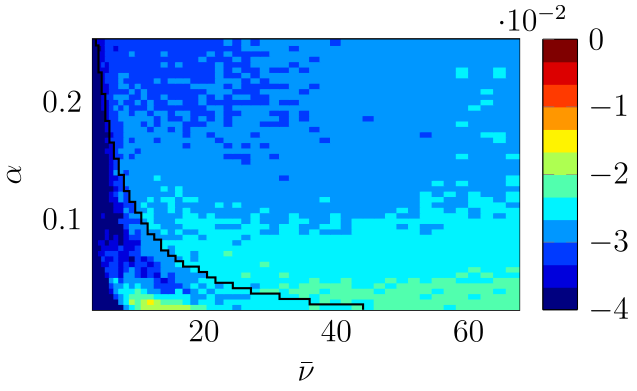


(h) Memory (3–12)

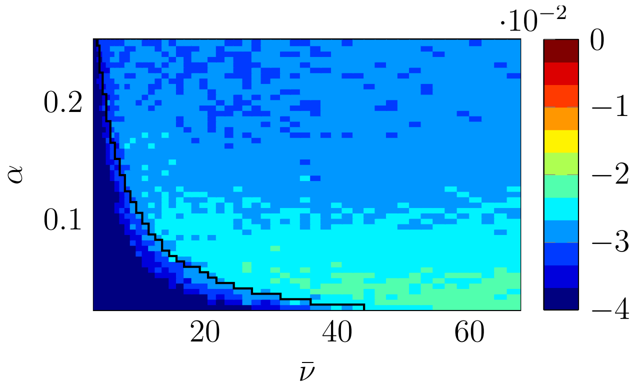
Figure B.9: Average fraction of profitable downward deviations.



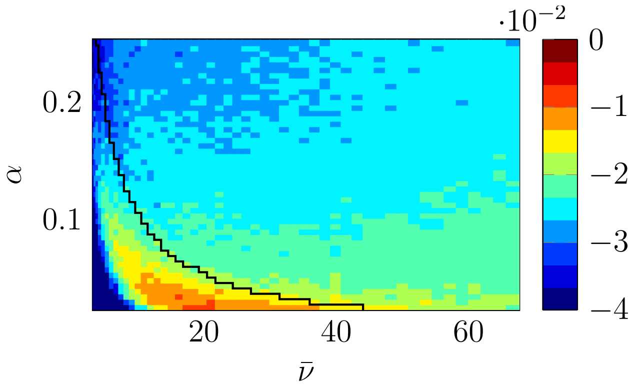
(a) Memory (3–3–3–3–3)



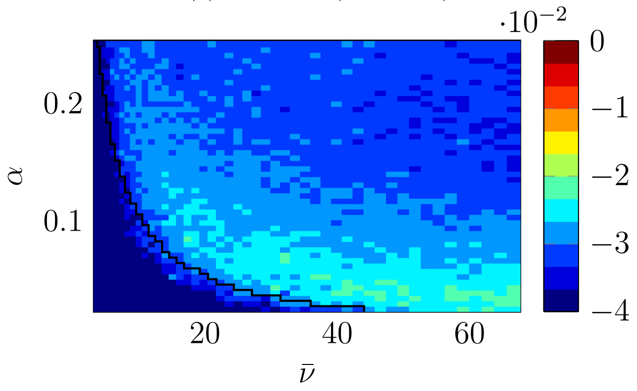
(b) Memory (3–3–3–3–3)



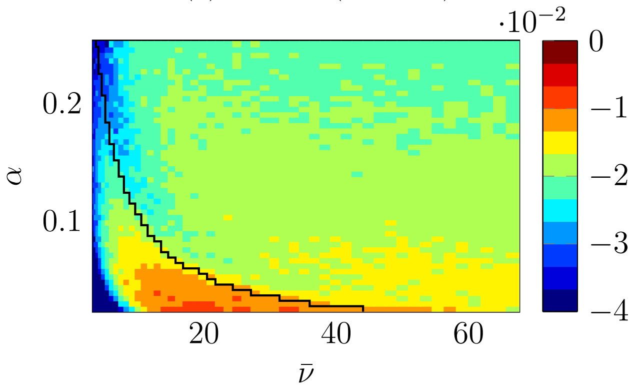
(c) Memory (6–3–3–3)



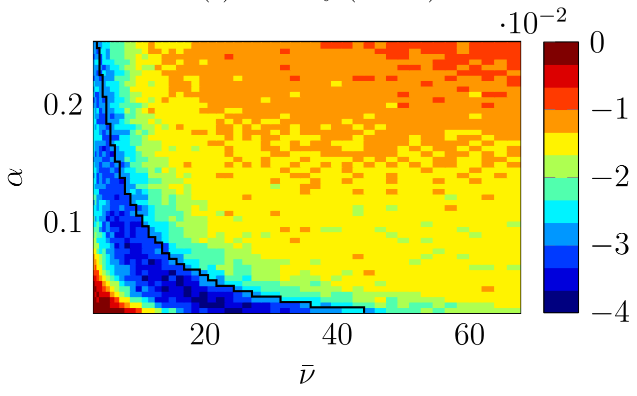
(d) Memory (3–3–3–6)



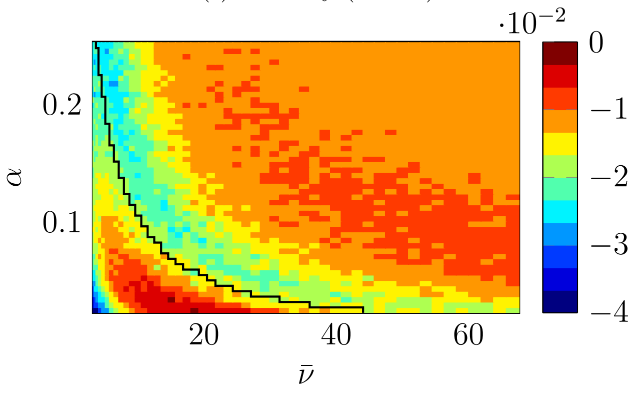
(e) Memory (9–3–3)



(f) Memory (3–3–9)



(g) Memory (12–3)



(h) Memory (3–12)

Figure B.10: Average percentage gain (loss) by deviating downward from the equilibrium.

Bibliography

Anderson, S. P. and de Palma, A. (1992). The logit as a model of product differentiation. *Oxford Economic Papers*, 44(1):51–67.

Assad, S., Clark, R., Ershov, D., and Xu, L. (2020). Algorithmic pricing and competition: Empirical evidence from the german retail gasoline market.

Bezanson, J., Edelman, A., Karpinski, S., and Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98.

Brown, Z. Y. and MacKay, A. (2021). Competition in pricing algorithms. Working Paper 28860, National Bureau of Economic Research.

Calvano, E., Calzolari, G., Denicolò, V., and Pastorello, S. (2020). Artificial intelligence, algorithmic pricing, and collusion. *American Economic Review*, 110(10):3267–97.

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, pages 1339–1349, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Degrave, J., Felici, F., Buchli, J., Neunert, M., Tracey, B., Carpanese, F., Ewalds, T., Hafner, R., Abdolmaleki, A., Casas, D., Donner, C., Fritz, L., Galperti, C., Huber, A., Keeling, J., Tsimpoukelli, M., Kay, J., Merle, A., Moret, J.-M., and Riedmiller, M. (2022). Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602:414–419.

Frick, K. M. (2022). Autonomous pricing using policy gradient reinforcement learning. Master's thesis, Bologna University, Bologna, Italy.

Hettich, M. (2021). Algorithmic collusion: Insights from deep learning. CQE Working Papers 9421, Center for Quantitative Economics (CQE), University of Muenster.

Jumper, J. M., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Zídek, A., Potapenko, A., Bridgland, A., Meyer, C., Kohl, S. A. A., Ballard, A., Cowie, A., Romera-Paredes, B., Nikolov, S., Jain, R., Adler, J., Back, T., Petersen, S., Reiman, D. A., Clancy, E., Zielinski, M., Steinegger, M., Pacholska, M., Berghammer, T., Bodenstein, S., Silver, D., Vinyals, O., Senior, A. W., Kavukcuoglu, K., Kohli, P., and Hassabis, D. (2021). Highly accurate protein structure prediction with alphafold. *Nature*, 596:583 – 589.

Klein, T. (2021). Autonomous algorithmic collusion: Q-learning under sequential pricing. *The RAND Journal of Economics*, 52(3):538–558.

Marschak, J. and Radner, R. (1972). The economic theory of teams. *Yale University Press.*

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.

Schwalbe, U. (2018). Algorithms, machine learning, and collusion. *Journal of Competition Law & Economics*, 14(4):568–607.

Silver, D., 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.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction.* A Bradford Book, Cambridge, MA, USA.

Waltman, L. and Kaymak, U. (2008). Q-learning agents in a cournot oligopoly model. *Journal of Economic Dynamics and Control*, 32(10):3275–3293.

Watkins, C. J. C. H. and Dayan, P. (1992). Q-learning. *Machine Learning*, 8(3-4):279–292.