Robust Algorithmic Collusion

Nicolas Eschenbaum Department of Economics
University of St. Gallen
St. Gallen, Switzerland
nicolas.eschenbaum@unisg.ch

Filip Melgren
Stockholm School of Economics
Stockholm, Sweden
filip.mellgren@gmail.com

Philipp Zahn
Department of Economics
University of St. Gallen
St. Gallen, Switzerland
philipp.zahn@unisg.ch

Abstract

This paper develops an approach to assess reinforcement learners with collusive pricing policies in a testing environment. We find that algorithms are unable to extrapolate collusive policies from their training environment to testing environments. Collusion consistently breaks down, and algorithms instead tend to converge to Nash prices. Policy updating with or without exploration re-establishes collusion, but only in the current environment. This is robust to repeated learning across environments. Our results indicate that frequent market interaction, coordination of algorithm design, and stable environments are essential for algorithmic collusion.

1 Introduction

Software systems that take over pricing decisions are spreading quickly. Pricing algorithms can allow firms to monitor and process large amounts of data and adjust prices quickly to changing circumstances. The ascent of such systems poses a potential challenge for the current regulatory landscape: pricing algorithms based on artificial intelligence (AI) may learn to autonomously collude without any previous intentional agreement or explicit instruction to do so.

A growing literature has shown algorithmic collusion to be possible in principle. The results documented so far are a clear warning sign: Even simple algorithms learn to tacitly collude and thereby harm consumers. However, existing analyses have studied the behavior of algorithms in their training environment. It is well-known that algorithms tend to overfit to the training environment, and results cannot easily be extrapolated to other environments (e.g. Lanctot et al., 2017). In practice, firms train their algorithms offline before using them and face substantial uncertainty about important parameters of the market and their competitors, as well as potentially significant cost from randomized learning in the marketplace. Thus, it is implicitly assumed that the training environment and market environment are symmetric and identical, and that results can be extrapolated from one environment to the other.

In this paper, we develop a simple approach to assess the behavior of reinforcement learning algorithms in pricing games when training is separated from the ensuing market interactions. To construct artificial testing environments, we consider small variations of structurally relevant parameters. We assess two dimensions of mis-specification of the

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

training environment: (i) rival parameters, and (ii) demand parameters. To study the impact of uncertainty over competitor parameters, we train algorithms in environments with dfferent marginal cost, and match them following convergence. To study the effect of uncertainty regarding the market environment, we vary the demand function parameters following convergence for a given cost level. We then study the outcome (i) in the first 100 iterations following the change, (ii) after convergence when learning is absent, and (iii) after convergence if learning is instead permitted.

Our results are as follows: First, we confirm existing findings of the literature. Algorithms which jointly learn overwhelmingly achieve collusive prices. Second, when we separate training from "market" interactions – by rematching learners after an initial learning phase – so that learners trained with one cost parameter may face a learner that trained with a different cost parameter, collusion breaks down. This happens even with small changes to the environment. Moreover, as cost differences become smaller, prices played converge towards Nash prices. Third, the breakdown of collusion is permanent and does not recover with further iterations. Fourth, subsequent learning in this new environment re-establishes collusive outcomes. Yet, this is again limited to that specific environment. In particular, if after collusion is re-established the player is rematched again – back to his original environment – collusion breaks down again and Nash play is reinforced. Fifth, re-establishing collusion in the new environment requires learning but not exploration. That is, when algorithms only exploit (and do not randomly explore) but continue to update their policy, they also re-establish collusive outcomes. Lastly, this breakdown of collusion, yet stubborn return if learning continues (with or without exploration) also arises when algorithms play against a previously unseen competitor in an unchanged environment.[1] This breakdown of collusion holds for repeated learning across different parameterized environments, implying that algorithmic collusion in this setting is a consequence of algorithm overfit.

Our findings illustrate a key problem with the current setup of the analysis of algorithmic collusion: results are reported based on the outcomes in the training environment. In practice however, algorithms are trained and deployed in separate environments. The tendency of machine learning algorithms to overfit to the training environment (or data), and that therefore a separate testing environment is required to assess their behavior is well-established (see e.g. Lanctot et al., 2017; Zhang et al., 2018b,a; Song et al., 2019). But this testing environment is not readily available with reinforcement learners.

Our work in this paper develops a possible approach to overcome this limitation in the context of pricing algorithms. Small variations in structurally relevant parameters, such as firms' cost levels, can serve to mimick the uncertainty firms face when training their respective algorithms. By testing the behavior in environments with slight parameter differences or against previously unseen competitors, we can assess the likelihood of a given algorithm to achieve collusive outcomes in the market. While we document here that collusion breaks down with the specific learning model that we consider, this may not apply to other models. Firms may even explicitly attempt to construct a learner that successfully extrapolates collusive strategies across environments and competitors. We also observe that the fragility of algorithmic collusion is balanced against its stubborn and relatively fast return when learning continues or is re-started.

This paper contributes to a growing literature on algorithmic collusion (for a recent survey of the economic literature on AI see Abrardi et al. (2021)). We employ simple Q-learning algorithms in line with related work in e.g. **?**Klein (2021). Our baseline scenario and parameterization is built on the environment studied in **?**. The repeated Bertrand setting with logit demand that we use implies that both the Nash and joint-profit maximizing profits are constant across marginal cost levels. Equally, the Nash price and joint-profit maximizing price stays constant when a competitors' cost level changes. Thus, differences in outcomes observed are not due to changes in possible payoffs for players. To the best of our knowledge, we are the first to explicitly study the behavior of algorithms that have learnt

[1] Calvano et al. (2020b) report a similar finding for symmetric algorithms. Our work shows that this arises due to continued updating.

collusive strategies outside their training environment, and investigate pricing algorithm overfit.

Our paper is related to the computer science literature that studies the overfitting of reinforcement learning algorithms. Lanctot et al. (2017) show that the overfitting to rival agents' policies we observe is a common problem in RL. Zhang et al. (2018b) examine different ways how deep RL algorithms overfit to the environment and show that attempted solutions in the literature of adding stochasticity to the environment do not necessarily prevent overfitting. Closely related to our approach is a strand of literature that assumes there exists a distribution of Markov-decision-problems of the scenario of interest, and then trains algorithms on a finite set of samples from this distribution before testing the behavior on the entire distribution (e.g. Zhang et al., 2018a; Nichol et al., 2018; Justesen et al., 2018).

Lastly, our paper contributes to the literature on competition policy and regulatory responses to algorithmic collusion. The potential challenge to policy has been previously discussed both by the European Commissioner for Competition (Vestager, 2017) and Commissioner of the Federal Trade Commission (Ohlhausen, 2017), and potential solutions have been suggested (e.g. Calvano et al., 2020a; Harrington, 2018; Beneke and Mackenrodt, 2021). Our results provide some novel perspectives and qualify existing results: A key ingredient for collusive behavior appears to be that the learning phase is shared between opponents in the same market. Thus, most prone to possible collusion are markets where learning in the market itself is feasible and not too costly. This comprises markets with frequent interactions, small unit prices, and a very stable set of competing companies. Our results also suggest that the actual danger of algorithmic collusion may not necessarily be in the market interaction itself but in coordinative moves beforehand. For instance, when companies of an industry buy pricing-services from the same dominant upstream supplier, creating scenarios where collusion results may be more likely.[2] Particular attention may therefore be warranted when there is evidence of coordination of algorithm *design*.

The remainder of this paper is organized as follows. section 2 introduces the economic model of the environment (in subsection 2.1), and learning model of the algorithm (in subsection 2.3), and explains the matching across different-cost environments that we study in subsection 2.6. section 3 presents our results. **??** presents robustness checks of our analysis. Finally, section 4 concludes.

2 Learning and Market Environment Model

2.1 Economic model

We model the economic environment as a standard repeated Bertrand setting. The specification of the demand function and baseline parameterizations are in line with the setup employed in Calvano et al. (2020b). This provides a benchmark of existing findings of algorithmic collusion that our results can be directly compared to.

In each round of the game, each player i obtains their profit $\pi_i(p_{i,t}, p_{j,t}) = (p_{i,t} - c_i)q_{i,t}$, where $i \neq j$, $i, j \in \{1, 2\}$, and $p_{i,t}, p_{j,t}$ denote the period-t prices of players i and j respectively, $q_{i,t}, q_{j,t}$ the corresponding quantities, and c_i, c_j the marginal cost. The demand function is given by a classic logit-demand specification of

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

where a_i denotes the quality parameter of the good supplied by firm i (vertical differentiation) with $a_0 = 0$ being the product quality of the outside good, and μ the index of horizontal differentiation, so that the goods are perfect substitutes in the limit when $\mu \to \infty$. For our baseline parameterization we set $a_i - c_i = 1$ and $\mu = 1/4$

[2] This has been noted before, see e.g. Harrington Jr (2021) for a model of sellers that outsource their pricing algorithms to a third-party.

We consider constant marginal cost levels in the range $c_i, c_j \in [1, 1.7]$. We stop at 1.7 to ensure that either players' cost remain below the monopoly price of a seller with the lowest cost $c = 1$, so that it is never optimal for the market to be served by only one firm.

This specification of demand is particularly well-suited for our analysis. It implies that both the Nash equilibrium profits and the joint profit maximizing profits are constant across all cost levels and all combinations of cost, since $a_i - c_i = 1$. Only the associated optimal prices change. Thus, when an algorithm faces a different-cost competitor than previously, no change in the equilibrium price is required by the agent. Only a strategy to *support* high, supra-competitive profits will require a different set of prices, but can be obtained by appropriately 'shifting' the prices along the grid in line with the change in cost of the competitor. Hence, the challenge the algorithm faces in playing against a rival with different cost is particularly simple and the profits that can be obtained remain constant, ensuring that differences in the outcome observed are not due to a change in possible payoffs for the players.

2.2 Action space

The learning model we consider requires a discretization of the action space. We construct the grid of prices algorithms can choose from as follows.

Let the set of one-shot Nash equilibrium prices corresponding to the cost levels we consider be $\mathbf{p}^N$ and similarly let the set of joint profit maximizing prices be $\mathbf{p}^C$. Then the grid of prices available is given by $k = 20$ equally spaced points in the interval $[\underline{\mathbf{p}}^N - \xi(\bar{\mathbf{p}}^C - \underline{\mathbf{p}}^N), \bar{\mathbf{p}}^C + \xi(\bar{\mathbf{p}}^C - \underline{\mathbf{p}}^N)]$ with $\xi = 0.1$, where $\underline{\mathbf{p}}^N = [\min\{p \in \mathbf{p}^N\}]^2$ and $\bar{\mathbf{p}}^C = [\max\{p \in \mathbf{p}^C\}]^2$ The cost levels under consideration imply that $\underline{\mathbf{p}}^N \approx (1.47, 1.47)$ and $\bar{\mathbf{p}}^C \approx (2.62, 2.62)$.

2.3 Learning model

We employ Q-learning as our learning model, a standard model-free reinforcement learning algorithm. Q-learning was designed to tackle Markov-decision problems and attempts to learn the value of an action a among the set of actions A for each state s among the set of states S in order to maximize a cumulative reward function. The algorithm computes a so-called Q-function of expected rewards for an action taken in a given state – $Q : S \times A \to \mathbb{R}$.

Q-learning stores the (current) computed Q-value of each state-action pair in a table and hence requires the action and state space to be discrete. In each period, the cell in this Q-matrix corresponding to the current periods state-action combination, $Q_t(s_t, a_{i,t})$, is updated based on the observed reward in the current period, $\pi_t(a_{i,t}, a_{j,t})$), and a learning rate $\alpha \in [0, 1]$, according to the following Bellman equation

$$Q_{t+1}(s_t, a_{i,t}) = (1 - \alpha)Q_t(s_t, a_{i,t}) + \alpha(\pi_t + \delta \max_{a \in A} Q_t(s_{t+1}, a)),$$

where δ is the discount factor.

The initial state and the initial Q-matrix must be specified by the programmer at the start of the learning process. We initialize the Q-matrix with the Q-values that would arise if both agents were to play entirely random. We also choose the initial state at random.

In each period, the algorithm chooses an action (a price) either in order to *explore* the environment or to *exploit* its current state of knowledge. When exploring, the agent chooses an action at random. When exploiting, it chooses the action with the highest Q-value in the current state. We employ standard ε_t- greedy exploration in which the agent explores with probability ε_t and exploits with probability $1 - \varepsilon_t$. We let ε_t vary according to $\varepsilon_t = e^{-\beta t}$ where $\beta = 4 \times 10^{-6}$, implying that agents explore relatively often at the start and focus on exploiting over time. We focus on a learning rate of $\alpha = 0.15$ and specify the state space to be the previous period prices, $s_t = (p_{i,t-1}, p_{j,t-1})$, implying that agents have a one-period long memory.[3]

[3] This is in line with existing work in the literature, allowing our results to be directly compared. In addition, a larger memory significantly increases the size of the Q-matrix.

We stop the learning process when we observe that the algorithms have converged. Specifically, a given run is stopped if for each player i the action $a_{i,t} = \arg\max\{Q_{i,t}(a, s_t)\}$ does not change for 100000 consecutive periods, or after one billion total repetitions. We obtain convergence for over 99 percent of runs.

2.4 Cost of learning

Because the algorithms must explore the environment in order to learn an optimal policy, a firm employing such a learner faces possible short-term costs from exploration that must be balanced against the possible long-term gains from the algorithms chosen optimal policy.[4] A useful benchmark for this cost of learning is the competitive outcome, the one-shot Nash equilibrium.

Table 1 shows the profit loss relative to Nash play in our setting with the baseline parameters when both players randomize. Players with a high cost level consistently lose from randomizing, compared to playing the unique Nash equilibrium. But they also lose if the opponent randomizes and the agent itself plays Nash or the best-response to random play. Low cost players on the other hand *benefit* from randomized play and achieve above-Nash profits.

Table 1: Profit gain or loss in percentage of Nash-equilibrium profit

		Opponent Randomizes	
Cost Level	Both Randomize	Nash	Best-response
1.00	0.13%	0.63%	0.80%
1.10	0.09%	0.50%	0.67%
1.20	0.02%	0.36%	0.53%
1.30	-0.08%	0.32%	0.39%
1.40	-0.21%	0.17%	0.25%
1.50	-0.36%	0.04%	0.11%
1.60	-0.54%	-0.04%	-0.02%
1.70	-0.73%	-0.17%	-0.15%

This important fact illustrates a key aspect of the price grid definition: for low cost agents, almost all prices lie above the Nash equilibrium price and on average these will yield a higher profit than the Nash equilibrium. For high-cost types this is not the case, since the Nash price already lies in the upper part of the price grid and thus most prices in the grid yield below-Nash profit.

For intermediate and high cost players our setup thus captures the trade-off that firms face: employing pricing algorithms may cost in the short run due to exploration, but may pay off in the long run due to seemingly collusive play when exploiting. These costs can be avoided by training offline first. For low-cost types instead, there is no cost to exploration and we would therefore expect that these types will learn to converge to above-Nash prices.

Figure 8 in the Appendix shows the time to convergence. Agents consistently require more than 1 million rounds of play and on average over 2 million rounds to achieve convergence. Thus, learning online is likely infeasible in practice. In light of the potential high per-period cost for the firm from exploration, there may be hundreds of thousands of periods of significant losses before the algorithm begins achieving supra-Nash profits.

[4] See also the 'cold-start problem' of model-free machine learners in general, e.g. Zhu et al. (2019); Yuan et al. (2016); Liu et al. (2021); Ding and Soricut (2017).

2.5 Measures of Collusion

To assess the propensity of algorithms to collude, we focus on two measures of profit: the collusion index M and the profit gain Δ. Both express the realized profit in relation to the static Nash equilibrium and joint profit maximizing profits. Specifically, the two metrics are defined as

$$M = \frac{\bar{\pi} - \pi^N}{\pi^C - \pi^N},$$
$$\Delta_i = \frac{\bar{\pi}_i - \pi_i^N}{\pi_i^C - \pi_i^N},$$

where $\bar{\pi}_i$ denotes the average profit of agent i, π_i^N the profit of agent i in the one-shot Nash equilibrium of the game, and π_i^C the profit of agent i in the joint profit maximizing outcome of the one-shot game. $\bar{\pi}$, π^N, and π^C are defined analogously, but always represent the (average of the) *sum* of profits of players i and $j \neq i$, $i, j \in \{1, 2\}$.

Thus, for both the collusion index and an individual player's profit gain, when the respective measure is zero the average profit is equal to the Nash profit, while a value of one implies that the average profit is equal to the joint profit maximizing profit. Note that by definition, the collusion index is equal to the average of the profit gains of the two players.

In addition, we investigate the actions played by agents in more detail. We classify outcomes based on the unique convergence to specific actions. If both agents choose the same price in more than 90% of rounds, we classify the outcome as *symmetric* convergence. If both agents choose a unique but different price in more than 90% of rounds, we classify the outcome as *asymmetric* convergence. Finally, if at least one agent plays the same sequence of prices repeatedly in over 90% of rounds, we classify the outcome as a price *cycle*.[5] If we cannot identify either unique convergence or a price cycle, we classify the outcome as *other*. The vast majority of our learning runs converge to unique symmetric or asymmetric prices and we barely observe any longer price cycles.

2.6 Learning Environment vs. Market Environment

To study the asymmetry between the learning environment of a learning agent and the market environment in which the agent competes, following convergence in the learning environment we rematch players from different learning environments. In the 'market' environment, no learning takes place and agents are just exploiting, i.e. choosing the optimal action given their Q-matrix. In the initial learning environment, we always match agents with the same parameters.[6]

We begin our analysis of behavior in the market environment by matching players with different cost parameters. For example, if we have two separate, symmetric learning environments e_1 and e_2 each with two players, p_1^1, p_1^2 in the first environment, and p_2^1, p_2^2 in the second, we rematch players by matching p_1^1 with p_2^2 and p_1^2 with p_2^1 respectively. That is, we always pair player 1 from one environment with player 2 from a different environment, as well as vice versa across all cost environments.

Thus, the 8 symmetric cost level environments agents learn in initially lead to 56 asymmetric cost matches for rematching. As we run numerous individual sessions of learning algorithms in each initial cost environment, there are an exponential number of possible matches of agents for any two cost levels. We ensure that the number of sessions for each asymmetric cost rematch is equal to the number of sessions that we run in each symmetric environment, by always pairing players from the same respective session number.[7]

[5] In principle, we can search for price cycles of any length. However, in practice we limit ourselves to a search for cycles with a maximum length of 15 to avoid costly computations.

[6] As we will see later, this is not a critical assumption.

[7] Note that the sessions per cost level run independently of one another. The session number has no further implications, but since there are an exponential number of possible player-cost-session-number matches, using it is a straightforward way to limit computations.

The cost level of any agent remains the same before and after rematching. It is only the cost level of the competing player that may change. As discussed above, the specification of demand and the quality parameters ensure that after being matched to a player with a different cost level, in order to keep its profit constant all the algorithm needs to do is to appropriately 'shift' its optimal actions across the grid in response. Both the level of the Nash and the joint profit maximizing profit, as well as the associated prices, are unchanged.

3 Results

3.1 Cost asymmetries

Table 2 shows the summary statistics for the initial learning phase with symmetric costs. Across all cost levels, we observe a high collusion index and predominantly convergence to unique prices. The average collusion index for the low and intermediate cost levels are in line with previous estimates. The average collusion index for high cost runs is slightly lower. However, it continues to be at a supra-competitive level, showing that findings of algorithmic collusion extend to and are stable in environments in which exploration is costly, and short-run costs from learning must be balanced by long-term benefits from algorithmic collusion. We further observe almost no longer price cycles across all cost levels.

Table 2: Summary Statistics Initial Learning

			Convergence Type					
	Collusion Index				Cycle			
Cost Level	Mean	SD	Symmetric	Asymmetric	2	3	4	5+
1.00	0.79	0.16	79	52	59	27	23	10
1.10	0.81	0.13	89	51	65	26	12	7
1.20	0.87	0.11	159	20	39	15	9	8
1.30	0.87	0.10	171	23	36	9	4	7
1.40	0.84	0.10	168	36	42	3	0	1
1.50	0.75	0.12	164	49	32	5	0	0
1.60	0.70	0.13	156	50	38	5	1	0
1.70	0.73	0.15	154	52	40	3	1	0

Main results from the rematched runs are shown in Figure 1. The figure shows an 8×8 matrix corresponding to the cost levels of the two players. Each entry in the matrix shows the mean collusion index obtained over 250 individual sessions for a given match, i.e. cost level of player 1 (indicated along x-axis) and cost level of player 2 (indicated along the y-axis). Cost levels are increasing from left to right and bottom to top. Hence, the entries on the diagonal show the average when learning and market environment are symmetric, while the off-diagonals show the averages when the two environments are asymmetric.

Figure 1 shows that while the collusion index along the diagonal is in line with previous estimates and shows evidence of algorithmic collusion, off-diagonal the profit agents obtain is substantially smaller. For many cost-combinations the average profit of players is only slightly above the Nash profit. The definition of the price grid implies that both the Nash and joint profit-maximizing prices are never one of the grid points, and thus collusion indices equal to 0 or 1 are (almost) impossible to obtain. Hence, the smallest estimates in Figure 1 of up to 0.1 and above can therefore be interpreted as effectively Nash play.

We further quantify the effect of asymmetry between environments on the profit algorithms obtain by computing the average proportional loss, L, which is given by $L = (\bar{D} - \bar{O})/\bar{D}$, where $\bar{D}$ is the mean value of the diagonal and $\bar{O}$ is the mean value of the off-diagonal entries. The average proportional loss for Figure 1 is $L = 0.72$. Thus, even if all parameterization except the rivals marginal cost level stays constant between environments, the collusion index is 72% lower on average compared to the outcome in the training environment.

Figure 1: Collusion index

This effect is surprisingly stronger for small cost differences than for larger ones. As the two agents' cost levels become very similar, the collusion index approaches 0. That is, the profit players obtain on average approaches the Nash equilibrium profits. This pattern can be seen in the prices played by learners. We classify the off-diagonal matches by the absolute difference in cost levels between the two players, and the prices by their position on the price grid relative to a given players Nash price. Hence, for each learner in every match the Nash price has position zero on the grid, while all available prices on the grid have a positive value (greater than Nash) or negative value (lower than Nash) corresponding to the number of grid points to the Nash price. The result is shown in Figure 2.

We observe in Figure 2 that the distribution of prices played increasingly converges towards the Nash equilibrium price between -1 and 1 as the absolute difference in cost levels becomes smaller. That is, algorithms tend towards Nash play, if the cost level of the competitor is marginally different in the market environment compared to the learning environment.

In addition to the collusion index, we also document the average profit gain in Figure 3. This shows how profit gains are distributed between relatively high cost and low cost firms.

We observe that with relatively small asymmetries in cost levels, profit gains are also correspondingly distributed relatively evenly. But as cost differences become large, the low-cost player obtains an increasingly dominant share of the overall profits and approaches the profit gains obtained along the diagonal. We quantify the relative cost of being the high-cost competitor relative to being the low-cost competitor in the same match in an analogous way to the average proportional loss as $C = (\bar{C}_- - \bar{C}_+)/\bar{C}_-)$, where $\bar{C}_-$ is the average profit gain of player i when $c_i < c_j$, $j \neq i$, and $\bar{C}_+$ the average profit gain of player i when $c_i > c_j$. We find that for Figure 3, $C = 0.3$. However, this value rises to to 0.54 when considering only the largest cost differences.

3.2 Re-learning restores collusion

Our results show that findings of algorithmic collusion are very sensitive to changes in the environment and resulting asymmetry between learning and market environment. Collusive results disappear almost entirely with small parameter asymmetries, and instead we observe a tendency to play Nash prices. We now re-start the learning process for the

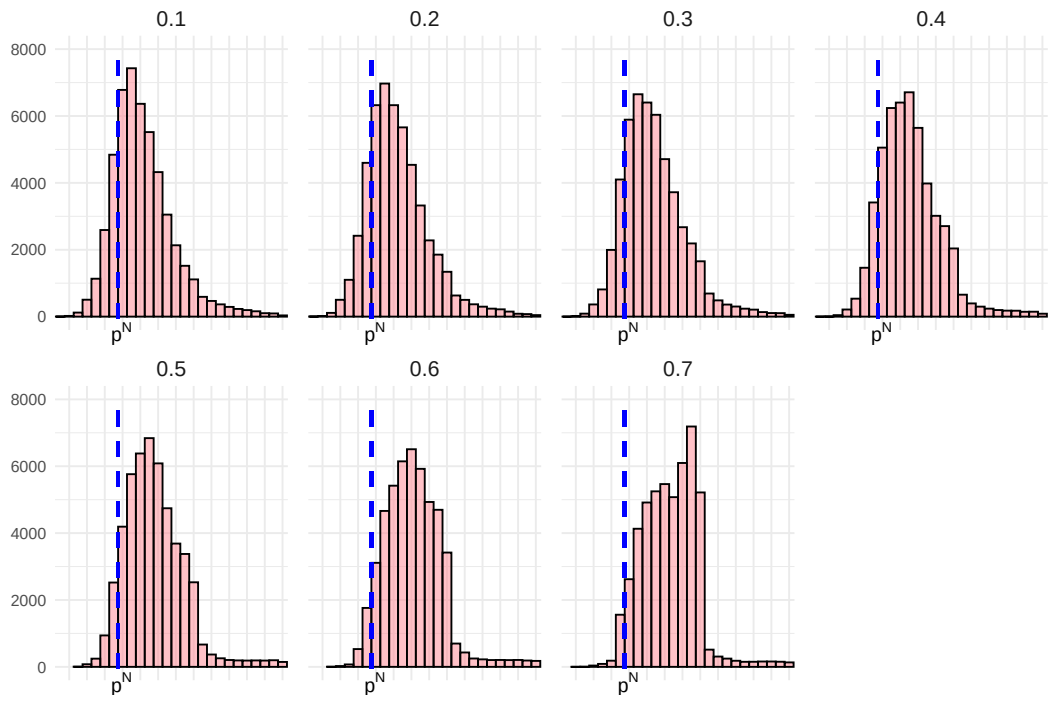


Figure 2: Price Distributions

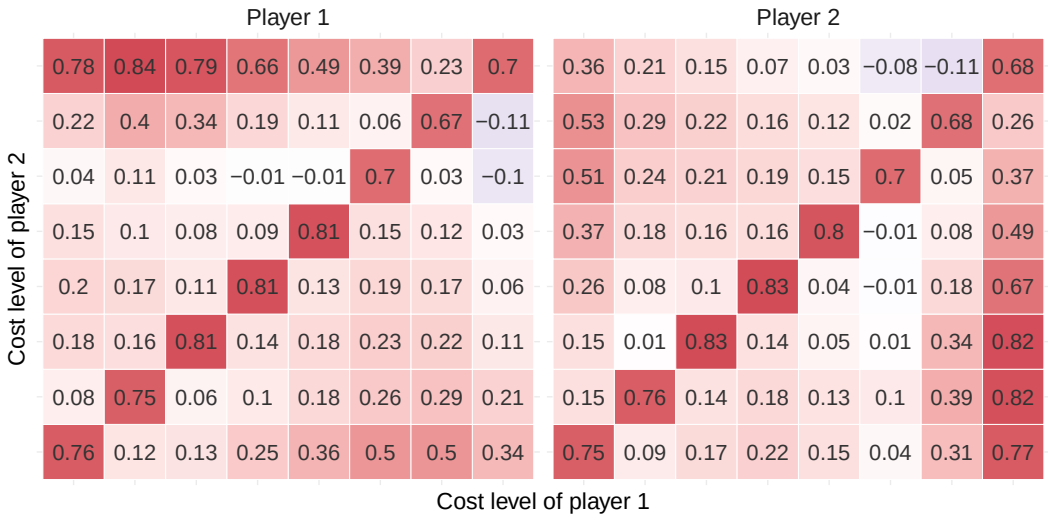


Figure 3: Profit gains by player.

matched algorithms in order to see if algorithms can learn to collude in asymmetric cost environments and overcome this breakdown of collusion. The summary statistics are reported in Table 3 and Table 4 in the Appendix.

Figure 4 shows the average collusion index for each match across the 250 sessions per match. We observe a complete return of collusive outcomes across all environments: agents consistently obtain very high profits irrespective of the exact cost levels, showing that asymmetric cost levels are no hindrance to algorithmic collusion in principle. We also observe that the highest returns arise with symmetric cost and that the larger the cost difference, the lower the collusion index after convergence. It appears that while high,

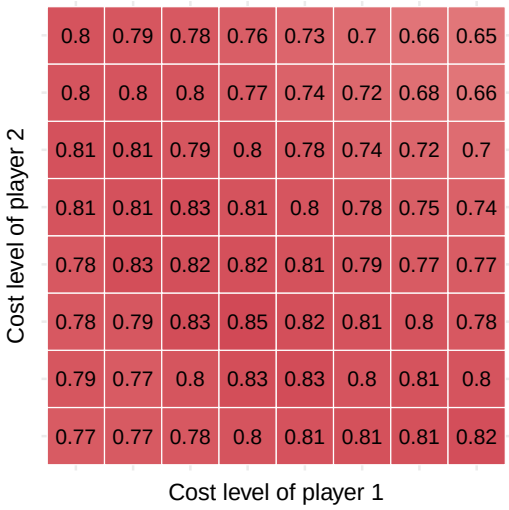


Figure 4: Collusion index after re-learning

seemingly collusive profits can always be obtained, more asymmetric agents find it harder to coordinate on a high price level.

This is reflected in the prices played. Figure 11 in the Appendix shows the distribution of prices across matches, and the respective Nash and joint-profit maximizing prices. For asymmetric cost pairs, for each player we observe convergence to the same distribution of prices that they converge to in their respective symmetric cost match.

We also report the time to convergence in Figure 8 in the Appendix. We observe on average significantly faster convergence in the re-learning phase compared to the initial learning phase. This shows that learning in the market becomes more feasible after previously learning offline, giving firms a possible avenue to overcome the (relatively) low profits when the market environment is different to the initial learning environment. However, learning continues to take on average more than a million rounds until convergence, likely making it still unfeasible and potentially costly in many markets.

3.3 Sequential asymmetric learning

Since re-learning can re-establish collusive outcomes, it may be that learning in multiple environments allows an algorithm to achieve collusive outcomes across environments. This would imply that a firm could train its algorithm in all possible environments in order to obtain a pricing policy robust to all possible competitors. We now show that this is not the case.

We sequentially match the same algorithm against different-cost competitors across all cost levels. Each time, we restart the learning process and following convergence study its behavior against competitors from all other cost levels.[8]

Figure 5 shows the mean value of the collusion index (in red), and the mean value of the profit gain for each player. With 8 cost levels, there are 8 separate learning phases and 8 subsequent exploit phases. We find that sequential learning across all possible parameterizations of the environment does not allow algorithms to establish collusion outside of their specific training environment. If anything, we observe a slight decrease in profits over the repeated learning phases. Thus, the breakdown of collusive outcomes we document is robust to repeated learning.

[8]We only investigate this for one specific cost level to limit computations. Otherwise, repeated learning for each cost level would lead to an exponential increase in possible asymmetric matches following convergence.

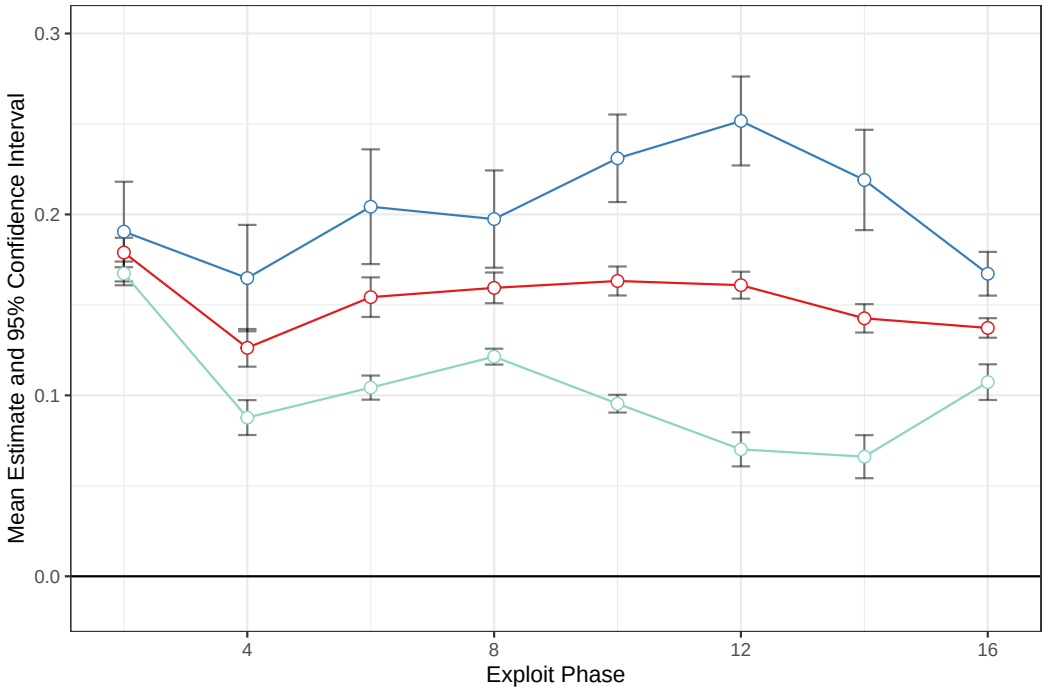


Figure 5: Outcomes after sequential learning and re-matching

We document the average collusion index and profit gains in Table 7 in the Appendix in detail. During each learning phase, the algorithms achieve high, collusive outcomes. As seen from Figure 5, when rematching across environments, each time there is no evidence of collusion. However, we also observe that the player repeatedly learning (shown in blue) consistently outperforms the opposing player, who only learnt once in the initial learning phase. Thus, there is a benefit from sequential learning.

3.4 Rematching identical agents

The breakdown of algorithmic collusion that we document is most severe when the structural asymmetry between algorithms is particularly small. We now show that the same finding arises when there are *no* structural differences. We run two instances of 250 sessions of the same experiment with symmetric cost differing only in the seed used to initialize the runs. To limit computations, we only consider three cost levels, low, medium, and high.[9]

Each of the two instances that we run yields converged policies for the two algorithms for each of the 250 sessions. We document our results in Figure 6. The entries of the matrix show the mean collusion index obtained across 250 sessions when players are matched across the two instances.

Along the diagonal, we observe the usual outcome of high, collusive profits. But across all three cost levels, off-diagonal we observe as before a breakdown of collusion with profits only slightly above Nash equilibrium profits. The average proportional loss over all three cost levels is equal to $L = 0.8$. Thus, even with an unchanged environment, playing against a previously unseen opponent results on average in a loss of 80% of the supra-competitive profits achieved before. This implies that the collusive outcomes observed and pricing policies learned are sensitive to the learning path.

[9]Since all our previous results are highly symmetric across cost levels and matches, we consider this to be without loss as it is highly likely that our findings here extrapolate to the remaining 5 cost levels.

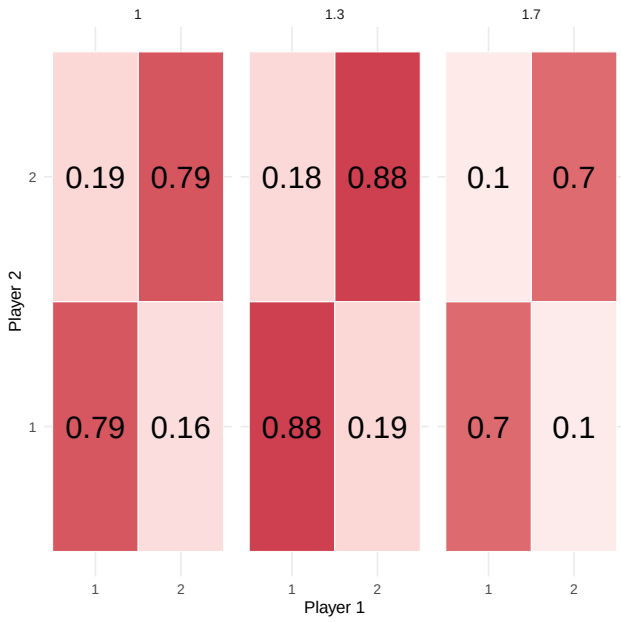


Figure 6: Mean Collusion Index

4 Conclusion

Companies worldwide increasingly make use of reinforcement learning (RL) and other machine learning techniques for their pricing decisions. The application of such tools and the resulting automation of pricing decisions has gained the attention of policy-makers and researchers. One major concern is that self-learning pricing algorithms may lead to collusion without any explicit instruction to do so by firms. A nascent literature that studies the behavior of RL in economic games indicates that concerns about algorithmic collusion are not unfounded and algorithms can indeed learn sophisticated strategies supporting supra-competitive pricing.

This paper shows that algorithms are unable to successfully use their learned, collusive policies outside their training environment. We train algorithms in environments with slight structural differences in the parameterization and match them following convergence to collusive policies. We find that collusion breaks down with even the smallest parameter asymmetries, and instead algorithms tend to play Nash prices. Re-learning in the new environment re-establishes collusive outcomes consistently, however it continues to be the case that algorithms are only able to collude in their latest training environment. We let algorithms learn repeatedly across the different environments and study their behavior across environments after each learning round. We observe a consistent breakdown of collusion in all but the most recent training environment.

We show that these findings are driven by algorithm overfit. We train algorithms in identical environments and match them across individual, separate runs. We find that when algorithms play against previously unseen competitors in an unchanged environment, they equally tend towards Nash play. This shows that the convergence proporties of the RL algorithms we consider depend on the learning path and the specific counterpart during learning. Our analysis illustrates a key challenge in the assessment of pricing algorithms: a testing environment is required that is not identical to the training environment. Our approach in this paper shows that economically relevant changes to the environment, and matching up converged algorithms across learning environments can serve as an artificial testing environment.

Our results provide a novel perspective on the existing findings of algorithmic collusion. A key ingredient for collusive behavior appears to be that the learning phase is shared

between opponents in the same market. Hence, when learning in the market is feasible, algorithms may be prone to collusion. In addition, we argue that our findings suggest that symmetry between algorithms and firms make algorithmic collusion more likely. Thus, policymakers may want to pay particular attention to any evidence of a coordination of algorithm design.

References

Abrardi, L., Cambini, C., and Rondi, L. (2021). Artificial intelligence, firms and consumer behavior: A survey. *Journal of Economic Surveys*, pages 1–23.

Beneke, F. and Mackenrodt, M.-O. (2021). Remedies for algorithmic tacit collusion. *Journal of Antitrust Enforcement*, 9(1):152–176.

Calvano, E., Calzolari, G., Denicolò, V., Harrington, J. E., and Pastorello, S. (2020a). Protecting consumers from collusive prices due to ai. *Science*, 370(6520):1040–1042.

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

Ding, N. and Soricut, R. (2017). Cold-start reinforcement learning with softmax policy gradient. *arXiv preprint arXiv:1709.09346*.

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

Harrington Jr, J. E. (2021). The effect of outsourcing pricing algorithms on market competition. *Available at SSRN 3798847*.

Justesen, N., Torrado, R. R., Bontrager, P., Khalifa, A., Togelius, J., and Risi, S. (2018). Illuminating generalization in deep reinforcement learning through procedural level generation.

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

Lanctot, M., Zambaldi, V., Gruslys, A., Lazaridou, A., Tuyls, K., Pérolat, J., Silver, D., and Graepel, T. (2017). A unified game-theoretic approach to multiagent reinforcement learning. *arXiv preprint arXiv:1711.00832*.

Liu, J., Zhang, Y., Wang, X., Deng, Y., and Wu, X. (2021). Dynamic pricing on e-commerce platform with deep reinforcement learning: A field experiment. *arXiv preprint arXiv:1912.02572*.

Nichol, A., Pfau, V., Hesse, C., Klimov, O., and Schulman, J. (2018). Gotta learn fast: A new benchmark for generalization in rl. *arXiv preprint arXiv:1804.03720*.

Ohlhausen, M. K. (2017). Should we fear the things that go beep in the night? some initial thoughts on the intersection of antitrust law and algorithmic pricing.

Song, X., Jiang, Y., Tu, S., Du, Y., and Neyshabur, B. (2019). Observational overfitting in reinforcement learning. *arXiv preprint arXiv:1912.02975*.

Vestager, M. (2017). Algorithms and competition. In *Bundeskartellamt 18th Conference on Competition, Berlin, 16 March 2017*.

Yuan, J., Shalaby, W., Korayem, M., Lin, D., AlJadda, K., and Luo, J. (2016). Solving cold-start problem in large-scale recommendation engines: A deep learning approach. In *2016 IEEE International Conference on Big Data (Big Data)*, pages 1901–1910.

Zhang, A., Ballas, N., and Pineau, J. (2018a). A dissection of overfitting and generalization in continuous reinforcement learning. *arXiv preprint arXiv:1806.07937*.

Zhang, C., Vinyals, O., Munos, R., and Bengio, S. (2018b). A study on overfitting in deep reinforcement learning. *arXiv preprint arXiv:1804.06893*.

Zhu, Y., Lin, J., He, S., Wang, B., Guan, Z., Liu, H., and Cai, D. (2019). Addressing the item cold-start problem by attribute-driven active learning. *IEEE Transactions on Knowledge and Data Engineering*, 32(4):631–644.

A Figures

Figure 7: Profit gain by player after re-learning

Figure 8: Time to convergence

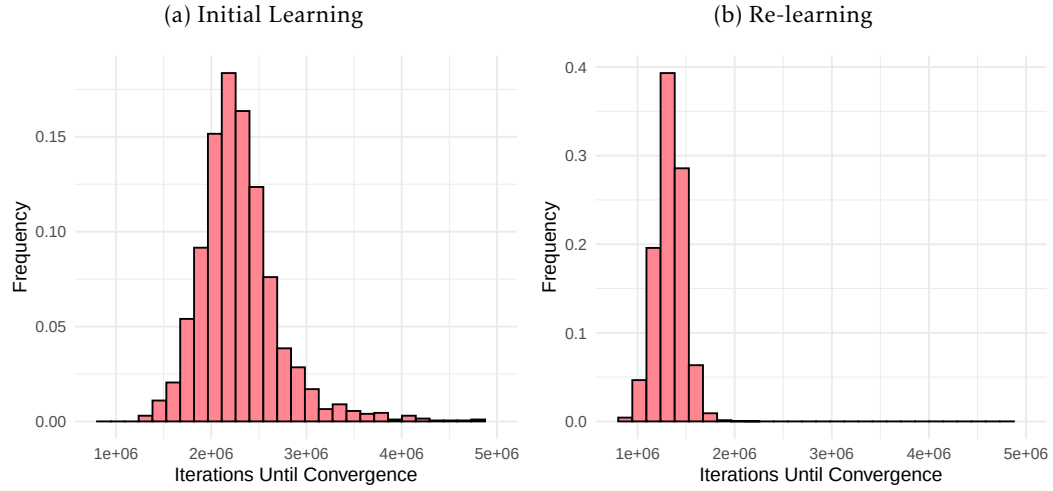
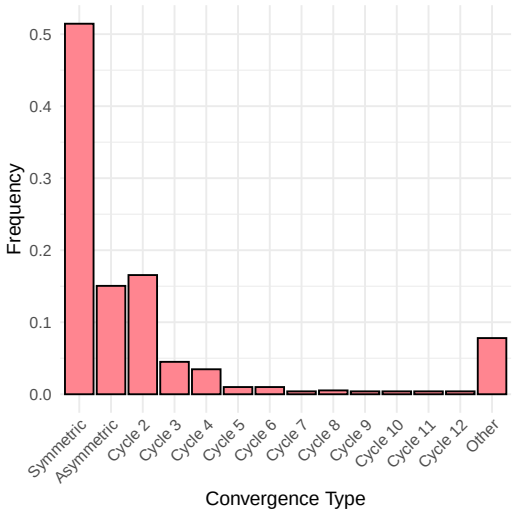
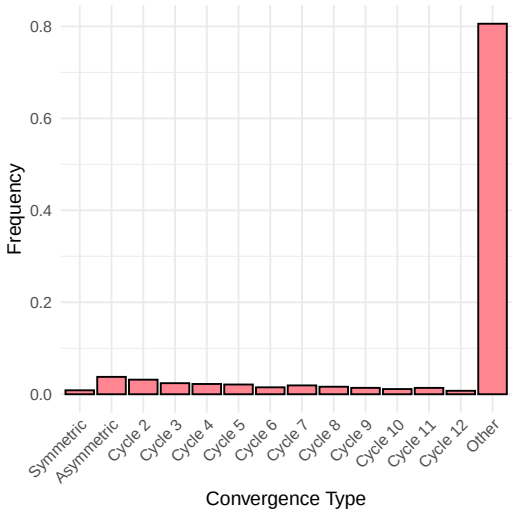


Figure 9: Convergence type frequency

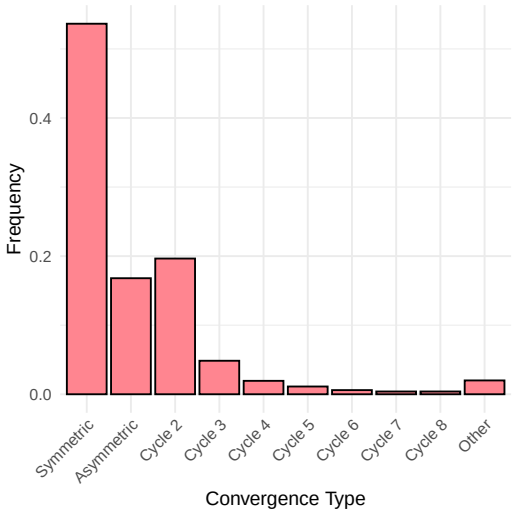
(a) On-diagonal after re-matching



(b) Off-diagonal after re-matching



(c) On-diagonal after re-learning



(d) Off-diagonal after re-learning

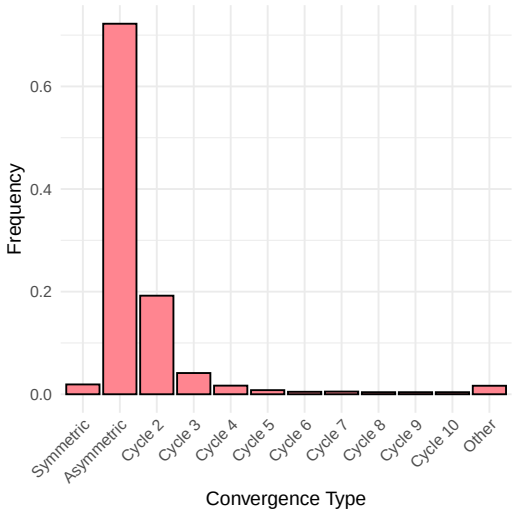


Figure 10: Frequency of prices after re-matching

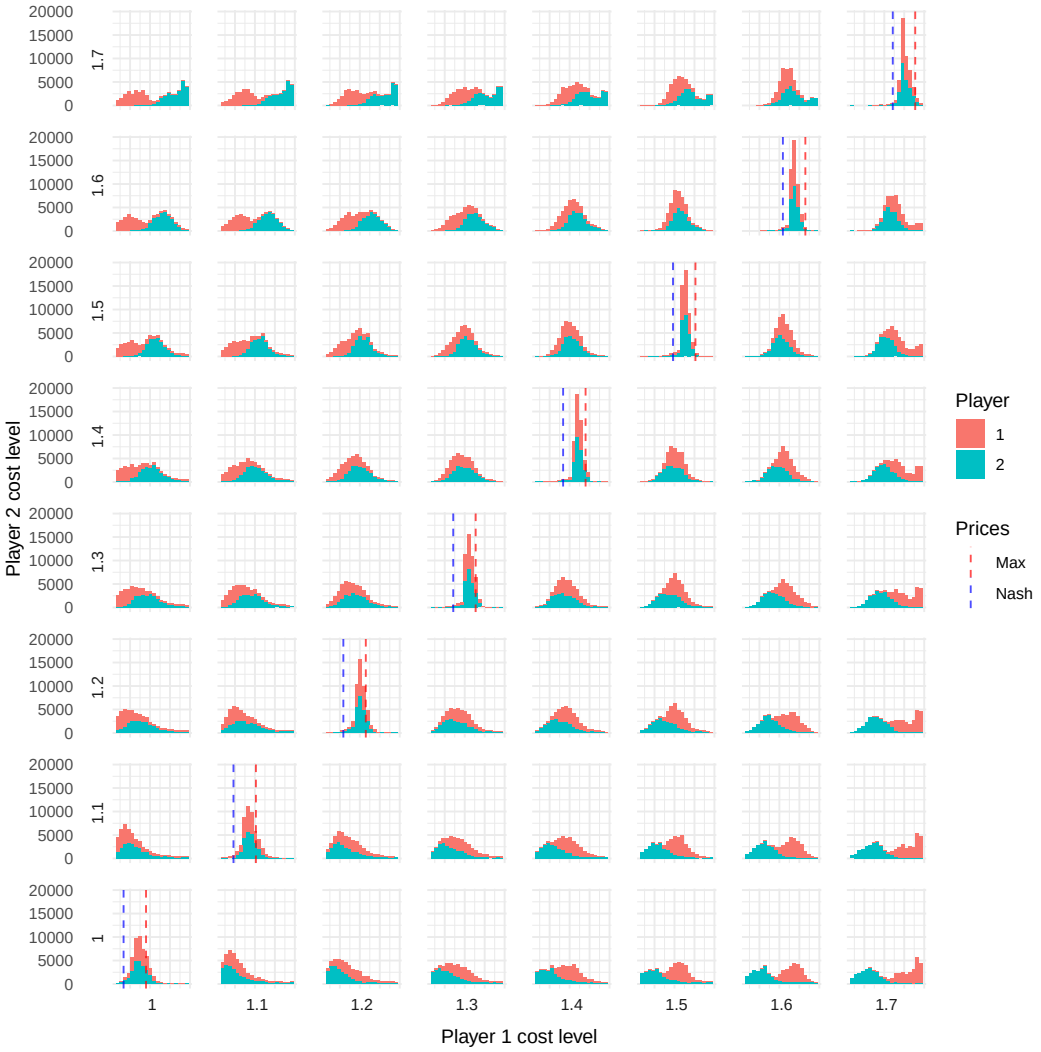


Figure 11: Frequency of prices after re-learning

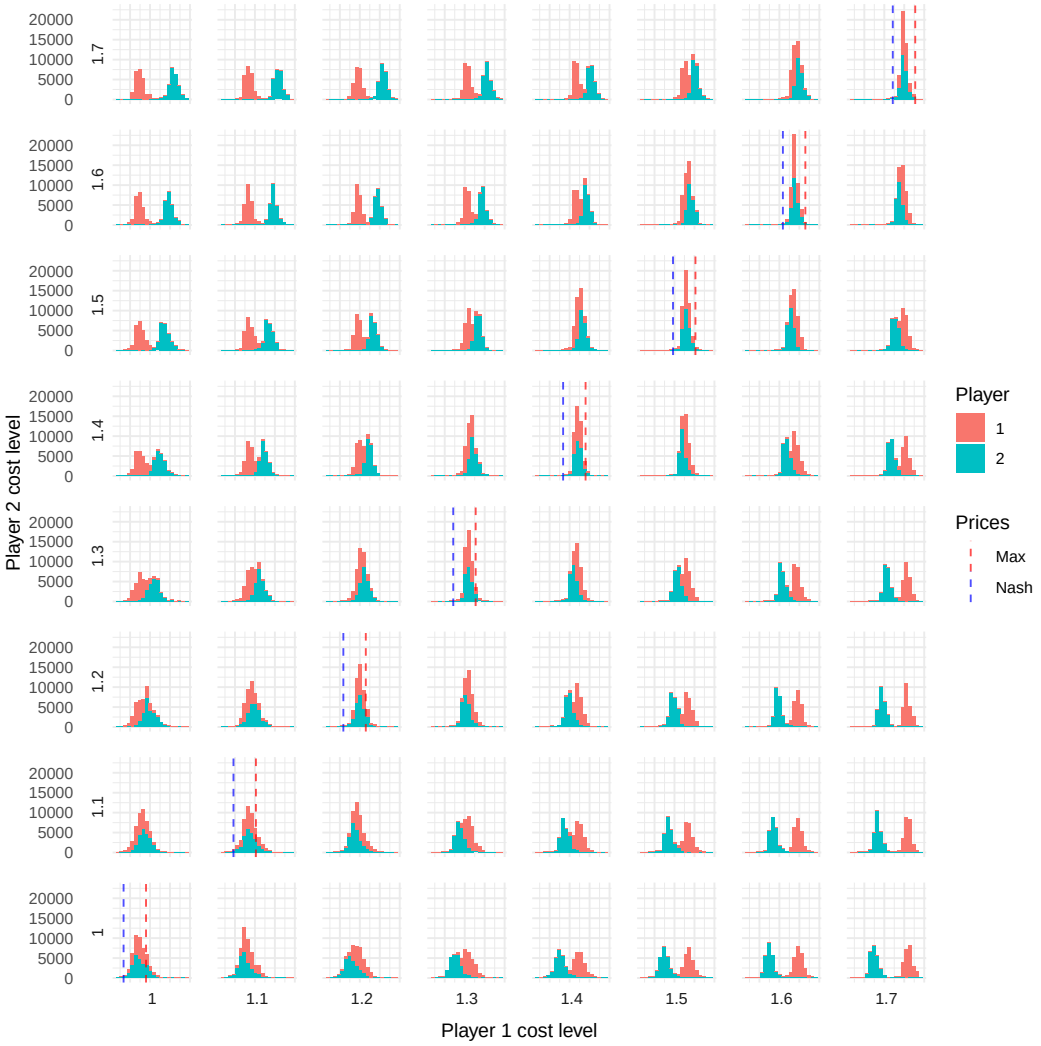
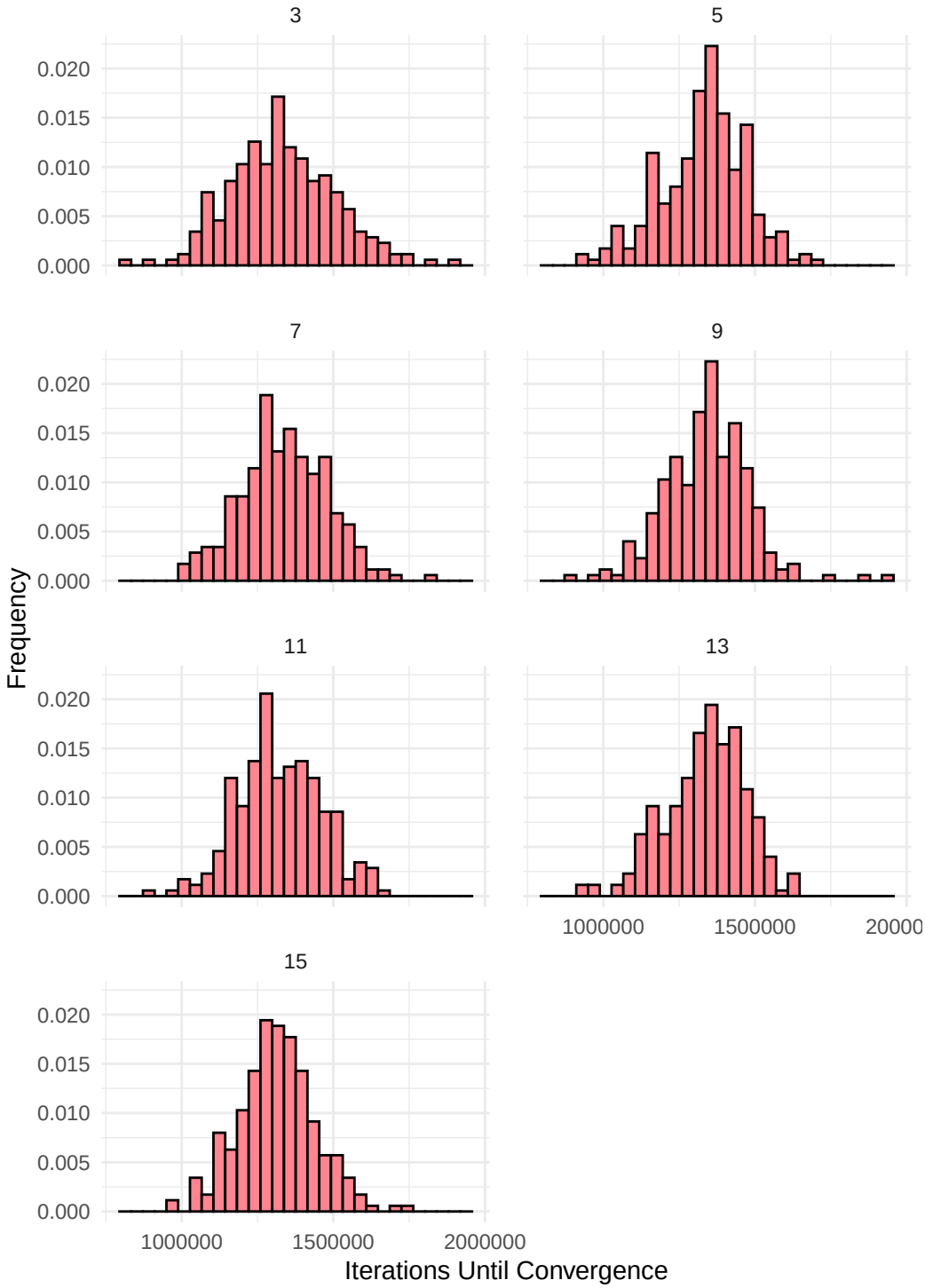


Figure 12: Time to convergence during sequential learning



B Tables

Table 3: Summary Statistics Symmetric Cost After Re-matching

Cost Player		Collusion Index		Convergence Type						
						Cycle				
1	2	Mean	SD	Symmetric	Asymmetric	2	3	4	5+	Other
1.00	1.00	0.76	0.19	73	48	51	27	25	12	14
1.10	1.10	0.76	0.22	80	45	62	24	12	6	21
1.20	1.20	0.82	0.19	144	15	39	16	8	8	20
1.30	1.30	0.82	0.17	150	25	33	9	5	7	21
1.40	1.40	0.80	0.19	158	30	38	3	0	3	18
1.50	1.50	0.70	0.19	147	43	28	4	0	3	25
1.60	1.60	0.67	0.18	140	47	40	4	1	0	18
1.70	1.70	0.69	0.20	137	48	40	3	1	2	19

Table 4: Summary Statistics Asymmetric Cost After Re-matching

Cost Player		Collusion Index		Convergence Type						
						Cycle				
1	2	Mean	SD	Symmetric	Asymmetric	2	3	4	5+	Other
1.00	1.10	0.11	0.18	0	5	6	1	0	2	236
1.00	1.20	0.16	0.16	0	4	1	1	1	1	242
1.00	1.30	0.23	0.17	0	5	3	2	1	5	234
1.00	1.40	0.26	0.17	0	13	1	4	3	9	220
1.00	1.50	0.28	0.19	0	17	8	5	2	13	205
1.00	1.60	0.37	0.19	0	24	11	8	2	26	179
1.00	1.70	0.57	0.23	0	25	30	21	19	56	99
1.10	1.00	0.11	0.16	2	3	2	1	1	5	236
1.10	1.20	0.08	0.17	2	2	0	0	1	1	244
1.10	1.30	0.13	0.18	0	3	5	2	1	3	236
1.10	1.40	0.14	0.16	0	3	4	3	2	3	235
1.10	1.50	0.17	0.17	0	17	1	1	4	12	215
1.10	1.60	0.35	0.21	0	25	8	15	9	30	163
1.10	1.70	0.53	0.22	0	18	32	25	18	68	89
1.20	1.00	0.15	0.16	0	3	1	3	0	1	242
1.20	1.10	0.10	0.20	3	5	1	3	0	3	235
1.20	1.30	0.11	0.18	2	2	1	0	2	4	239
1.20	1.40	0.12	0.19	0	10	0	2	1	4	233
1.20	1.50	0.12	0.16	0	8	7	0	4	8	223
1.20	1.60	0.28	0.18	0	15	11	7	2	38	177
1.20	1.70	0.47	0.17	0	5	24	14	23	77	107
1.30	1.00	0.24	0.22	0	14	6	4	2	4	220
1.30	1.10	0.14	0.21	0	7	4	2	0	4	233
1.30	1.20	0.14	0.19	2	4	4	0	1	8	231
1.30	1.40	0.12	0.21	0	7	5	2	2	7	227
1.30	1.50	0.09	0.18	0	7	3	4	1	6	229
1.30	1.60	0.18	0.13	0	6	5	6	0	20	213
1.30	1.70	0.36	0.18	0	7	12	13	11	69	138
1.40	1.00	0.25	0.19	0	17	3	1	1	7	221
1.40	1.10	0.16	0.18	0	6	5	1	0	9	229
1.40	1.20	0.11	0.16	0	5	0	4	0	3	238
1.40	1.30	0.09	0.17	0	4	1	0	1	6	238
1.40	1.50	0.07	0.19	3	4	3	4	0	7	229
1.40	1.60	0.11	0.13	0	3	5	5	3	12	222
1.40	1.70	0.26	0.15	0	8	8	8	6	57	163
1.50	1.00	0.27	0.17	0	16	5	6	3	11	209
1.50	1.10	0.18	0.16	0	10	4	5	4	9	218
1.50	1.20	0.12	0.14	0	4	1	3	3	13	226
1.50	1.30	0.09	0.16	0	5	1	1	3	8	232
1.50	1.40	0.07	0.18	0	5	0	4	1	8	232
1.50	1.60	0.04	0.15	0	3	5	4	3	10	225
1.50	1.70	0.16	0.15	0	10	5	5	5	27	198
1.60	1.00	0.41	0.21	0	31	18	7	10	29	155
1.60	1.10	0.34	0.20	0	26	14	12	7	33	158
1.60	1.20	0.28	0.19	0	19	8	6	14	17	186
1.60	1.30	0.17	0.13	0	4	5	5	2	20	214
1.60	1.40	0.10	0.13	0	3	3	2	3	16	223
1.60	1.50	0.04	0.16	0	9	4	1	2	8	226
1.60	1.70	0.06	0.12	0	1	4	1	6	15	223
1.70	1.00	0.55	0.22	0	19	39	17	11	61	103
1.70	1.10	0.52	0.22	0	19	32	17	15	64	103
1.70	1.20	0.47	0.16	0	5	21	14	29	79	102
1.70	1.30	0.36	0.17	0	8	13	9	10	70	140
1.70	1.40	0.26	0.15	0	9	4	9	8	56	164
1.70	1.50	0.14	0.14	0	8	3	2	3	30	204
1.70	1.60	0.07	0.13	1	4	2	5	2	15	221

Table 5: Summary Statistics Asymmetric Cost After Re-learning

Cost Player		Collusion Index		Convergence Type						
						Cycle				
1	2	Mean	SD	Symmetric	Asymmetric	2	3	4	5+	Other
1.00	1.00	0.77	0.14	51	72	72	25	12	9	9
1.10	1.10	0.77	0.16	87	64	60	15	6	6	12
1.20	1.20	0.83	0.11	154	25	38	20	6	3	4
1.30	1.30	0.82	0.10	154	25	45	12	5	4	5
1.40	1.40	0.80	0.11	159	35	38	10	2	3	3
1.50	1.50	0.74	0.11	164	33	44	7	0	1	1
1.60	1.60	0.68	0.13	154	43	44	5	2	0	2
1.70	1.70	0.65	0.13	150	39	52	3	1	1	4

Table 6: Summary Statistics Asymmetric Cost After Re-learning

Cost Player		Collusion Index		Convergence Type						
						Cycle				
1	2	Mean	SD	Symmetric	Asymmetric	2	3	4	5+	Other
1.00	1.10	0.79	0.14	17	129	63	20	8	6	7
1.00	1.20	0.78	0.13	0	148	63	26	7	4	2
1.00	1.30	0.78	0.15	0	158	57	17	6	5	7
1.00	1.40	0.81	0.12	0	162	51	18	8	3	8
1.00	1.50	0.81	0.11	0	165	60	13	7	2	3
1.00	1.60	0.80	0.11	0	178	51	13	4	1	3
1.00	1.70	0.80	0.11	0	183	48	8	4	4	3
1.10	1.00	0.77	0.13	22	135	62	19	6	2	4
1.10	1.20	0.79	0.13	6	159	50	19	6	6	4
1.10	1.30	0.83	0.12	0	180	46	8	6	3	7
1.10	1.40	0.81	0.12	0	176	53	14	2	1	4
1.10	1.50	0.81	0.11	0	175	52	10	6	2	5
1.10	1.60	0.80	0.10	0	194	43	5	3	2	3
1.10	1.70	0.79	0.11	0	177	59	7	3	0	4
1.20	1.00	0.78	0.14	1	139	68	24	5	9	4
1.20	1.10	0.80	0.12	8	159	50	13	12	3	5
1.20	1.30	0.82	0.10	1	175	47	13	8	4	2
1.20	1.40	0.83	0.10	0	196	37	10	3	1	3
1.20	1.50	0.79	0.10	0	193	39	7	3	1	7
1.20	1.60	0.80	0.11	0	191	43	10	1	3	2
1.20	1.70	0.78	0.11	0	175	60	5	3	2	5
1.30	1.00	0.80	0.13	0	170	50	18	6	1	5
1.30	1.10	0.83	0.11	0	184	40	10	8	3	5
1.30	1.20	0.85	0.10	1	183	48	8	3	3	4
1.30	1.40	0.81	0.10	1	188	45	6	1	3	6
1.30	1.50	0.80	0.10	0	183	46	12	2	1	6
1.30	1.60	0.77	0.11	0	188	48	7	4	2	1
1.30	1.70	0.76	0.11	0	188	47	8	4	1	2
1.40	1.00	0.81	0.12	0	164	57	10	8	5	6
1.40	1.10	0.83	0.10	0	167	61	11	6	1	4
1.40	1.20	0.82	0.10	0	189	39	13	2	4	3
1.40	1.30	0.81	0.10	0	187	42	10	6	2	3
1.40	1.50	0.78	0.10	2	196	43	4	1	1	3
1.40	1.60	0.74	0.11	0	185	55	5	0	1	4
1.40	1.70	0.73	0.12	0	194	45	8	1	0	2
1.50	1.00	0.81	0.11	0	167	47	23	5	2	6
1.50	1.10	0.80	0.11	0	180	47	9	5	2	7
1.50	1.20	0.81	0.11	0	193	36	13	3	0	5
1.50	1.30	0.79	0.11	0	196	38	10	3	1	2
1.50	1.40	0.78	0.10	4	205	33	5	0	0	3
1.50	1.60	0.72	0.12	1	196	41	6	1	1	4
1.50	1.70	0.70	0.12	0	192	46	4	3	0	5
1.60	1.00	0.81	0.11	0	177	41	16	5	4	7
1.60	1.10	0.81	0.11	0	188	40	12	3	2	5
1.60	1.20	0.80	0.10	0	189	46	7	0	4	4
1.60	1.30	0.77	0.11	0	202	38	5	0	1	4
1.60	1.40	0.75	0.11	0	197	40	7	1	1	4
1.60	1.50	0.72	0.11	1	195	46	6	0	0	2
1.60	1.70	0.66	0.12	1	195	52	1	0	0	1
1.70	1.00	0.82	0.11	0	188	46	8	3	1	4
1.70	1.10	0.80	0.11	0	184	47	10	4	3	2
1.70	1.20	0.78	0.11	0	191	46	6	3	0	4
1.70	1.30	0.77	0.11	0	191	46	7	1	1	4
1.70	1.40	0.74	0.11	0	197	43	6	1	0	3
1.70	1.50	0.70	0.12	0	193	51	2	0	0	4
1.70	1.60	0.66	0.12	1	182	52	7	2	0	6

Table 7: Summary Statistics Sequential Learning

Phase		Collusion Index		Profit Gain			
				Learning Player		Opposing Player	
Number	Type	Mean	SD	Mean	SD	Mean	SD
1	learn	0.79	0.14	0.79	0.19	0.79	0.19
2	exploit	0.18	0.20	0.19	0.35	0.17	0.25
3	learn	0.79	0.13	0.86	0.21	0.72	0.20
4	exploit	0.13	0.24	0.16	0.38	0.09	0.28
5	learn	0.82	0.12	0.82	0.18	0.81	0.17
6	exploit	0.15	0.22	0.20	0.38	0.10	0.27
7	learn	0.84	0.11	0.85	0.16	0.82	0.17
8	exploit	0.16	0.21	0.20	0.36	0.12	0.26
9	learn	0.82	0.10	0.81	0.17	0.83	0.15
10	exploit	0.16	0.20	0.23	0.33	0.10	0.26
11	learn	0.78	0.11	0.77	0.17	0.79	0.16
12	exploit	0.16	0.19	0.25	0.32	0.07	0.26
13	learn	0.77	0.11	0.77	0.16	0.77	0.15
14	exploit	0.14	0.19	0.22	0.33	0.07	0.27
15	learn	0.77	0.11	0.76	0.17	0.77	0.16
16	exploit	0.14	0.19	0.17	0.27	0.11	0.29

Table 8: Summary Statistics Identical Agents

				Convergence Type						
		Collusion Index				Cycle				
Cost	Instance	Mean	SD	Symm.	Asymm.	2	3	4	5+	Other
low	same	0.79	0.16	326	232	228	102	48	104	4
low	different	0.17	0.20	34	18	16	14	8	104	866
med	same	0.88	0.09	692	54	168	30	28	58	0
med	different	0.19	0.28	104	8	8	2	8	58	840
high	same	0.70	0.15	590	182	202	16	4	84	0
high	different	0.10	0.19	62	16	18	8	16	84	802

TODO