Pseudo-collusion due to algorithmic pricing with transient demand response

Lukasz Sliwinski[1], David Siska[1]

Abstract

The use of learning algorithms for automatic price adjustments is on the rise. However, these algorithms, developed under idealised conditions, can yield unexpected outcomes when put into practice. This paper presents simulations of pricing competitions between autonomous learning agents. We introduce a basic pricing algorithm and apply it within a standard pricing environment. We demonstrate that pricing algorithms that do not consider competitor prices may converge to pseudo-collusive equilibria. In these equilibria, the margins they charge are slightly higher than the competitive margins corresponding to the Nash equilibrium. Subsequently, we modify the environment to introduce a delay in the demand's reaction to price changes. With this transient demand response, the observed pricing inefficiency significantly increases. This study thus illustrates how model misspecification can lead to supra-competitive pricing induced by simple, stateless learning algorithms.

Keywords: algorithmic pricing; pseudo-collusion; model misspecification

[1]School of Mathematics, University of Edinburgh

1 Introduction

An increasing number of pricing decisions are now made by algorithms. Algorithmic pricing use cases range from high-frequency trading (Seth 2023), to online retail (Amazon) (Rathore 2022) and demand-based pricing of taxi service (Uber) (Singh 2024). Most recently, a fast food restaurant in the US has attracted attention for introducing surge pricing, where the price of a meal can change by the hour depending on the time of day and demand (Foroohar 2024).

Although algorithmic pricing can, in theory, provide benefits to the seller and the buyer (better price discovery), the benefits are not guaranteed. Specifically, when some learning algorithms continuously adapt to new information, their performance might be difficult to predict. There have been a few cases where the use of algorithms has led to market inefficiencies and supra-competitive pricing. In Assad et al. 2020, the authors inspected real-world data and showed that the introduction of automated pricing at gas stations is linked to higher gas prices with an increase ranging from 9% to 28%. The topic of algorithmic collusion was further investigated by numerical studies (Calvano et al. 2020; Calvano et al. 2021; Klein 2021; Kastius et al. 2022; Boer et al. 2022; Cartea, Chang, Penalva, et al. 2022) and empirical studies (Musolff 2022; Brown et al. 2023). Calvano et al. 2020 presented a numerical study in which the competition between Q-learning algorithms converged to tacit collusion. That is, dynamic equilibrium in which both agents quoted prices higher than one-stage Nash equilibrium prices and punished any deviations from those prices. Consequently, the authors concluded that the algorithms have learned to collude without explicit communication. While other studies such as Boer et al. 2022 contested the conclusion that the algorithms in fact learned to collude, it cannot be disputed that algorithmic pricing can lead to higher prices.

In this paper, we draw from the numerical experiments carried out in Calvano et al. 2020. Similarly, we perform simulations of pricing competitions between autonomous learning agents. However, we make a few important modifications. For the learning algorithm, we use a special type of multi-armed bandit algorithm. The agents do not possess any state and cannot observe the prices of the competitors. Secondly, we impose a constraint on the exploration of the algorithms so that the explored price remains within a specified distance from the current quoted price. The used algorithm is both rational and non-stationary. When the environment is stationary, the algorithm will converge towards a local maximiser of the reward. Furthermore, if the environment changes, the algorithm will converge towards a new maximiser.

In addition to the modifications to the learning algorithm, we introduce a modification to the pricing environment. Specifically, we allow for a transient demand response. We define steady-state demand as a demand calculated given fixed, stationary prices. Then, the true demand experienced by the market participants depends on the history of the theoretical steady-state demands. This is motivated by the intuition that customers cannot always easily observe, and thus compare prices from different competitors. As an example, let us consider gas stations. If a gas station increases the price of gas, customers cannot immediately identify that there is a gas station with a better offer. Change in customer behaviour may take time.

Calvano et al. 2020 defines tacit collusion as a supra-competitive pricing equilibrium in which agents punish opponents when they deviate from quoting high prices. Since the bandit agents used here do

not have state and do not observe competitor prices, they are not be able to explicitly respond to price deviations. Consequently, any potential price increase cannot be classified as tacit collusion. Therefore, this work uses the terms "psuedo-collusion" and "pricing inefficiency" to emphasise the distinction.

In addition to the literature regarding collusion, our work builds up on non-stationary, multi-armed bandit algorithms (Sutton et al. 1998; Garivier et al. 2008) and their application to pricing environments (Cartea, Chang, Mroczka, et al. 2022; Treetanthiploet et al. 2023; Sliwinski et al. 2024).

Our main contributions are as follows:

1. We demonstrate that even without transient demand response, autonomous agents that cannot observe the competitor prices can converge to supra-competitive equilibrium. In this equilibrium, the margins they charge slightly exceed the competitive margins corresponding to the Nash equilibrium.

2. We show that when the transient demand response is introduced into the environment, the observed pricing inefficiency can increase significantly, with the margins increasing by the order of 50% with respect to the competitive levels.

3. We conduct sensitivity analysis and indicate how the hyperparameters of the simulations (number of agents, market parameters, learning parameters, noise) influence observed pseudo-collusion.

The numerical experiments were implemented using Python. The code necessary to reproduce the obtained results can be accessed via a Github repository[1].

2 Setup

We model the market competition as a discrete simultaneous multi-agent stage game with N agents and T stages. We index the time with $t \in \{1, 2, 3, ...\}$ and the agents with $i \in \{1, 2, 3, ..., N\}$. At each time step t, each agent submits an action to the environment, which then returns the reward r_t^i. The goal of the agents is to maximise their own expected profit

$$E\left[\sum_{t=1}^{T} r_t^i\right]. \tag{2.1}$$

The agents do not have any prior knowledge and have to learn based on the observed own rewards only.

2.1 Economic environment

We start with a simple model of price competition with constant marginal costs and logit demand as in Calvano et al. 2020. Each agent has a constant cost c^i. At step t, the agents submit the

[1] See `https://github.com/ls2716/AlgorithmicPricingWithTransientDemand`.

prices $\mathbf{p}_t = \{p_t^1, p_t^2, ..., p_t^N\} \in \mathbb{R}^N$. Then, given the prices, the steady-state demands for the agents $\hat{\mathbf{d}}_t = \{\hat{d}_t^1, \hat{d}_t^2, ..., \hat{d}_t^N\} \in \mathbb{R}^N$ are calculated as follows

$$\hat{d}_t^i = \frac{\exp\left(\frac{a^i - p_t^i}{\mu}\right)}{\sum_{j=1}^{N} \exp\left(\frac{a^j - p_t^i}{\mu}\right) + \exp\left(\frac{a^0}{\mu}\right)}, \tag{2.2}$$

where $\mathbf{a} = \{a^1, a^2, ..., a^N\} \in \mathbb{R}^N$ are the agent quality indices, a^0 is the inverse index of aggregate demand and $\mu > 0$ is the index of horizontal differentiation. $\mathbf{a}$, a^0 and μ are fixed environment parameters and are not given to the agents. For conciseness, a^0 will be referred to as the "demand index" and μ as the "substitution index".

2.1.1 Delay mechanism

We now introduce a delay mechanism which encodes the possibility that the true demands $\mathbf{d}_t = \{d_t^1, d_t^2, ..., d_t^N\} \in \mathbb{R}^N$ at the current step depend on the history of the episode. The true demand will be equal to the average of the steady-state demands from the last m time steps (including the current step):

$$d_t^i := \frac{\sum_{k=\min(1,t-m+1)}^{t} \hat{d}_k^i}{\sum_{k=\min(1,t-m+1)}^{t} 1} \tag{2.3}$$

Then, the rewards for the agents, are calculated as:

$$r_t^i = (p_t^i - c^i) \cdot d_t^i. \tag{2.4}$$

All simulations presented in this work employ one of two possible values of the delay parameter m. Either there is no transient demand response ($m = 1$), or the true demands are two-step averages of steady-step demands from the current and previous step ($m = 2$).

2.1.2 Single-stage Nash equilibrium

Setting the demand delay m to 1 (no delay; true demands equal steady-state demands $\hat{\mathbf{d}}_t = \mathbf{d}_t$) and the action space to $\mathbb{R}$ (price can be any real number), we obtain a N-player single-stage pricing game. This pricing game has a unique Nash equilibrium. Furthermore, for any fixed competitor prices

$$p^{-i} := \{p^1, p^2, ..., p^{i-1}, p^{i+1}, ..., p^N\} \tag{2.5}$$

the reward function of i-th agent

$$r^i(p^i) = (p^i - c^i) \cdot d_t^i(p^i, p^{-i}) \tag{2.6}$$

has a unique maximiser and no other stationary points.

For a proof, see Sliwinski et al. 2024.

2.2 Pricing algorithm

The agents will follow a simple learning algorithm, a non-stationary sliding-window Eps-Greedy bandit with localised exploration. Algorithm 1 presents the algorithm. The bandit is initialised with an exploration rate ε, a finite action set $A \subset \mathbb{R}$, a width of the exploration range w, and a size of the sliding window τ. The action set is formed of equispaced real numbers (e.g. $[1., 1.01, 1.02, ..., 2.]$). Let us denote the j-th element of the action set as A^j.

The bandit maintains a time-indexed expected reward vector $\mathbf{m}_t \in \mathbb{R}^{[A]}$, initialised to 0. At step t, with probability $1-\varepsilon$, the bandit submits any action that maximises the expected reward from $\mathbf{m}_{t-1}$. The ties are broken randomly. With probability ε, the bandit explores. First, it chooses any maximising action $p_{t,\max}$ and then uniformly samples a random action from $[p_{t,\max} - w/2, p_{t,\max} + w/2] \cap A$.

After submitting the action and receiving the reward r_t, the expected reward vector is updated. Let us index the actions with j and let us denote the indicator function that j-th action was chosen at time step k as $I_k(j)$. Then:

$$m_t^j = \begin{cases} \frac{\sum_{k=\min(1,t-\tau+1)}^{t} I_k(j) r_k}{\sum_{k=\min(1,t-\tau+1)}^{t} I_k(j)} & \text{if } \sum_{k=\min(1,t-\tau+1)}^{t} I_k(j) > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{2.7}$$

That is, the expected reward is the average reward over the last τ steps with value 0 if an action was not taken in the last τ steps.

The algorithm design is based on the Eps-Greedy bandit algorithm from Sutton et al. 1998. The non-stationary modification, that is, the sliding window approach, is sourced from Garivier et al. 2008. The local exploration constraint is introduced for the purpose of this work. The application of the relatively simple Eps-Greedy with a sliding window is motivated by the previous work on Sliwinski et al. 2024 where this algorithm (without localised exploration) achieved the best performance in a competitive pricing problem.

Algorithm 1 Non-stationary sliding-window localised-exploration Eps-Greedy bandit algorithm

Input: environment, exploration rate ε, finite action set A, width of the exploration range w, sliding window size τ

Initialise $\mathbf{m}_0 = 0$.
for $t = 1, 2, ..., T-1, T$ steps **do**
    Sample the maximising action $p_{t,\max}$ uniformly from $\{A^j : j \in \arg\max_l m_{t-1}^l\}$.
    Sample X_t from $U(0, 1)$.
    if $X_t > \varepsilon$ **then**
        Submit the action $p_t := p_{t,\max}$.
    else
        Sample the action p_t uniformly from the set $A \cap [p_{t,\max} - w/2, p_{t,\max} + w/2]$ and submit it.
    end if
    Obtain the reward r_t and compute the expected reward vector $\mathbf{m}_t$ using Equation (2.7).
end for

We assume that in the action set, all the actions yield non-negative rewards. Then, the presented

bandit algorithm has following characteristics:

- The algorithm is rational. In a stationary environment with a reward function $R : \mathbb{R} \to \mathbb{R}$, the maximising action $p_{t,\max}$ will eventually reach a local maximiser of the reward function R i.e.

$$\exists_{t'} \forall_{t \geq t'} \forall_{p \in A \cap [p_{t,\max} - w/2, p_{t,\max} + w/2]} R(p_{t,\max}) \geq R(p). \tag{2.8}$$

  If the reward function has a unique maximiser and no other stationary points, the algorithm will converge to that maximiser.

- The exploration of the bandit is limited to the neighborhood of the best action. This considerably limits the temporal variations of the quoted price.

- The algorithm is non-stationary - if the reward function changes, the bandit will converge towards a new local maximiser.

3 Results

Having specified the environment and the agents, we run the simulations of the pricing competition. All simulations followed the same procedure:

1. Initialise the environment.
2. Initialise the agents.
3. For T steps:
   - i Get actions from the agents.
   - ii Submit the actions to the environment and receive the corresponding rewards.
   - iii Update the agents based on the observed rewards.

Although the setup allows for asymmetric agents with different costs and quality indices, all the simulations presented show symmetric cases where both the costs and quality indices are the same for all the agents.

3.1 Normalised profit and margin

To measure the collusion between the pricing algorithms, it is useful to juxtapose the collected rewards with the payoffs corresponding to the competitive Nash equilibrium and the monopolistic Pareto equilibrium, in which the agents work together to maximise the sum of the profits. Thus, in the presentation of the results, we compute a normalised reward per step Δ_t^i, calculated as

$$\Delta_t^i = \frac{r_t^i - r_{Nash}}{r_{Pareto} - r_{Nash}}, \tag{3.1}$$

where r_{Nash} is the Nash payoff and r_{Pareto} is the Pareto payoff.

The above definition works only when the Nash and Pareto payoffs are unique. This is satisfied for symmetric competition, where all agents quote the same price. For proof, see Appendix A. Note that we define the equilibria in the steady-state sense using steady-state demands.

In addition to the normalised profit, we also quantify the degree of collusion by comparing the quoted margin $s_t^i := p_t^i - c^i$ to the margin corresponding to the Nash equilibrium. For this purpose, we calculate the percent increase of the margin with respect to the Nash margin

$$\gamma_t^i := \frac{s_t^i - s_{Nash}}{s_{Nash}} \cdot 100\%, \tag{3.2}$$

where $s_{Nash} := p_{Nash} - c^1$ (costs are identical).

3.2 Long term behaviour analysis

In addition to price/reward graphs that display the time evolution of the pricing competition, we want to analyse the average long-term behaviour of the prices and rewards. Specifically, we want to inspect following quantities:

- average normalised reward $\overline{\Delta}$,
- standard deviation of the long-term reward $\mathrm{std}(r)$
- percent increase of the average margin with respect to the Nash margin $\overline{\gamma} = \frac{\overline{s} - \overline{s}_{Nash}}{\overline{s}_{Nash}} \cdot 100\%$,
- standard deviation of the percent increase in margin $\mathrm{std}(\gamma)$.

Averages and standard deviations were computed over different instances of simulations, over agents, and over time. There were $n_{sim} = 10$ simulations, each with $T = 20000$ time steps. To ensure that the values are not affected by the initial learning phase, the computation excluded the first $T_{in} := 5000$ time steps. Denoting the averaged quantity with overbar $\overline{(\cdot)}$, we can write:

$$\overline{(\cdot)} := \frac{1}{n_{sim} N (T - T_{in})} \sum_{j=1}^{n_{sim}} \sum_{i=1}^{N} \sum_{t=T_{in}+1}^{T} [(\cdot)_t^i]_j, \tag{3.3}$$

where $[\cdot]_j$ corresponds to j-th simulation instance. Then, for the standard deviation

$$\mathrm{std}(\cdot) := \sqrt{\frac{1}{n_{sim} N (T - T_{in})} \sum_{j=1}^{n_{sim}} \sum_{i=1}^{N} \sum_{t=T_{in}+1}^{T} \left(\overline{(\cdot)} - [(\cdot)_t^i]_j \right)^2}. \tag{3.4}$$

Note that the standard deviation does not indicate the error in the estimation of the average quantity. Rather, it indicates the spread of the rewards/margins.

3.3 Baseline simulations

We start with a baseline experiment. The following sections explore how the results might change for different parameters of the environment and the learning algorithms. Table 1 presents the parameters of the baseline experiment.

Environment		Bandit	
Parameter	Value	Parameter	Value
Number of agents N	3	Action set	$[1.001, 1.002, ..., 3.999, 4.]$
Quality indices $\mathbf{a}$	$[1, 1, 1]$	Exploration rate ε	0.25
Costs $\mathbf{c}$	$[1, 1, 1]$	Sliding window τ	50
Demand index a^0	-1	Exploration width w	0.01
Substitution index μ	0.25		
Delay m	$\in \{1, 2\}$		
Total steps T	$\in \{2000, 20000\}$		

Table 1: *Parameters for the baseline numerical experiment. The left part of the table shows the environment parameters, while the right part shows the parameters of the bandit algorithms.*

We aim to analyse the steady state of price competition. To decrease the time of initial learning, the maximising actions at the first step were set to the Nash prices. Figures 1 and 2 present the results of the simulations for a scenarios with no demand delay ($m = 1$) and where the true demand equals two-step average of steady-state demands ($m = 2$), respectively.

Table 2 presents the long-term averaged results obtained for the baseline experiment parameters.

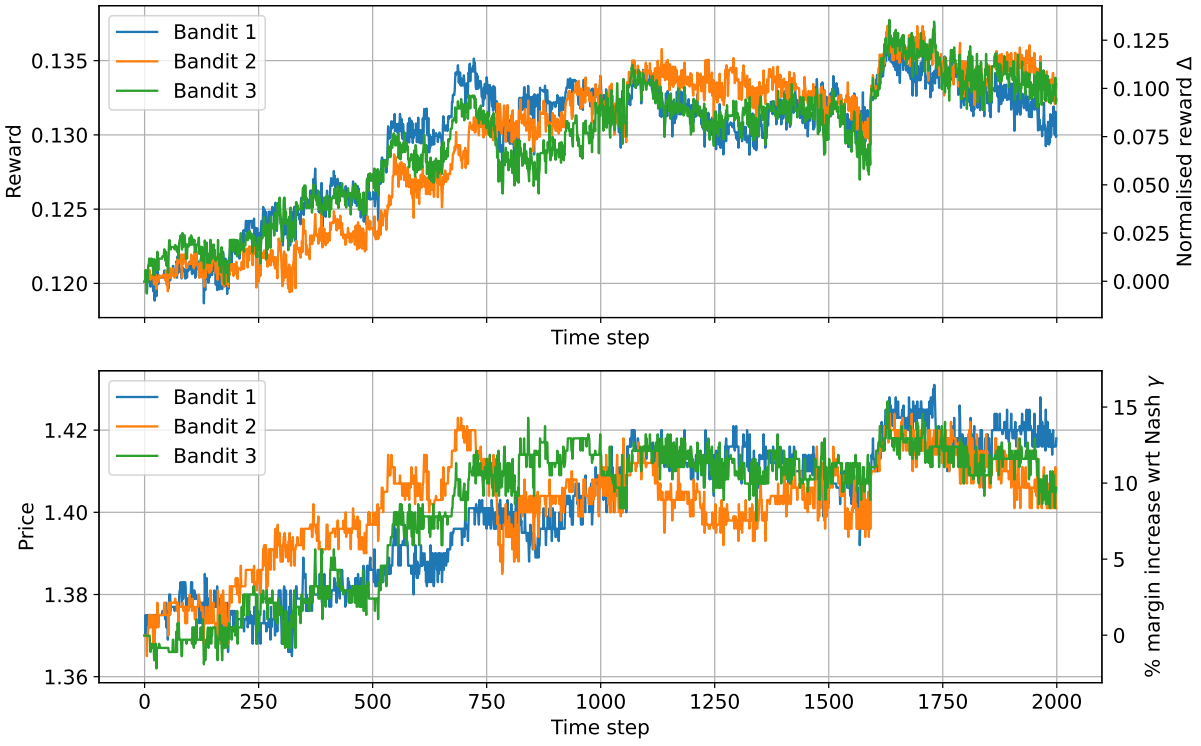

Figure 1: *Symmetric competition between three learning agents for an episode with $T = 2000$ and no delay in demand response ($m = 1$). The top plot shows the reward per step with the left axis indicating the reward and the right axis the normalised reward. The bottom plot shows the corresponding action - the left axis indicates the price, and the right axis the % increase of margin w.r.t. Nash margin. The initial action indicates the Nash price.*

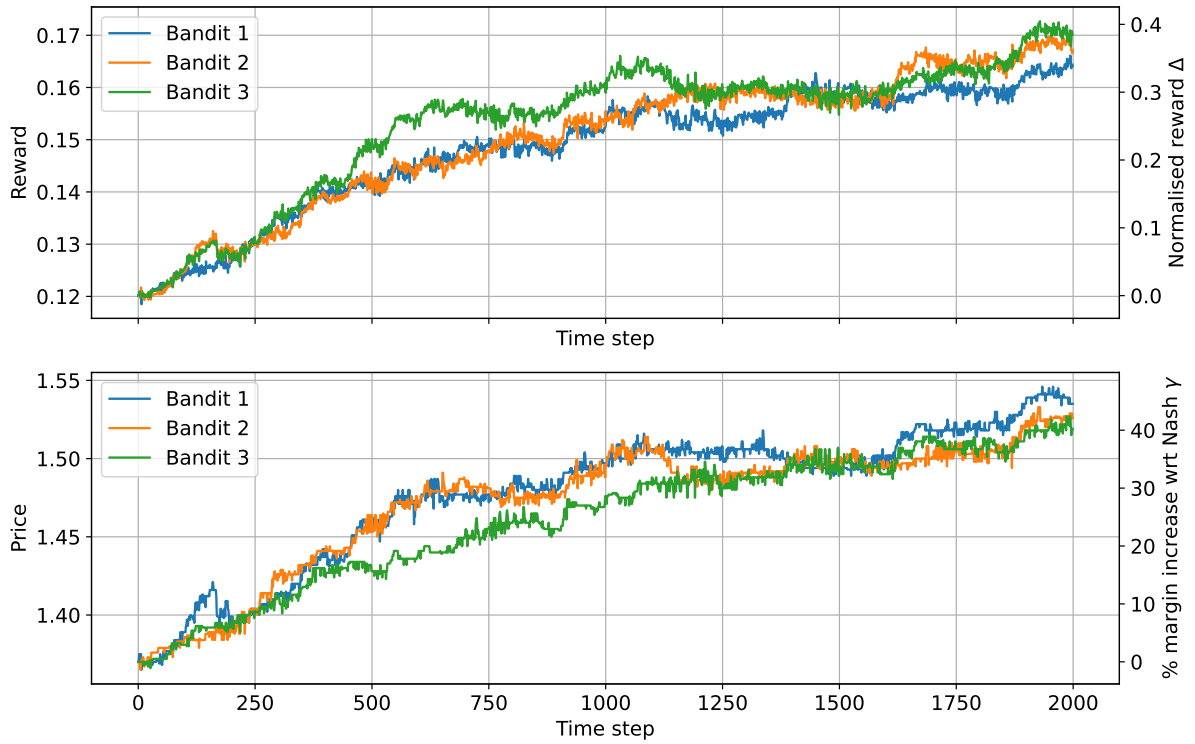

Figure 2: *Symmetric competition between three learning agents for an episode with* $T = 2000$ *and one step delay* ($m = 1$) *- the true demand is an average of the steady-state demands from the current and previous step. The top plot shows the reward per step with the left axis indicating the reward and the right axis the normalised reward. The bottom plot shows the corresponding action - the left axis indicates the price, and the right axis the % increase of margin w.r.t. Nash margin. The initial action indicates the Nash price.*

The plots in Figures 1 and 2 indicate that the agents' actions become correlated. Whenever there is an increase or decrease in the quoted price for one agent, it is accompanied by a similar increase/decrease in the price of the other agents. For the case with no delay, it can be seen that the agents leave the Nash equilibrium towards supra-competitive, inefficient pricing. Table 2 indicates that in the long term, the time-averaged margins increase by 8.2% which corresponds to 0.072 normalised reward $\overline{\Delta}$.

When a delay is added to the environment (Figure 2), we can still observe correlation between agents' actions. Moreover, the long-term increase in the average reward and price is significantly greater compared to the case without delay. In the long term, the average margins increase by 51% which corresponds to the normalised profit $\overline{\Delta} = 0.43$. We note that in the simulations, the delay included a two-step averaging window, which is much smaller than the agents' reward memory $\tau = 50$ (sliding window size).

The results above show that for the given learning algorithm, even when there is no delay in the demand response, the agents might converge to supra-competitive pricing. This pricing inefficiency can be greatly exacerbated if the demand response is delayed.

Delay m	Average normalised reward $\overline{\Delta}$ [std(Δ)]	Average % increase w.r.t. Nash margin $\overline{\gamma}$ [std(γ)]
1 (no delay)	0.072 [0.024]	8.2 [3.0]%
2 (2 step averaging)	0.432 [0.041]	51.1 [4.4]%

Table 2: *Average reward and price for the symmetric competition between three learning agents. For the given environment parameters, the Nash reward $r_{Nash} = 0.12$, Pareto reward $r_{Pareto} = 0.25$ and Nash margin $s_{Nash} = 0.37$.*

3.4 Number of agents

In this section, we analyse how the number of agents influences the averaged steady state reached in the pricing competition. We run simulations for the same parameters as in the baseline experiments, but with a different number of agents N. All quality indices and costs are still equal

$$a^1 = a^2 = ... = a^N = 1, \qquad c^1 = c^2 = ... = c^N = 1. \tag{3.5}$$

Table 3 presents the results for the case with no delay ($m = 1$) and Table 4 presents the results for the case with a one-step delay ($m = 2$). Firstly, we note that for one agent without any delay (first row of Table 3, the average collected reward $\overline{r}$ is equal to the Nash reward r_{Nash}, indicating that Algorithm 1 achieves the maximum reward (within the rounding error). The rest of the table shows that increasing the number of agents from $N = 2$ increases market competitiveness. Not only does it decrease the competitive Nash margin, but it also reduces pricing inefficiency.

When the delay is added to the environment (Table 4), we again observe a significant increase in prices. For a single agent (first row), the algorithm no longer finds the optimal Nash price. Instead, it increases the margin by 11% w.r.t. Nash margin and as a result receives a smaller reward. For $N \geq 2$, we observe margin increases of order 50%. While increasing the number of agents decreases pricing inefficiency, the effect remains significant. For simulation with 10 agents, the average margins are still 46% higher than the Nash level and the average normalised reward is 0.176.

Number of agents N	Nash payoff	Pareto payoff	Average normalised reward $\overline{\Delta}$ [std(Δ)]	Nash margin	Average % increase w.r.t. Nash margin $\overline{\gamma}$ [std(γ)]
1	0.552	0.552	NA ($\overline{r} = 0.552$)	0.802	0 [0.2]%
2	0.223	0.338	0.154 [0.045]	0.473	9.1 [2.6]%
3	0.120	0.250	0.072 [0.024]	0.370	8.2 [3.0]%
4	0.0814	0.201	0.048 [0.017]	0.331	7.3 [3.2]%
5	0.0615	0.169	0.035 [0.012]	0.312	6.3 [3.0]%
7	0.0413	0.130	0.021 [0.008]	0.291	4.8 [3.1]%
10	0.0276	0.0983	0.014 [0.006]	0.278	3.6 [3.0]%

Table 3: *Time-averaged rewards and margins in pricing competition with no delay for varying number of agents.*

Number of agents N	Nash payoff	Pareto payoff	Average normalised reward $\overline{\Delta}$ [std(Δ)]	Nash margin	Average % increase w.r.t. Nash margin $\overline{\gamma}$ [std(γ)]
1	0.552	0.552	NA ($\overline{r} = 0.540$)	0.802	11.5 [1.1]%
2	0.223	0.338	0.701 [0.069]	0.473	47.4 [4.0]%
3	0.120	0.250	0.432 [0.041]	0.370	51.1 [4.4]%
4	0.0814	0.201	0.322 [0.029]	0.331	49.8 [4.4]%
5	0.0615	0.169	0.269 [0.023]	0.312	48.9 [4.6]%
7	0.0413	0.130	0.215 [0.017]	0.291	47.4 [4.6]%
10	0.0276	0.0983	0.176 [0.012]	0.278	45.9 [4.4]%

Table 4: *Time-averaged rewards and margins in pricing competition with one-step delay* ($m = 2$) *for varying number of agents.*

3.5 Environment parameters

In this section, we analyse how the pricing inefficiency changes as we change the parameters of the environment, specifically the substitution index μ and the demand index a^0. For clarity, we will focus just on the percent increase of the average margin with respect to the Nash margin $\overline{\gamma}$ and the corresponding standard deviation std(γ).

We start with the parameters from the baseline simulation in Table 1 and then vary μ and a^0. Table 5 displays the equilibrium information for different environment parameters. This information is useful for understanding the relative pricing inefficiency. Table 6 presents the results for simulations with no delay, and Table 7 presents the results for simulations with one-step delay ($m = 2$). For simulations without delay, increasing a^0 decreases the pricing inefficiency, and this effect becomes more pronounced as the substitution index μ decreases. This result is particularly important as a^0 can be regarded as fixed competitors who do not vary their actions. Consequently, as more agents quote fixed (competitive) prices, the possibility of pricing inefficiency decreases. Table 6 appears to indicate that for low a^0 (high demand) increasing the substitution index μ (the products are substituted less easily) decreases the pricing inefficiency. However, looking at Nash margins in Table 5, increasing μ corresponds to a significant increase in the competitive margin. Therefore, when μ increases from 0.1 to 1, the inefficiency decreases from 9% to 6% but the Nash margin increases almost tenfold from 0.15 to 1.4. Thus, it is difficult to conclude whether μ has a positive or negative impact on pseudo-collusion.

The results from the simulations with one-step delay (Table 7) again indicate that the introduction of the delay corresponds to higher pricing inefficiency. Similarly to the case with no delay, increasing a^0 roughly corresponds to a decrease in pricing inefficiency, while the effect of μ is hard to gauge. However, as was the case with increasing the number of market participants, the pseudo-collusion due to the transient demand response is only moderately affected by the market parameters - the best-case margin is still 35% higher than the Nash level.

a^0 / μ	-2			-1			0			1		
	r_{Nash}	r_{Pareto}	s_{Nash}	-	-	-	-	-	-	-	-	-
0.1	0.050	0.575	0.150	0.050	0.267	0.150	0.019	0.020	0.119	0	0	0.1
0.25	0.125	0.522	0.375	0.120	0.250	0.370	0.047	0.050	0.297	0.017	0.017	0.252
0.5	0.240	0.500	0.740	0.196	0.270	0.696	0.095	0.101	0.595	0.022	0.022	0.522
1.0	0.393	0.539	1.393	0.299	0.350	1.299	0.189	0.210	1.189	0.099	0.100	1.099

Table 5: *Nash payoff (left value), Pareto payoff (middle value) and Nash margin (right value) for symmetric pricing competition with three agents and varying environment parameters.*

a^0 / μ	-2	-1	0	1
0.1	9.4 [5.0]%	9.3 [5.0]%	2.8 [3.8]%	0.0 [1.6]%
0.25	9.5 [3.0]%	8.2 [3.0]%	2.7 [2.2]%	0.0 [0.7]%
0.5	8.1 [2.2]%	6.6 [2.0]%	2.8 [1.5]%	0.6 [0.9]%
1.0	6.2 [1.3]%	4.6 [1.2]%	2.9 [1.0]%	1.5 [0.8]%

Table 6: *Average percent increase of margin* $\overline{\gamma}$ [*std*(γ)] *in pricing competition with three agents and no demand delay* ($m = 1$) *for varying environment parameters.*

a^0 / μ	-2	-1	0	1
0.1	54.9 [7.7]%	55.7 [8.0]%	37.8 [5.5]%	43.0 [6.2]%
0.25	54.1 [5.0]%	51.1 [4.4]%	37.3 [3.4]%	42.3 [3.8]%
0.5	50.1 [3.6]%	43.1 [2.6]%	37.1 [2.4]%	40.2 [2.7]%
1.0	40.5 [2.4]%	37.1 [2.2]%	35.3 [2.1]%	36.1 [2.7]%

Table 7: *Average percent increase of margin* $\overline{\gamma}$ [*std*(γ)] *in pricing competition with three agents and one-step demand delay* ($m = 2$) *for varying environment parameters.*

3.6 Learning algorithm parameters

In this section, we investigate how pricing inefficiency is influenced by the parameters of the learning algorithm. We again focus on the percent increase of the average margin with respect to the Nash margin.

For the simulations, we start with the parameters from the baseline simulation (see Table 1). We first vary the exploration parameter ε and then the size of the sliding window τ. Table 8 shows the results for the simulations with variable exploration rate ε while Table 9 shows the results for the simulations with variable sliding window size τ. Both tables include the results for the cases with no delay ($m = 1$) and the cases with one-step delay ($m = 2$). Inspecting Table 8, we observe that the pricing inefficiency is greatest for moderate values ε. Again, without the delay, margin increases are

of the order 7%, and when the delay is introduced, pricing inefficiency increases significantly and is only moderately affected by changes in the exploration rate ε.

Looking at Table 9 with the results corresponding to varying window size τ, we observe an interesting phenomenon. In the case without the delay, increasing τ increases the quoted margins. With the delay, increasing τ has the opposite effect - the pricing inefficiency is the greatest for the smallest value of τ and decreases as τ becomes larger. However, akin to the previous simulations, we observe that changing value of τ cannot prevent supra-competitive pricing.

delay \ ε	0.05	0.1	0.2	0.25	0.3	0.5
no delay	4.9 [1.9]%	7.2 [2.2]%	8.8 [3.0]%	8.2 [3.0]%	8.8 [3.3]%	7.3 [3.9]%
one-step delay	46.5 [5.2]%	51.4 [4.5]%	51.9 [3.8]%	51.1 [4.4]%	50.2 [4.2]%	48.2 [5.4]%

Table 8: *Effect of changing the exploration parameter ε on the average percent increase of the average quoted margin w.r.t. Nash margin $\overline{\gamma}$ [std(γ)] for pricing competition with three agents.*

delay \ τ	10	25	50	75	100	150
no delay	5.2 [3.1]%	6.9 [3.1]%	8.2 [3.0]%	9.1 [3.0]%	10.0 [3.4]%	10.3 [3.5]%
one-step delay	74.1 [5.3]%	58.1 [4.4]%	51.1 [4.4]%	49.1 [4.4]%	46.9 [4.6]%	45.1 [5.5]%

Table 9: *Effect of changing the sliding window size τ on the average percent increase of the average quoted margin w.r.t. Nash margin $\overline{\gamma}$ [std(γ)] for pricing competition with three agents.*

3.7 Adding noise

In the last section, we analyse the effect of noise on the pricing inefficiency. For this purpose, we modify the rewards received by the agents such that agent i's reward becomes:

$$r_t^i + l\eta_t^i, \tag{3.6}$$

where η_t^i follows a standard normal distribution $\mathcal{N}(0,1)$ and l is the noise level. Note that $\eta_t^1, ..., \eta_t^N$ are independent.

Table 10 reports the average percent increase in the quoted margin w.r.t. Nash margin for different levels of noise and for simulations with and without delay in the demand response. The table indicates that adding noise effectively prevents pseudo-collusion. For simulations without the transient demand response, when the noise level becomes of the order of the exploration range w, the average margins are almost equal to the competitive Nash margins.

For simulations with delayed demand response, we can still observe high pricing inefficiency, but it becomes significantly smaller as more noise is added to the reward. That said, when the noise level is large (of the order of the exploration width $w = 0.01$), we can argue whether the parameters of the bandit algorithm used for the simulations are still appropriate.

delay \ l	0.0002	0.0005	0.001	0.002	0.003	0.005	0.01
no delay	8.3 [3.0]%	6.9 [3.3]%	4.3 [3.3]%	1.8 [3.9]%	1.5 [4.8]%	1 [6]%	1 [8]%
one-step delay	50.1 [4.6]%	47.0 [4.2]%	41.8 [4.8]%	34.3 [5.4]%	30.6 [5.8]%	26 [7]%	20 [9]%

Table 10: *Effect of noise level l on the average percent increase of the average quoted margin w.r.t. Nash margin $\overline{\gamma}$ [$std(\gamma)$] for pricing competition with three agents. The results for $l \in \{0.005, 0.01\}$ were calculated across* 40 *different simulation instances due to high variance.*

4 Conclusions

In this work, we present simulations of pricing competition between bandit algorithms. The bandit algorithms, while simple, are rational, non-stationary, and with local exploration. When deployed, they converge to the local maximum of the reward function and can adapt to changes in the environment, at the same time limiting the temporal variation of the quoted price. Furthermore, we modify the standard logistic pricing environment so that it allows for a transient response of demand to the price changes. The demand used to calculate the rewards can depend on the price history, not just the current prices.

We show that in this simple setting, even without delay in demand response, the agents can converge to supra-competitive pricing and increase the quoted margins by around 8% with respect to competitive Nash levels.

When a small delay is added to the environment and the experienced demand is a two-step average of the current and previous true demand, the pricing inefficiency significantly increases. The quoted margins can increase by approximately 50%.

In addition to the baseline experiment, the study presents a simple sensitivity analysis of the pseudo-collusion due to simulation hyperparameters: the number of participating agents, environment parameters, learning parameters and noise in the observed reward. While the hyperparameters affect the pricing inefficiency for simulations with and without transient demand response, the effect of the delay is always significant. The factor that reduces the pricing inefficiency the most is the stochasticity of the reward. When the noise is of the order of the exploration width w, the observed pseudo-collusion is greatly diminished. Furthermore, since the pricing inefficiency is measured relative to the Nash equilibrium, the magnitude of price increases can be reduced by improving market competitiveness, particularly through increasing the number of market participants.

4.1 Further Work

An interesting avenue for future work would be the addition of past action memory to the agents while keeping the restrictions applied in this work. The agents should be ready for deployment, and their exploration should be constrained so that the temporal variation in the quoted price is still limited. A possible approach would be to use Q-learning (Watkins et al. 1992) or Soft-Actor-Critic (Haarnoja et al. 2018), identify state-action pairs that correspond to high price variations, and manually set their values to a large negative number. Such a study would be an extension of Calvano et al. 2020

and would provide insight into whether tacit collusion can arise when restrictions are imposed on the start-up and exploration of these algorithms.

References

Assad, Stephanie et al. (2020). "Algorithmic pricing and competition: Empirical evidence from the German retail gasoline market". In.

Boer, Arnoud V den, Janusz M Meylahn, and Maarten Pieter Schinkel (2022). "Artificial collusion: Examining supracompetitive pricing by Q-learning algorithms". In: *Amsterdam Law School Research Paper* 2022-25.

Brown, Zach Y and Alexander MacKay (2023). "Competition in pricing algorithms". In: *American Economic Journal: Microeconomics* 15.2, pp. 109–156.

Calvano, Emilio et al. (2020). "Artificial intelligence, algorithmic pricing, and collusion". In: *American Economic Review* 110.10, pp. 3267–3297.

— (2021). "Algorithmic collusion with imperfect monitoring". In: *International journal of industrial organization* 79, p. 102712.

Cartea, Álvaro, Patrick Chang, Mateusz Mroczka, et al. (2022). "AI-driven liquidity provision in OTC financial markets". In: *Quantitative Finance* 22.12, pp. 2171–2204.

Cartea, Álvaro, Patrick Chang, José Penalva, et al. (2022). "Learning to Collude: A Partial Folk Theorem for Smooth Fictitious Play". In: *Available at SSRN*.

Foroohar, Rana (Mar. 10, 2024). "To surge or not to surge, the algorithm is the question". In: *Financial Times*. URL: `https://on.ft.com/4a9xGmc` (visited on 03/25/2024).

Garivier, Aurélien and Eric Moulines (2008). "On upper-confidence bound policies for non-stationary bandit problems". In: *arXiv preprint arXiv:0805.3415*.

Haarnoja, Tuomas et al. (2018). "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor". In: *International conference on machine learning*. PMLR, pp. 1861–1870.

Kastius, Alexander and Rainer Schlosser (2022). "Dynamic pricing under competition using reinforcement learning". In: *Journal of Revenue and Pricing Management* 21.1, pp. 50–63.

Klein, Timo (2021). "Autonomous algorithmic collusion: Q-learning under sequential pricing". In: *The RAND Journal of Economics* 52.3, pp. 538–558.

Musolff, Leon (2022). "Algorithmic pricing facilitates tacit collusion: Evidence from e-commerce". In: *Proceedings of the 23rd ACM Conference on Economics and Computation*, pp. 32–33.

Rathore, Waleed (2022). *Dynamic Pricing on Amazon: A Brief and Easy Explainer*. `https://www.zonguru.com/blog/dynamic-pricing-on-amazon`. [Online; accessed 26-March-2024].

Seth, Shobhit (2023). *Basics of Algorithmic Trading: Concepts and Examples*. `https://www.investopedia.com/articles/active-trading/101014/basics-algorithmic-trading-concepts-and-examples.asp`. [Online; accessed 26-March-2024].

Singh, Abhijeet P. (2024). *Uber Pricing Strategy*. `https://ginzerr.com/uber-pricing-strategy/`. [Online; accessed 25-March-2024].

Sliwinski, Lukasz et al. (2024). "Competitive Insurance Pricing Using Model-Based Bandits". In: *Available at SSRN: https://ssrn.com/abstract=4755027*.

Sutton, RS and AG Barto (1998). *Reinforcement learning: An introduction. Cambridge, MA: MIT Press.*

Treetanthiploet, Tanut et al. (2023). "Insurance pricing on price comparison websites via reinforcement learning". In: *arXiv preprint arXiv:2308.06935.*

Watkins, Christopher JCH and Peter Dayan (1992). "Q-learning". In: *Machine learning* 8, pp. 279–292.

Declarations

Funding

Lukasz Sliwinski was supported by the EPSRC Centre for Doctoral Training in Mathematical Modelling, Analysis and Computation (MAC-MIGS) funded by the UK Engineering and Physical Sciences Research Council (grant EP/S023291/1), Heriot-Watt University and the University of Edinburgh.

Code availability

Numerical experiments presented in this work have been implemented in Python. All the code necessary to reproduce the results can be accessed at the Github repository `https://github.com/ls2716/AlgorithmicPricingWithTransientDemand`.

A Proofs of symmetric equilibrium uniqueness

A.1 Proof of uniqueness of symmetric Nash equilibrium

We want to prove that for a symmetric environment with $a^1 = a^2 = ... = a^N := a$ and $c^1 = c^2 = ... = c^N =: c$ there exists exactly one symmetric Nash equilibrium $\mathbf{p}^* = \{p^*, p^*, ..., p^*\} \in \mathbb{R}^N$. Specifically, we have to show that there exists exactly one p^* such that the expression

$$r(p, p^*) := \frac{(p-c)\exp\left(\frac{a-p}{\mu}\right)}{\exp\left(\frac{a-p}{\mu}\right) + \sum_{j=2}^{N}\exp\left(\frac{a-p^*}{\mu}\right) + \exp\left(\frac{a^0}{\mu}\right)} \tag{A.1}$$

has a unique global maximiser w.r.t p which is equal to p^*.

We start by computing the derivative with respect to p

$$\frac{\partial r}{\partial p} = \frac{\left[-\frac{1}{\mu}(p-c)\exp\left(\frac{a-p}{\mu}\right) + \exp\left(\frac{a-p}{\mu}\right)\right]A + \frac{1}{\mu}(p-c)\exp\left(\frac{a-p}{\mu}\right)\exp\left(\frac{a-p}{\mu}\right)}{A^2}, \tag{A.2}$$

where

$$A := \exp\left(\frac{a-p}{\mu}\right) + \sum_{j=2}^{N}\exp\left(\frac{a-p^*}{\mu}\right) + \exp\left(\frac{a^0}{\mu}\right). \tag{A.3}$$

We equate the derivative to 0, divide by $\exp\left(\frac{a-p}{\mu}\right) > 0$ and multiply by $A^2 > 0$. Then

$$0 = -\frac{1}{\mu}(p-c)\left[A - \exp\left(\frac{a-p}{\mu}\right)\right] + A \tag{A.4}$$

Now, we note that

$$A - \exp\left(\frac{a-p}{\mu}\right) = \sum_{j=2}^{N}\exp\left(\frac{a-p^*}{\mu}\right) + \exp\left(\frac{a^0}{\mu}\right) \tag{A.5}$$

is positive and constant with respect to p. Furthermore,

$$\left[A - \exp\left(\frac{a-p}{\mu}\right)\right] > 0 \tag{A.6}$$

and

$$\frac{\partial}{\partial p}\exp\left(\frac{a-p}{\mu}\right) < 0 \implies \frac{\partial A}{\partial p} < 0. \tag{A.7}$$

(Note that $\mu > 0$.)

Therefore

$$-\frac{1}{\mu}(p-c)\left[A - \exp\left(\frac{a-p}{\mu}\right)\right] + A \tag{A.8}$$

is monotonically decreasing with respect to p. We then see that

$$\lim_{p\to-\infty}\frac{\partial r}{\partial p} = \infty \quad \text{and} \lim_{p\to+\infty}\frac{\partial r}{\partial p} = -\infty. \tag{A.9}$$

Thus, $\frac{\partial r}{\partial p} = 0$ has a unique solution and so r has a unique maximiser with respect to p.

Let us denote the maximiser as p' and note that it is a function of p^*. Next, we have to prove that there is exactly one solution to

$$p'(p^*) = p^*. \tag{A.10}$$

To show that, we first write the implicit equation for p' and p^* (A is a function of p^*):

$$-\frac{1}{\mu}(p' - c)\left[A - \exp\left(\frac{a - p'}{\mu}\right)\right] + A = 0. \tag{A.11}$$

Since $\exp\left(\frac{a-p'}{\mu}\right) > 0$, we have

$$\frac{1}{\mu}(p' - c)\left[A - \exp\left(\frac{a - p'}{\mu}\right)\right] = A - \exp\left(\frac{a - p'}{\mu}\right) + \exp\left(\frac{a - p'}{\mu}\right) \tag{A.12}$$

$$\frac{1}{\mu}(p' - c) - 1 > 0. \tag{A.13}$$

We now take a derivative of Equation (A.11) with respect to p^*. We get

$$\begin{aligned} -\frac{1}{\mu}\frac{\partial p'}{\partial p^*}\left[A - \exp\left(\frac{a - p'}{\mu}\right)\right] + \frac{1}{\mu^2}(p' - c)(N-1)\exp\left(\frac{a - p^*}{\mu}\right) \\ - \frac{1}{\mu}(N-1)\exp\left(\frac{a - p^*}{\mu}\right) - \frac{1}{\mu}\frac{\partial p'}{\partial p^*}\exp\left(\frac{a - p'}{\mu}\right) = 0. \end{aligned} \tag{A.14}$$

Solving for the partial derivative, we get

$$\frac{\partial p'}{\partial p^*} = \frac{\left[\frac{1}{\mu}(p' - c) - 1\right](N-1)\exp\left(\frac{a-p^*}{\mu}\right)}{A}. \tag{A.15}$$

Now, Inequality (A.13) implies $\frac{\partial p'}{\partial p^*} > 0$ always.

Furthermore, from Inequality (A.13), we know that $p' > c$ and by inspection

$$\lim_{p^* \to \infty}\left[-\frac{1}{\mu}(p' - c)\left(A - \exp\left(\frac{a - p'}{\mu}\right)\right) + A\right] = -\frac{1}{\mu}(p' - c)\exp\left(\frac{a^0}{\mu}\right) + \exp\left(\frac{a - p'}{\mu}\right) + \exp\left(\frac{a^0}{\mu}\right) \tag{A.16}$$

has a unique finite solution (same argument as before). Thus, continuity implies that there exists exactly one p^* such that $p'(p^*) = p^*$. This concludes the proof.

A.2 Proof of uniqueness of symmetric Pareto equilibrium

We want to prove that for a symmetric environment with $a^1 = a^2 = ... = a^N := a$ and $c^1 = c^2 = ... = c^N =: c$ there exists exactly one symmetric Pareto equilibrium $\mathbf{p}^* = \{p^*, p^*, ..., p^*\} \in \mathbb{R}^N$. Specifically, we have to show that

$$r := \frac{(p - c)\exp\left(\frac{a-p}{\mu}\right)}{\sum_{j=1}^{N}\exp\left(\frac{a-p}{\mu}\right) + \exp\left(\frac{a^0}{\mu}\right)}, \tag{A.17}$$

has a unique global maximiser.

We again compute the derivative:

$$\frac{\partial r}{\partial p} = \frac{\left[-\frac{1}{\mu}(p-c)\exp\left(\frac{a-p}{\mu}\right) + \exp\left(\frac{a-p}{\mu}\right)\right] A + \frac{1}{\mu}(p-c)N\exp\left(\frac{a-p}{\mu}\right)\exp\left(\frac{a-p}{\mu}\right)}{A^2}, \tag{A.18}$$

where

$$A := N\exp\left(\frac{a-p}{\mu}\right) + \exp\left(\frac{a^0}{\mu}\right). \tag{A.19}$$

We equate the derivative to 0, divide by $\exp\left(\frac{a-p}{\mu}\right) > 0$ and multiply by $A^2 > 0$.

$$0 = -\frac{1}{\mu}(p-c)\exp\left(\frac{a^0}{\mu}\right) + A \tag{A.20}$$

Now, we note that $\exp\left(\frac{a^0}{\mu}\right)$ is positive and constant with respect to p. Furthermore,

$$A > 0 \tag{A.21}$$

and

$$\frac{\partial}{\partial p}\exp\left(\frac{a-p}{\mu}\right) < 0 \implies \frac{\partial A}{\partial p} < 0. \tag{A.22}$$

Therefore

$$-\frac{1}{\mu}(p-c)\exp\left(\frac{a^0}{\mu}\right) + A \tag{A.23}$$

is monotonically decreasing with respect to p. We note that

$$\lim_{p\to-\infty}\frac{\partial r}{\partial p} = \infty \quad \text{and} \lim_{p\to+\infty}\frac{\partial r}{\partial p} = -\infty. \tag{A.24}$$

Thus, $\frac{\partial r}{\partial p} = 0$ has a unique solution and so r has a unique maximiser with respect to p. This concludes the proof.