When Asymmetric Pricing Algorithms Collide[*]

William B. Brasic[†]

April 26, 2024

Abstract

Algorithms are increasingly superseding humans in the pricing of goods and services, enabling firms to swiftly adapt to shifting market dynamics with greater precision. Despite the widespread adoption of these algorithms, there remains a scarcity of knowledge regarding their specific configurations and their impact on competition. I investigate algorithmic heterogeneity to assess whether asymmetric reinforcement learning-based pricing algorithms can effectively learn to engage in tacit collusion within a repeated Bertrand-Markov pricing environment. My analysis reveals that diverse algorithms can indeed learn to tacitly collude, consistently setting prices above competitive levels and sustaining such supracometitive outcomes via the implementation of learned trigger strategies. This practice results in enhanced firm profitability, while concurrently diminishing consumer welfare.

Keywords: Reinforcement Learning, Pricing Algorithms, Collusion
JEL codes: C730, D430, L130, L410

I. Introduction

Artificial Intelligence (AI) is increasingly assuming responsibility for pricing goods and services, relegating humans from this crucial task. This shift is expected to persist as suppliers embrace algorithmic solutions that alleviate the traditional burden of manually determining prices. However, the evolving landscape of algorithmic pricing introduces a pressing concern—can these algorithms, inherently designed to optimize firm objectives, inadvertently learn to collude?

The apprehension lies in the potential collusion among algorithms without explicit programming for such behavior and being able to sustain collusive outcomes more effectively than human price setters (Calvano et al., 2019). Unlike traditional antitrust concerns centered around firms communicating with an intent to collude, algorithmic collusion may occur without direct instruction or communication (Harrington, 2018),

[*] I am especially grateful to Matthijs Wildenbeest for his feedback, support, and guidance. All errors are my own. Code for the main results of this paper can be found at `https://github.com/willbrasic/Asymmetric_Multi-Agent_Reinforcement_Learning_Pricing_Competition`.

[†] Department of Economics, The University of Arizona. Email: wbrasic@arizona.edu

although there is an active research domain devoted to algorithms learning to communicate to generate better policies than their non-communicative counterparts (Zhu et al. 2024). Legal scholars such as Ezrachi and Stucke (2017, 2020) have named such tacit collusion scenarios as *Artificial Intelligence and the Digital Eye*, expressing great concern regarding how current anti-trust law may be deemed antiquated to handle these problems. Mazumdar (2022) agrees with the general consensus that the Sherman Act is likely ill-equipped to handle algorithmic price fixing and argues that section 5 of the Federal Trade Comission (FTC) Act may be better suited to do so as it provides a broader means to handle anti-trust issues. United States government officials are already taking action to remedy the concerns of these and other legal scholars[1] and the FTC is making progress via discussions[2] and notes[3] while bringing charges to Amazon citing their alleged illegal use of pricing technology.[4] This paradigm shift poses a challenge to antitrust authorities, as their conventional means of prosecuting cartels through evidence of explicit communication may become obsolete as algorithms continue to supplant humans in the pricing of goods and services.

Experimental studies, such as those conducted by Calvano et al. (2020) and Klein (2021), have explored the collusion potential among identical AI agents. This research has revealed that through repeated interactions with an unknown economic environment, identical AI agents can achieve supracompetitive prices and profits. However, it remains unclear whether heterogeneous algorithms—distinct AI systems with varied architectures—can also learn to collude. Given the diverse nature of AI implementations in real-world scenarios due to, for instance, differing investments in pricing technologies, the assumption that firms utilize symmetric AI software for pricing is likely unrealistic.

To address this gap in understanding, my research delves into the experimental AI pricing literature, aiming to investigate the question of whether algorithms with asymmetric characteristics can learn to collude. By exploring the dynamics of diverse AI agents navigating the complexities of pricing within an economic environment, this study contributes to a comprehension of the potential collusion risks associated with the proliferation of algorithmic pricing in contemporary markets. The goal of this paper is to contribute to the current experimental pricing algorithm literature along with the existing legal discourse surrounding competition policy implications of algorithmic tacit collusion.

At the core of algorithmic pricing software lies the sophisticated realm of reinforcement learning, a branch of machine learning that thrives on reinforcing actions yielding high returns while discouraging otherwise less promising actions. Among the prominent algorithms investigated in the AI experimental pricing literature, Q-learning takes cen-

[1] https://www.klobuchar.senate.gov/public/index.cfm/2024/2/klobuchar-colleagues-introduce-antitrust-legisl

[2] https://www.ftc.gov/media/70163

[3] https://www.ftc.gov/system/files/attachments/us-submissions-oecd-2010-present-other-international-compe
algorithms.pdf

[4] https://www.ftc.gov/news-events/news/press-releases/2023/09/
ftc-sues-amazon-illegally-maintaining-monopoly-power

ter stage. This foundational reinforcement learning algorithm, rooted in the theory of dynamic optimization, leverages action-value function estimates to derive a policy that maximizes the expected discounted cumulative return.

While Q-learning has been a focal point in understanding the dynamics of algorithmic pricing, it is crucial to broaden our perspective to encompass a spectrum of reinforcement learning algorithms. One such noteworthy counterpart to Q-learning is SARSA, sharing its fundamental principles, but diverging as an on-policy algorithm compared to the off-policy nature of Q-learning (further elucidated in Section IV.). In the pursuit of a comprehensive understanding, my investigation extends beyond the confines of identical Q-learning agents to explore the collusion potential of SARSA interacting with Q-learning. The exploration aims not only to discern if such behavior is attainable, but also to shed light on the distinctive characteristics that define collusive actions within the framework of reinforcement learning.

To explore the potential collusive effects arising from the interaction of diverse algorithms, I adopt a methodological approach reminiscent of Calvano et al. (2020). In this framework, agents engage in an infinitely repeated Bertrand-Markov pricing game where they condition these actions on past history. This investigation incorporates a realistic differentiated demand framework to model the stage game of pricing competition, leveraging the logit model which is unknown to the AI agent. This modeling choice enhances the real-world applicability of the analysis, allowing for a more realistic understanding of the collusive dynamics at play.

The crux of my findings lies in the revelation that asymmetric algorithms do exhibit a capacity to learn and engage in collusion, echoing the collusive behavior established by their homogeneous counterparts. When SARSA and Q-learning agents engage in price competition, they converge towards a policy consisting of supracompetitive profits that are sustained through the employment of learned trigger strategies. These findings are robust to a variety of specifications, including extending the market to three agents. The results in this work largely corroborate the existing experimental and empirical literature regarding tacit algorithmic collusion as well as further justify the concerns of legal scholars and antitrust authorities.

This paper proceeds with a discussion of the prior literature, a primer on reinforcement learning followed by detailing the specific algorithms deployed, a description of the economic environment, the results of the research, and a section of concluding remarks with possible future extensions.

II. Literature Review

The literature on algorithmic pricing collusion unfolds across three distinct dimensions: firstly, an exploration of antitrust concerns; secondly, experimental studies that present compelling proofs-of-concept regarding the feasibility of tacit algorithmic collusion; and thirdly, empirical investigations delving into the tangible effects of pricing software on real-world market prices. Despite the valuable contributions from existing studies, the literature remains relatively sparse, leaving ample room for further research and insights. This section navigates through a comprehensive overview of prominent studies that collectively shape our understanding of AI-based collusion.

In the realm of antitrust regulation, Section I of the Sherman Act comes into play when firms engage in agreements to restrict competition. Traditionally, collusion among humans involves a stepwise process as outlined by Harrington (2018): (1) communication between competitors regarding collusive intent and conduct, (2) mutual adoption of collusive conduct, and (3) resultant higher prices. Enforcement of antitrust laws has historically relied on tangible evidence of explicit communication to prosecute firms for such violations. However, the landscape shifts significantly when it comes to algorithmic pricing. Unlike human collusion, the communication element is notably absent in the deployment of AI pricing software, as firms may be unaware of the collusive potential inherent in these algorithms once operational. Harrington (2018) argues that AI pricing software, autonomously learning collusive behavior without human intervention, may not be deemed in violation of Section I of the Sherman Act. This raises a pivotal question: if AI agents indeed learn to collude, how should current antitrust law be restructured to address this evolving challenge? While the second step of mutual adoption in human collusion is elusive due to the inaccessibility of firm managers' minds, the scenario is different in the case of algorithmic collusion. Here, the code of AI pricing software can be inspected and potentially tested for collusive conduct. This distinction provides antitrust authorities with an unique avenue to construct a framework targeting prohibited pricing algorithms. By establishing a set of algorithms deemed to have collusive potential, antitrust authorities could conduct tests to determine their compliance, ensuring a more adaptive and effective approach to regulating algorithmic collusion.

Embedded within the works of Mehra (2016), Harrington (2018), and Ezrachi and Stucke (2017, 2020), to name a few, is a crucial concern: the necessity to establish whether algorithms have the inherent capacity to learn collusion and, if so, what such collusive outcomes look like. This concern is pivotal, as addressing a problem that does not authentically exist would render these scholarly endeavors void. There is not a general consensus regarding this as works such as Miklós-Thal and Tucker (2019) form a theoretical model showing that algorithmic pricing can lead to lower prices and higher levels of consumer welfare. However, Calvano et al. (2020) conduct a groundbreaking experiment, marking the first systematic study to investigate the learnability of collusion by AI pricing software. In a stylized environment featuring two firms utilizing symmetric Q-learning agents in simultaneous price competition over a discrete pricing space, the authors unveil compelling findings. Their study demonstrates that Q-learning algorithms not only have the potential to achieve supracompetitive prices and profits, but can also effectively learn reward-punishment schemes—a prerequisite for collusion, as elucidated by Harrington (2018). When a specific algorithm deviates in one period by undercutting its competitor and is promptly punished in the next period, prices quickly return to the non-competitive level seen prior to this deviation. Although these algorithms fall short of attaining the joint-profit maximizing level, they consistently converge to prices well above the static Bertrand-Nash equilibrium, indicative of a departure from the purely competitive outcome. Notably, a few issues exist that question the applicability this study has on real-world AI pricing collusion: collusive tendencies decrease monotonically as the number of market participants rises, convergence rates of Q-learning agents are incredibly slow, the Q-learning algorithm can only handle a discrete action space, and the work represents only homogeneous algorithms interacting. However, the seminal Calvano et al. (2020) study still represents a compelling proof-of-concept for algorithmic collusion, shedding light on the dynamics at play when AI pricing software engages in strategic interactions. The findings contribute significantly to the ongoing discourse surrounding the implications and regulation of algorithmic behavior in competitive markets.

While employing a simultaneous move framework, as utilized in both my study and Calvano et al. (2020), provides valuable insights into the potential for algorithms to acquire collusive behavior, Klein (2021) elevates this exploration by extending it to a sequential move game. This sequential approach introduces a more dynamic setting, enhancing our understanding of AI collusive behavior. In a setup akin to Calvano et al., Klein considers two firms utilizing identical Q-learning agents. However, in Klein's sequential move game, firms now establish prices sequentially rather than simultaneously. Klein's work not only reaffirms the findings of Calvano et al., but also adds depth to our comprehension of algorithmic collusion in a dynamic context. Through repeated interactions with the environment, Q-learning algorithms in this sequential setup demonstrate an ability to achieve supracompetitive prices and learn reward-punishment schemes. Interestingly, Klein's study finds that as the pricing space increases, prices display asymmetric Edgeworth cycles similar to that discussed in Maskin and Tirole (1988). While these findings are robust to variations in the discount factor and learning rate, a notable insight from Klein's investigation is that, similar to the simultaneous move setting, the collusive capacity of these algorithms diminishes as the number of firms in the market increases and these Q-learning algorithms take remarkably long time to converge to supracompetitive levels. Overall, this compelling paper contributes to the evolving discourse on algorithmic collusion, providing valuable insights into how the strategic dynamics of AI agents evolve in response to varying market structures.

Mellgren's (2020) and Hettich's (2021) exploration into the collusive potential of deep reinforcement learning (DRL) methods, specifically the deep Q-network (DQN) formally introduced by Mnih et al. (2015) at Google DeepMind, and Frick's (2023) investigation of Soft Actor-Critic, a novel algorithm developed by Haarnoja et al. (2018) that learns both policies and value functions simultaneously while acting with a continuum of actions, has yielded compelling insights. Importantly, Hettich and Frick reveal that

DRL algorithms operating within a simultaneous Bertrand pricing game exhibit evidence of faster acquisition of supracompetitive profits and a reward-punishment scheme compared to Q-learning agents alleviating a potent concern of Calvano et al. (2020) and Klein (2021) to a certain degree. This suggests that more advanced algorithms possess a heightened capacity to learn collusive behavior, outpacing their less sophisticated counterparts. Notably, Mellgren (2020) investigates the collusive capacity of deep Q-learning within a sequential move pricing game finding evidence of algorithms achieving anti-competitive prices, but unable to learn reward-punishment schemes, a necessary condition for collusion. Furthermore, Hettich (2021) delves into the interplay between Q-learning and DQN agents. The results indicate a strategic exploitation by the DQN, effectively driving the Q-learning agent out of the market. Importantly, the study discerns that the higher profits of the DQN agent do not necessarily signal collusive behavior, but stem from an enhanced market power. While this finding provides valuable insights, it is acknowledged that DQNs are inherently more powerful than conventional tabular-based reinforcement learning algorithms such as Q-learning, particularly when the state space is large. Consequently, the test for collusive behavior among diverse algorithms, limited to Q-learning and DQN, may not fully capture the nuances of collusive behaviors. I extend and refine this investigation by broadening the spectrum of algorithms considered to SARSA and Q-learning, two algorithms that are quite similar, but with innate differences.

While experimental studies provide valuable insights into the potential for AI collusion and elucidate the nature of collusive behavior, the current literature faces a notable gap in empirical evidence substantiating the conclusions drawn from these experiments. Intending to help address this void, Assad et al. (2023) contribute a pivotal empirical study examining the capacity of algorithms to collude, utilizing high frequency retail German gasoline pricing data. With the widespread availability of AI pricing software in Germany from 2017 onwards, the authors confront the challenge of unobserved gasoline station adoption of such software. To overcome this, they employ a novel test for adoption, focusing on structural breaks in three key variables: the number of price changes in a day, average size of price changes, and rival response time. To mitigate concerns about endogeneity in station adoption, the authors utilize an instrumental variable and two-way fixed effects approach. They employ the share of a given station's brand that adopted algorithmic pricing software as an instrument, addressing potential endogeneity arising from unobserved station and time-specific factors, such as managerial skill and changing local market conditions. The results of their empirical analysis reveal compelling evidence of collusive behavior. Specifically, adopting stations with competitors in their ZIP code experience a statistically significant mean margin increase, while adopting stations without competitors in their ZIP code show no statistically significant change in mean margins. This outcome suggests that algorithmic collusion, facilitated by the adoption of AI pricing software, influences margins through its impact on competition among non-monopolist firms. The authors' findings not only contribute crucial empirical evidence to the discourse on algorithmic collusion, but also lay a robust foundation for future empirical research examining the real-world effects of algorithmic pricing on competition and market dynamics.

III. Reinforcement Learning

A. *Markov Decision Processes*

As discussed in Hettich (2021), repeated Bertrand competition can be modeled via a Markov decision process (MDP) giving rise to a Bertrand-Markov pricing game. In a MDP tailored to reinforcement learning, the learner and decision maker, called the agent, continually interacts with everything outside of the agent, called the environment, to learn optimal behavior in order to maximize the expected cumulative discounted return over time through its choice of actions. Note, the agent is unaware of the underlying dynamics of the MDP and must learn about it; this is where reinforcement learning comes into play. Reinforcement learning agents want to *learn* about the underlying MDP and *reinforce* good actions. MDPs lay the foundation for reinforcement learning so I give an overview from this perspective following the setup of Agarwal et al. (2022).

In reinforcement learning, the interactions between the agent and the environment can be described by an infinite-horizon MDP $M = (\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \delta, \mu)$. Here, $\mathcal{S}$ is the state space, $\mathcal{A}$ is the actions space, and $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function. In general, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$ is a stochastic state-transition function mapping the current state and action at time step t, s_t and a_t, respectively, into probabilities of observing all possible next states $s_{t+1} \in \mathcal{S}$ where $\mathcal{P}(\mathcal{S}) \subseteq [0,1]^{|\mathcal{S}|}$ is the probability simplex over $\mathcal{S}$. Lastly, $\delta \in [0,1)$ is the discount factor which is bounded away from one to ensure infinite summations converge, and $\mu \in \mathcal{P}(\mathcal{S})$ is the initial state distribution specifying how the initial state s_0 is drawn.

For a given MDP M, the agent starts at an initial state $s_0 \sim \mu$. At each time step $t \in \{0, 1, 2, \ldots\}$, the agent takes action $a_t \in \mathcal{A}$ upon observing state s_t, consequently receives an immediate reward $\mathcal{R}(s_t, a_t) \triangleq r_t$, and then observes the next state $s_{t+1} \sim \mathcal{T}(\cdot|s_t, a_t)$.[5] The stochastic state-transition function $\mathcal{T}$ as well as the reward function $\mathcal{R}$ together comprise a model of the environment. The agent repeats this process indefinitely starting from time step $t = 0$ forming a trajectory of experiences: $\tau = (s_0, a_0, r_0, s_1, \ldots)$ with the objective of finding a policy $\pi : \mathcal{S} \to \mathcal{P}(\mathcal{A})$ [6] representing a decision-making strategy in which the agents' action selections are only based on the current state, i.e., $a_t \sim \pi(\cdot|s_t)$. Notably, reinforcement learning algorithms make use of the Markov property that the state-transition function only depends on the state and action from the current time

[5] In the deterministic transition case, $s_{t+1} = \mathcal{T}(s_t, a_t)$

[6] This policy is deemed *stationary* if the randomness in action selections stays constant over time, e.g., ϵ-greedy exploration. However, if randomness varies over each time step, e.g., time decaying ϵ-greedy, the policy is no longer stationary.

step: $\mathcal{T}(\cdot|s_t, a_t, s_{t-1}, a_{t-1}, \ldots) = \mathcal{T}(\cdot|s_t, a_t)$. The agent's goal is to maximize, at each time step t, the expected discounted cumulative return

$$R_t \triangleq \mathbb{E}_\pi \left[\sum_{h=0}^{\infty} \delta^h r_{t+h+1} \right]$$

and this can be accomplished by finding the optimal policy π^*, which achieves the maximum expected discounted cumulative return from all states:

$$\pi^* = \arg\max_\pi \mathbb{E}_\pi \left[\sum_{h=0}^{\infty} \delta^h r_{t+h+1} \right].$$

Therefore, the task of the agent is to maximize its long-run performance while only receiving feedback from the environment about its one-step reward, namely r_t.[7] The agent can achieve this goal via the use of value functions and their natural extension to reinforcement learning, action-value or Q-value functions.

B. *Value and Action-Value Functions*

For a fixed policy π starting from state s_t, the value function $V_\pi : \mathcal{S} \to \mathbb{R}$ is defined as

$$V_\pi(s_t) \triangleq \mathbb{E}_\pi \left[\sum_{h=0}^{\infty} \delta^h r_{t+h+1} \middle| s = s_t \right].$$

The above value function can be written in the recursive manner giving rise to the Bellman equation for V_π:

$$V_\pi(s_t) \triangleq \mathbb{E}_\pi \left[r_{t+1} + \delta V_\pi(s_{t+1}) \middle| s = s_t \right].$$

This is a measure of the quality of the policy π representing the expected cumulative discounted return following policy π from state s_t forever onward. The optimal policy, π^*, has the corresponding value function V_{π^*}:

$$V_{\pi^*}(s_t) \triangleq \max_\pi V_\pi(s_t).$$

[7] It is important to note the difference between the immediate reward at time t, r_t, and what is commonly to referred to as a return. An agent's goal is not to maximize their immediate reward, but their expected discounted cumulative return over the long-run. This expected return over the long-run is the value that is attributed to following a given policy. Thus, the agent must balance the benefit of receiving a large immediate reward signal versus the overall goal of maximizing their return. For an in depth treatment of this difference, see Morales (2020)

Thus, π^* is defined to be the policy such that $V_{\pi^*}(s_t) \geq V_\pi(s_t)$ for all policies π. As noted in Sutton and Barto (2018), this equation can be written recursively as

$$V_{\pi^*}(s_t) = \mathbb{E}_{\pi^*}[r_{t+1} + \delta V_{\pi^*}(s_{t+1})|s = s_t].$$

which is commonly referred to as the Bellman equation for V_{π^*}. This Bellman equation can be solved for by finding its unique solution V_{π^*}. Existence and uniqueness results for V_{π^*} for finite Markov decision processes can be found in Szepesvári (2010) and for continuous space/action and discrete-time Markov decision processes in Bertsekas (2019).

When the model of the environment is unknown, reinforcement learning can be used to try to find the optimal policy π^*. For a fixed policy π starting from taking an arbitrary action a_t in state s_t, the action-value function $Q_\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is defined as

$$Q_\pi(s_t, a_t) \triangleq \mathbb{E}_\pi \left[\sum_{h=0}^{\infty} \delta^h r_{t+h+1} \middle| s = s_t, a = a_t \right]$$

The above action-value function can be written in the recursive manner[8] giving rise to the Bellman equation for Q_π:

$$Q_\pi(s_t, a_t) \triangleq \mathbb{E}_\pi \left[r_{t+1} + \delta Q_\pi(s_{t+1}, a_{t+1}) \middle| s = s_t, a = a_t \right].$$

This is the expected cumulative discounted return from taking any action a_t in state s_t and following policy π forever thereafter.

The reinforcement learning agent wishes to find the optimal policy, π^*, has the corresponding action-value function Q_{π^*}:

$$Q_{\pi^*}(s_t, a_t) \triangleq \max_\pi Q_\pi(s_t, a_t).$$

Therefore, π^* is defined to be the policy such that $Q_{\pi^*}(s_t, a_t) \geq Q_\pi(s_t, a_t)$ for all policies π. Q_{π^*} can be written in the recursive manner

$$Q_{\pi^*}(s_t, a_t) = \mathbb{E}_{\pi^*} \left[r_{t+1} + \delta \max_{a' \in \mathcal{A}} Q_{\pi^*}(s_{t+1}, a') \middle| s = s_t, a = a_t \right]$$

which is commonly referred to as the Bellman equation for Q_{π^*}. Notably, it may seem redundant to include the maximum over actions when we are already considering the optimal policy π^*. This is done to ensure robustness to stochasticity and approximation errors as well as to force the agent to consider all possible outcomes when estimating the expected return.

[8] For the derivation of the Bellman equation for Q_π, see the Appendix. The derivation of the Bellman equation for V_π follows similarly.

A convenient result[9] exists demonstrating the similarity between the value and action-value functions:

$$V_\pi(s_t) = \mathbb{E}\left[Q_\pi(s_t, a_t) | s = s_t\right].$$

Dynamic programming methods can be used to solve for Q_{π^*} when the underlying MDP is known. However, this is rarely the case in practical settings. Thus, given the state-transition probabilities are unknown, the agent must try to learn Q_{π^*} via repeated interactions with the environment. Reinforcement learning algorithms aim to achieve this goal via the use of model-free based algorithms, ones that require no such knowledge of the underlying environment.

C. *Multi-Agent Reinforcement Learning*

While the setup described here is for a single agent, the framework illustrated can easily be extended to numerous players thereby leading to multi-agent reinforcement learning[10] which is the methodology used in this article and commonly seen as an intersection of reinforcement learning and game theory (Yang and Wang, 2020). Expanding the Markov decision process to the multi-agent case gives rise to what is commonly referred to as a Markov or stochastic game $G = (P, \mathcal{S}, \boldsymbol{\mathcal{A}}, \mathcal{R}, \mathcal{T}, \boldsymbol{\delta}, \mu)$ in game theory.[11] Here, P is the set of n players, $\mathcal{S}$ is the set of states, $\boldsymbol{\mathcal{A}} \triangleq \times_{i=1}^{n} \mathcal{A}_i$ is the joint action set and $\mathcal{R} : \mathcal{S} \times \boldsymbol{\mathcal{A}} \to \mathbb{R}$ is the reward function. The stochastic state-transition function is defined as $\mathcal{T} : \mathcal{S} \times \boldsymbol{\mathcal{A}} \to \mathcal{P}(\mathcal{S})$. Lastly, $\boldsymbol{\delta} \in [0, 1)^n$ holds each player's discount factor, and $\mu \in \mathcal{P}(\mathcal{S})$ is the initial state distribution. Now, the state-transitions are a mapping from the state space $\mathcal{S}$ along with the joint action set $\boldsymbol{\mathcal{A}}$, i.e., the transition probability of observing $s_{t+1} \sim \mathcal{T}(\cdot | s_t, a_{1t}, \ldots, a_{nt})$. Consequently, the reward at each time step t for each agent is a function of the joint action of all players. The policies for each agent now form a joint stationary policy $\boldsymbol{\pi} : \mathcal{S} \to \mathcal{P}(\boldsymbol{\mathcal{A}})$ and the Q-function for each agent now depends on this joint policy $\boldsymbol{\pi}$, i.e., $Q_{i,\boldsymbol{\pi}} : \mathcal{S} \times \boldsymbol{A} \to \mathbb{R}$. When the agents rewards are perfectly positively correlated this is a fully cooperative game while when the agents rewards are perfectly inversely correlated this results in a zero-sum game.

Notably, the transition function can be a deterministic function of the current state and each agent's action. Moreover, when the current state is completely characterized by the current actions, the deterministic state-transition function can be defined as $\mathcal{T} : \boldsymbol{\mathcal{A}} \to \mathcal{S}$. This is the transition mapping used in my experiments.

[9] For the proof, see the Appendix.

[10] For an overview of this domain, see Busoniu et al. (2008) and Zhang et al. (2021)

[11] These games are thoroughly described in Maschler et al. (2020) and, from the reinforcement learning perspective, in Tuyls and Weiss (2012)

IV. Algorithms

A. *SARSA*

SARSA $(s_t, a_t, r_t, s_{t+1}, a_{t+1})$, first formally proposed by Rummery and Niranjan (1994), is an example of an on-policy temporal difference (TD) learning algorithm. TD algorithms use a combination of Monte Carlo (MC) methods and dynamic programming to achieve a middle ground between the two. More specifically, SARSA uses bootstrapped estimates of the target value similar to MC methods and unlike Dynamic programming methods, but does not wait until an episode is finished to calculate a return like dynamic programming and unlike the MC approach. For an on-policy method, we must estimate $Q_\pi(s_t, a_t)$ for the *current* behavior policy π for all $(s_t, a_t) \in \mathcal{S} \times \mathcal{A}$. That is, the current policy π that is actively generating experiences. The update rule[12] for the SARSA algorithm is

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \delta Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right]. \tag{1}$$

where $\alpha \in [0, 1]$ is the learning rate and $\delta \in [0, 1)$ is the discount factor. SARSA is defined as an on-policy algorithm because the action a_{t+1} chosen in s_{t+1} per policy π to update $Q(s_t, a_t)$ is the action actually taken in the next period $t + 1$ per policy π. Furthermore, SARSA is a model-free algorithm as (1) does not require knowledge of the underlying environment such as the state-transition function. The SARSA pseudocode using ϵ-greedy (discussed further below) as the exploration strategy with infinite time horizon is given in Algorithm 1.

Algorithm 1 SARSA w/ ϵ-greedy

```
Require: $\alpha \in [0,1]$, $\delta \in [0,1)$, and $\epsilon > 0$ small
Require: $Q(s_t, a_t)$ initialized arbitrarily for all $(s_t, a_t) \in \mathcal{S} \times \mathcal{A}$
for all episodes $e = 1, \ldots, E$ do
    Initialize $t = 0$ and maximum time steps allowed $max_t$
    Initialize $s_0$
    Choose action $a_0$ in state $s_0$ according to $\epsilon$-greedy
    while $t < max_t$ and not converged do
        Take action $a_t$ according to $\epsilon$-greedy
        Observe $r_t$ and $s_{t+1}$
        Choose action $a_{t+1}$ in $s_{t+1}$ according to $\epsilon$-greedy
        $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha [r_{t+1} + \delta Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$
    end while
end for
```

B. *Q-learning*

[12] For the intuition of the update rule which is common to all TD algorithms, see the Appendix.

Q-learning is a fundamental reinforcement learning algorithm developed by Watkins and Dayan (1992). Studies using Q-learning to investigate collusion go as far back as Waltman and Kaymack (2008) who looked at the potential for Q-learning agents to collude in a Cournot oligopoly game setting and this algorithm is still being investigated in works such as Calvano et al. (2020) and Klein (2021) who looked at simultaneous and sequential oligopoly environments, respectively. Like SARSA, Q-learning is a TD algorithm. However, it is off-policy meaning we estimate $Q_\pi(s_t, a_t)$ using a *different* policy π' for all $(s_t, a_t) \in \mathcal{S} \times \mathcal{A}$. The update rule for Q-learning is

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \delta \max_{a \in \mathcal{A}} Q(s_{t+1}, a) - Q(s_t, a_t) \right]. \tag{2}$$

where $\alpha \in [0, 1]$ is the learning rate and $\delta \in [0, 1)$ is the discount factor. Q-learning is defined as an off-policy algorithm because the action $a \in \arg\max_{a' \in \mathcal{A}} Q(s_{t+1}, a')$ is *not* necessarily the action actually taken in the next period $t + 1$ per policy π. Furthermore, Q-learning is a model-free algorithm as (2) does not require knowledge of the underlying environment such as the state-transition function. The Q-learning pseudocode using ϵ-greedy as the exploration strategy with infinite time horizon is given in Algorithm 2.

```
Algorithm 2 Q-learning w/ $\epsilon$-greedy
Require: $\alpha \in [0, 1]$, $\delta \in [0, 1)$, and $\epsilon > 0$ small
Require: $Q(s_t, a_t)$ initialized arbitrarily for all $(s_t, a_t) \in \mathcal{S} \times \mathcal{A}$
for all episodes $e = 1, \dots, E$ do
    Initialize $t = 0$ and maximum time steps allowed $max_t$
    Initialize $s_0$
    Choose action $a_0$ in state $s_0$ according to $\epsilon$-greedy
    while $t < max_t$ and algorithm not converged do
        Take action $a_t$ according to $\epsilon$-greedy
        Observe $r_t$ and $s_{t+1}$
        $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[ r_{t+1} + \delta \max_{a' \in \mathcal{A}} Q(s_{t+1}, a') - Q(s_t, a_t) \right]$
    end while
end for
```

C. *Exploration Versus Exploitation*

Within the domain of reinforcement learning lies the pivotal challenge of balancing the pursuit of actions that promise immediate rewards with the imperative to explore alternatives that might yield even greater long-run returns. This equilibrium is encapsulated in the exploitation-exploration trade-off, a cornerstone concept in the literature of reinforcement learning. The overarching objective of reinforcement learning algorithms is to strategically select actions that maximize the expected discounted cumulative reward. However, the challenge lies in avoiding excessive exploitation, where actions yielding current benefits are favored, at the risk of neglecting unexplored actions that could potentially offer superior returns.

Various strategies have been devised to address this fundamental problem, ranging from Boltzmann experimentation to Upper Confidence Bounds, as outlined by Auer (2002). One prevalent method for navigating the exploitation-exploration trade-off is the ϵ-greedy algorithm. This approach introduces an element of randomness by selecting a random action $a_t \in \mathcal{A}$ with small probability ϵ. In contrast, it leans towards exploitation by choosing the currently perceived best action in a given state s_t with probability $1 - \epsilon$. An important modification used in this paper is ϵ-greedy with time decay where exploration is encouraged more for early time steps in an episode and diminishes as the algorithm gains experience operating in the environment. Specifically, this study uses $e^{-\beta t}$, where t is time step and β is an exploration parameter, as this approaches 0 as t tends to infinity at an exponential rate so exploration occurs earlier in any given episode.

The ϵ-greedy algorithm encapsulates the essence of the exploration-exploitation trade-off, offering a pragmatic and widely employed solution to strike a balance between immediate rewards and the potential for discovering more rewarding actions. For a comprehensive understanding of the exploration-exploitation trade-off, see Morales (2020).

D. *Convergence*

Notably, while convergence results for SARSA and Q-learning agents individually exist in single player systems, similar proofs do not when extending to multi-agent reinforcement learning environments. To that end, I allow the agents to engage in price competition each episode for a maximum of ten million time steps. Convergence is checked for every 100 time steps and deemed to be achieved when both algorithms' optimal actions in any state have not changed for 1,000 consecutive convergence checks. That is, if for each player i and each state s, the set $\arg\max_{a \in \mathcal{A}_i} Q_i(a, s)$ has not changed for 1,000 consecutive time steps that convergence is checked for. This effectively suggests that each agent's action has not changed for 100,000 consecutive time steps. In my baseline analysis outlined in section VI.A., each of the $E = 100$ episodes converged well before the maximum ten million time step mark, but many repetitions were still needed. This number depends on factors such as the discount factor of each agent and the total number of players in the market.

V. Economic Environment

I follow a similar economic environment to that of Calvano et al. (2020). Firms engage in an infinitely repeated Bertrand-Markov pricing game where they set prices simultaneously and condition these actions on past history. The stage game models price competition via a logit demand model, as used in a wide variety of empirical applications across many different industries, with constant marginal costs.

To apply the MDP setup described in the prior section to an infinitely repeated simultaneous move Bertrand environment, an agent represents each firm producing a differentiated good $i \in \{0, 1, \ldots, n\}$ where $i = 0$ represents the outside option whose price is normalized to zero and n is both the number of products and the number of firms in the market. At each time step t, firm j has demand

$$d_{jt} = \frac{\exp\left(\frac{a_j - p_{jt}}{\mu}\right)}{\exp\left(\frac{a_0}{\mu}\right) + \sum_{i=1}^{n} \exp\left(\frac{a_i - p_{it}}{\mu}\right)}.$$

where parameters μ and a_i are product quality indices capturing horizontal and vertical differentiation, respectively. Since product 0 is the outside good, a_0 is an inverse index of aggregate demand. c_i is marginal cost, and I assume no fixed costs. Upon each agent i selecting an action p_{it} in their action space $\mathcal{A}_i$ at time step t, they receive reward $r_{it} = (p_{it} - c_i)d_{it}$ and transition to the next state $s_{t+1} = \{p_{1,t-1}, \ldots, p_{n,t-1}\}$.

To ensure the state space is finite, I use a bounded memory of length k_i for each player i so that a given state can be represented as $s_t = \{\boldsymbol{p}_{t-1}, \ldots, \boldsymbol{p}_{t-k}\}$ where each $\boldsymbol{p}_{t-h} \in \bigtimes_{i=1}^{n} \mathcal{A}_i$ for $1 \leq h \leq k$ is the vector of all firm prices set in period $t - h$. Unless noted otherwise, I assume $k_i = 1 \; \forall i \in \{1, \ldots, n\}$ so that $s_t = \boldsymbol{p}_{t-1}$. Notably, the state space $\mathcal{S}_i = \bigtimes_{i=1}^{n} \mathcal{A}_i$, with cardinality $|\mathcal{S}_i| = m^{n*k_i}$, is completely characterized by all possible price combinations each firm can set. When $k_i = 1 \; \forall i \in \{1, \ldots, n\}$, each firm bases its choice of actions at time step t on the history of each firms' actions at time step $t - 1$ meaning they have a one period recall.

For each of the parameter values, I compute the static Bertrand-Nash equilibrium prices $\boldsymbol{p}_N$ and the collusive prices $\boldsymbol{p}_C$ via fixed point iteration until convergence is achieved. When considering collusive behavior of SARSA and Q-learning as both require a discrete number of possible actions, I discretize the action space $\mathcal{A}_i$ to contain fifteen equally spaced price points from the minimum to the maximum price firm i can set where $\xi \in \mathbb{R}_{++}$ is the step size between each price. These minimum and maximum prices are 1.0 and 2.1, respectively.

To determine the level of collusion and stay consistent with the existing literature, I use the following measure profit measure:

$$\Delta_{ite} \triangleq \frac{r_{ite} - r_{iN}}{r_{iC} - r_{iN}}.$$

r_{ite} is firm i's profit at time step t in episode e, r_{iN} is firm i's competitive profit from the one-shot Bertrand-Nash equilibrium when firms set prices are $\mathbf{p}_N$, and r_{iC} is firm i's collusive profit when firms use prices $\boldsymbol{p}_C$. Notably, $\Delta_{ite} = 0$ and $\Delta_{ite} = 1 \; \forall i \in \{1, \ldots, n\}$ imply perfectly competitive and perfectly collusive markets, respectively. Throughout this article I define

$$\Delta = \frac{1}{100000(nE)} \sum_{i=1}^{n} \sum_{e=1}^{E} \sum_{t=T-100000}^{T} \Delta_{ite}$$

as the profit measure Δ_{ite} averaged across the last 100,000 time steps prior to convergence and then subsequently averaged across all firms and episodes.

Unless otherwise noted, the economic environment consists of a symmetric duopoly ($n = 2$) with $c_i = 1$, $a_i = 2$, $\delta_i = 0.95$, and $k_i = 1 \; \forall i \in \{1, \ldots, n\}$, while $a_0 = 0$, $\mu = 1/4$, $m = 15$, and $\xi = (2.1 - 1.0)/(m - 1)$. ϵ-greedy time decay parameter and learning rate is fixed for each i at $\beta_i = 0.00001$ and $\alpha_i = 0.15$. Moreover, each firms pricing space is identical so that $\mathcal{A}_i = \mathcal{A}_{-i} \; \forall i \in \{1, \ldots, n\}$ and, consequently, $\mathcal{S}_i = \mathcal{S}_{-i} \; \forall i \in \{1, \ldots, n\}$. The initial Q-matrix at $t = 0$ for firm i is the average profit given when setting each of the possible m prices. Notably, each agent i's Q-matrix is an element in $\mathbb{R}^{|\mathcal{S}_i|} \times \mathbb{R}^{|\mathcal{A}_i|}$ which corresponds to a 225 x 15 matrix. I test for robustness under differing initialized Q-matrices across the two agents and obtain similar results to that discussed below. Furthermore, I allow the agents to interact for $E = 100$ episodes.

It is crucial to underscore that the algorithms under examination operate in a knowledge vacuum concerning the economic environment, possessing only the capacity to compute profits.

VI. Results

A. *SARSA and Q-learning*

Table 1 and the learning curves depicted in Figure 1 reveal compelling evidence that SARSA and Q-learning agents, engaged in price competition, demonstrate the capability to achieve outcomes that surpass traditional competitive benchmarks. The data in Table 1 specifically emphasizes the collusion measure Δ and the percentage change observed in transitioning from the Bertrand-Nash equilibrium to the levels achieved over $E = 100$ episodes average over the last 100,000 time steps prior to algorithmic convergence.

Table 1

	Total/Average	**SARSA**	**Q-learning**
Δ	0.4549	0.4866	0.4233
Percentage Change from Bertrand-Nash Outcome			
Profits	23.38%	25.01%	21.75%
Demand	-4.51%	-2.81%	-6.20%
Prices	9.82%	9.49%	10.15%
Revenue	4.45%	6.12%	2.77%
CS	-18.27%		

Results averaged across $E = 100$ episodes.
Average number of time steps until convergence is 723,230.

Recalling that $\Delta = 0$ implies a perfectly competitive market, while $\Delta = 1$ implies a perfectly collusive outcome, the agents collectively achieve $\Delta = 0.4549$ at convergence, averaged across the $E = 100$ episodes. This indicates that while perfectly collusive levels were not attained, the algorithms' profit gains converge to values well above the competitive outcome. Additionally, consumer welfare decreases by 18.27% relative to the Bertrand-Nash outcome. This result underscores the capacity of heterogeneous algorithms, namely SARSA and Q-learning, to yield supracompetitive outcomes in the realm of price competition.

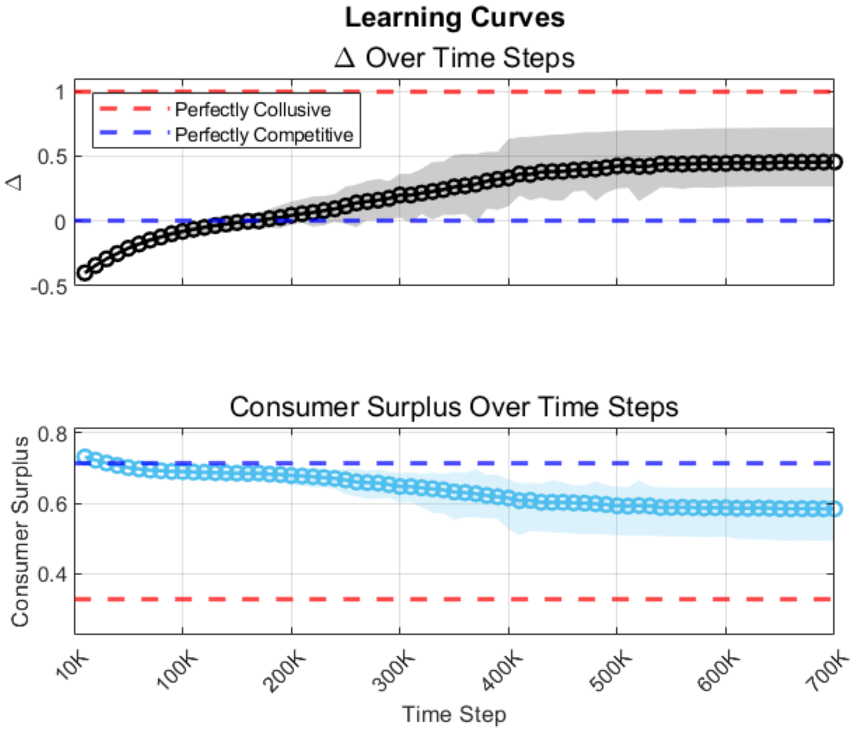
These algorithms achieve convergence at time step 723,230 on average across the E episodes. This convergence point is over 100,000 time steps less to that observed when two SARSA agents compete against each other and roughly 35,000 time steps more than when two Q-learning agents compete against each other. This suggests that algorithmic heterogeneity may not be slowing down learning times but rather that the SARSA algorithm itself contributes to the observed convergence delay. This delay could be attributed to SARSA's on-policy nature, requiring learning to be done exclusively using experiences generated by the policy the algorithm is currently employing.

Moreover, in the scenario where two SARSA agents compete, convergence leads to $\Delta = 0.2706$, while in the case of two Q-learning agents, it converges to $\Delta = 0.6301$, and the Q-learning agents converge much faster than the two SARSA agents. Due to the differing updating procedures of the two algorithms, Q-learning learns an optimal policy directly via off-policy learning allowing Q-matrix updates to applied by using a different policy than the one currently being optimized. This leads to Q-learning gaining higher profit levels quicker while trained offline while SARSA likely performs better in an online setting due to its on-policy nature. To that end, these Δ differentials align with the findings of Brown and MacKay (2023) in that if both firms initially employed SARSA and one switched to the superior pricing technology Q-learning (as these algorithms are trained offline in this setting), all firms obtain higher prices and profits. Interestingly, this effect seems so intense to where SARSA converges to higher profits, on average, relative to Q-learning. A deeper exploration into pricing games between on-policy and off-policy algorithms may provide further insights into this intriguing finding.

A common concern when running reinforcement learning experiments is that results can differ widely across varying random seeds. To that end, I run an experiment analogous to that above averaging results over ten differing random seeds. These averages correspond very well to that observed in Table 1. For instance, the convergence rate averaged across the random seeds is 729,557, only 6,000 time steps more than that noted in Table 1. Moreover, consumer surplus experiences a 19.06% decline relative to the Bertrand-Nash outcome which is less than a single percentage point away from that seen above. This establishes further credibility of the results seen in Table 1 in that these outcomes are not solely a function of the stochastic process that a certain seed may generate.

Figure 1 vividly presents the learning trajectories of the collusion measure Δ and the evolution of consumer welfare, each accompanied by their respective 95% confidence bands. The algorithms showcase a learning trajectory where Δ progressively surpasses the competitive baseline while concurrently witnessing a decline in consumer welfare. Remarkably, both variables exceed competitive levels at approximately time step 150,000, stabilizing around time step 500,000. By this point, the lower bound of the 95% confidence band is significantly above the competitive benchmark, suggesting a robust rejection of the hypothesis of a competitive market. The widening of this confidence band as convergence approaches can be partially attributed to the injected heterogeneity in the experiment, where SARSA and Q-learning, though initialized with identical Q-matrices, gradually develop distinct Q-values for each state-action pair over time. This divergence contributes to the increasing uncertainty reflected in the widening confidence band as the algorithms evolve.

Figure 1



Results averaged across $E = 100$ episodes.
Shaded area represents 95% confidence bands.

Examining Figure 2, which portrays the distributions of actions taken for each agent, I average these results across all episodes for each firm. This distribution is fairly normally distributed with firms demonstrating a proclivity to take the profit maximizing action. Importantly, the algorithms exhibit a level of flexibility by choosing alternative prices, a phenomenon attributed to the incorporation of the ϵ-greedy procedure. This mechanism guarantees ample exploration, enabling the agents to delve into a diverse range of actions beyond the seemingly optimal choice. As a result, agents delve into a diverse range of actions beyond the apparent optimal choice, thereby enriching the adaptive capabilities of the learning process.

Figure 2

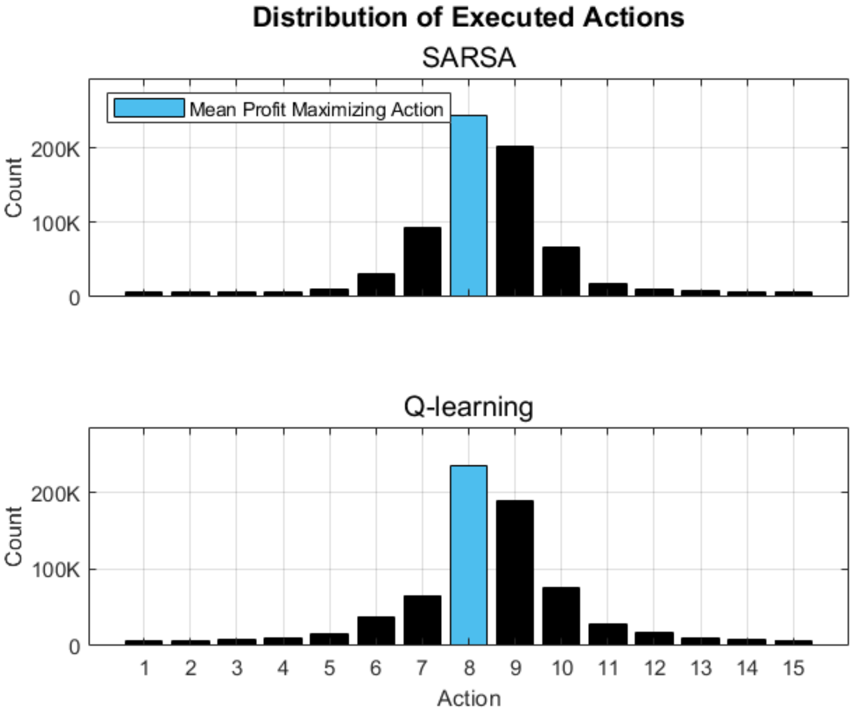
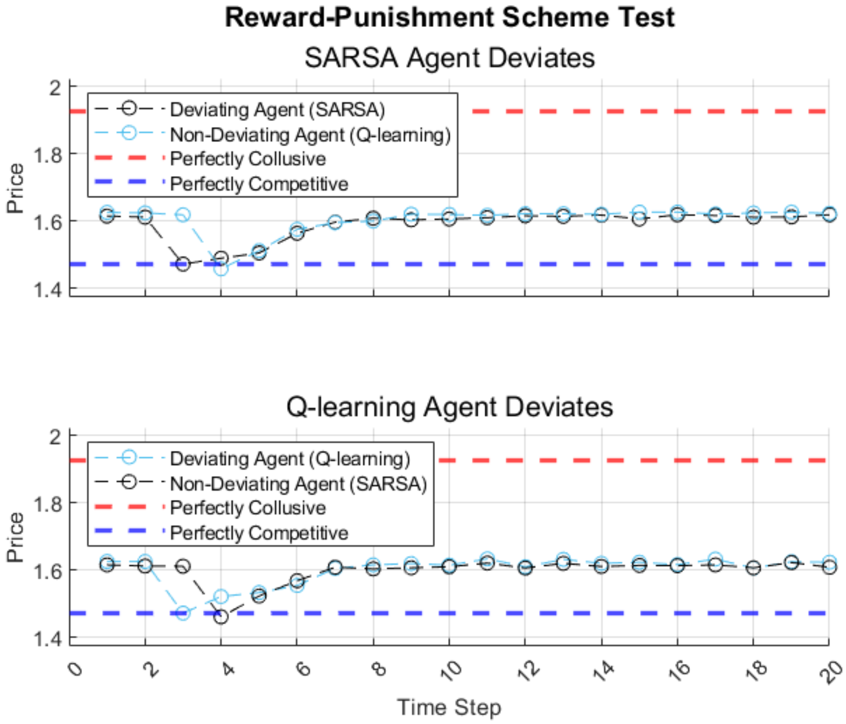


Figure 3 depicts the dynamic interplay between the two distinct algorithms, SARSA and Q-learning, engaged in price competition. Notably, the graphical representation reveals a strategic implementation of reward-punishment schemes. In a noteworthy instance, when the SARSA (Q-learning) agent deviates by undercutting to the one-shot Bertrand-Nash price level at $t = 3$, the Q-learning (SARSA) agent responds promptly in $t = 4$ with a retaliatory undercutting punishment. Over subsequent time intervals, both agents gradually revert to the initial prices upon algorithmic convergence, echoing patterns observed in the current experimental algorithmic pricing literature that algorithms do implement trigger strategies, but they are not grim as first described in Friedman (1971). This visualization, coupled with the depictions of supracompetitive profits and prices in Figures 1 and 2, serves as a compelling proof-of-concept. It signifies that asymmetric AI pricing software has the capacity to learn and sustain collusive outcomes through the strategic deployment of reward-punishment mechanisms. These findings not only align with established experimental literature regarding homogeneous AI pricing systems, but also underscore the nuanced dynamics and potential implications of algorithmic collusion in the domain of pricing strategies.

Figure 3



A..1 *More Than Two Players*

In each of the above settings I have considered a duopoly market. As noted in Ezrachi and Stucke (2020), algorithmic tacit collusion is likely to appear in concentrated markets so the algorithms can more easily monitor competitor prices and adjust their own. While the duopoly case provides a proof-of-concept that asymmetric reinforcement learning algorithms in such concentrated markets can indeed learn collusive strategies, it is imperative to extend this framework to consider the implications of incorporating a market with more than two firms. In this section I extend the baseline setup involving a SARSA and Q-learning agent engaging in price competition to a market with three players: two SARSA agents and one Q-learning agent.

Table 2

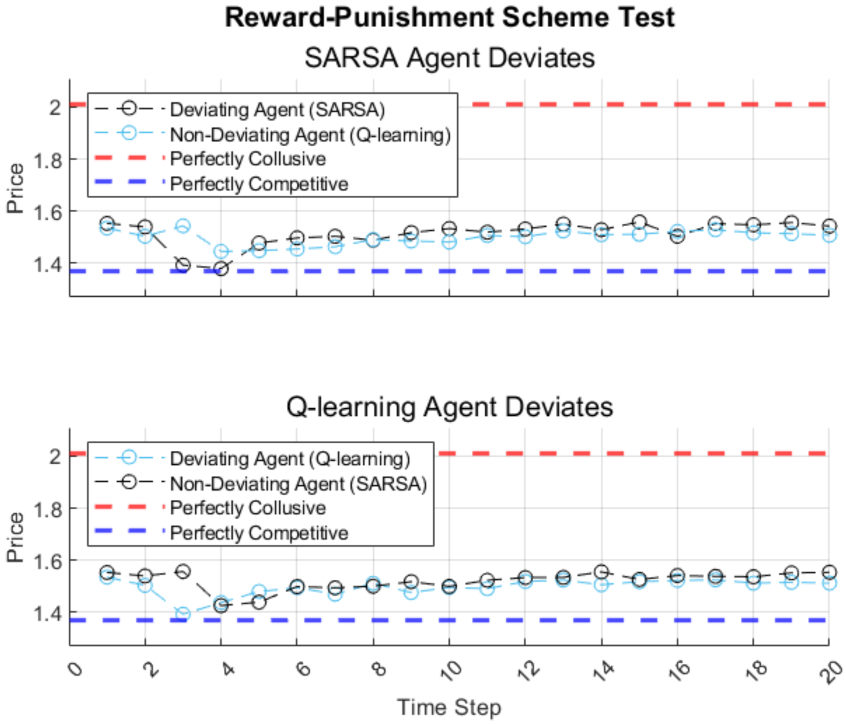
	Total/Average	**SARSA**	**SARSA**	**Q-learning**
Δ	0.3197	0.3023	0.2935	0.3634
Percentage Change from Bertrand-Nash Outcome				
Profits	34.54%	32.65%	31.71%	39.25%
Demand	-2.71%	-5.09%	-6.51%	3.46%
Prices	12.39%	12.85%	13.22%	11.11%
Revenue	7.35%	5.10%	3.82%	13.13%
CS	-16.44%			

Results averaged across $E = 100$ episodes.
Average number of time steps until convergence is 1,089,440.

Table 2 gives results from this experiment. Notably, Δ still lies above the competitive outcome, but well-below the joint collusive value of one, aligning well with the results of Table 1. Relative the baseline model of one SARSA and one Q-learning agent, Δ decreases by 29.72%. Moreover, the average number of time steps increases by roughly 50% to 1,089,440. This suggests that while collusive outcomes are inhibited in terms of the profit measure Δ and learning times, asymmetric reinforcement learning procedures with greater than two agents can still learn anti-competitive behavior given enough time.

Figure 4 visualizes the algorithm's ability to implement reward-schemes in order to sustain converged strategies in equilibrium. A deviation in period $t = 3$ by SARSA (Q-learning) is met with a swift punishment in period $t = 4$ by Q-learning (SARSA). Shortly thereafter, these algorithms revert back to their pre-deviation prices. Seemingly, these algorithms retain the power to implement trigger strategies even in the face of more competition within the market.

Figure 4



A..2 *Reward-Punishment Schemes Prior to Convergence*

Even though trigger strategies are successfully implemented by both algorithms upon convergence[13] at around 723,000 time steps, it's worth exploring whether such time is necessary to sustain collusive outcomes. Notably, in Figure 1, the value of Δ comfortably surpasses the competitive benchmark with 95% confidence by time step 400,000. To further investigate, I conducted tests to check if reward-punishment schemes could be effectively implemented at this time step, as well as at time step 200,000. It's important to note that in each case for each episode, the algorithms had not yet converged.

[13] Recall that convergence is checked for every 100 time steps and is said to be achieved when neither algorithms' optimal actions in any state have changed for 1,000 consecutive time steps that convergence is checked for.

Figure 5

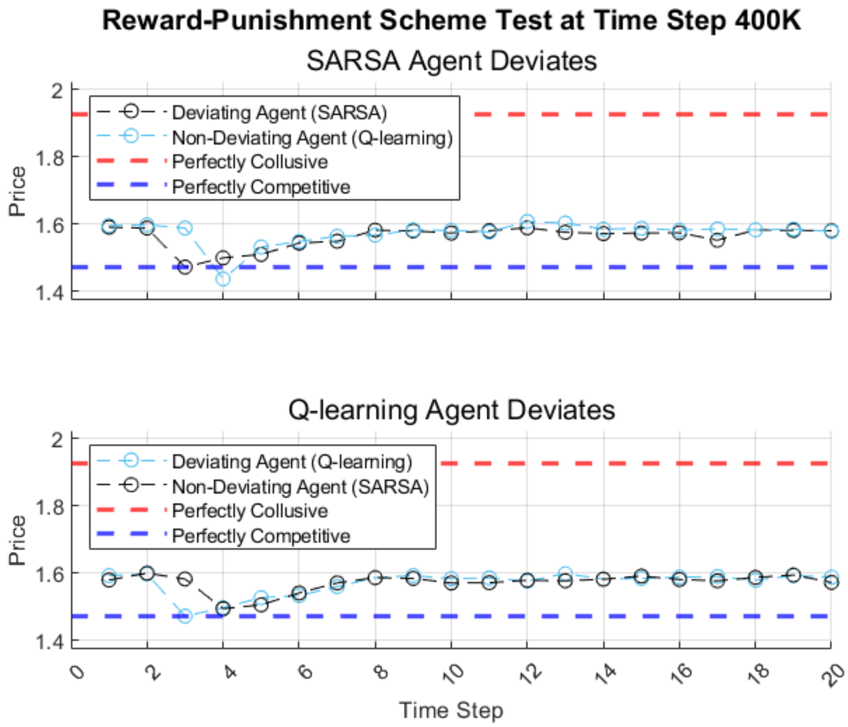
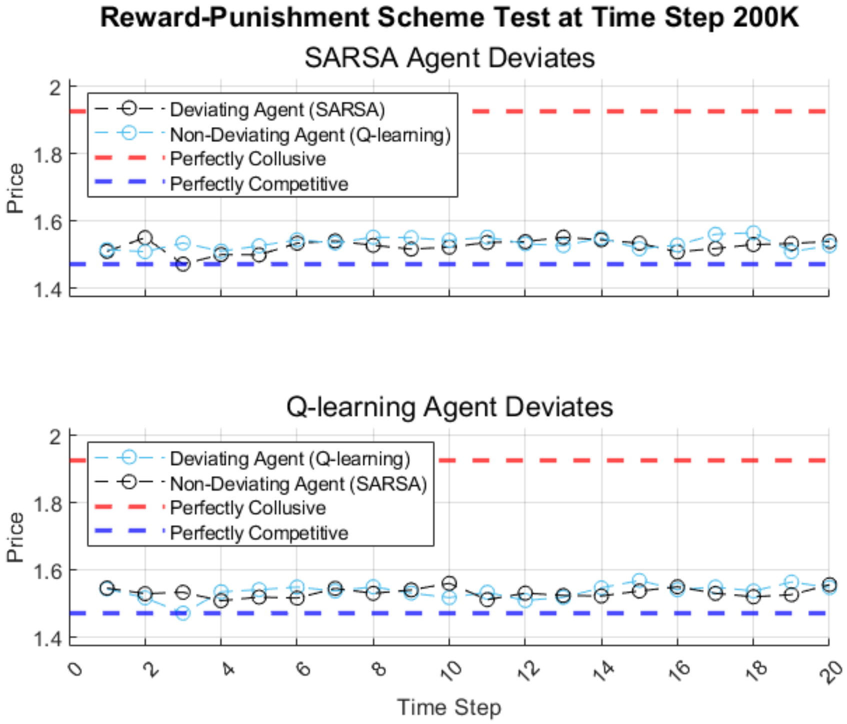


Figure 4 illustrates that convergence is not a prerequisite for either algorithm to implement trigger strategies. In this scenario, reward-punishment schemes are tested approximately 320,000 time steps prior to algorithmic convergence, and both algorithms exhibit behavior identical to that shown in Figure 3. However, when this setup is extended to allow the algorithms to interact for only 200,000 time steps, their behavior becomes more erratic, as demonstrated by Figure 5.

Figure 6



A..3 *Varying Learning Parameters*

A critical component of these algorithms' collusive capabilities is their respective learning parameters: the discount factor δ_i, the learning rate α_i, and the ϵ-greedy experimentation parameter β_i. In this setting, I allow these parameters to vary across algorithms to determine outcome differentials from the baseline model.

Firstly, I let δ_i vary across 0.00, 0.50, 0.75, and 0.99 for agent i while keeping it fixed at 0.95 for agent j. In both cases where the SARSA agent's discount factor varies while the Q-learner's stays constant and vice versa, Δ surpasses the competitive baseline for each of the four scenarios scenario. Moreover, convergence rates drop from roughly 975,000 to 720,000 when the discount factor of one agents transitions from zero to 0.99. Figure 7 visualizes learning over time steps when the SARSA agent's discount factor varies, and Figure 8 depicts the converse scenario. In the case of the latter, the algorithms converge to a higher level of Δ when SARSA values future returns moderately, while Figure 8 shows the converged value of Δ increases strictly monotonically in the Q-learner's discount factor.

Figure 7

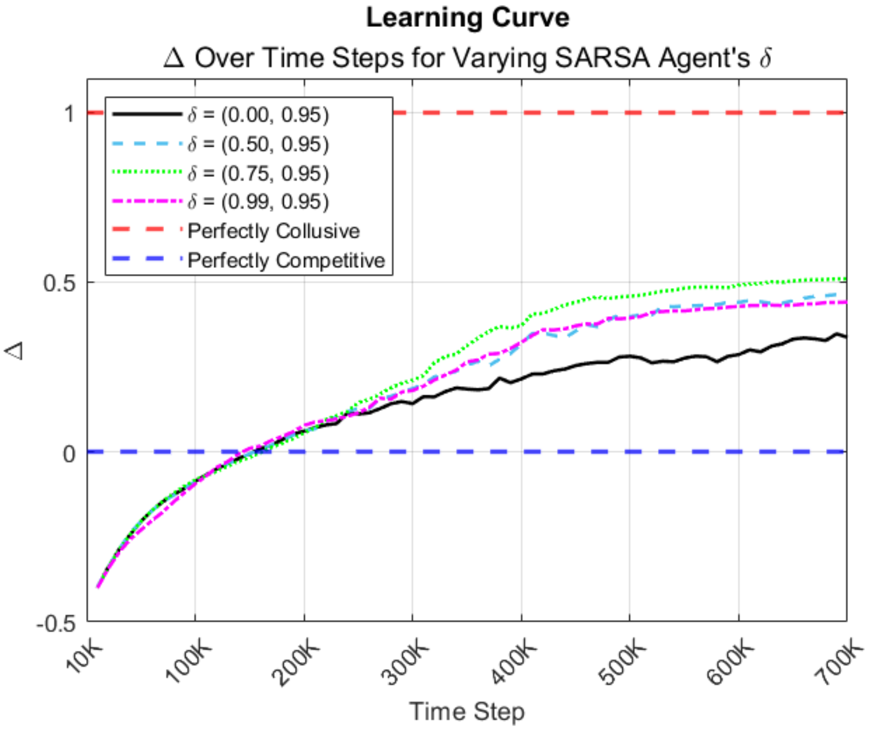
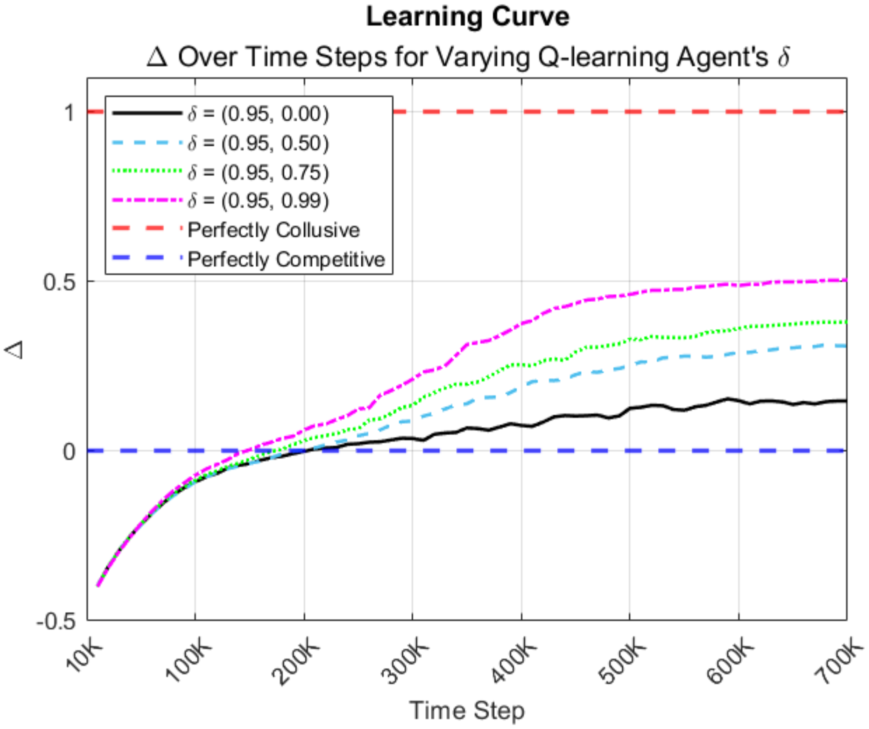
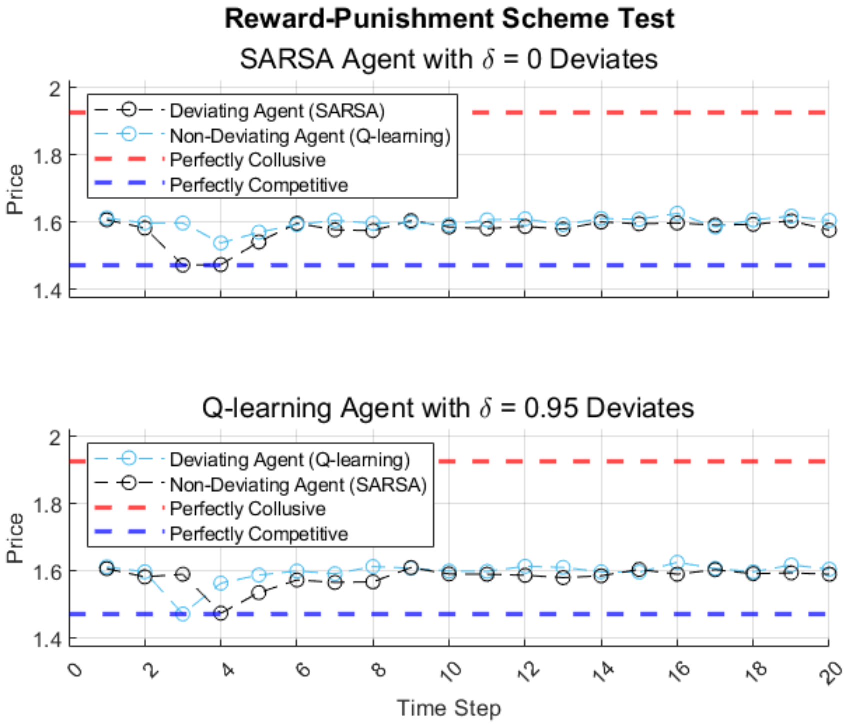


Figure 8



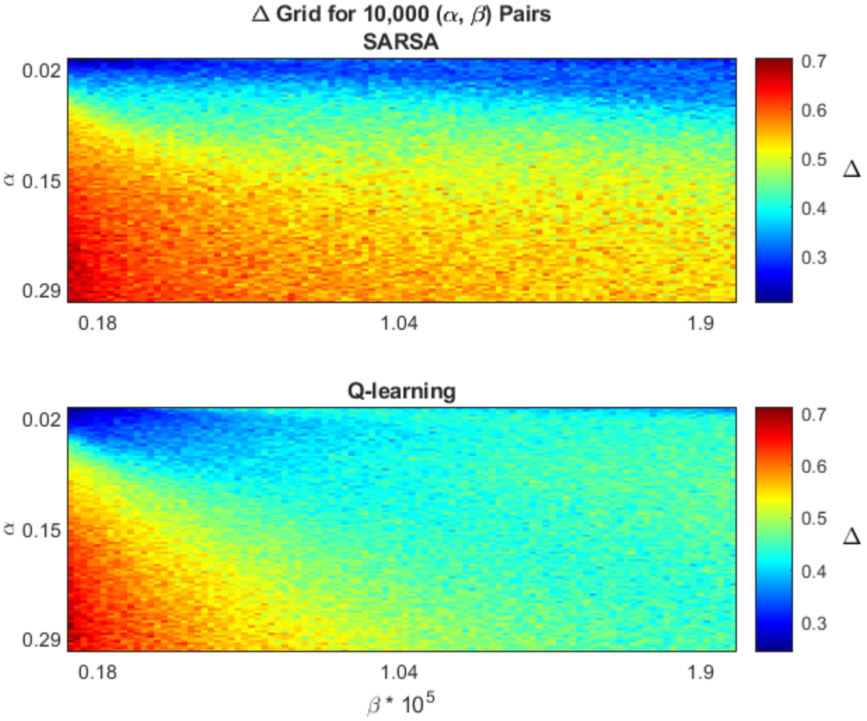
When either algorithm's discount rate varies across 0.50, 0.75, and 0.99, trigger strategies are implemented similarly to the baseline model. However, when the SARSA (Q-learning) agent's discount factor is zeror, the Q-learning (SARSA) agent, whose discount factor is fixed at 0.95, has difficulty implementing trigger strategies. The case of the SARSA agent have a 0.00 discount factor is illustrated in Figure 9.

Figure 9



The selection of learning rate (α) and exploration rate (β) plays a pivotal role in shaping the efficacy of reinforcement learning algorithms. In Figure 10, I crafted a grid comprising 10,000 unique (α, β) pairs, spanning learning rates from 0.01 to 0.30 and experimentation parameters ranging from 10^{-6} to $2 * 10^{-5}$. For every (α, β) pairing, I computed the average Δ of each agent over the final 100,000 time steps leading up to convergence across 100 episodes ($E = 100$).

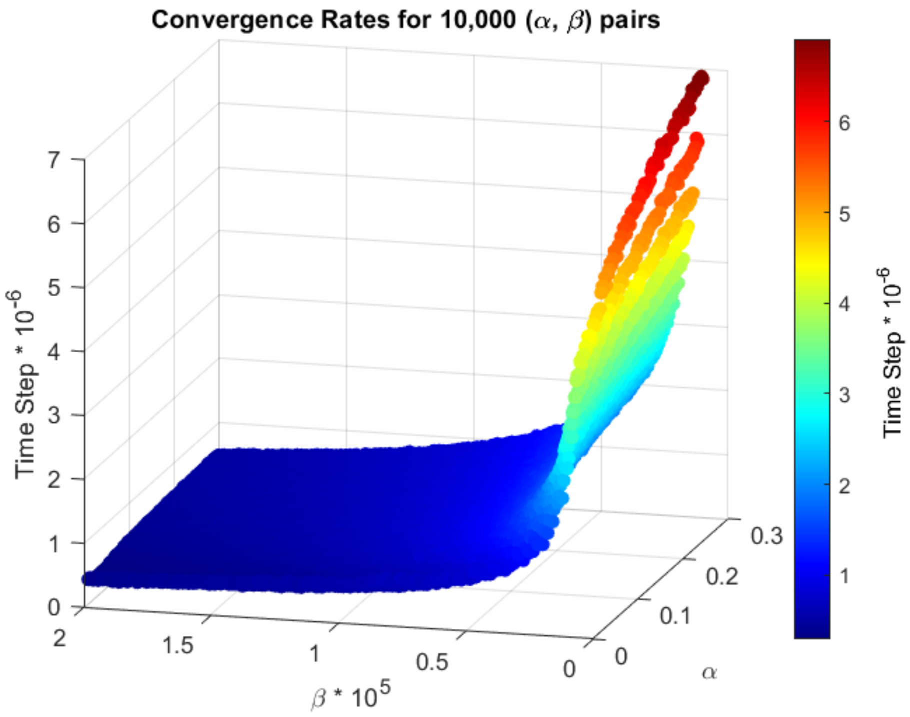
Figure 10



The discerned patterns revealed a noteworthy divergence between SARSA and Q-learning algorithms. Notably, Q-learning exhibited superior resilience to low learning rates, while SARSA sustained Δ values above 0.50 as long as α approximated values greater than 0.15, irrespective of β. The overarching trend unveiled an optimal performance domain for both algorithms, characterized by elevated learning rates coupled with diminished experimentation rates. This strategic configuration facilitated swift exploitation of the most lucrative actions, particularly advantageous in environments featuring compact state spaces akin to the one under scrutiny in this experiment. Additionally, amplifying α magnifies the influence of novel Q-matrix updates, accelerating the learning process and enhancing algorithmic efficiency.

From an economic standpoint, these findings underscore the potential of finely tuned algorithms to surpass profit thresholds that far exceed the Bertrand-Nash equilibrium. Such a revelation should serve as a warning to antitrust authorities, signaling the emergence of a landscape where algorithmic optimization can yield profit margins going well beyond traditional economic benchmarks.

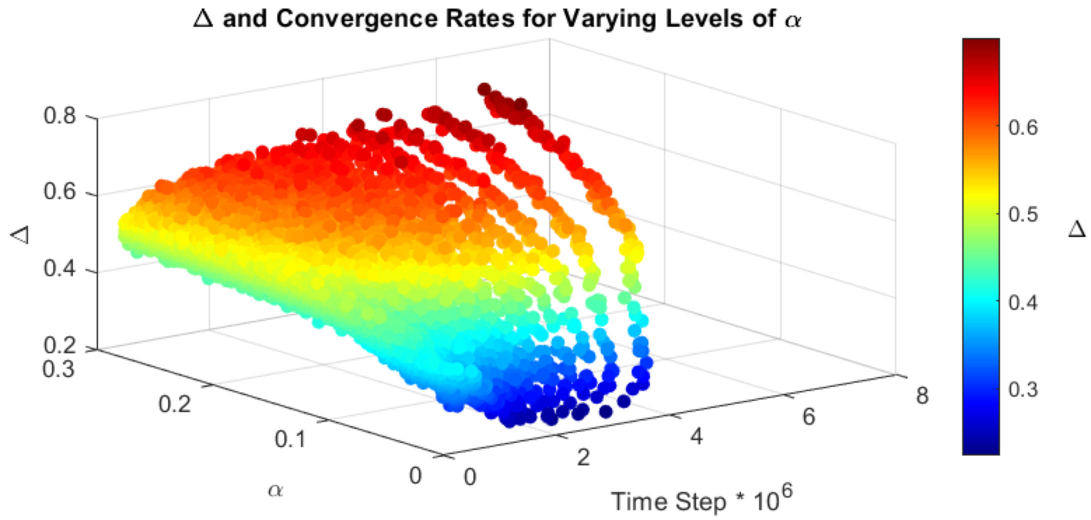
Figure 11

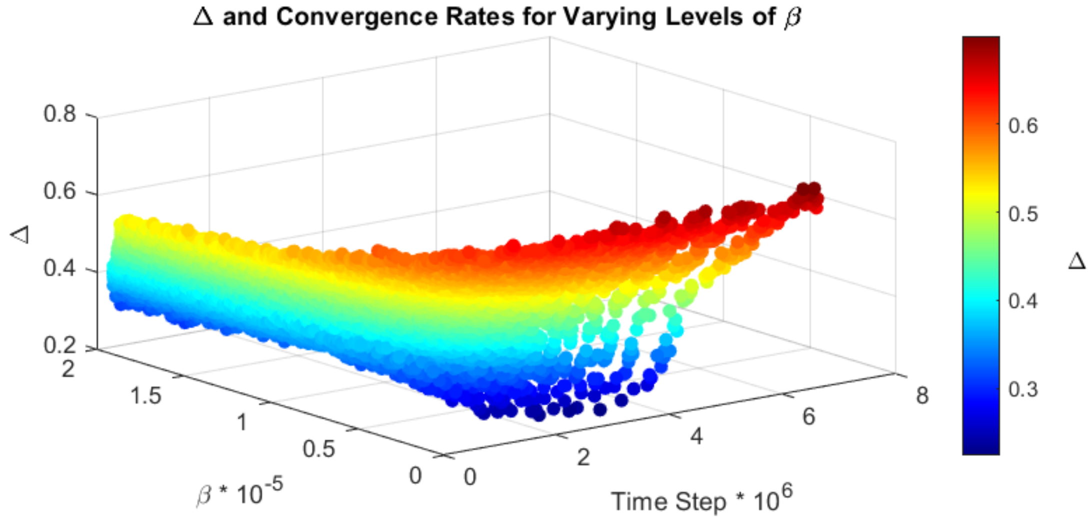


The convergence rates depicted in Figure 11, representing a comprehensive grid of 10,000 combinations of learning and experimentation rates, unveil intriguing insights. Notably, across various values of α for a given β, convergence rates exhibit uniformity. However, a pivotal observation emerges as the experimentation parameter approximates $0.5 * 10^{-5}$: a vertical asymptote materializes, instigating an exponential surge in convergence rates. This distinctive behavior underscores that β remains the predominant determinant of convergence rates. Nonetheless, as elucidated in Figure 10, careful calibration of α proves indispensable to guide these algorithms towards optimal convergence outcomes.

Figure 12 illustrates the interplay between Δ and the time step required for convergence concerning either α or β. The top panel reveals a noteworthy trend: as both the learning and convergence rates escalate, the profit gain measure Δ demonstrates a corresponding increase. Additionally, the bottom panel unveils a compelling observation: as the exploration parameter decreases and simultaneously the convergence rate escalates, Δ attains elevated values. This dual-panel representation underscores the relationship between the models' parameters and the resulting convergence dynamics, offering insights into the optimization of algorithmic performance.

Figure 12





A..4 *Zero Recall*

Originally, I considered a scenario where each firm had a one-period recall, denoted as $k_i = 1 \ \forall i \in \{1, \ldots, n\}$. This implies that at time step t, firm i retained memory of the price vector $\boldsymbol{p}_{t-k_i}$ but not of $\boldsymbol{p}_{t-h}$ for all $h > k_i$. Building upon this framework, I introduced a more challenging setting where agent i has no memory ($k_i = 0$), while agent j maintains a one-period recall ($k_j = 1$). In this extended setup, the state space for agent i is reduced to a singleton, implying that, from its perspective, all states are indistinguishable (Asker et al., 2022). Meanwhile, the state space for agent j remains of size $m^{n \cdot k_j} = 225$.

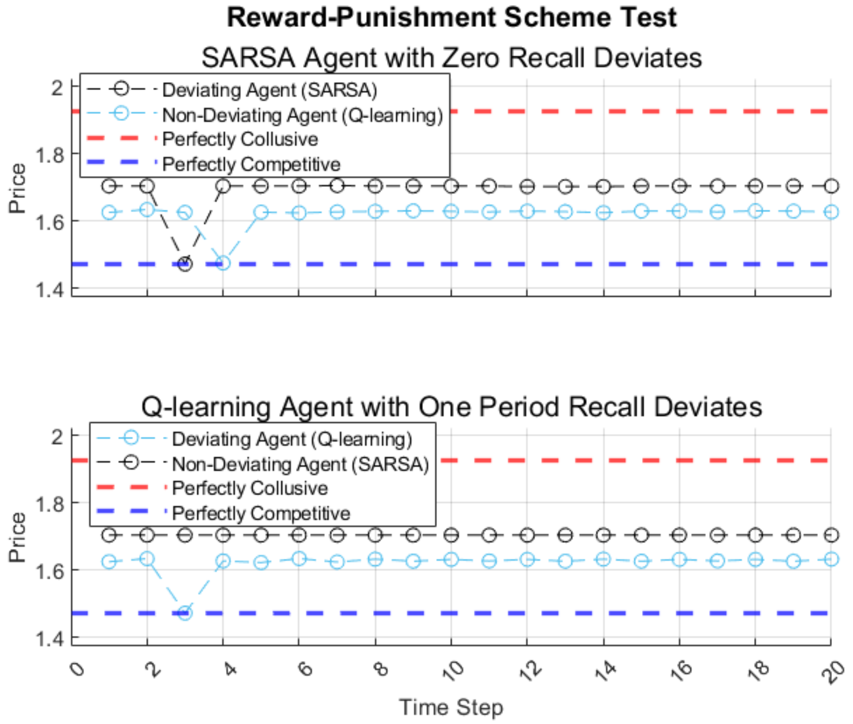
In experiments where either a SARSA or Q-learning agent is subjected to zero-recall, the results are notably similar. When the SARSA agent has zero-recall, Δ_i and Δ_j, on average across $E = 100$ episodes, are 0.3654 and 0.8341 for the SARSA and Q-learning agents, respectively. Similarly, when the Q-learner has no memory, Δ_i and Δ_j are 0.5771 and 0.2873, respectively. This suggests that even when one algorithm is severely inhibited, supracompetitive profit levels are still attainable.

However, the quantity for the agent with zero recall, in both cases, falls significantly below the competitive outcome, indicating that the algorithm with one-period recall might effectively be dominating the market, potentially leading to market control. This observation aligns with Hettich (2021), where a comparison between a more powerful algorithm (Deep Q-network) and a less powerful one (Q-learning) showed that the former gained substantial market power.

Additionally, it is noteworthy that the agent with zero recall is unable to implement trigger strategies. This limitation arises because, if the agent with one-period recall deviates at time step t, the other agent has no recollection of this deviation, as it interprets every state as identical. Conversely, the algorithm with memory of time step t can effectively implement trigger strategies following a deviation at time step $t + 1$, promptly reverting to its price set prior to the deviation shortly thereafter. Figure 13 below depicts the scenario of the SARSA agent having zero-recall, and the visualization is practically identical when the roles are reversed.

Lastly, convergence takes roughly 70,000 time steps longer relative to the baseline model, indicating that learning collusive outcomes is inhibited when one algorithm is constrained. This adds further credence to algorithms learning tacit collusion via repeated interactions with each other in the environment.

Figure 13



A..5 *Stochastic Demand Setting*

In the baseline case, I considered deterministic demand, where each of the parameters within the economic environment was fixed. However, it's more realistic to assume that demand is stochastic, with these parameters fluctuating due to random shocks. In this section, I allow the inverse index of aggregate demand, denoted as a_0, to vary.

In this stochastic setting, a_0 is distributed uniformly over the set $\{a_0^L, 0, a_0^H\}$, where $a_0^L = -a_0^H$. To illustrate, when $a_0 = a_0^L$, demand experiences a negative shock, and when $a_0 = a_0^H$, demand experiences a positive shock. These shocks occur at each time step for all episodes entirely idiosyncratically.

I consider two cases: $a_0^L = 0.15$ and $a_0^L = 0.30$. When the former is true, Δ converges to 0.4364, aligning well with the case of no demand shocks. When the latter holds, Δ decreases to 0.3568 at convergence, roughly 27% lower relative to when $a_0 = 0$ with probability one. Convergence in the event of a_0^L equal to 0.15 and 0.30 is achieved, on average, at $t = 747,482$ and $t = 794,613$, respectively. This convergence rate lies, on average, about 47,000 time steps above the case when the inverse index a_0 is deterministically equal to zero. These results suggest that demand shocks hamper algorithmic collusion to a slight degree, but do not completely prevent it. Moreover, reward-punishment schemes are implemented in a manner practically identical to that observed in Figure 3, ensuring that anti-competitive outcomes are sustained even in the presence of demand shocks.

A..6 *Differing Pricing Spaces*

In practice, it is highly implausible that each firm tunes its respective algorithm with an identical action space, even when dealing with similar products. To capture this realistic scenario, I extend the baseline model, allowing firms to differ in the action spaces each algorithm is trained on. Both firms still have $m = 15$ actions to choose from, but now the firm whose action space differs from the baseline model only has a greater average profit when setting five of the fifteen possible prices. Thus, the firm with this action space can be seen as being at a disadvantage.

When the action space of SARSA or Q-learning changes, the profit gain measure Δ does not vary significantly to that seen in Table 1. However, when the SARSA agent's action space is altered, convergence rates increase by approximately 190,000 time steps, while when the Q-learner is subjected to such a change convergence rates only increase by roughly 10,000 time steps. These differing convergence rates provide further evidence that SARSA has slower learning times relative to Q-learning, especially when at a disadvantage regarding how it is trained. Importantly, in both cases, each algorithm can implement a trigger strategy in a similar manner to that visualized in Figure 3.

VII. Conclusion

This paper discusses the exploration of heterogeneous algorithmic collusion attempting to unravel the intricacies of such collusive behavior. The findings reveal that relatively simple SARSA and Q-learning agents possess the capability to learn pricing strategies that exceed the competitive level, leading to the attainment of supracompetitive outcomes. Additionally, these agents demonstrate the ability to retain anti-competitive profits and prices via the implementation of learned trigger strategies. The ensuing results shed light on the persistence and efficiency of collusive strategies from asymmetric AI systems, offering insights into the speed with which such strategies can be acquired.

The findings of this study should raise concerns for antitrust authorities, adding to the growing body of experimental and empirical evidence that strongly indicates the existence of tacit algorithmic collusion. While the focus on homogeneous algorithmic collusion is noteworthy, in practice diverse firms are likely engaging with different software companies to implement varied pricing technologies and such a choice of technology is conditional the varied costs of implementation. This paper serves as a compelling proof-of-concept, demonstrating that algorithms with distinct architectures not only have the capacity to learn collusive behavior, but also sustain such outcomes in equilibrium with the use of trigger strategies. This underscores the plausibility of pricing technologies engaging in collusion within real-world environments. Importantly, these algorithms don't rely on a shared language or explicit communication among managers to achieve anti-competitive prices and profits. The implication is that existing antitrust measures may be insufficient in addressing the complexities introduced by algorithmic collusion. The absence of communication channels and the ability of algorithms with diverse structures to independently learn collusive strategies challenges the traditional tools used by antitrust authorities. This aligns with the perspectives from a wide-range of legal scholars suggesting a pressing need to re-evaluate current antitrust laws to effectively navigate cases involving tacit algorithmic collusion.

A critical challenge confronting algorithms like SARSA and Q-learning lies in their notably sluggish learning times. The average convergence iteration for price competition, as observed with these two agents, stands at 723,230, with exploration demanding approximately 500,000 time steps before the algorithms begin stabilization. This temporal demand raises concerns, especially when considering practical applications as the state spaces increases exponentially in the number firms. As highlighted by Calvano et al. (2020), even if each time step lasted only a few minutes, the cumulative learning times would extend over several years, rendering the practical adoption of these algorithms seemingly unrealistic. In reality, while training is likely done offline before the algorithms are deployed, learning likely continues thereafter to ensure the offline work does not become antiquated. After these algorithms are put into the real-world, such an environment could be differing from the one it was trained in. Adding to the complexity, SARSA and Q-learning agents encounter limitations in handling continuous action spaces—a prevalent feature likely employed in real-world pricing software. This underscores a crucial inadequacy in their potential applicability to practical scenarios. Therefore, a substantial extension would be to extend this framework to algorithms that are capable of handling a continuum of actions to determine: (1) if asymmetric reinforcement learning algorithms acting in that space can still obtain collusive outcomes and (2) if the knowledge accumulated in the learning environment is transferable to a slightly different market environment as would likely be the case in practice. Furthermore, such algorithms may be much more likely to learn at increased rate given they rely on advanced technology, namely neural networks and, in particular, policy gradient-based reinforcement learning procedures in which there is a current scarcity in the experimental AI pricing literature. Such a study corroborating this works' results would render the idea of tacit algorithmic collusion much more practically plausible.

References

Abadi, Martín and **David G. Andersen** (2016). "Learning to Protect Communications with Adversarial Neural Cryptography". DOI: https://doi.org/10.48550/arXiv.1610.06918.

Agarwal, Alekh, **Nan Jiang**, **Sham M. Kakade**, and **Wen Sun** (2022). *Reinforcement Learning: Theory and Algorithms.* 1st. TBD: TBD. ISBN: TBD. URL: TBD.

Asker, John, **Chaim Fershtman**, and **Ariel Pakes** (2022). "Artifical Intelligence, Algorithmic Design, and Pricing". *American Economic Association* 112, pp. 452–56. DOI: 10.1257/pandp.20221059.

Assad, Stephanie, **Robert Clark**, **Daniel Ershov**, and **Lei Xu** (2024). "Algorithmic Pricing and Competition: Empirical Evidence from the German Retail Gasoline Market". *Journal of Politcal Economy (Forthcoming)* 132.2, TBD. DOI: `https://doi.org/10.1086/726906`.

Auer, Peter (2002). "Using Confidence Bounds for Exploitation-Exploration Trade-offs". *Journal of Machine Learning Research* 3, pp. 397–422. URL: `https://dl.acm.org/doi/10.5555/944919.944941`.

Bertsekas, Dimitri P. (2019). *Reinforcement Learning and Optimal Control.* 1st. Nashua, NH: Athena Scientific. ISBN: 9781886529397. URL: `http://athenasc.com/rlbook_athena.html`.

Brown, Zach Y. and **Alexander MacKay** (2023). "Competition in Pricing Algorithms". *American Economic Journal: Microeconomics* 15.2, pp. 109–156. DOI: `https://doi.org/10.1257/mic.20210158`.

Busoniu, Lucian, **Bart De Schutter**, and **Robert Babuska** (2008). "A Comprehensive Survey of Multiagent Reinforcement Learning". *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 38.2, pp. 156–172. DOI: `https://doi.org/10.1109/TSMCC.2007.913919`.

Calvano, Emilio, **Giacoma Calzolari**, **Vincenzo Denicolò**, and **Sergio Pastorello** (2019). "Algorithmic Pricing: What Implications for Competition Policy?" *Review of Industrial Organization* 55.9, pp. 155–171. DOI: `https://doi.org/10.1007/s11151-019-09689-3`.

— (2020). "Artificial Intelligence, Algorithmic Pricing, and Collusion". *American Economic Review* 110.10, pp. 3267–3297. DOI: `https://doi.org/10.1257/aer.20190623`.

Ezrachi, Ariel and **Maurice E. Stucke** (2017). "Artificial Intelligence and Collusion: When Computers Inhibit Competition". *University of Illinois Law Review* 2017.5, pp. 1775–1810. URL: `https://www.illinoislawreview.org/wp-content/uploads/2017/10/Ezrachi-Stucke.pdf`.

— (2020). "Sustainable and Unchallenged Algorithmic Tacit Collusion". *Northwestern Journal of Technology and Intellectual Property* 17.2, pp. 217–260. URL: `https://scholarlycommons.law.northwestern.edu/njtip/vol17/iss2/2`.

Frick, Kevin Michael (2023). "Convergence Rates and Collusive Outcomes of Pricing Algorithms". *Social Science Research Network (SSRN).* DOI: `https://dx.doi.org/10.2139/ssrn.4527452`.

Friedman, James (1971). "A Non-cooperative Equilibrium for Supergames". *Review of Economic Studies* 38.1, pp. 1–12. DOI: `https://doi.org/10.2307/2296617`.

Haarnoja, Tuomas, **Aurick Zhou**, **Pieter Abbeel**, and **Sergey Levine** (2018). "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor". DOI: `https://doi.org/10.48550/arXiv.1801.01290`.

Harrington, Joseph E. (2018). "Developing Competition Law for Collusion by Autonomous Artifical Agents". *Journal of Competition Law and Economics* 14.3, pp. 331–363. DOI: `https://doi.org/10.1093/joclec/nhy016`.

Hettich, Matthias (2021). "Algorithmic Collusion: Insights from Deep Learning". *Social Science Research Network (SSRN).* DOI: `https://dx.doi.org/10.2139/ssrn.3785966`.

Klein, Timo (2021). "Autonomous Algorithmic Collusion: Q-learning Under Sequential Pricing". *The RAND Journal of Economics* 52.3, pp. 538–558. DOI: `https://doi.org/10.1111/1756-2171.12383`.

Maschler, Michael, **Eilon Solan**, and **Shmuel Zamir** (2020). *Game Theory.* 2nd. Cambridge University Press. ISBN: 9781108825146. URL: `https://www.cambridge.org/us/universitypress/subjects/economics/microeconomics/game-theory-2nd-edition?format=PB`.

Maskin, Eric and **Jean Tirole** (1988). "A Theory of Dynamic Oligopoly, II: Price Competition, Kinked Demand Curves, and Edgeworth Cycles". *Econometrica* 56.3, pp. 571–599. DOI: `https://doi.org/10.2307/1911701`.

Mazumdar, Aneesa (2022). "Algorithmic Collusion: Reviving Section 5 of the FTC Act". *Columbia Law Review* 122.2, pp. 449–488. URL: `https://www.jstor.org/stable/10.2307/27114356`.

Mehra, Salil K. (2016). "Antitrust and the Robo-Seller: Competition in the Time of Algorithms". *Minnesota Law Review* 100, pp. 1323–1375. URL: `https://scholarship.law.umn.edu/mlr/204`.

Mellgren, Filip (2020). "Tacit Collusion with Deep Multi-Agent Reinforcement Learning". URL: https://www.semanticscholar.org/paper/Tacit-collusion-with-deep-multi-agent-reinforcement-Mellgren/716ffdc23e36748f66e8e481077f2e9004150748.

Miklós-Thal, Jeanine and **Catherine Tucker** (2019). "Collusion by Algorithm: Does Better Demand Prediction Facilitate Coordination Between Sellers?" *American Economic Review* 65.4, pp. 1552–1561. URL: https://hdl.handle.net/1721.1/134402.

Mnih, Volodymyr, **Koray Kavukcuoglu**, **David Silver**, and **et al.** (2015). "Human Level Control Through Deep Reinforcement Learning". *Nature* 518, pp. 529–533. DOI: https://doi.org/10.1038/nature14236.

Morales, Miguel (2020). *Deep Reinforcement Learning.* 1st. Shelter Island, NY: Manning Publications Co. ISBN: 9781617295454. URL: https://www.manning.com/books/grokking-deep-reinforcement-learning.

Rummery, G.A. and **Mahesan Niranjan** (1994). "On-Line Q-Learning Using Connectionist Systems". URL: https://www.researchgate.net/publication/2500611_On-Line_Q-Learning_Using_Connectionist_Systems.

Sutton, Richard S. and **Andrew G. Barto** (2018). *Reinforcement Learning: An Introduction.* 2nd. Cambridge, MA: The MIT Press. ISBN: 9780262039246. URL: https://dl.acm.org/doi/10.5555/3312046.

Szepesvári, Csaba (2010). *Aglorithms for Reinforcement Learning.* 1st. Spring Cham. ISBN: 9783031004230. URL: https://doi.org/10.1007/978-3-031-01551-9.

Tuyls, Karl and **Gerhard Weiss** (2021). "Multiagent Learning: Basics, Challenges, and Prospects". *AI Magazine* 33.3, pp. 41–52. DOI: https://doi.org/10.1609/aimag.v33i3.2426.

Waltman, Ludo and **Uzay Kaymack** (2008). "Q-learning Agents in a Cournot Oligopoly Model". *Journal of Economic Dynamics and Control* 32.10, pp. 3275–3293. DOI: https://doi.org/10.1016/j.jedc.2008.01.003.

Watkins, Christopher J.C.H and **Peter Dayan** (1992). "Q-learning". *Machine Learning* 8, pp. 279–292. DOI: https://doi.org/10.1007/BF00992698.

Yaodong, Yang and **Jun Wang** (2020). "An Overview of Multi-agent Reinforcement Learning from Game Theoretical Persepctive". URL: https://arxiv.org/abs/2011.00583.

Zhang, Kaiqing, **Zhuoran Yang**, and **Tamer Basar** (2021). "Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms". *Handbook of Reinforcement Learning and Control* 325, pp. 321–384. DOI: https://doi.org/10.1007/978-3-030-60990-0_12.

Zhu, Changxi, **Mehdi Dastani**, and **Shihan Wang** (2024). "A Surve of Multi-agent Deep Reinforcement Learning with Communication". *Autonomous Agents and Multi-Agent Systems* 38.4. DOI: https://doi.org/10.1007/s10458-023-09633-6.

VIII. Appendix

A. *Temporal Difference (TD) Update*

To understand the intuition of the TD update, note that

$$
\begin{aligned}
Q_\pi(s_t, a_t) &\triangleq \mathbb{E}_\pi \left[\sum_{h=0}^{\infty} \delta^h r_{t+h+1} \Big| s = s_t, a = a_t \right] && (1)\\
&= \mathbb{E}_\pi \left[r_{t+1} + \sum_{h=1}^{\infty} \delta^h r_{t+h+1} \Big| s = s_t, a = a_t \right] && (2)\\
&= \mathbb{E}_\pi \left[r_{t+1} | s = s_t, a = a_t \right] + \mathbb{E}_\pi \left[\sum_{h=1}^{\infty} \delta^h r_{t+h+1} \Big| s = s_t, a = a_t \right] && (3)\\
&= \mathbb{E}_\pi \left[r_{t+1} | s = s_t, a = a_t \right] + \mathbb{E}_\pi \left[\sum_{h=0}^{\infty} \delta^{h+1} r_{t+h+2} \Big| s = s_t, a = a_t \right] && (4)\\
&= \mathbb{E}_\pi \left[r_{t+1} | s = s_t, a = a_t \right] + \mathbb{E}_\pi \left[\delta \sum_{h=0}^{\infty} \delta^h r_{t+h+2} \Big| s = s_t, a = a_t \right] && (5)\\
&= \mathbb{E}_\pi \left[r_{t+1} + \delta \sum_{h=0}^{\infty} \delta^h r_{t+h+2} \Big| s = s_t, a = a_t \right] && (6)\\
&= \mathbb{E}_\pi \left[\mathbb{E}_\pi \left[r_{t+1} + \delta \sum_{h=0}^{\infty} \delta^h r_{t+h+2} \Big| s = s_t, a = a_t, s = s_{t+1}, a = a_{t+1} \right] \Big| s = s_t, a = a_t \right] && (7)\\
&= \mathbb{E}_\pi \left[\mathbb{E} \left[r_{t+1} | s = s_t, a = a_t, s = s_{t+1}, a = a_{t+1} \right] | s = s_t, a = a_t \right] && (8)\\
&+ \delta \mathbb{E}_\pi \left[\mathbb{E}_\pi \left[\sum_{h=0}^{\infty} \delta^h r_{t+h+2} \Big| s = s_t, a = a_t, s = s_{t+1}, a = a_{t+1} \right] \Big| s = s_t, a = a_t \right] && (9)\\
&= \mathbb{E}_\pi \left[\mathbb{E} \left[r_{t+1} | s = s_t, a = a_t \right] | s = s_t, a = a_t \right] && (10)\\
&+ \delta \mathbb{E}_\pi \left[\mathbb{E}_\pi \left[\sum_{h=0}^{\infty} \delta^h r_{t+h+2} \Big| s = s_{t+1}, a = a_{t+1} \right] \Big| s = s_t, a = a_t \right] && (11)\\
&= \mathbb{E}_\pi \left[r_{t+1} | s = s_t, a = a_t \right] + \delta \mathbb{E}_\pi \left[\mathbb{E}_\pi \left[\sum_{h=0}^{\infty} \delta^h r_{t+h+2} \Big| s = s_{t+1}, a = a_{t+1} \right] \Big| s = s_t, a = a_t \right] && (12)\\
&= \mathbb{E}_\pi \left[r_{t+1} + \delta \mathbb{E}_\pi \left[\sum_{h=0}^{\infty} \delta^h r_{t+h+2} \Big| s = s_{t+1}, a = a_{t+1} \right] \Big| s = s_t, a = a_t \right] && (13)\\
&= \mathbb{E}_\pi \left[r_{t+1} + \delta Q_\pi(s_{t+1}, a_{t+1}) \Big| s = s_t, a = a_t \right]. && (14)
\end{aligned}
$$

where we move from line (6) to line (7) via the so-called Tower Property of Conditional Expectation. An analogous derivation exists for the value function, $V_\pi(s_t)$.

The term $r_{t+1} + \delta Q_\pi(s_{t+1}, a_{t+1})$ is called the TD target for SARSA and reflects the expected return based on the current policy π, while the difference between the TD target and $Q_\pi(s_t, a_t)$ is SARSA's TD error. Given the TD-target for SARSA involves the current policy π, this gives rise to SARSA's on-policy nature.

On the other hand, Q-learning uses the TD target of $r_{t+1} + \delta \max_{a' \in \mathcal{A}} Q_\pi(s_{t+1}, a')$ which is essentially the estimated Bellman equation for Q_π and can be derived by taking the maximum of the above equation over all polices π given for intuition of the TD update. This implies we take the maximum over actions $a' \in \mathcal{A}$ since this is what the optimal policy would do and want to want our policy π to get as close as possible to the optimal policy π^*. Consequently, Q-learning constructs its TD-target based on the greedy policy, i.e., the policy that gives rise to the Q-value maximizing action in state s_{t+1} which is not necessarily the policy actually being followed giving rise to the off-policy nature of Q-learning.

The goal of TD-based reinforcement learning algorithms is to minimize their respective TD errors. Note that in the update rules for SARSA and Q-learning, once their respective TD errors are zero no further updating occurs.

B. *Relationship Between the Value and Action-Value Functions*

By applying the law of total expectation, it follows that

$$\begin{aligned} V_\pi(s_t) &= \mathbb{E}_\pi\left[\sum_{h=0}^{\infty} \delta^h r_{t+h+1} \middle| s = s_t\right] \\ &= \mathbb{E}_\pi\left[\mathbb{E}_\pi\left[\sum_{h=0}^{\infty} \delta^h r_{t+h+1} \middle| s = s_t, a = a_t\right] \middle| s = s_t\right] \\ &= \mathbb{E}_\pi\left[Q_\pi(s_t, a_t) \middle| s = s_t\right]. \end{aligned}$$