Autonomous Algorithmic Collusion: Q-Learning Under Sequential Pricing

Timo Klein
Tinbergen Institute; University of Amsterdam

Bergen Competition Policy Conference 2019
BECCLE, Norway

26 April 2019


ALPHAGO
TRIBECA FILM FESTIVAL 2017
BFI LONDON FILM FESTIVAL OFFICIAL SELECTION 2017
WINNER TRAVERSE CITY FILM FESTIVAL 2017
OFFICIAL SELECTION MILL VALLEY FILM FESTIVAL 2017
Official Selection WARSAW FILM FESTIVAL
VIFF Official Selection 2017
NETFLIX

1. Can AI-driven pricing algorithms learn to collude?
2. Would this be a competition law infringement?

1. Can AI-driven pricing algorithms learn to collude?
2. Would this be a competition law infringement?

- Concerns mostly based on intuitive interpretation of AI
- Many skeptical that this is even a problem

Outline

Literature

Literature (1/2)

- Calvano, Calzolari, Denicolo and Pastorello (working paper, 2019)
  - Also look at Q-learning collusion
  - Results generally aligned
  - Differences:
    1. Updates occur simultaneously instead of sequentially
    2. Allow for and require self-reactive conditioning (non-Markov)
    3. Explicit analysis of punishment strategies

Literature (2/2)

- Kuhn and Tadelis (2017), Schwalbe (2018)
  - Humans and algorithms similarly ill-equiped to tacitly coordinate
  - Would assume similar cognition for humans and AI
- Tesauro and Kephart (2002), Huck, Normann and Oechssler (2003), Waltman and Kaymak (2008)
  - Use forms of Q-learning in oligopoly environments
  - Full knowledge; Not robust; Do not produce equilibrium behavior
- Cooper *et al.* (2015)
  - Certain revenue management convention may lead to collusion
  - Not equilibrium behavior
- Salcedo (2015)
  - Collusion inevitable if short-run strategy commitments and 'decode'
  - May be framed as communication; Conditions may not hold
- Miklos-Thal and Tucker (2019)
  - Better demand prediction may require lower cartel prices

Primer

Reinforcement Learning

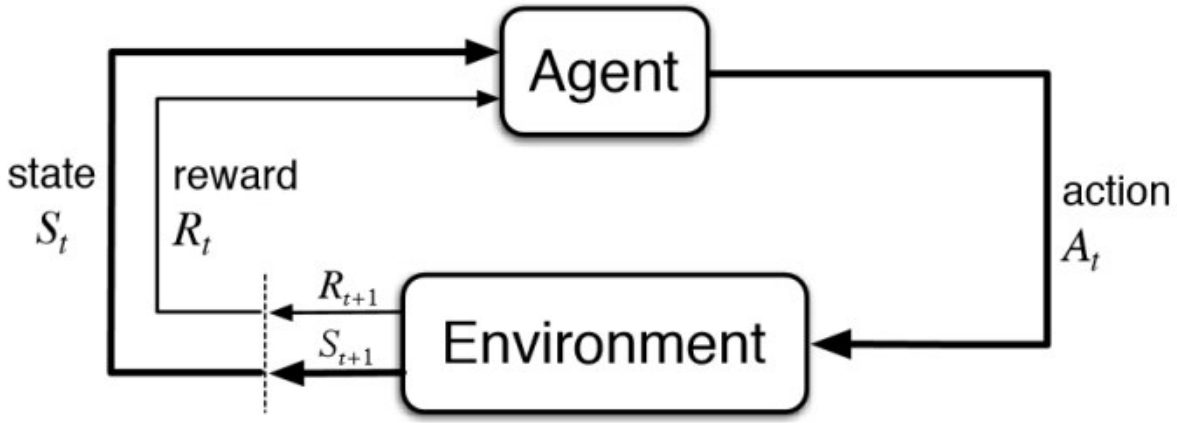


Figure: Sutton and Barto (2018)

Reinforcement Learning

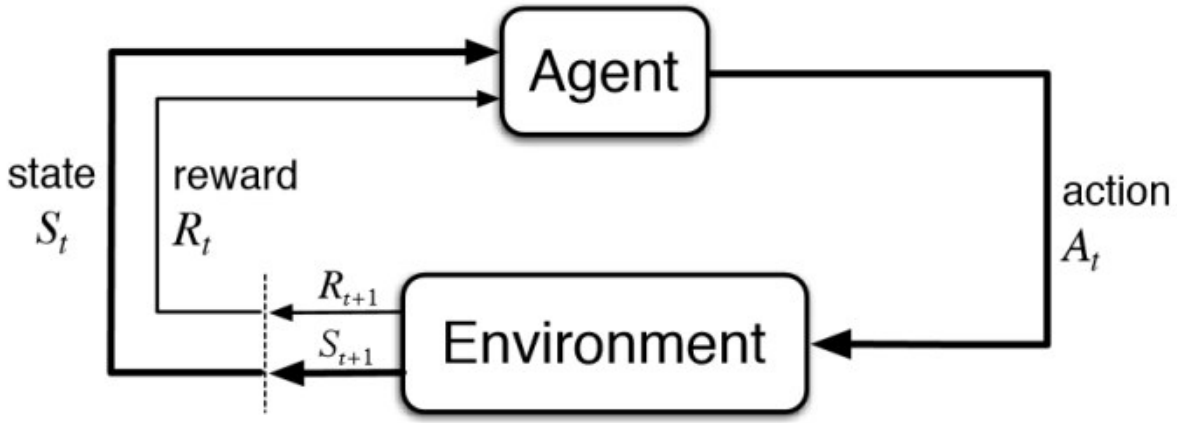


Figure: Sutton and Barto (2018)

Q-Learning (Watkins, 1989)

- Popular and well-established type of reinforcement learning
- Aims to maximize sum of discounted rewards in unknown environment
- Strong theoretical properties in single-agent settings

Primer

Q-Learning

- $Q(a, s)$ estimates discounted rewards from action $a \in A$ in state $s \in S$
- Tabular case: Q is a $|A| \times |S|$ matrix

Primer

Q-Learning

- $Q(a, s)$ estimates discounted rewards from action $a \in A$ in state $s \in S$
- Tabular case: Q is a $|A| \times |S|$ matrix

Learning Module

- Take s as old state and s' as new state
- Recursive updating:

$$Q(a, s) \leftarrow (1 - \alpha) Q(a, s) + \alpha \left(R(a, s, s') + \delta \max_{a} Q(a, s') \right)$$

Action-Selection Module

- Exogenously programmed to trade off exploitation-exploration

Primer

Q-Learning

- $Q(a, s)$ estimates discounted rewards from action $a \in A$ in state $s \in S$
- Tabular case: Q is a $|A| \times |S|$ matrix

Learning Module

- Take s as old state and s' as new state
- Recursive updating:

$$Q(a, s) \leftarrow (1 - \alpha) Q(a, s) + \alpha \left(R(a, s, s') + \delta \max_{a} Q(a, s') \right)$$

Action-Selection Module

- Exogenously programmed to trade off exploitation-exploration

- Provably converges to optimal policy under single-agent learning
- No theoretical guarantee under multi-agent learning

Environment and Algorithm

Environment and Algorithm

Environment: Sequential Competition

- Maskin and Tirole (1988), firms $i \in \{1, 2\}$ set prices in turn

Environment and Algorithm

Environment: Sequential Competition

- Maskin and Tirole (1988), firms $i \in \{1, 2\}$ set prices in turn
- Prices $p^i \in \{0, \frac{1}{k}, \frac{2}{k}, ..., 1\}$, so k intervals between 0 and 1

Environment and Algorithm

Environment: Sequential Competition

- Maskin and Tirole (1988), firms $i \in \{1, 2\}$ set prices in turn
- Prices $p^i \in \{0, \frac{1}{k}, \frac{2}{k}, ..., 1\}$, so k intervals between 0 and 1
- Per-period profit $\pi^i = (p^i - c^i)q^i$

Environment and Algorithm

Environment: Sequential Competition

- Maskin and Tirole (1988), firms $i \in \{1, 2\}$ set prices in turn
- Prices $p^i \in \{0, \frac{1}{k}, \frac{2}{k}, ..., 1\}$, so k intervals between 0 and 1
- Per-period profit $\pi^i = (p^i - c^i)q^i$
- Objective $\max \sum_{s=0} \delta^s \pi^i_{t+s}$

Environment and Algorithm

Environment: Sequential Competition

- Maskin and Tirole (1988), firms $i \in \{1, 2\}$ set prices in turn
- Prices $p^i \in \{0, \frac{1}{k}, \frac{2}{k}, ..., 1\}$, so k intervals between 0 and 1
- Per-period profit $\pi^i = (p^i - c^i)q^i$
- Objective $\max \sum_{s=0} \delta^s \pi^i_{t+s}$
- Scope: homogeneous good, linear demand, 2 firms

$$q^i = \begin{cases} 1 - p^i & \text{if } p^i < p^j \\ 0.5(1 - p^i) & \text{if } p^i = p^j \\ 0 & \text{if } p^i > p^j \end{cases}$$

Environment and Algorithm

Algorithm: Sequential Q-Learning

Learning Module

- Take old state $s = p^j_{t-1}$ and new state $s' = p^j_{t+1}$,

$$Q(p^i_t, s) \leftarrow (1-\alpha)Q(p^i_t, s) + \alpha\left(\pi(p^i_t, s) + \delta\pi(p^i_t, s') + \delta^2 \max_p Q(p, s')\right)$$

Action-Selection Module

Environment and Algorithm

Algorithm: Sequential Q-Learning

Learning Module

- Take old state $s = p^j_{t-1}$ and new state $s' = p^j_{t+1}$,

$$Q(p^i_t, s) \leftarrow (1-\alpha)Q(p^i_t, s) + \alpha\left(\pi(p^i_t, s) + \delta\pi(p^i_t, s') + \delta^2 \max_p Q(p, s')\right)$$

Action-Selection Module

- Explores with probability ε_t $\Rightarrow$ Pick any p
- Exploits with probability $1-\varepsilon_t$ $\Rightarrow$ Pick p that maximizes $Q(p, s)$

Environment and Algorithm

Algorithm: Sequential Q-Learning

Learning Module

- Take old state $s = p^j_{t-1}$ and new state $s' = p^j_{t+1}$,

$$Q(p^i_t, s) \leftarrow (1-\alpha)Q(p^i_t, s) + \alpha\left(\pi(p^i_t, s) + \delta\pi(p^i_t, s') + \delta^2 \max_p Q(p, s')\right)$$

Action-Selection Module

- Explores with probability ε_t $\Rightarrow$ Pick any p
- Exploits with probability $1-\varepsilon_t$ $\Rightarrow$ Pick p that maximizes $Q(p, s)$

- Still very basic algorithm:
  1. Slow and inefficient learning
  2. Untargetted exploration

Performance Metrics

(1) Profitability: $\Delta^i \doteq \dfrac{\text{Expected profit gains}}{\text{Joint-profit maximizing gains}} = \dfrac{Q^i(p^i, s) - Q^N}{Q^C - Q^N}$

- $\Delta^i = 1$ joint-profit maximizing outcome
- $\Delta^i = 0$ competitive outcome (defined as static Nash)

–

Performance Metrics

(1) Profitability: $\Delta^i \doteq \dfrac{\text{Expected profit gains}}{\text{Joint-profit maximizing gains}} = \dfrac{Q^i(p^i, s) - Q^N}{Q^C - Q^N}$

- $\Delta^i = 1$ joint-profit maximizing outcome
- $\Delta^i = 0$ competitive outcome (defined as static Nash)

–

(2) Optimality: $\Gamma^i \doteq \dfrac{\text{Expected future profits}}{\text{Best-response future profits}} = \dfrac{Q^i(p^i, s)}{\max_p Q^{i*}(p, s)}$

- Q^{i*} are the optimal Q-values given current competitor strategy
- $\Gamma^i = 1$ best response
- $\Gamma^i < 1$ shows degree below best response

Simulation Results

Simulation Results

Two cases:

- (1) Q-learning versus fixed-strategy tit-for-tat
- (2) Q-learning versus Q-learning

Simulation set-up:

Simulation Results

Two cases:

- (1) Q-learning versus fixed-strategy tit-for-tat
- (2) Q-learning versus Q-learning

Simulation set-up:

- Price set: $k = \{6, 12, 50\}$ possible prices

Simulation Results

Two cases:

- (1) Q-learning versus fixed-strategy tit-for-tat
- (2) Q-learning versus Q-learning

Simulation set-up:

- Price set: $k = \{6, 12, 50\}$ possible prices
- $R = 1000$ runs of $T = 300(k+1)^2$ periods each

Simulation Results

Two cases:

- (1) Q-learning versus fixed-strategy tit-for-tat
- (2) Q-learning versus Q-learning

Simulation set-up:

- Price set: $k = \{6, 12, 50\}$ possible prices
- $R = 1000$ runs of $T = 300(k+1)^2$ periods each
- Learning parameters: $\alpha = 0.5$, $\delta = 0.95$ and $\varepsilon_t = (1-\theta)^t$

Simulation Results

Two cases:

- (1) Q-learning versus fixed-strategy tit-for-tat
- (2) Q-learning versus Q-learning

Simulation set-up:

- Price set: $k = \{6, 12, 50\}$ possible prices
- $R = 1000$ runs of $T = 300(k+1)^2$ periods each
- Learning parameters: $\alpha = 0.5$, $\delta = 0.95$ and $\varepsilon_t = (1-\theta)^t$
- θ such that $\varepsilon_t = 0.5\%$ halfway and $\varepsilon_t = 0.0025\%$ at the end

Simulation Results

Two cases:

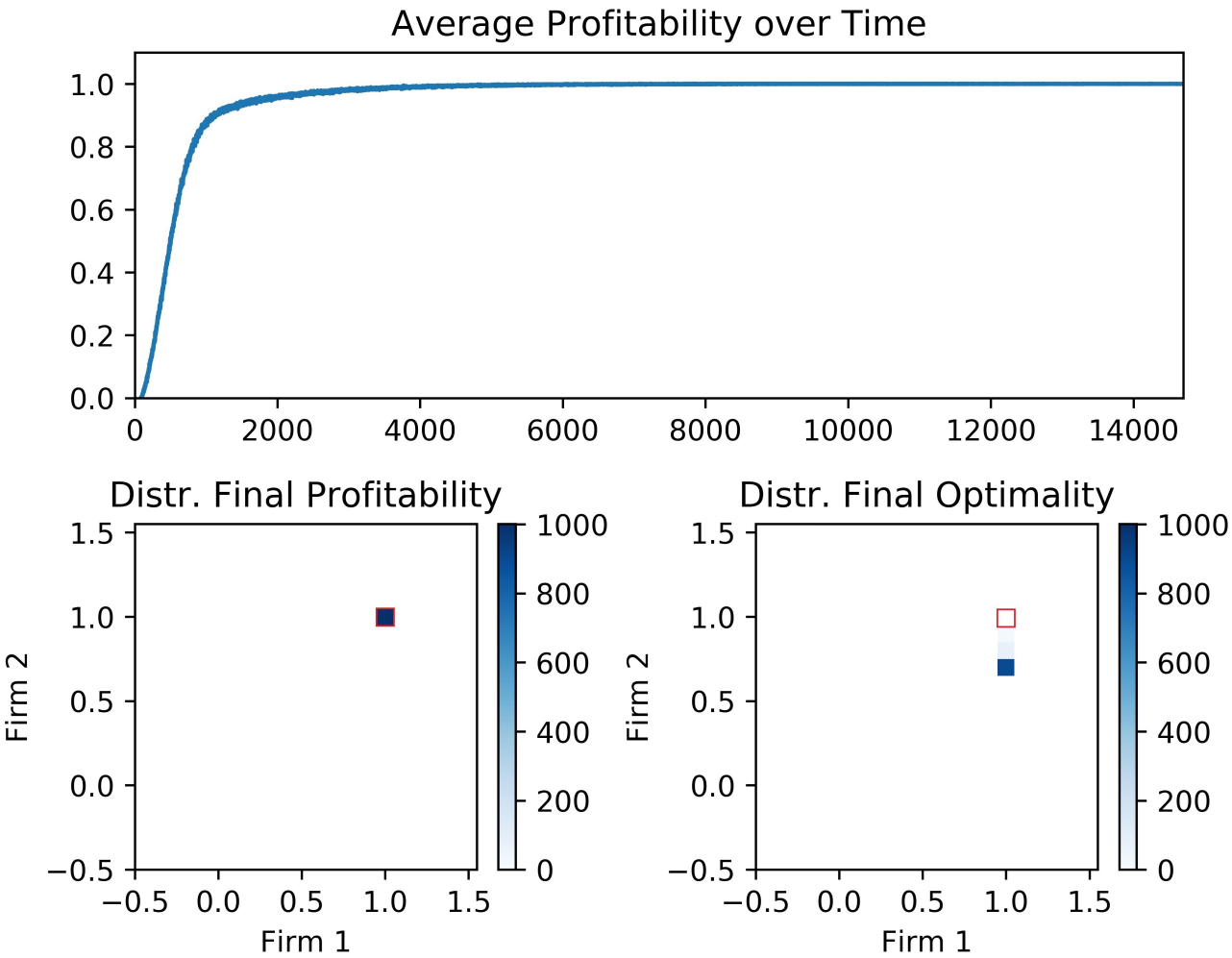
- (1) Q-learning versus fixed-strategy tit-for-tat
- (2) Q-learning versus Q-learning

Simulation set-up:

- Price set: $k = \{6, 12, 50\}$ possible prices
- $R = 1000$ runs of $T = 300(k+1)^2$ periods each
- Learning parameters: $\alpha = 0.5$, $\delta = 0.95$ and $\varepsilon_t = (1-\theta)^t$
- θ such that $\varepsilon_t = 0.5\%$ halfway and $\varepsilon_t = 0.0025\%$ at the end
- Initiate Q with discounted perpetuity static Nash (not necessary)

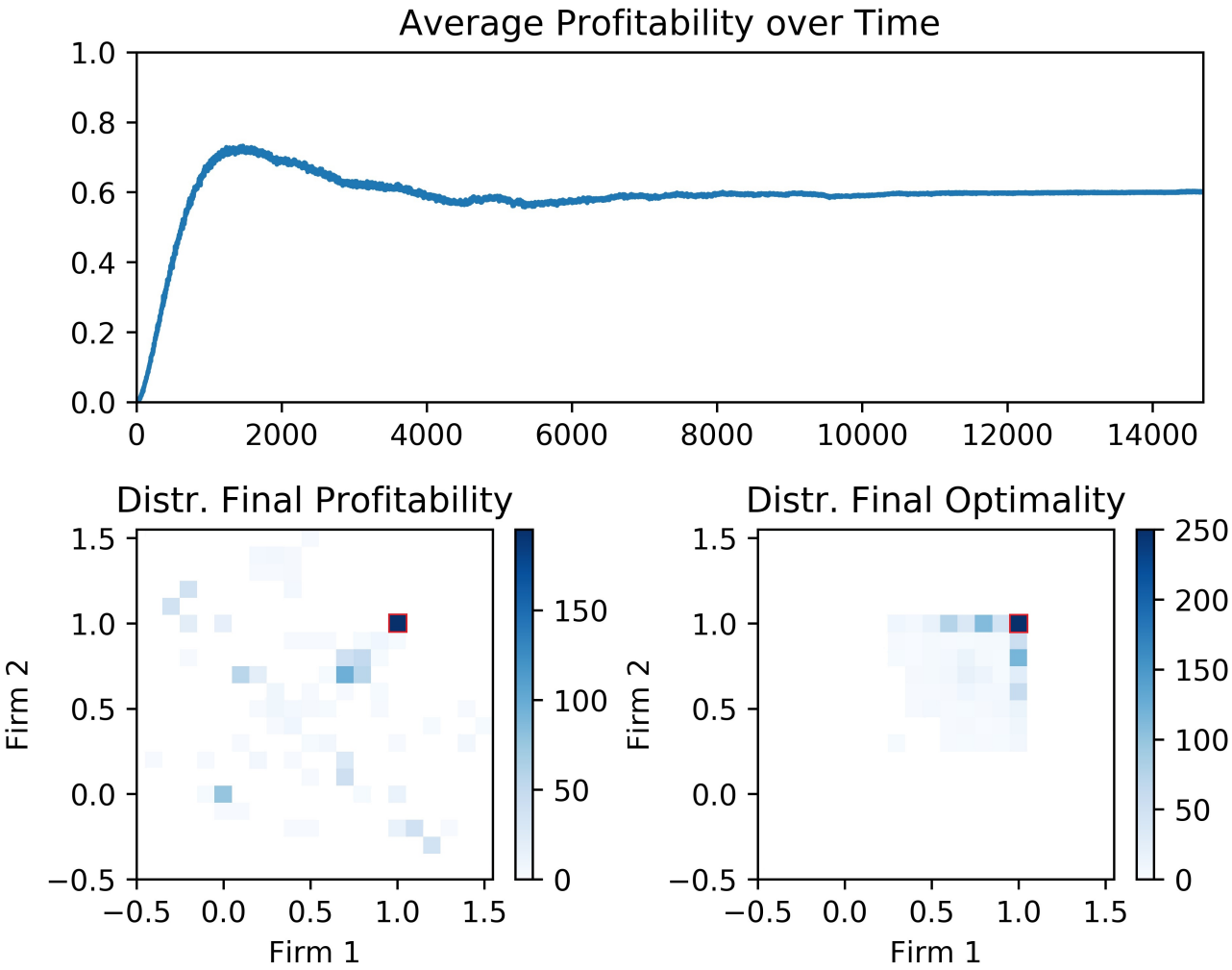
Simulation Results

(1) Q-learning versus fixed-strategy tit-for-tat, $k = 6$



Simulation Results

(2) Q-learning versus Q-learning, $k = 6$



Conclusions

-Profitability high

-194 with both $\Delta = 1$

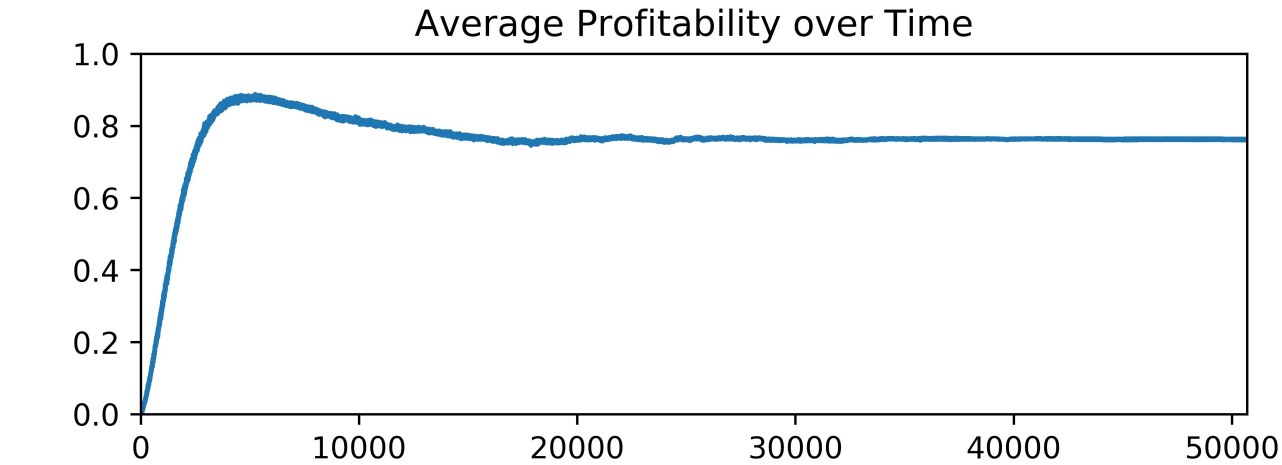
-Often optimal

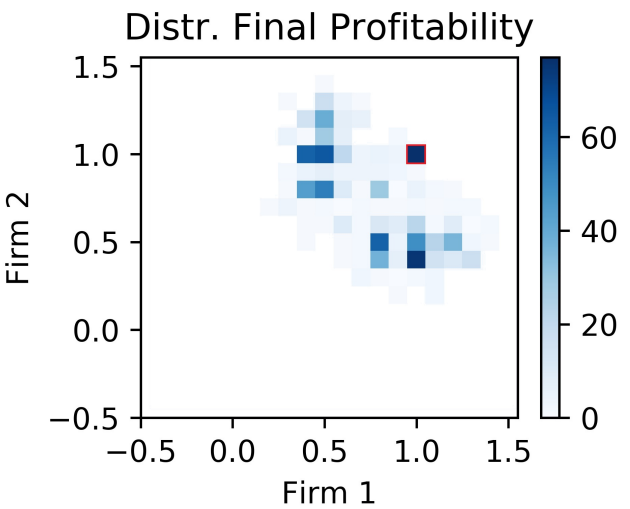
-164 with both $\Gamma = 1$

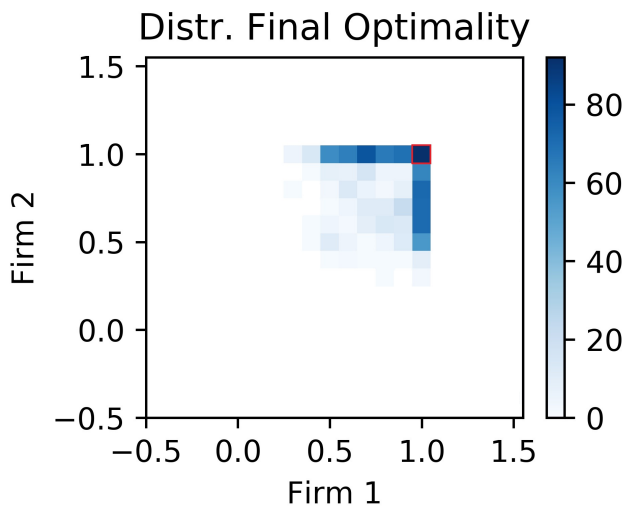
-63 with $\Delta = \Gamma = 1$

Simulation Results

(2) Q-learning versus Q-learning, $k = 12$







Conclusions

-More profitable

-34 with both $\Delta = 1$

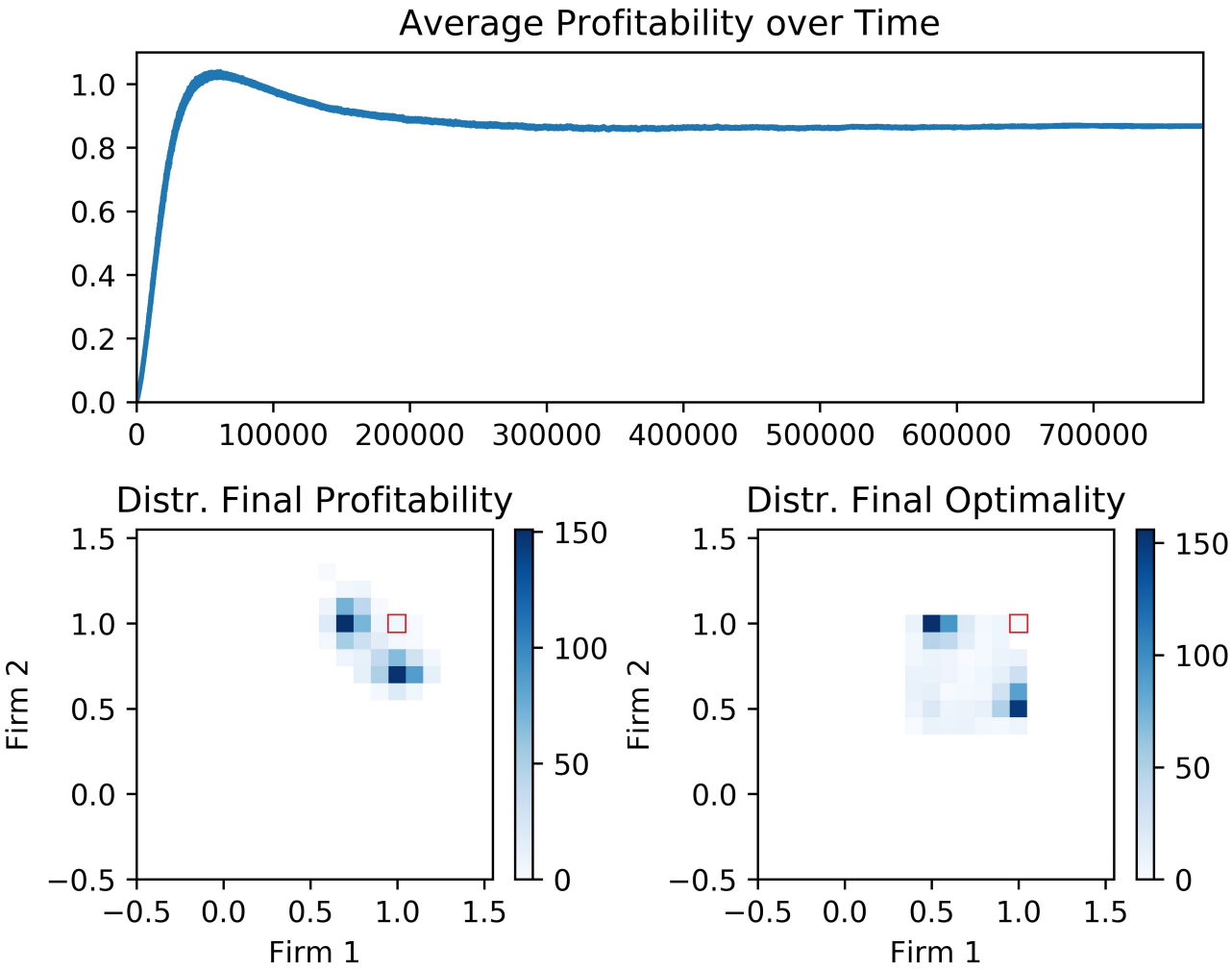
-Often optimal for only one firm

-27 with both $\Gamma = 1$

-6 with $\Delta = \Gamma = 1$

Simulation Results

(2) Q-learning versus Q-learning, $k = 50$



Conclusions

-Even more profitable

-Less optimal

-0 with $\Delta = \Gamma = 1$

-Price dynamics?

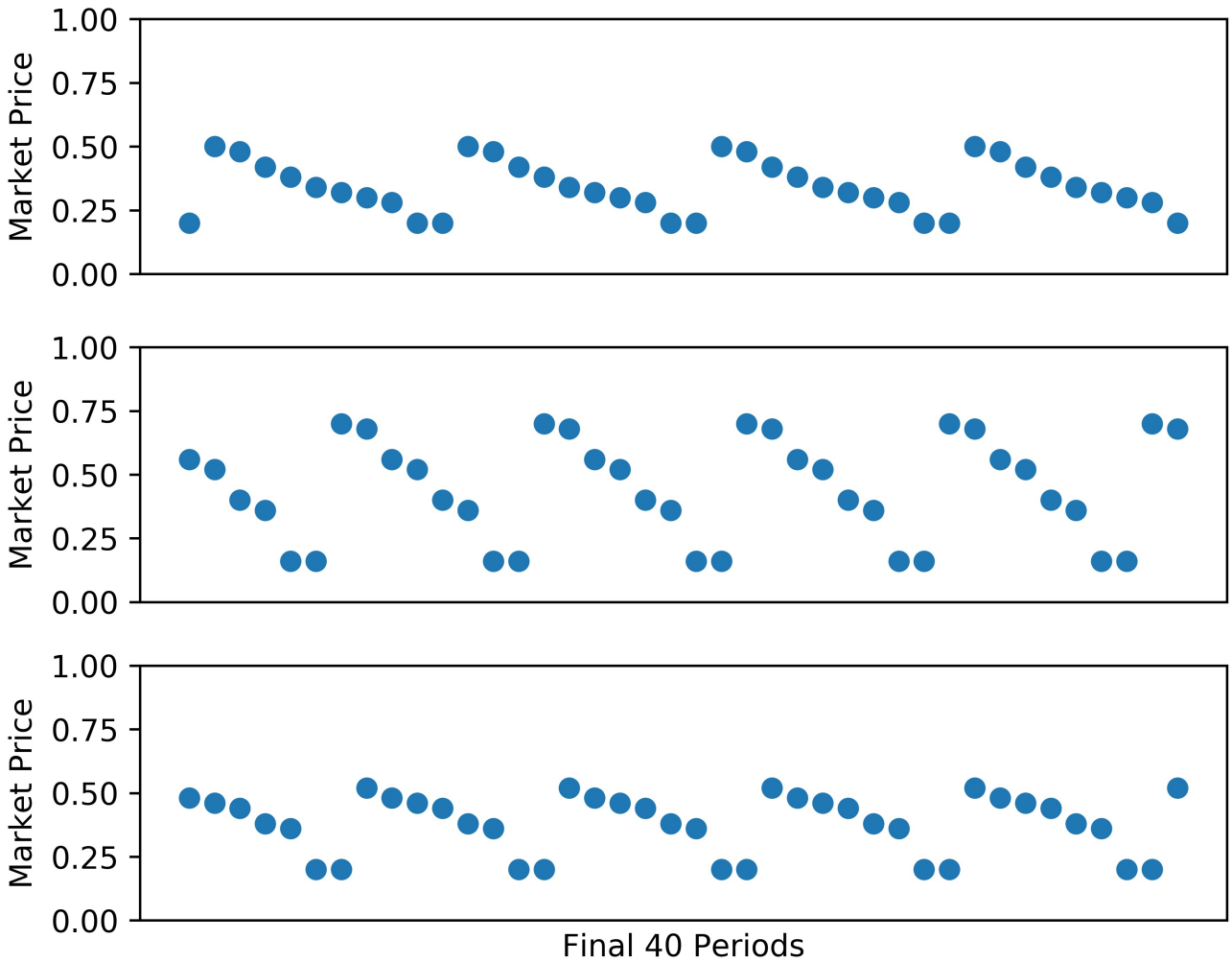
Simulation Results

	$k = 6$	$k = 12$	$k = 50$
Runs with a fixed price	508/1,000	111/1,000	11/1,000
Runs with monopoly fixed price	194/1,000	35/1,000	0/1,000
Runs without a fixed price	492/1,000	889/1,000	989/1,000
Periods with a price decrease	47%	63%	76%
Periods with a price increase	22%	17%	11%

Table 1: Price dynamics final 100 periods

- Adopts a fixed price or asymmetric price cycles
- More asymmetric price cyles if k is larger

Simulation Results



- Market price dynamics final 40 periods, 3 random runs, $k = 50$
- Jumps before reaching lower bound, to price above monopoly

Conclusions

Conclusions

- Autonomous algorithmic collusion in principle possible

Conclusions

- Autonomous algorithmic collusion in principle possible
- Sequential Q leads to higher prices, only programmed to max *own* profits

Conclusions

- Autonomous algorithmic collusion in principle possible
- Sequential Q leads to higher prices, only programmed to max *own* profits
- Outcomes resemble equilibrium behavior ...

Conclusions

- Autonomous algorithmic collusion in principle possible
- Sequential Q leads to higher prices, only programmed to max *own* profits
- Outcomes resemble equilibrium behavior ...
- ... but scope for more advanced algorithms
  1. to guarantee optimality
  2. to deal with less stylized environments

- Many exciting areas for future research!
  - Multi-Agent Reinforcement Learning $\Rightarrow$ see appendix
  - Deep Reinforcement Learning
  - Supervised Learning (function approximation)
  - (...)