Multiagent Learning and Equilibrium in Pricing Games

Katerina Papadaki

Joint work with:

Bernhard von Stengel, Galit Ashkenazi-Golan

Edward Plumb, Sahar Jahani

Department of Mathematics

London School of Economics

Overview

This is work in progress

- Aim of this study
- Pricing game
- Learning framework
- Learning environment
- Some initial results
- Conclusion and continuation

Aim the of study

- To explore larger games with machine learning
- We consider a duopoly game (2 firms) with demand inertia (price this period affects demand next period).
- We use this base game to develop a *learning framework* that we can apply to other games.

Pricing game

Duopoly with demand inertia: multistage pricing game.

- This was analysed theoretically by Selten (oligopoly version):

  R. Selten (1965), Game-theoretic analysis of an oligopolic model with buyers' interia. [German] Zeitsch. gesammte Staatswiss. 21, 301–304

  One of the first subgame perfect equilibriums computed (by Selten in 1965).

- Keser (1993) used this game in an experimental study where she run a tournament between game theorists.

  C. Keser (1993), Some results of experimental duopoly markets with demand inertia. Journal of Industrial Economics 41, 133–151

  1992 PhD thesis: Springer Lecture Notes Econ. Math. Systems 391

Duopoly game with demand inertia

The game is played between two producing firms with costs and .

The **demand potential** of 400 is split as between the two firms.

At each period firm

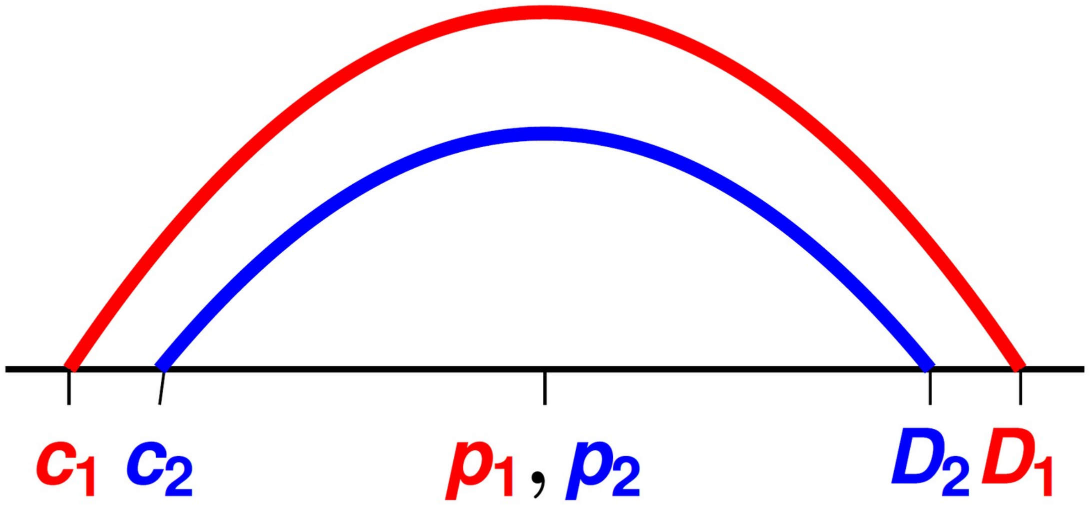
- chooses price
- sells units
- gets profit

Optimal **myopic price**:

Example: , ,

Then myopic prices are

And profits are and



Duopoly game with demand inertia

, ,

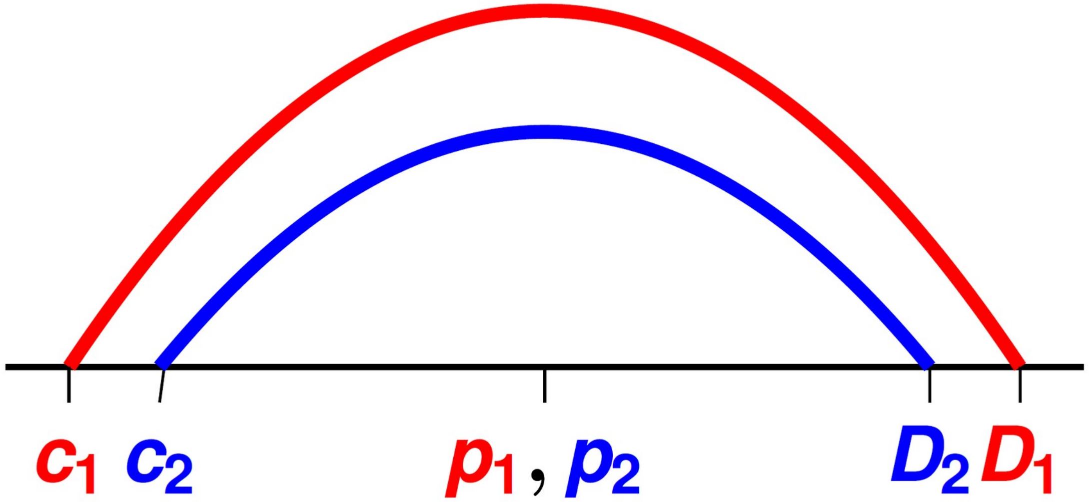
Units sold = , profit =

Optimal **myopic price**:

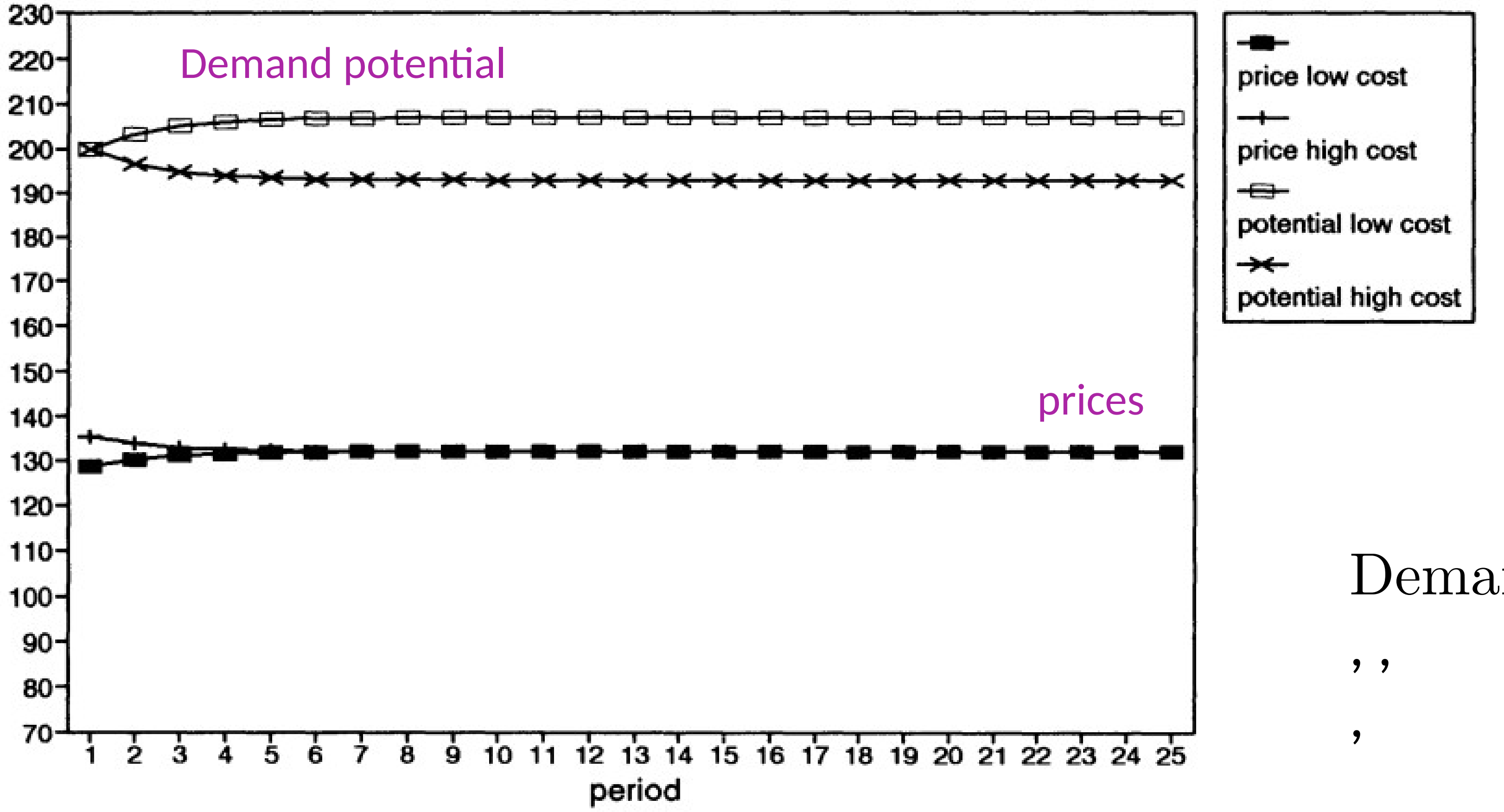
Played over 25 periods :

)/2

)/2



Myopic policy – cooperative solution



Myopic price:

Total profits:

156K

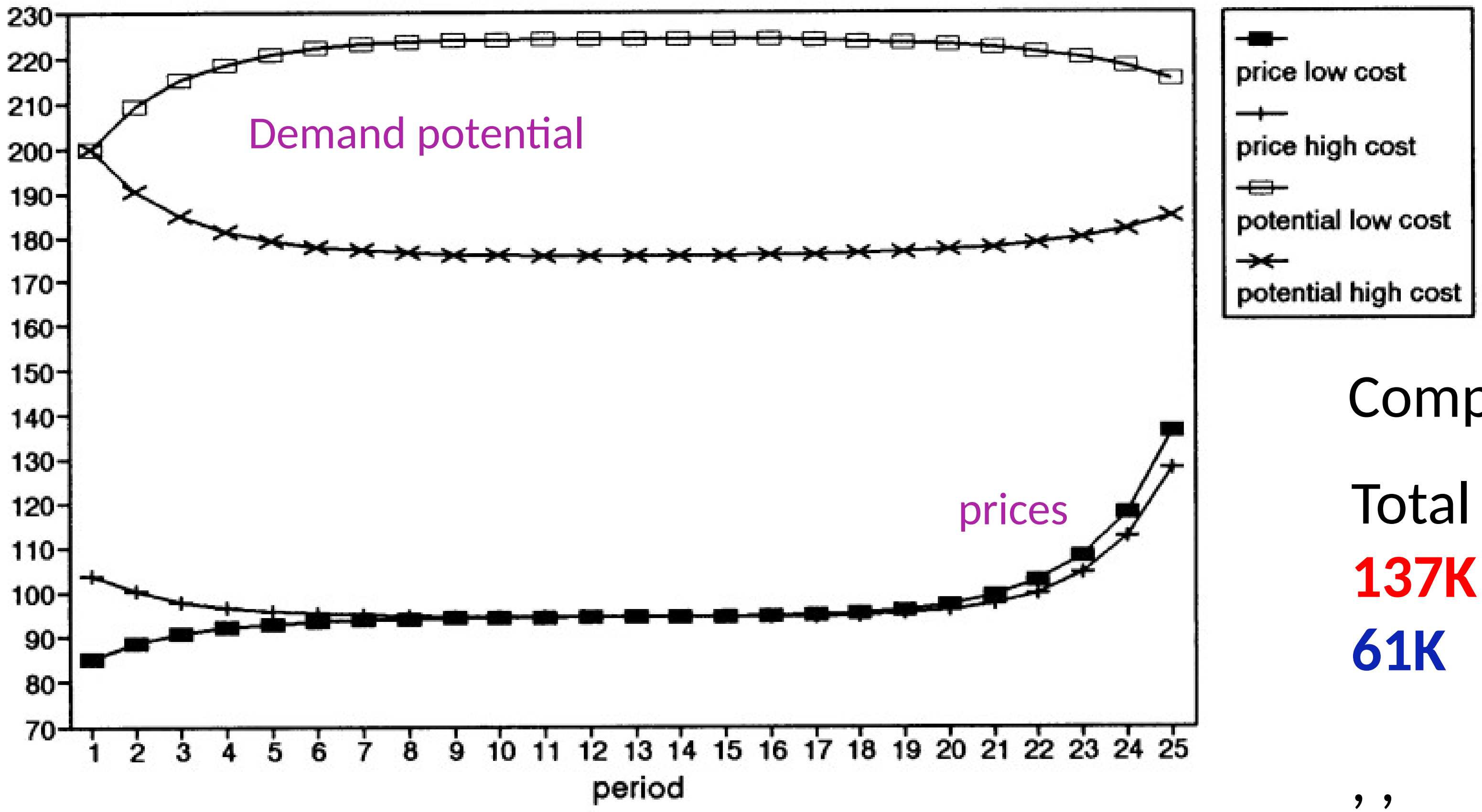
109K

Demands and prices converge to:

, ,

,

Subgame Perfect Equilibrium



Computed via backward Induction.

Total profits:
137K
61K

, ,

Keser’s tournament

Each participant submitted a strategy in the form of a *flow chart* both for low cost and high cost firms.

First round 45 participants submitted:

- strategies were played against each other
- cumulative payoffs ranked and sent to participants as feedback

Second round 34 participants

Evolutionary dynamics:

- Keser applied *replicator dynamics* to 34x34 matrix
- Eliminated most strategies
- Leaving 4x4 with positive probability.
- We look at this mixed equilibrium later.

	H_1	H_2	H_n
L_1		P_{12}^{H} P_{12}^{L}	
L_2			
L_n			

Learning framework

Base game: Duopoly pricing game with demand inertia played over 25 periods.

- Suppose we have agents, where each agent has one strategy for low cost and one for high cost firms.

Population game: An , bimatrix game (low cost firm vs high cost firm) between these strategies.

Agent has strategy for low cost firms and for high cost firms.

Population game

	H_1	H_2	H_n
L_1		P^H_{12} P^L_{12}	
L_2			
L_n			

Learning framework

Suppose we have already trained agents and added their strategies to the population game.

Agent: is a function that maps data from the current period (and possibly previous periods) to the current price.

- We want to train the next agent (against these agents).
- We compute a *mixed equilibrium* of the existing strategies of the population game.
- We train the next agent against this mixed equilibrium: *agent is trained by repeatedly meeting another random agent, drawn from the mixed equilibrium*

- Learning environment constant but random.
- If the newly trained agent produces payoffs equilibrium payoffs, we add the agent to the population game:
  - new entrant has payoffs against each existing strategy
  - defines a bimatrix game and computes a *new equilibrium* as next learning environment

Example of the learning framework

		0.05	0.03	0.58	0.34
	L \ H	*a*	*b*	*c*	*d*
0.02	***A***	$\boxed{100}$ / 152	86 / $\boxed{180}$	98 / 157	99 / 154
0.01	***B***	$\boxed{110}$ / 74	66 / 170	47 / $\boxed{178}$	75 / 130
0.67	***C***	102 / $\boxed{155}$	$\boxed{103}$ / 160	$\boxed{103}$ / 156	$\boxed{103}$ / 157
0.30	***D***	$\boxed{105}$ / 154	$\boxed{105}$ / 158	$\boxed{105}$ / 155	104 / $\boxed{159}$

equilibrium payoffs:

103

156

The new agent is trained

- as a low cost agent against (0.05,0.03,0.58,0.34) to produce:
- as a high cost agent against (0.02,0.01,0.67,0.30) to produce

- We would then test against **a, b, c, d, e**
- And test against **A, B, C, D, E**
- And this will add a **row** and a **column** to the bimatrix game.

Example of the learning framework

How do we decide if we are keeping new row ( ) or new column ( )?

		0.3	0.3	0.4	
		H_1	H_2	H_3	H_4
0.5	L_1		P^H_{12} P^L_{12}		1 / 1
0.4	L_2				2 / 1
0.1	L_3				3 / 2
	L_4	2 / 3	3 / 2	0 / 1	2 / 3

New agent trained against shown mixed equilibrium:

- New row ( ) and column ( ) added
- against equilibrium

1.5

- against equilibrium

1.9

1.6

- In both cases beats the equilibrium so we add both and to the population game.

1.1

Learning framework

Which equilibrium?

- Equilibriums are found by Lemke’s algorithm (mimics the Harsanyi–Selten tracing procedure)
- Finds an odd number of equilibria.
- Out of which are positive index equilibria (for dynamic stability).

Typically the algorithm finds equilibria with small support.

Learning framework - advantages

It is **modular** rather than a huge simulation:

- the **base game** (pricing game)
  - is complex (too complex?) as an interesting learning scenario
  - allows competition and cooperation
  - potentially has “hand-made” good strategies
  - can be replaced by another game

- the population game . . . uses game theory
  - provides via equilibria a “stable” learning environment
  - has typically mixed, non-unique equilibria
  - allows different equilibrium concepts (mixed, evolutionary)

$\Rightarrow$ can independently investigate different aspects

Learning environment

Agent: the agent we are training (assume low cost):

Adversary: the adversary we are training against (high cost):

, = demand potential of agent/adversary at beginning of period

, = price set of agent /adversary at period

, , ,

Learning environment

We model this as a Partially Observable Markov Decision Process (POMDP):

State: , ,)

Action:

Observation:

Transition: ,)

Immediate reward: = = profit from period

State value function:

State-action value function:

Learning environment

We use Q-learning:

For each episode *(episode is a 25 period pricing game)*

For

Agent is in state , and picks action as follows:

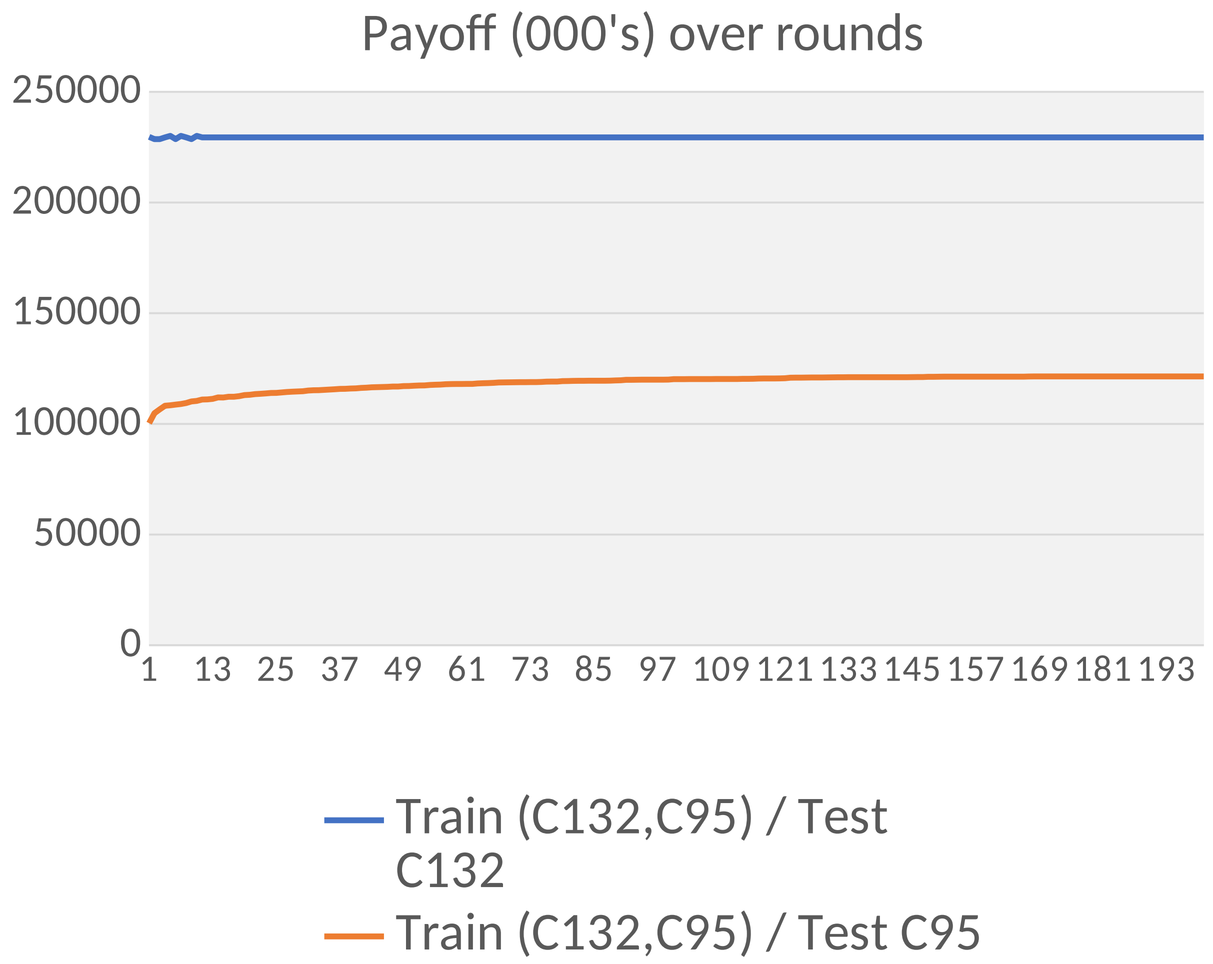
prob

prob 1-

Then receives payoff , observes , moves to state and updates as follows:

where is the learning rate.

Some initial Results

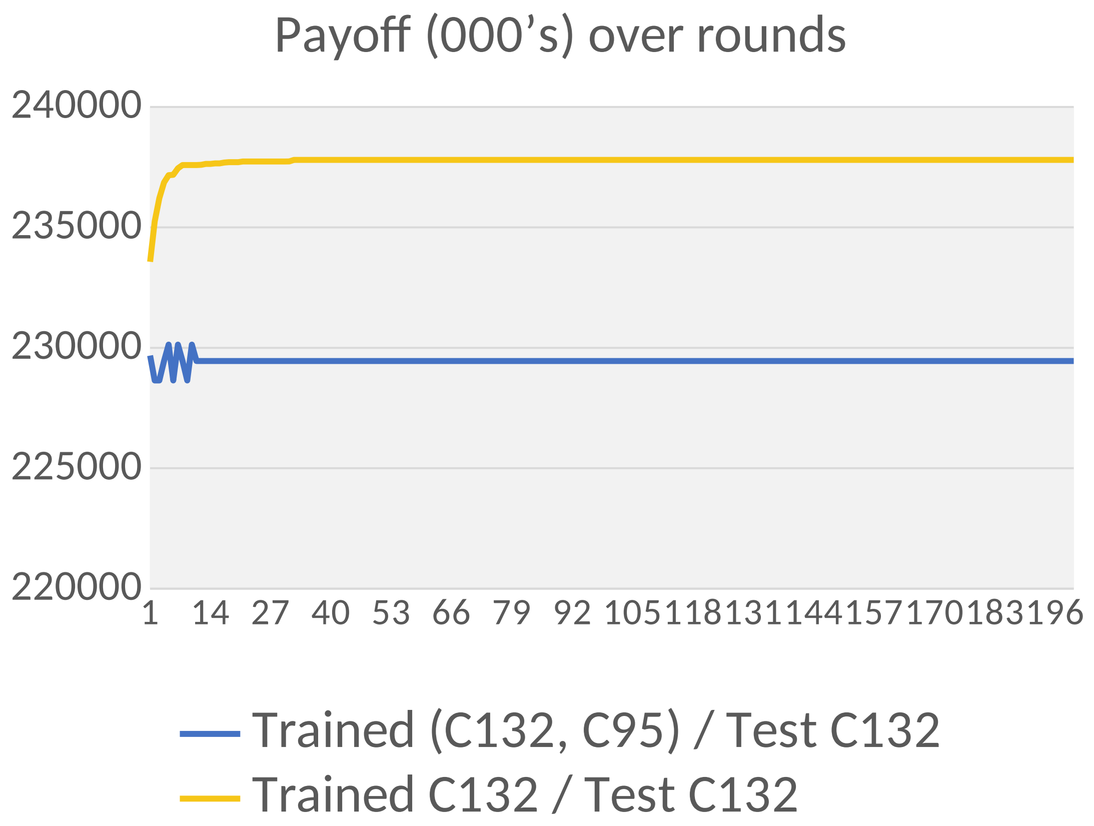


C132 = constant price 132
C95 = constant price 95

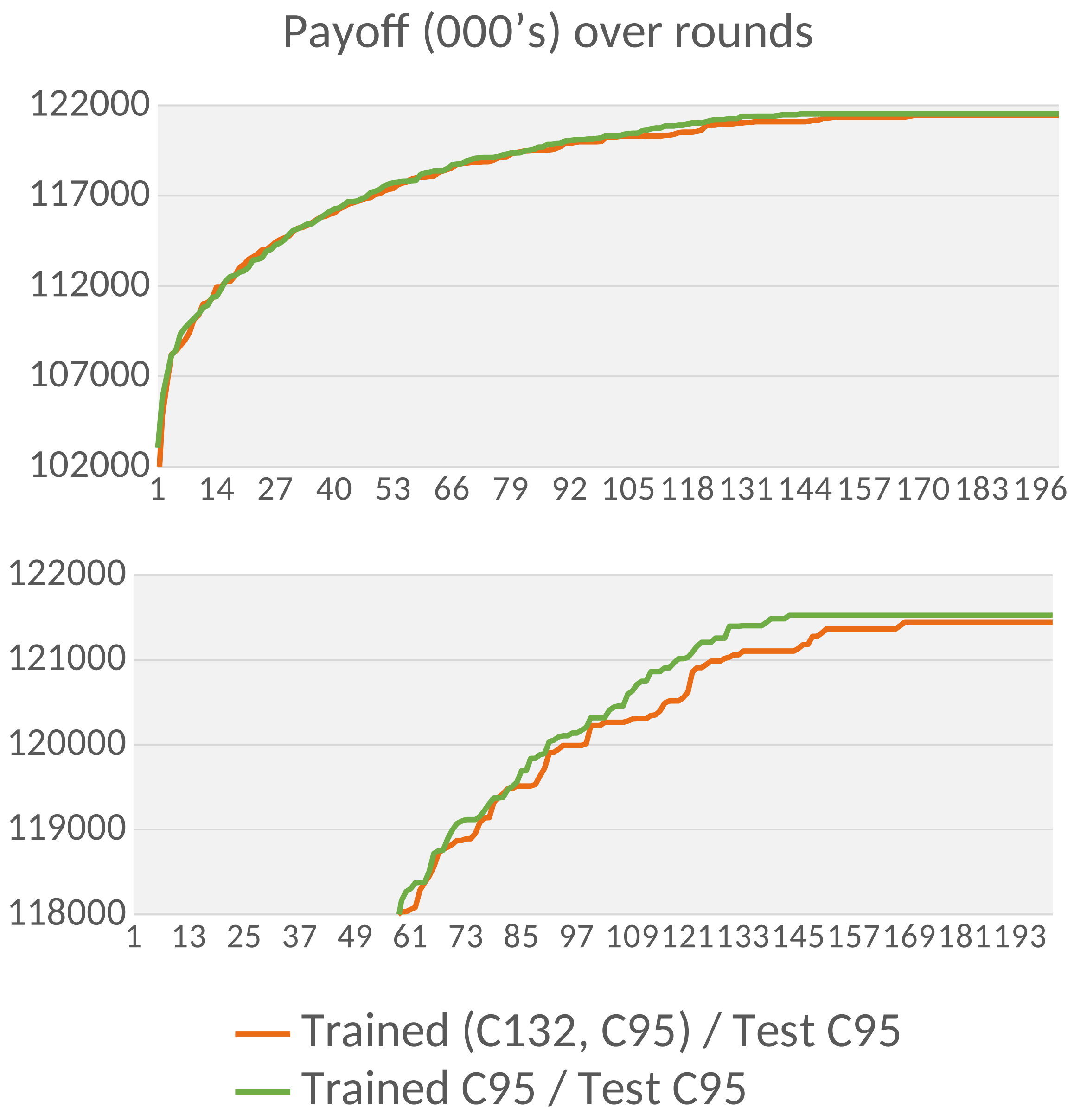
Training against:
(0.5,0.5) of (C132,C95)

While training, we play against C132 and C95

TC132 = train/play against C132
TC95 = train/play against C95

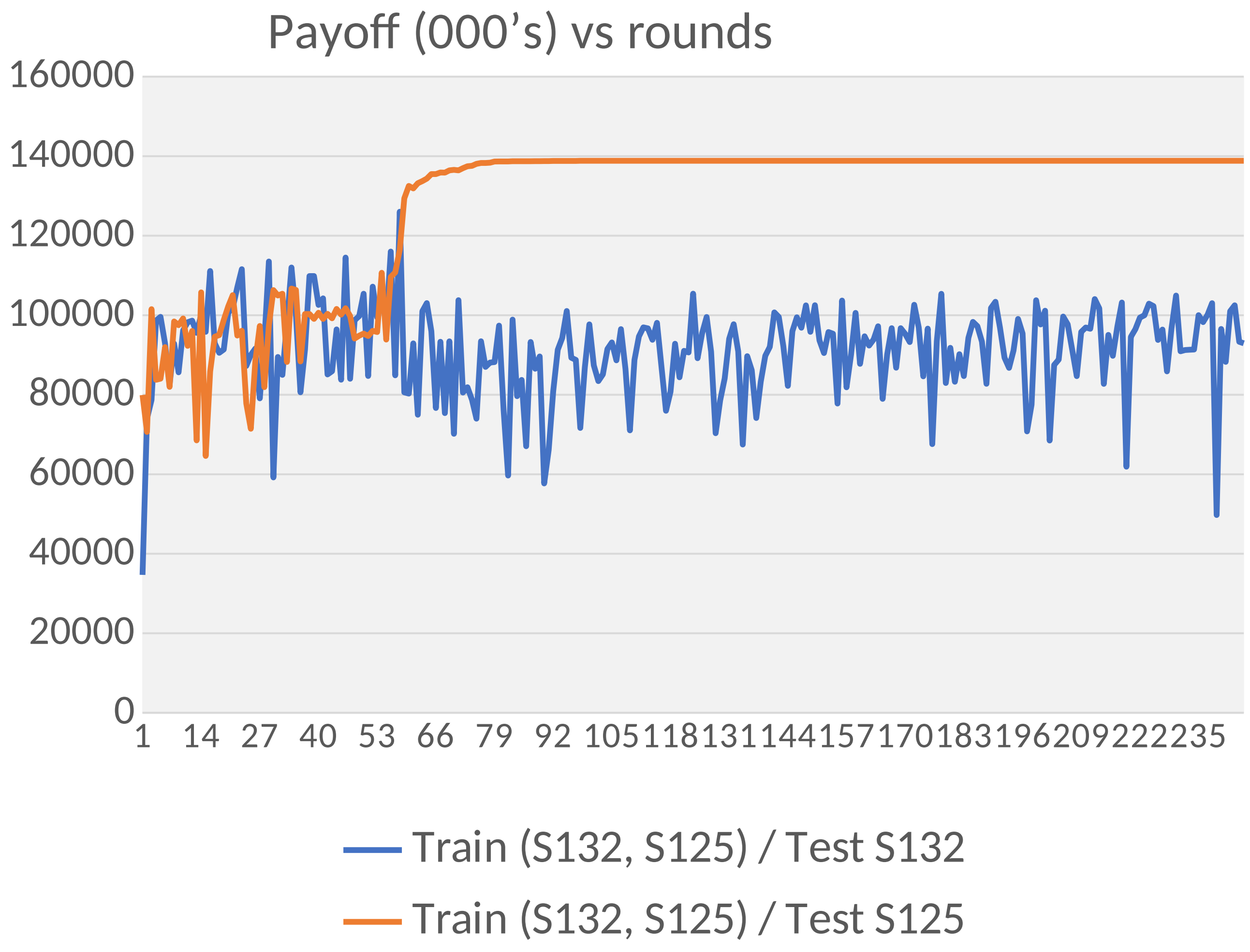


Payoff (000's) over rounds
240000
235000
230000
225000
220000
1 14 27 40 53 66 79 92 105 118 131 144 157 170 183 196
Trained (C132, C95) / Test C132
Trained C132 / Test C132



Payoff (000's) over rounds
122000
117000
112000
107000
102000
1 14 27 40 53 66 79 92 105 118 131 144 157 170 183 196
122000
121000
120000
119000
118000
1 13 25 37 49 61 73 85 97 109 121 133 145 157 169 181 193
Trained (C132, C95) / Test C95
Trained C95 / Test C95

Some initial Results

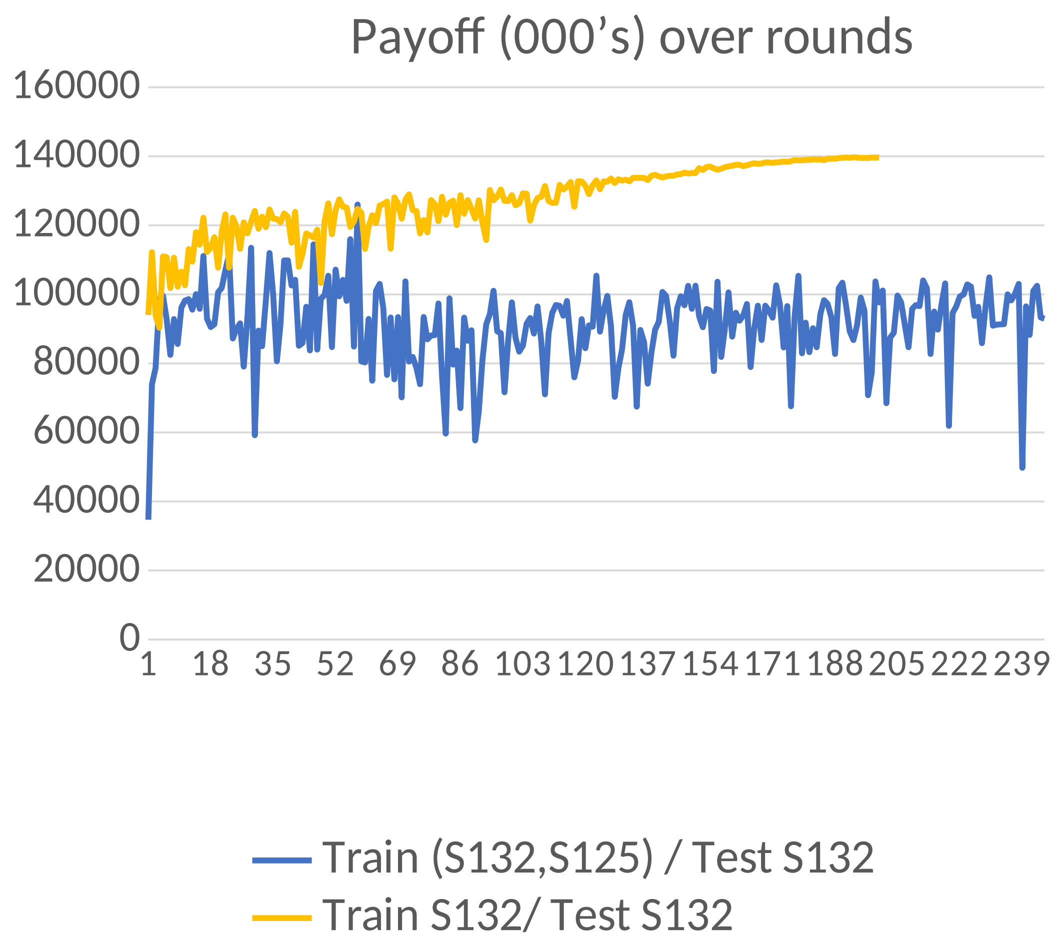


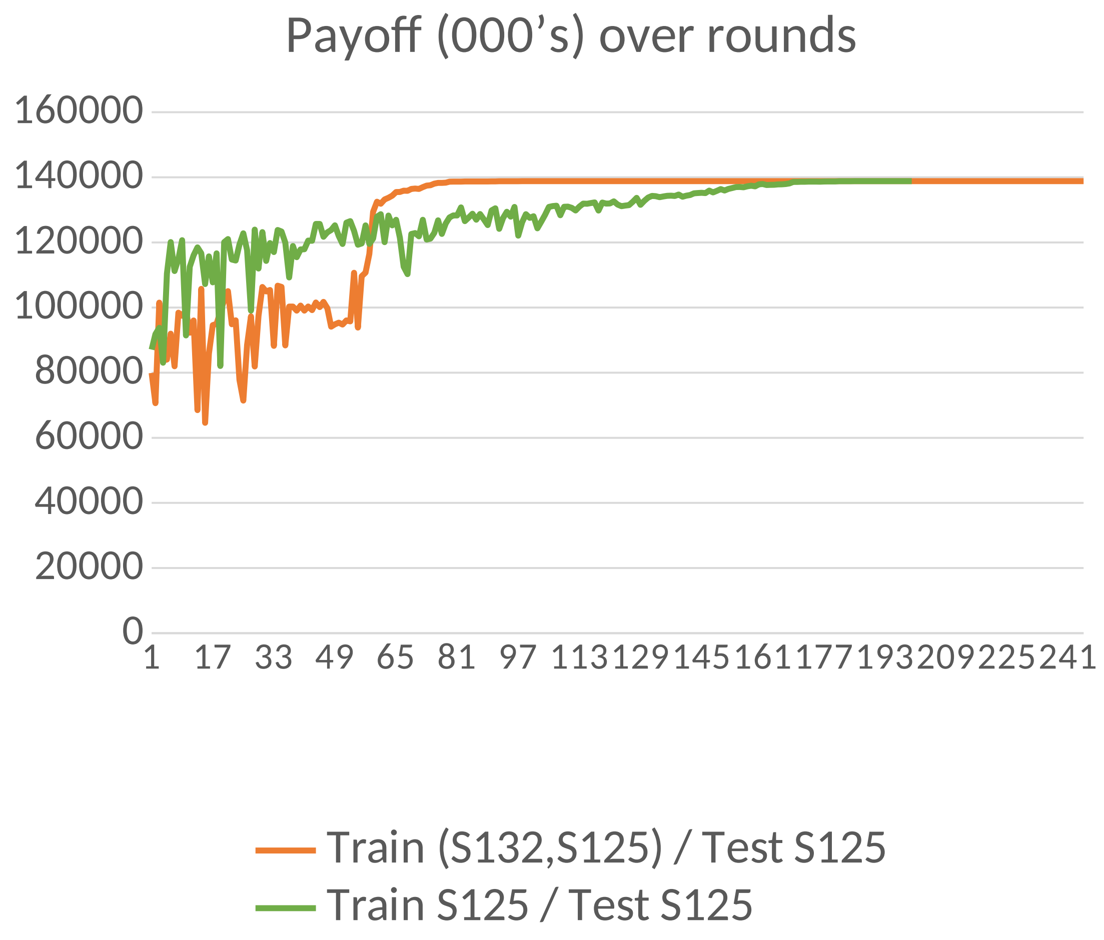
S132 = sophisticated strategy starts at 132
S125 = sophisticated strategy starts at 125

Training against:
(0.5,0.5) of (S132,S125)

While training, we play against S132 and S125

Some initial Results





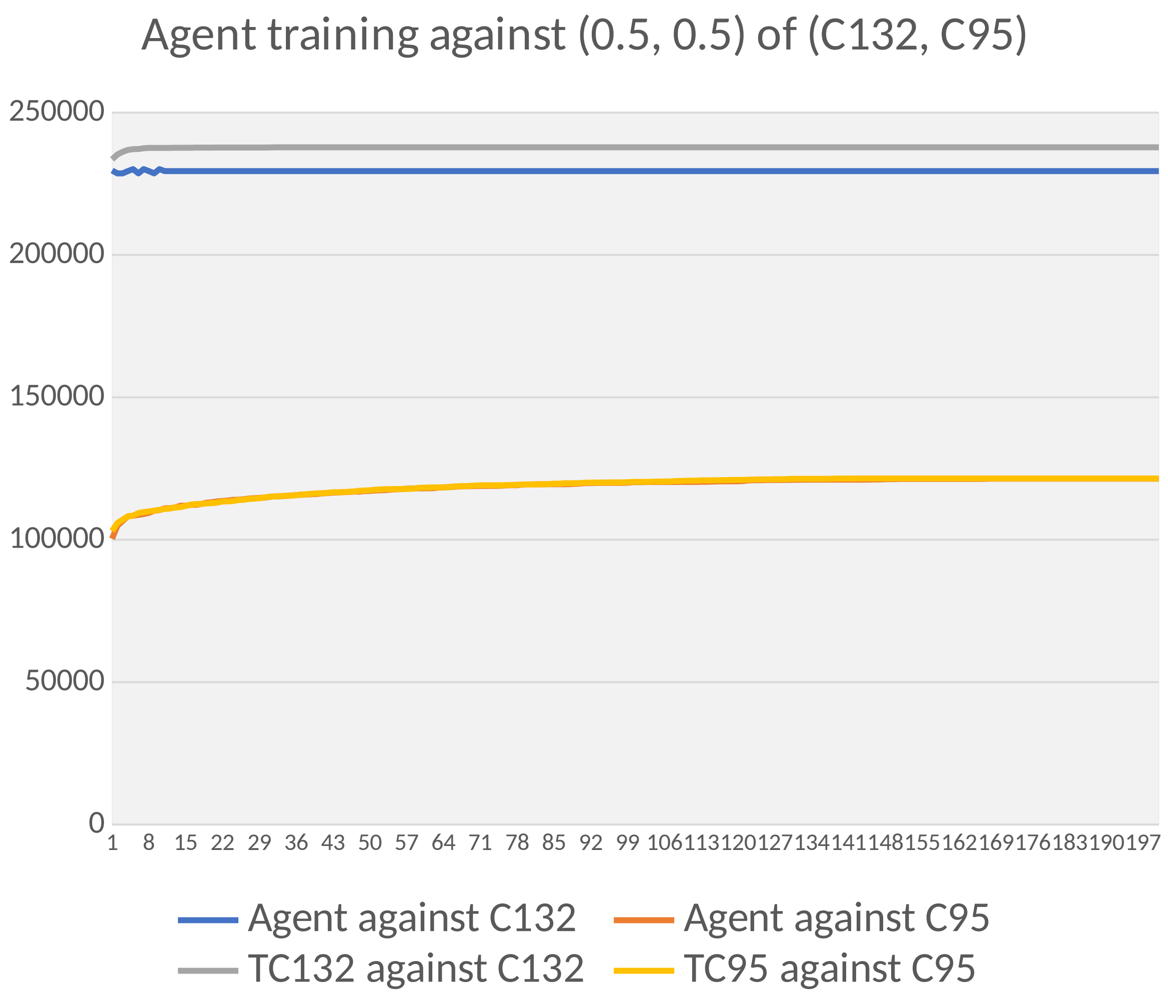
Conclusion and continuation

- Distinguishes between very different simple strategies.
- Complex strategies? Train longer?

- Include more memory in the state.
- Exploration probability depend on number of visits to a state.
- Q-table already too big.
- Q-table needs to be replaced by a neural network (deep Q-learning).
- Other RL methods such as policy gradient.

Thank you

Some initial Results



C132 = constant price 132
C95 = constant price 95

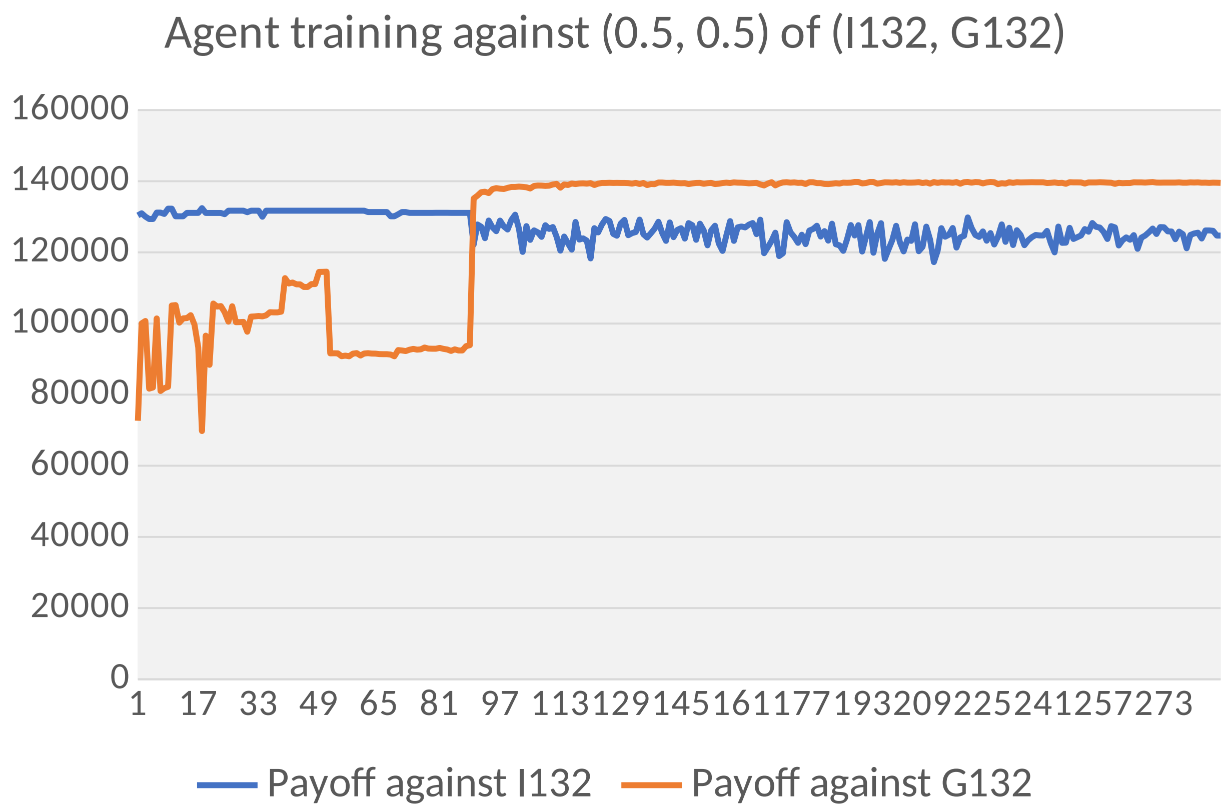
Training against:
(0.5,0.5) of (C132,C95)

While training, we play against C132 and C95

TC132 = train/play against C132
TC95 = train/play against C95

Some initial Results

Training agents using Q-learning:



Round = 500,000 episodes
I132 = starts at 132 and imitates opponents price
G132 = complicated strategy that starts at the price of 132

Training against:
(0.5,0.5) of (I132,G132)

While training, we play against I132 and G132

Policy gradient

lr = 0.0001

Aadversary's strategy	low, agent's payoff	low, adversary's payoff	high, agent's payoff	high, adversary's payoff
myopic	176443	50539	118022	95734
constant 132	276049	-25963	207197	-13449
constant 95	117703	52607	66850	98427
guess	118905	95285	86611	131345

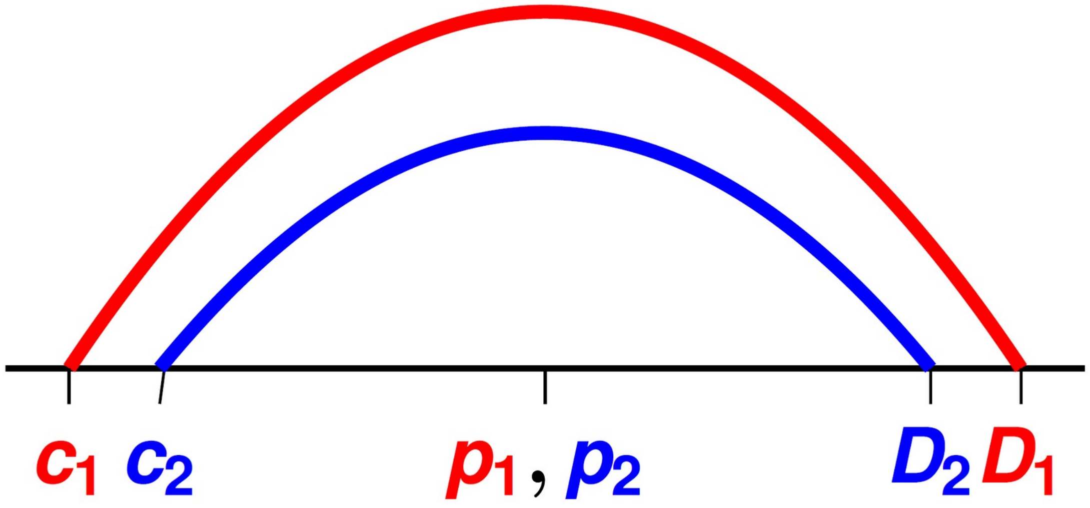
lr = 0.0005

Aadversary's strategy	low, agent's payoff	low, adversary's payoff	high, agent's payoff	high, adversary's payoff
myopic	176099	51548	118321	94330
constant 132	276049	-25963	186477	17177
constant 95	111855	54384	59063	110050
guess	132293	85868	88671	133895

low cost/ high cost	mixed agent	lr=0.00005 / myopic	lr=0.00005 / const 96	lr=0.00005 / guess
low	myopic/ const 95/ guess	173423	94059	116460
high	myopic/ const 95/ guess	115274	47784	76926

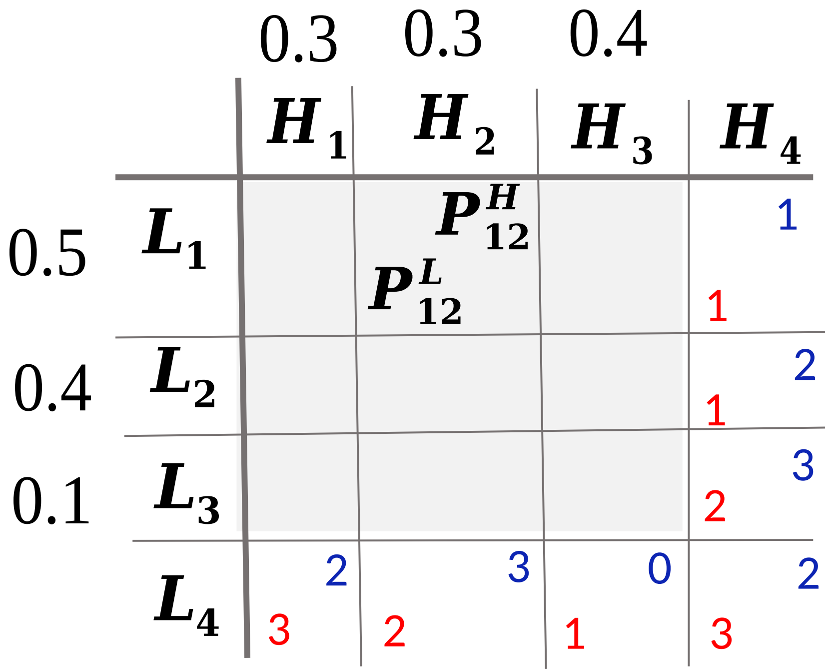
Example of a mixed equilibrium

L \ H	*a*	*b*	*c*	*d*
A	152, **100**	**180**, 86	157, 98	154, 99
B	74, **110**	170, 66	**178**, 47	130, 75
C	**155**, 102	160, **103**	156, **103**	157, **103**
D	154, **105**	158, **105**	155, **105**	**159**, 104



c_1
c_2
p_1, p_2
D_2
D_1

	H_1	H_2	H_n
L_1		P_{12}^{H} P_{12}^{L}	
L_2			
L_n			



New agent trained against shown mixed equilibrium:

- New row and column added
- against equilibrium

  1.5

  1.9
- against equilibrium

  1.6

  1.1
- In both cases beats the equilibrium so we add both and to the population game.