Multiagent Reinforcement Learning (MARL)

September 27, 2013 - ECML'13

Presenters

- Daan Bloembergen
- Daniel Hennes
- Michael Kaisers
- Peter Vrancx

Schedule

- Fundamentals of multi-agent reinforcement learning
  - **15:30 - 17:00**, Daan Bloembergen and Daniel Hennes
- Dynamics of learning in strategic interactions
  - **17:15 - 17:45**, Michael Kaisers
- Scaling multi-agent reinforcement learning
  - **17:45 - 18:45**, Peter Vrancx

Who are you?

We would like to get to know our audience!

Fundamentals of Multi-Agent Reinforcement Learning

Daan Bloembergen and Daniel Hennes

Outline (1)

Single Agent Reinforcement Learning

- Markov Decision Processes
  - Value Iteration
  - Policy Iteration
- Algorithms
  - Q-Learning
  - Learning Automata

Outline (2)

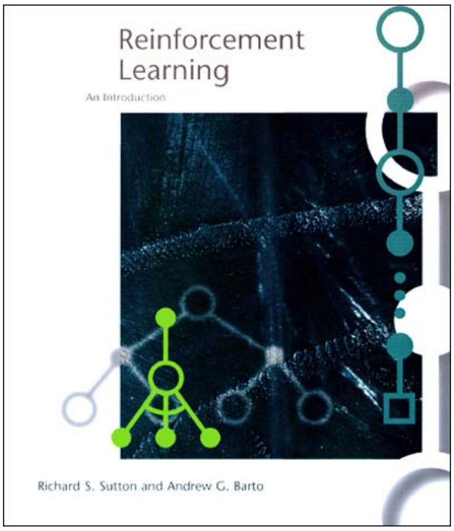
Multiagent Reinforcement Learning

- Game Theory
- Markov Games
  - Value Iteration
- Algorithms
  - Minimax-Q Learning
  - Nash-Q Learning
  - Other Equilibrium Learning Algorithms
  - Policy Hill-Climbing

Part I: Single Agent Reinforcement Learning

Richard S. Sutton and Andrew G. Barto
Reinforcement Learning: An Introduction
MIT Press, 1998

Available on-line for free!



Why reinforcement learning?

Based on ideas from psychology

- Edward Thorndike's **law of effect**
  - Satisfaction strengthens behavior, discomfort weakens it
- B.F. Skinner's **principle of reinforcement**
  - Skinner Box: train animals by providing (positive) feedback

Learning by interacting with the environment

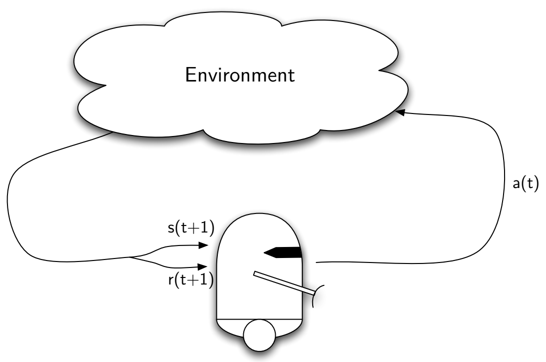
Why reinforcement learning?

Control theory

- Design a controller to minimize some measure of a dynamical systems's behavior
- Richard Bellman
  - Use system state and value functions (optimal return)
  - **Bellman equation**
- Dynamic programming
  - Solve optimal control problems by solving the Bellman equation

These two threads came together in the 1980s, producing the modern field of reinforcement learning

The RL setting



- Learning from interactions
- Learning what to do - **how to map situations to actions** - so as to maximize a numerical reward signal

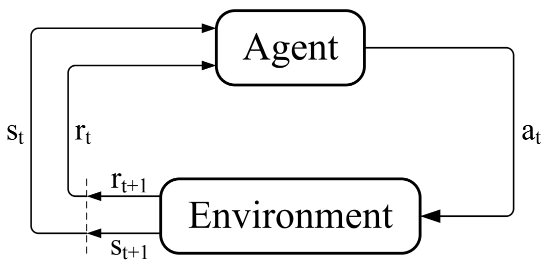
Key features of RL

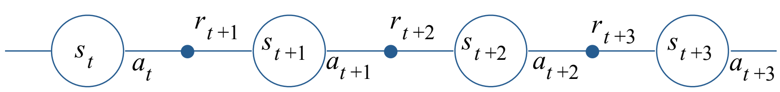
- Learner is **not** told which action to take
- Trial-and-error approach
- Possibility of **delayed reward**
  - Sacrifice short-term gains for greater long-term gains
- Need to balance **exploration** and **exploitation**
- In between **supervised** and **unsupervised** learning

The agent-environment interface

Agent interacts at discrete time steps $t = 0, 1, 2, \ldots$

- Observes state $s_t \in S$
- Selects action $a_t \in A(s_t)$
- Obtains immediate reward $r_{t+1} \in \Re$
- Observes resulting state s_{t+1}





Elements of RL

- Time steps need not refer to fixed intervals of real time
- **Actions** can be
  - low level (voltage to motors)
  - high level (go left, go right)
  - "mental" (shift focus of attention)
- **States** can be
  - low level "sensations" (temperature, (x, y) coordinates)
  - high level abstractions, symbolic
  - subjective, internal ("surprised", "lost")
- The **environment** is not necessarily known to the agent

Elements of RL

- **State transitions** are
  - changes to the internal state of the agent
  - changes in the environment as a result of the agent's action
  - can be nondeterministic
- **Rewards** are
  - goals, subgoals
  - duration
  - ...

Learning how to behave

- The agent's **policy** π at time t is
  - a mapping from states to action probabilities
  - $\pi_t(s, a) = P(a_t = a | s_t = s)$
- Reinforcement learning methods specify **how** the agent changes its policy as a result of experience
- Roughly, the agent's goal is to get **as much reward** as it can over the long run

The objective

Suppose the sequence of rewards after time t is

$$r_{t+1}, r_{t+2}, r_{t+3}, \ldots$$

- The goal is to maximize the **expected return** $E\{R_t\}$ for each time step t
- **Episodic tasks** naturally break into episodes, e.g., plays of a game, trips through a maze

$$R_t = r_{t+1} + r_{t+2} + \ldots + r_T$$

The objective

- **Continuing tasks** do not naturally break up into episodes
- Use **discounted return** instead of total reward

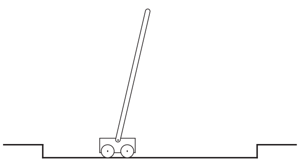
$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

where γ, $0 \leq \gamma \leq 1$ is the **discount factor** such that

shortsighted $0 \leftarrow \gamma \rightarrow 1$ farsighted

Example: pole balancing

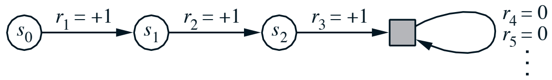
- As an **episodic** task where each episode ends upon failure
  - reward = +1 for each step before failure
  - return = number of steps before failure



- As a **continuing** task with discounted return
  - reward = -1 upon failure
  - return = $-\gamma^k$, for k steps before failure
- In both cases, return is maximized by avoiding failure for as long as possible

A unified notation

- Think of each episode as ending in an **absorbing state** that always produces a reward of zero



- Now we can cover both episodic and continuing tasks by writing

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

Markov decision processes

- It is often useful to a assume that all relevant information is present in the current state: **Markov property**

$$P(s_{t+1}, r_{t+1} | s_t, a_t) = P(s_{t+1}, r_{t+1} | s_t, a_t, r_t, s_{t-1}, a_{t-1}, \ldots, r_1, s_0, a_0)$$

- If a reinforcement learning task has the Markov property, it is basically a **Markov Decision Process (MDP)**
- Assuming finite state and action spaces, it is a finite MDP

Markov decision processes

An MDP is defined by

- **State and action sets**
- One-step dynamics defined by state **transition probabilities**

$$\mathcal{P}_{ss'}^{a} = P(s_{t+1} = s' | s_t = s, a_t = a)$$

- **Reward probabilties**

$$\mathcal{R}_{ss'}^{a} = E(r_{t+1} | s_t = s, a_t = a, s_{t+1} = s')$$

Value functions

- When following a fixed policy π we can define the **value** of a state s under that policy as

$$V^{\pi}(s) = E_{\pi}(R_t | s_t = s) = E_{\pi}(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s)$$

- Similarly we can define the value of taking action a in state s as

$$Q^{\pi}(s, a) = E_{\pi}(R_t | s_t = s, a_t = a)$$

Value functions

- The value function has a particular recursive relationship, defined by the **Bellman equation**

$$V^{\pi}(s) = \sum_{a} \pi(s,a) \sum_{s'} \mathcal{P}^{a}_{ss'} [\mathcal{R}^{a}_{ss'} + \gamma V^{\pi}(s')]$$

- The equation expresses the recursive relation between the value of a state and its successor states, and averages over all possibilities, weighting each by its probability of occurring

Optimal policy for an MDP

- We want to find the policy that maximizes long term reward, which equates to finding the optimal value function

$$V^{*}(s) = \max_{\pi} V^{\pi}(s) \qquad \forall s \in S$$

$$Q^{*}(s,a) = \max_{\pi} Q^{\pi}(s,a) \qquad \forall s \in S, a \in A(s)$$

- Expressed recursively, this is the **Bellman optimality equation**

$$\begin{aligned} V^{*}(s) &= \max_{a \in A(s)} Q^{\pi *}(s,a) \\ &= \max_{a \in A(s)} \sum_{s'} \mathcal{P}^{a}_{ss'} [\mathcal{R}^{a}_{ss'} + \gamma V^{*}(s')] \end{aligned}$$

Solving the Bellman equation

- We can find the **optimal policy** by solving the Bellman equation
  - Dynamic Programming
- Two approaches:
  - Iteratively improve the value function: **value iteration**
  - Iteratively evaluate and improve the policy: **policy iteration**
- Both approaches are proven to converge to the optimal value function

Value iteration

Initialize V arbitrarily, e.g.,$V(s) = 0$, for all $s \in \mathcal{S}^+$

Repeat
    $\Delta \leftarrow 0$
    For each $s \in \mathcal{S}$:
        $v \leftarrow V(s)$
        $V(s) \leftarrow \max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)

Output a deterministic policy, π, such that
    $\pi(s) = \arg\max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$

Policy iteration

- Often the optimal policy has been reached long before the value function has converged
- Policy iteration calculates a new policy **based on the current value function**, and then calculates a new value function based on this policy
- This process often converges faster to the optimal policy

Policy iteration

1. Initialization
   $V(s) \in \Re$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Repeat
   $\Delta \leftarrow 0$
   For each $s \in \mathcal{S}$:
   $v \leftarrow V(s)$
   $V(s) \leftarrow \sum_{s'} \mathcal{P}_{ss'}^{\pi(s)} \left[\mathcal{R}_{ss'}^{\pi(s)} + \gamma V(s') \right]$
   $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   $b \leftarrow \pi(s)$
   $\pi(s) \leftarrow \arg\max_a \sum_{s'} \mathcal{P}_{ss'}^{a} \left[\mathcal{R}_{ss'}^{a} + \gamma V(s') \right]$
   If $b \neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop; else go to 2

Learning an optimal policy online

- Both previous approaches require to know the dynamics of the environment
- Often this information is not available
- Using **temporal difference (TD)** methods is one way of overcoming this problem
  - Learn directly from raw experience
  - No model of the environment required (model-free)
  - E.g.: **Q-learning**
- Update predicted state values based on new observations of immediate rewards and successor states

Q-learning

- Q-learning updates state-action values based on the immediate reward and the optimal expected return

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right]$$

- Directly learns the optimal value function independent of the policy being followed
  - In contrast to on-policy learners, e.g. **SARSA**
- Proven to converge to the optimal policy given "sufficient" updates for each state-action pair, and decreasing learning rate α [Watkins92]

Q-learning

Initialize $Q(s, a)$ arbitrarily
Repeat (for each episode):
  Initialize s
  Repeat (for each step of episode):
    Choose a from s using policy derived from Q (e.g., ε-greedy)
    Take action a, observe r, s'
    $Q(s, a) \leftarrow Q(s, a) + \alpha\big[r + \gamma \max_{a'} Q(s', a') - Q(s, a)\big]$
    $s \leftarrow s'$;
  until s is terminal

Action selection

- How to select an action based on the values of the states or state-action pairs?
- Success of RL depends on a **trade-off**
  - Exploration
  - Exploitation
- **Exploration** is needed to prevent getting stuck in local optima
- To ensure convergence you need to **exploit**

Action selection

Two common choices

- **ϵ-greedy**
  - Choose the best action with probability $1-\epsilon$
  - Choose a random action with probability ϵ
- **Boltzmann exploration** (softmax) uses a temperature parameter τ to balance exploration and exploitation

$$\pi_t(s, a) = \frac{e^{Q_t(s,a)/\tau}}{\sum_{a' \in A} e^{Q_t(s,a')/\tau}}$$

pure exploitation $\quad 0 \leftarrow \tau \rightarrow \infty \quad$ pure exploration

Learning automata

- **Learning automata** [Narendra74] directly modify their policy based on the observed reward (policy iteration)
- Finite action-set learning automata learn a policy over a finite set of actions

$$\pi'(a) = \pi(a) + \begin{cases} \alpha r(1-\pi(a)) - \beta(1-r)\pi(a) & \text{if } a = a_t \\ -\alpha r \pi(a) + \beta(1-r)[(k-1)^{-1} - \pi(a)] & \text{if } a \neq a_t \end{cases}$$

  where $k = |A|$, and α and β are reward and penalty parameters respectively, and $r \in [0, 1]$
- **Cross learning** is a special case where $\alpha = 1$ and $\beta = 0$

Networks of learning automata

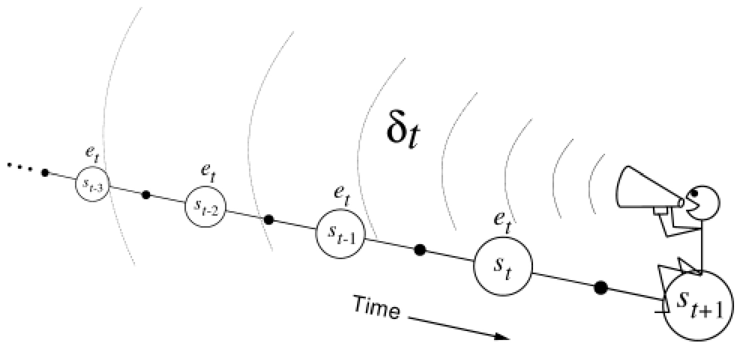
- A single learning automaton ignores any state information
- In a **network of learning automata** [Wheeler86] control is passed on from one automaton to another
  - One automaton $\mathcal{A}$ is active for each state
  - The immediate reward r is replaced by the average cumulative reward $\bar{r}$ since the last visit to that state

$$\bar{r}_t(s) = \frac{\Delta r}{\Delta t} = \frac{\sum_{i=l(s)}^{t-1} r_i}{t - l(s)}$$

where $l(s)$ indicates in which time step state s was last visited

Extensions

- Multi-step TD: **eligibility traces**
  - Instead of observing one immediate reward, use n consecutive rewards for the value update
  - Intuition: your current choice of action may have implications for the future
  - State-action pairs are eligible for future rewards, with more recent states getting more credit



Extensions

- **Reward shaping**
  - Incorporate domain knowledge to provide additional rewards during an episode
  - Guide the agent to learn faster
  - (Optimal) policies preserved given a potential-based shaping function [Ng99]
- **Function approximation**
  - So far we have used a tabular notation for value functions
  - For large state and actions spaces this approach becomes intractable
  - Function approximators can be used to generalize over large or even continuous state and action spaces

Questions so far?

Part II:
Multiagent Reinforcement Learning

Preliminaries: Fundamentals of Game Theory

Game theory

- Models **strategic interactions** as games
- In **normal form games**, all players simultaneously select an action, and their joint action determines their individual payoff
  - One-shot interaction
  - Can be represented as an n-dimensional payoff matrix, for n players
- A player's **strategy** is defined as a probability distribution over his possible actions

Example: Prisoner's Dilemma

- Two prisoners (A and B) commit a crime together
- They are questioned separately and can choose to confess or deny
  - If both confess, both prisoners will serve 3 years in jail
  - If both deny, both serve only 1 year for minor charges
  - If only one confesses, he goes free, while the other serves 5 years

	C	D
C	-3, -3	-0, -5
D	-5, -0	-1, -1

Example: Prisoner's Dilemma

- What should they do?
- If both deny, their total penalty is lowest
  - But is this individually rational?
- Purely selfish: regardless of what the other player does, confess is the optimal choice
  - If the other confesses, 3 instead of 5 years
  - If the other denies, free instead of 1 year

	C	D
C	-3, -3	-0, -5
D	-5, -0	-1, -1

Solution concepts

- **Nash equilibrium**
  - Individually rational
  - No player can improve by unilaterally changing his strategy
  - Mutual confession is the only Nash equilibrium of this game
- Jointly the players could do better
  - **Pareto optimum**: there is no other solution for which all players do at least as well and at least one player is strictly better off
  - Mutual denial Pareto dominates the Nash equilibrium in this game

	C	D
C	-3, -3	-0, -5
D	-5, -0	-1, -1

Types of games

- **Competitive** or zero-sum
  - Players have opposing preferences
  - E.g. Matching Pennies
- **Symmetric games**
  - Players are identical
  - E.g. Prioner's Dilemma
- **Asymmetric games**
  - Players are unique
  - E.g. Battle of the Sexes

Matching Pennies

	H	T
H	+1, -1	-1, +1
T	-1, +1	+1, -1

Prisoner's Dilemma

	C	D
C	-3, -3	-0, -5
D	-5, -0	-1, -1

Battle of the Sexes

	B	S
B	2, 1	0, 0
S	0, 0	1, 2

Part II: Multiagent Reinforcement Learning

MARL: Motivation

- MAS offer a solution paradigm that can cope with complex problems
- Technological challenges require decentralised solutions
  - Multiple autonomous vehicles for exploration, surveillance or rescue missions
  - Distributed sensing
  - Traffic control (data, urban or air traffic)
- Key advantages: Fault tolerance and load balancing
- But: highly dynamic and nondeterministic environments!
- Need for adaptation on an individual level
- Learning is crucial!

MARL: From single to multiagent learning

- Inherently more challenging
- Agents interact with the environment and each other
- Learning is simultaneous
- Changes in strategy of one agent might affect strategy of other agents
- Questions:
  - One vs. many learning agents?
  - Convergence?
  - Objective: maximise common reward or individual reward?
  - Credit assignment?

Independent reinforcement learners

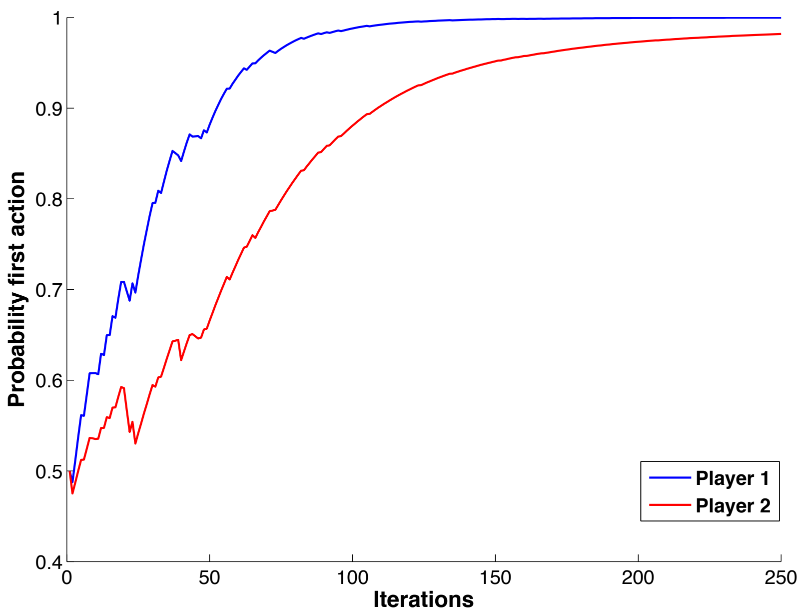
- Naive extension to multi agent setting
- Independent learners mutually ignore each other
- Implicitly perceive interaction with other agents as noise in a stochastic environment

Learning in matrix games

- Two Q-learners interact in Battle of the Sexes
  - $\alpha = 0.01$
  - Boltzmann exploration with $\tau = 0.2$
- They only observe their immediate reward
- Policy is gradually improved

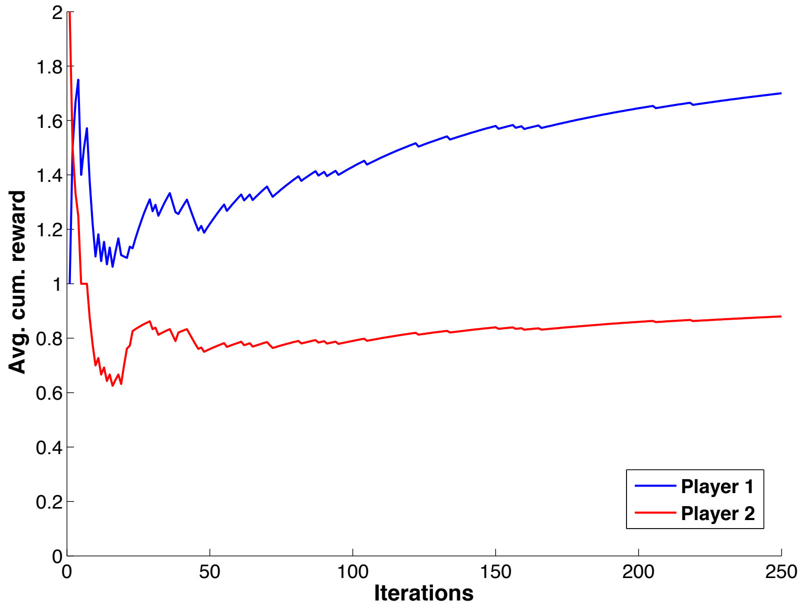
	B	S
B	2, 1	0, 0
S	0, 0	1, 2

Learning in matrix games



	B	S
B	2, 1	0, 0
S	0, 0	1, 2

Learning in matrix games



	B	S
B	2, 1	0, 0
S	0, 0	1, 2

Markov games

n-player game: $\langle n, S, A^1, \ldots, A^n, \mathcal{R}^1, \ldots, \mathcal{R}^n, \mathcal{P} \rangle$

- S : set of states
- A^i : action set for player i
- $\mathcal{R}^i$: reward/payoff for player i
- $\mathcal{P}$: transition function

The payoff function $\mathcal{R}^i : S \times A^1 \times \cdots \times A^n \mapsto \mathbb{R}$ maps the joint action $a = \langle a^1 \ldots a^n \rangle$ to an immediate payoff value for player i.

The transition function $\mathcal{P} : S \times A^1 \times \cdots \times A^n \mapsto \triangle(S)$ determines the probabilistic state change to the next state s_{t+1}.

Value iteration in Markov games

Single agent MDP:

$$V^*(s) = \max_{a \in A(s)} Q^{\pi*}(s, a)$$
$$= \max_{a \in A(s)} \sum_{s'} \mathcal{P}^a_{ss'} [\mathcal{R}^a_{ss'} + \gamma V^\pi(s')]$$

2-player zero-sum stochastic game:

$$Q^*(s, \langle a^1, a^2 \rangle) = \mathcal{R}(s, \langle a^1, a^2 \rangle) + \gamma \sum_{s' \in S} \mathcal{P}_{s'}(s, \langle a^1, a^2 \rangle) V^*(s')$$

$$V^*(s) = \max_{\pi \in \triangle(A^1)} \min_{a^2 \in A^2} \sum_{a^1 \in A^1} \pi_{a^1} Q^*(s, \langle a^1, a^2 \rangle)$$

Minimax-Q

- Value iteration requires knowledge of the reward and transition functions
- Minimax-Q [Littman94]: learning algorithm for zero-sum games
- Payoffs balance out, each agent only needs to observe its own payoff
- Q is a function of the joint action:

$$Q(s, \langle a^1, a^2 \rangle) = \mathcal{R}(s, \langle a^1, a^2 \rangle) + \gamma \sum_{s' \in S} \mathcal{P}_{s'}(s, \langle a^1, a^2 \rangle) V(s')$$

- A joint action learner (JAL) is an agent that learns Q-values for joint actions as opposed to individual actions.

Minimax-Q (2)

Update rule for agent 1 with reward function $\mathcal{R}_t$ at stage t:

$$Q_{t+1}(s_t, \langle a_t^1, a_t^2 \rangle) = (1 - \alpha_t)\; Q_t(s_t, \langle a_t^1, a_t^2 \rangle) + \alpha_t \left[\mathcal{R}_t + \gamma\, V_t(s_{t+1})\right]$$

The value of the next state $V(s_{t+1})$:

$$V_{t+1}(s) = \max_{\pi \in \triangle(A^1)} \min_{a^2 \in A^2} \sum_{a^1 \in A^1} \pi_{a^1}\, Q_t(s, \langle a^1, a^2 \rangle)\;.$$

Minimax-Q converges to Nash equilibria under the same assumptions as regular Q-learning [Littman94]

Nash-Q learning

- Nash-Q learning [Hu03]: joint action learner for general-sum stochastic games
- Each individual agent has to estimate Q values for all other agents as well
- Optimal Nash-Q values: sum of immediate reward and discounted future rewards under the condition that all agents play a specified Nash equilibrium from the next stage onward

Nash-Q learning (2)

Update rule for agent i:

$$Q_{t+1}^{i}(s_t, \langle a^1, \dots, a^n \rangle) = (1 - \alpha_t)\, Q(s_t, \langle a^1, \dots, a^n \rangle) \\ + \alpha_t \left[\mathcal{R}_t + \gamma\, Nash V_t^i(s_{t+1}) \right]$$

A Nash equilibrium is computed for each stage game $\left(Q_t^1(s_{t+1}, \cdot), \dots, Q_t^n(s_{t+1}, \cdot)\right)$ and results in the equilibrium payoff $Nash V_t^i(s_{t+1}, \cdot)$ to agent i

Agent i uses the same update rule to estimate Q values for all other agents, i.e., $Q^j\ \forall j \in \{1, \dots, n\} \backslash i$

Other equilibrium learning algorithms

- Friend-or-Foe Q-learning [Littman01]
- Correlated-Q learning (CE-Q) [Greenwald03]
- Nash bargaining solution Q-learning (NBS-Q) [Qiao06]]
- Optimal adaptive learning (OAL) [Wang02]
- Asymmetric-Q learning [Kononen03]

Limitations of MARL

- Convergence guarantees are mostly restricted to stateless repeated games
- ... or are inapplicable in general-sum games
- Many convergence proofs have strong assumptions with respect to a-priori knowledge and/or observability
- Equilibrium learners focus on stage-wise solutions (only indirect state coupling)

Summary

In a multi-agent system

- be aware what information is available to the agent
- if you can afford to try, just run an algorithm that matches the assumptions
- proofs of convergence are available for small games
- new research can focus either on engineering solutions, or advancing the state-of-the-art theories

Questions so far?

Thank you!

Daan Bloembergen | daan.bloembergen@gmail.com
Daniel Hennes | daniel.hennes@gmail.com

Dynamics of Learning in Strategic Interactions

Michael Kaisers

Outline

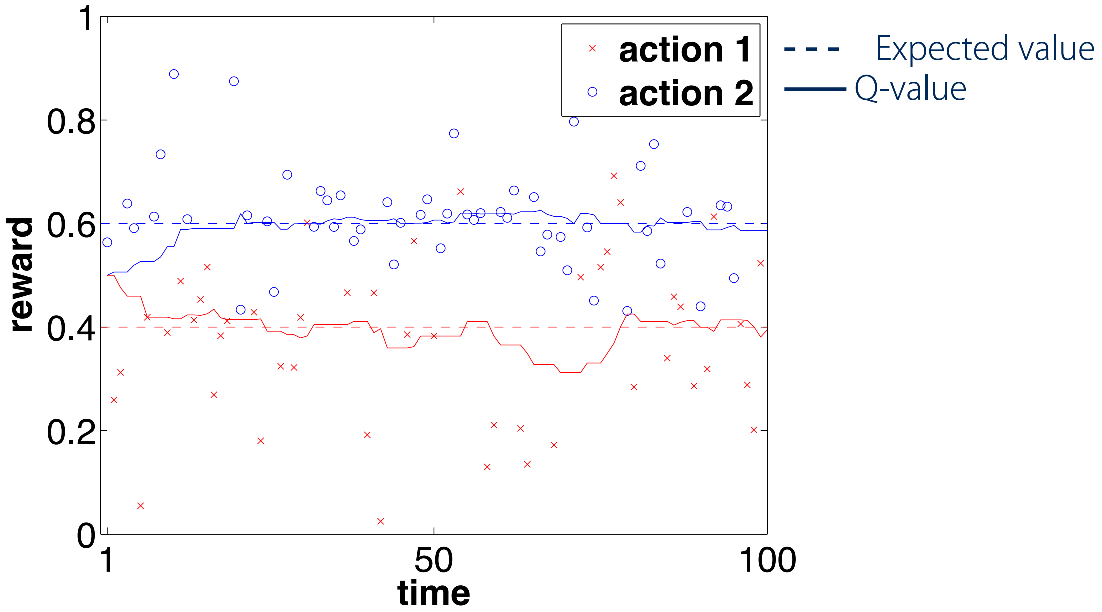
- Action values in dynamic environments
- Deriving learning dynamics
- Illustrating Convergence
- Comparing dynamics of various algorithms
- Replicator dynamics as models of evolution, swarm intelligence and learning
- Summary

Action values in dynamic environments

Action values are estimated by sampling from interactions with the environment, possibly in the presence of other agents.

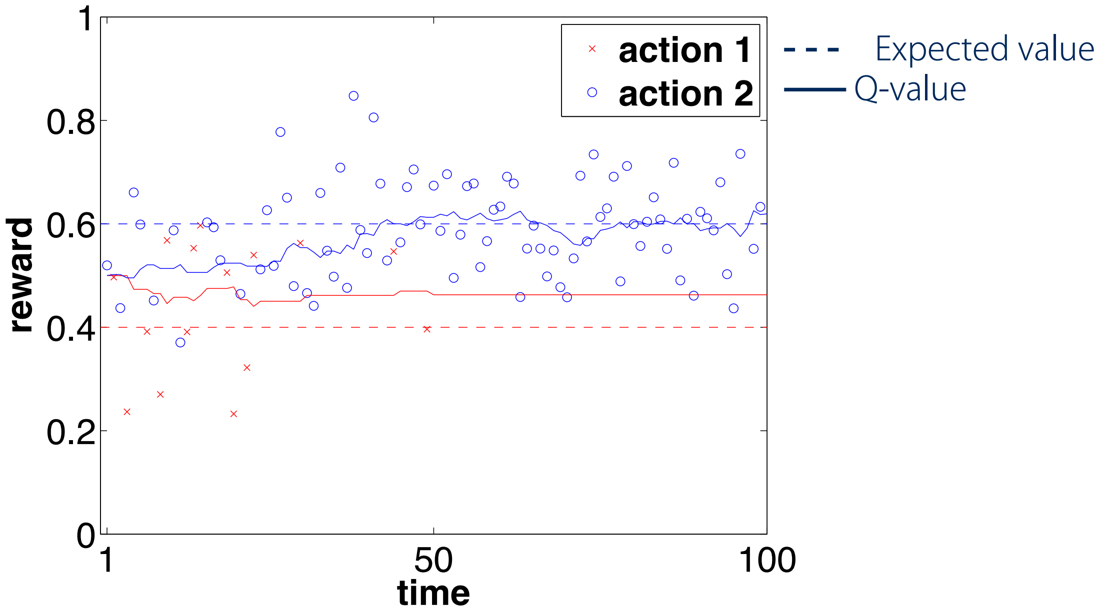
Action values in dynamic environments

Static environment, off-policy Q-value updates



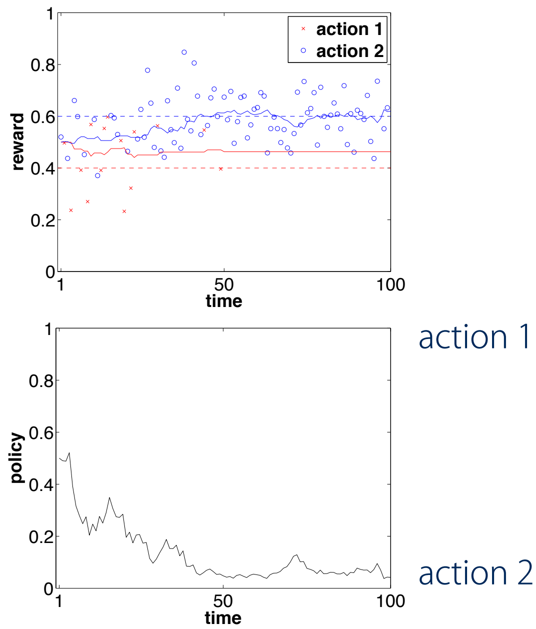
Action values in dynamic environments

Static environment, on-policy Q-value updates



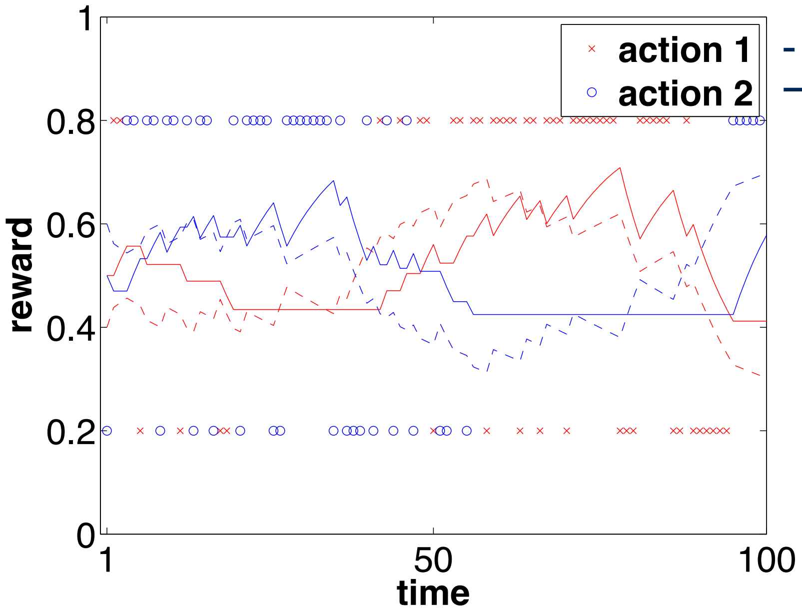
Action values in dynamic environments

Static environment, on-policy Q-value updates



Action values in dynamic environments

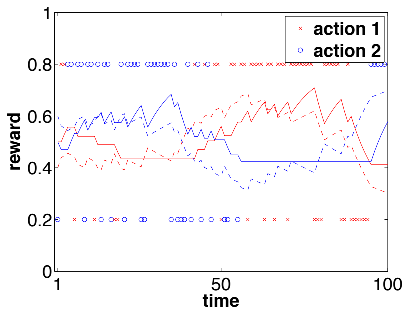
Adversarial environment, on-policy Q-value updates

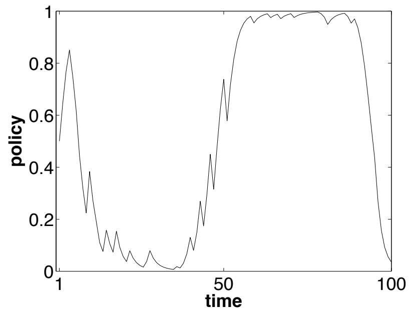


- - - Expected value
—— Q-value

Action values in dynamic environments

Adversarial environment, on-policy Q-value updates





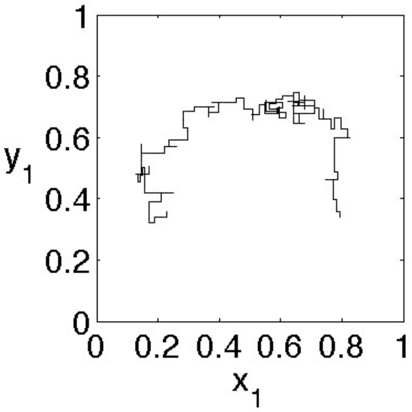
action 1

action 2

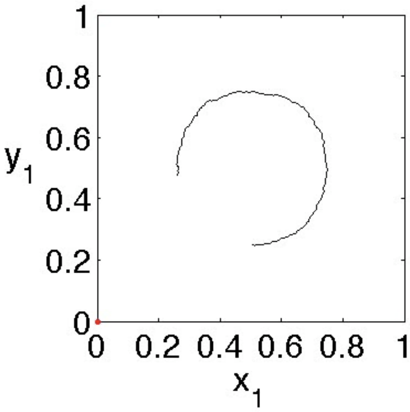
Deriving Learning Dynamics

Matching Pennies

	H	T
H	$1, 0$	$0, 1$
T	$0, 1$	$1, 0$



$\alpha = 0.1$



$\alpha = 0.001$

Deriving Learning Dynamics

Learning algorithm $\lim_{\alpha \to 0} \left(E(\Delta x) \right) = \frac{dx}{dt} = \dot{x}$ Dynamical system

Advantages of dynamical systems

- Deterministic
- Convergence guarantees using Jacobian
- Vast related body of literature (e.g., bifurcation theory)

Deriving Learning Dynamics

Example: Cross learning [Boergers97]

$$x_i(t+1) \leftarrow \begin{cases} (1-\alpha r_i)x_i + \alpha r_i & \text{if } i \text{ selected} \\ (1-\alpha r_j)x_i & \text{for other action } j \text{ selected} \end{cases}$$

$$E(\Delta x_i) = x_i\left[(1-\alpha r_i)x_i + \alpha r_i - x_i\right] + \sum_{k \neq i}^{n} x_k\left[(1-\alpha r_k)x_i - x_i\right]$$

$$= \alpha x_i[(1-x_i)r_i - \sum_{k \neq i}^{n} x_k r_k] = \alpha x_i[r_i - \sum_{k}^{n} x_k r_k]$$

Learning algorithm $\lim_{\alpha \to 0} \left(E(\Delta x)\right) = \frac{dx}{dt} = \dot{x}$ Dynamical system

$$\dot{x}_i = x_i \left[E[r_i] - \sum_{k}^{n} x_k E[r_k] \right] = \underbrace{x_i\left[(Ay)_i - xAy\right]}_{\text{replicator dynamics}}$$

Deriving Learning Dynamics

Prisoners' Dilemma

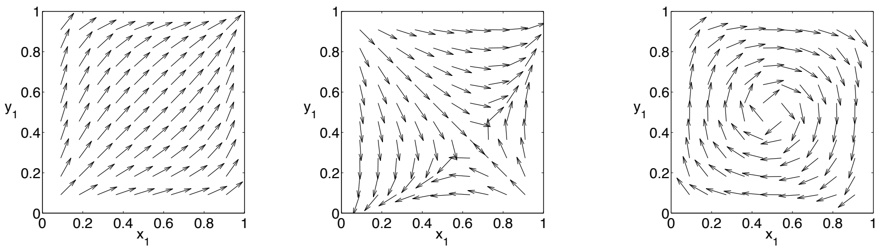
	D	C
D	$1,1$	$5,0$
C	$0,5$	$3,3$
	y_1	$1-y_1$

Battle of Sexes

	B	S
B	$2,1$	$0,0$
S	$0,0$	$1,2$
	y_1	$1-y_1$

Matching Pennies

	H	T	
H	$1,-1$	$-1,\ 1$	x_1
T	$-1,\ 1$	$1,-1$	$1-x_1$
	y_1	$1-y_1$	



$$\dot{x}_i = x_i\left[(Ay)_i - xAy\right]$$
$$\dot{y}_i = y_i\left[(xB)_i - xBy\right]$$

Deriving Learning Dynamics

Dynamics have been derived for

- Learning Automata (Cross Learning)
- Regret Matching (RM)
- Variations of Infinitesimal Gradient Ascent
  - Infinitesimal Gradient Ascent (IGA)
  - Win-or-Learn-Fast (WoLF) IGA
  - Weighted Policy Learning (WPL)
- Variations of Q-learning
  - Repeated Update Q-learning
    See our talk on Thursday, Session F4 Learning 1, 13:50
  - Frequency Adjusted Q-learning

Illustrating Convergence

Q-learning [Watkins92]

x_i probability of playing action i

α learning rate

r reward

τ temperature

Update rule

$$Q_i(t+1) \leftarrow Q_i(t) + \alpha \left(r_i(t) + \gamma \max_j Q_j(t) - Q_i(t) \right)$$

Policy generation function

$$x_i(Q, \tau) = \frac{e^{\tau^{-1} Q_i}}{\sum_j e^{\tau^{-1} Q_j}}$$

Illustrating Convergence

Frequency Adjusted Q-learning (FAQ-learning) [Kaisers2010]

x_i probability of playing action i

α learning rate

r reward

τ temperature

Update rule

$$Q_i(t+1) \leftarrow Q_i(t) + \alpha \frac{1}{x_i} \left(r_i(t) + \gamma \max_j Q_j(t) - Q_i(t) \right)$$

Policy generation function

$$x_i(Q, \tau) = \frac{e^{\tau^{-1} Q_i}}{\sum_j e^{\tau^{-1} Q_j}}$$

Illustrating Convergence

Cross Learning [Boergers97]

$$\dot{x}_i = x_i \left[E\left[r_i(t)\right] - \sum_k^n x_k E\left[r_k(t)\right] \right]$$

Frequency Adjusted Q-learning [Tuyls05, Kaisers2010]

$$\dot{x}_i = \alpha x_i \left(\tau^{-1} \left[E\left[r_i(t)\right] - \sum_k^n x_k E\left[r_k(t)\right] \right] - \log x_i + \sum_k x_k \log x_k \right)$$

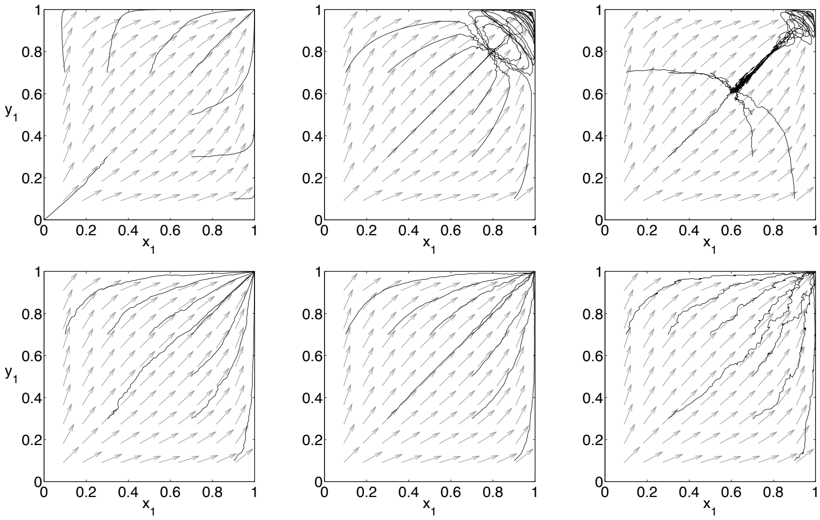
Proof of convergence in two-player two-action games [Kaisers2011, Kianercy2012]

Illustrating Convergence

Prisoners' Dilemma

Q

FAQ



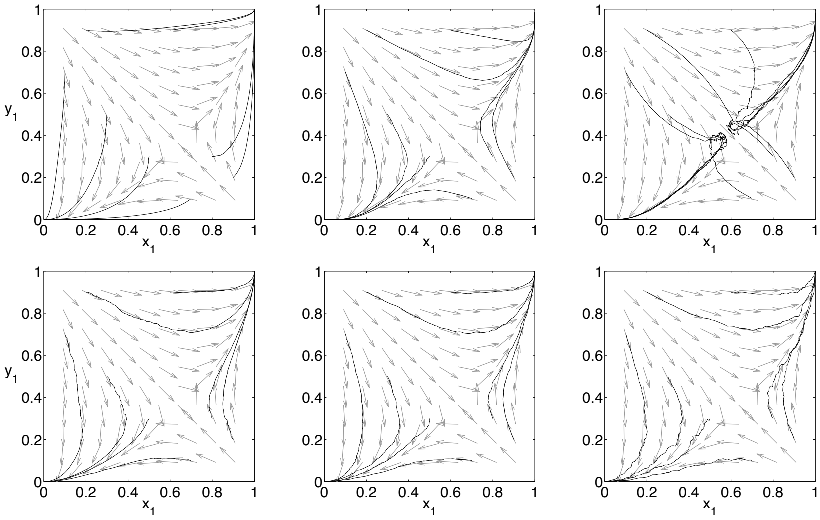
Q_{init} pessimistic neutral optimistic

Illustrating Convergence

Battle of Sexes

Q

FAQ

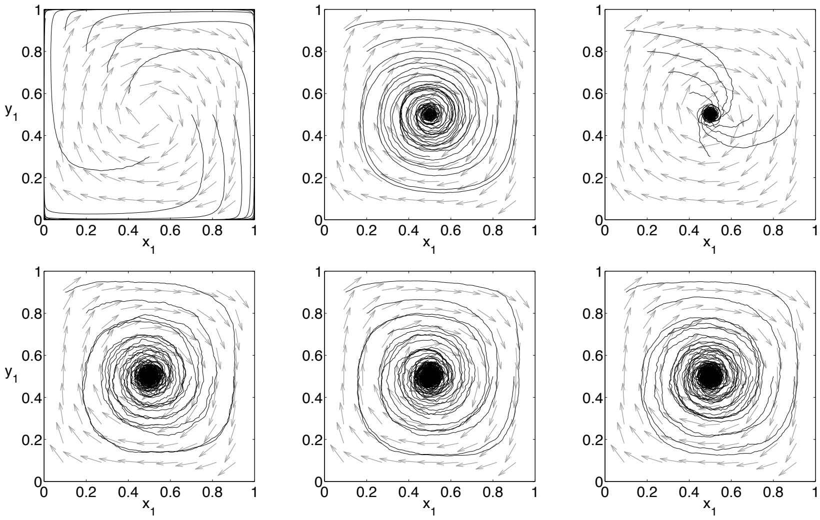


Q_{init} pessimistic neutral optimistic

Illustrating Convergence

Matching Pennies

Q



FAQ

Q_{init} pessimistic neutral optimistic

Comparing Dynamics

Dynamical systems have been associated with

- Infinitesimal Gradient Ascent (IGA)
- Win-or-Learn-Fast Infinitesimal Gradient Ascent (WoLF)
- Weighted Policy Learning (WPL)

- Cross Learning (CL)
- Frequency Adjusted Q-learning (FAQ)
- Regret Matching (RM)

Comparing Dynamics

Learning dynamics for two-agent two-action games. The common gradient is abbreviated $\eth = \left[yhAh^T + A_{12} - A_{22}\right]$.

Algorithm	$\dot{x}$
IGA	$\alpha\eth$
WoLF	$\eth \cdot \begin{cases} \alpha_{min} & \text{if } V(x,y) > V(x^e,y) \\ \alpha_{max} & \text{otherwise} \end{cases}$
WPL	$\alpha\eth \cdot \begin{cases} x & \text{if } \eth < 0 \\ (1-x) & \text{otherwise} \end{cases}$
CL	$\alpha x(1-x)\ \eth$
FAQ	$\alpha x(1-x)\left[\eth \cdot \tau^{-1} - \log\frac{x}{1-x}\right]$
RM	$\alpha x(1-x)\ \eth \cdot \begin{cases} (1+\alpha x\eth)^{-1} & \text{if } \eth < 0 \\ (1-\alpha(1-x)\eth)^{-1} & \text{otherwise} \end{cases}$

Comparing Dynamics

Cross Learning is linked to the replicator dynamics

$$\dot{x}_i = x_i \left[E\left[f_i(t)\right] - \sum_k^n x_k E\left[f_k(t)\right] \right]$$

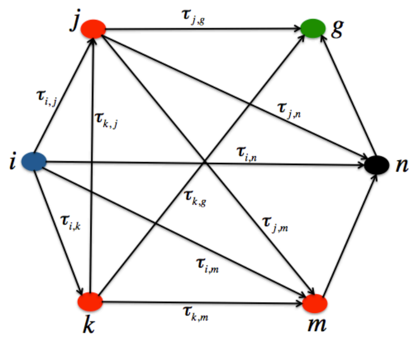
Gradient based algorithms use the orthogonal projection function, leading to the following dynamics for IGA:

$$\dot{x}_i = \alpha \left[E\left[f_i(t)\right] - \sum_k^n \frac{1}{n} E\left[f_k(t)\right] \right]$$

Gradient dynamics use information about all actions, and are equivalent to the replicator dynamics under the uniform policy (i.e., also given off-policy updates).

Replicator Dynamics

Path finding with Ant Colony Optimization



Pheromones τ and travel cost heuristic η lead to probabilistic selection of state transition $x_{i,j}$ from i to j:

$$x_{i,j} = \frac{\tau_{i,j}^{\alpha}\eta^{\beta}}{\sum_c \tau_{i,c}^{\alpha}\eta_{i,c}^{\beta}}, \text{ with } \alpha, \beta \text{ tuning parameters}$$

Replicator Dynamics

Pheromone trail reinforcement in Ant Colony Optimization

$$\tau_{i,j}(t+1) = (1-\rho)\tau_{i,j}(t) + \sum_{m=1}^{M} \delta_{i,j}(t,m),$$

where ρ denotes the pheromone evaporation rate, M is the number of ants and $\delta_{i,j}(t,m) = Q\frac{n_{i,j}}{L}$ with Q being a constant, $n_{i,j}$ being the number of times edge (i,j) has been visited.

$$\dot{x}_{i,j} = \left(\frac{\tau_{i,j}^{\alpha}\eta^{\beta}}{\sum_c \tau_{i,c}^{\alpha}\eta_{i,c}^{\beta}}\right)' = \alpha x_{i,j}\frac{\dot{\tau}_{i,j}}{\tau_{i,j}} - \alpha x_{i,j}\sum_c \frac{\dot{\tau}_{i,c}}{\tau_{i,c}} x_{i,c}$$

$$= \underbrace{\alpha\, x_{i,j}\left(\Theta_{i,j} - \sum_k x_{i,k}\Theta_{i,k}\right)}_{\text{replicator dynamics}}, \Theta_{i,j} = \frac{\dot{\tau}_{i,j}}{\tau_{i,j}}$$

Replicator Dynamics

$$\dot{x}_i = x_i \left[E\left[f_i(t)\right] - \sum_k^n x_k E\left[f_k(t)\right] \right]$$

Relative competitiveness as encoded by the replicator dynamics models

- the selection operator in evolutionary game theory
- pheromone trail reinforcement in swarm intelligence
- and exploitation in reinforcement learning dynamics.

Summary

In strategic interactions

- the action values change over time
- the joint learning is a complex stochastic system
- dynamics can be captured in dynamical systems
  - proof of convergence
  - similarity of dynamics despite different implementations
  - link between learning, evolution and swarm intelligence
- different assumptions about observability give rise to a menagerie of algorithms to choose from

Questions?

Thank you!

Michael Kaisers | michaelkaisers@gmail.com

Scaling Multi-agent Reinforcement Learning

Peter Vrancx

Joint work with: Y-M De Hauwere, A. Rodriguez, A. Nowe

CoMo
Computational Modeling Lab

Vrije
Universiteit
Brussel

Sparse Interactions in Multi-agent reinforcement learning

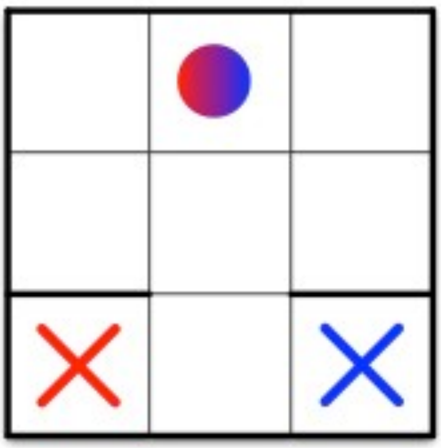
CoMo
Computational Modeling Lab

Vrije
Universiteit
Brussel

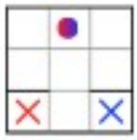
Motivation

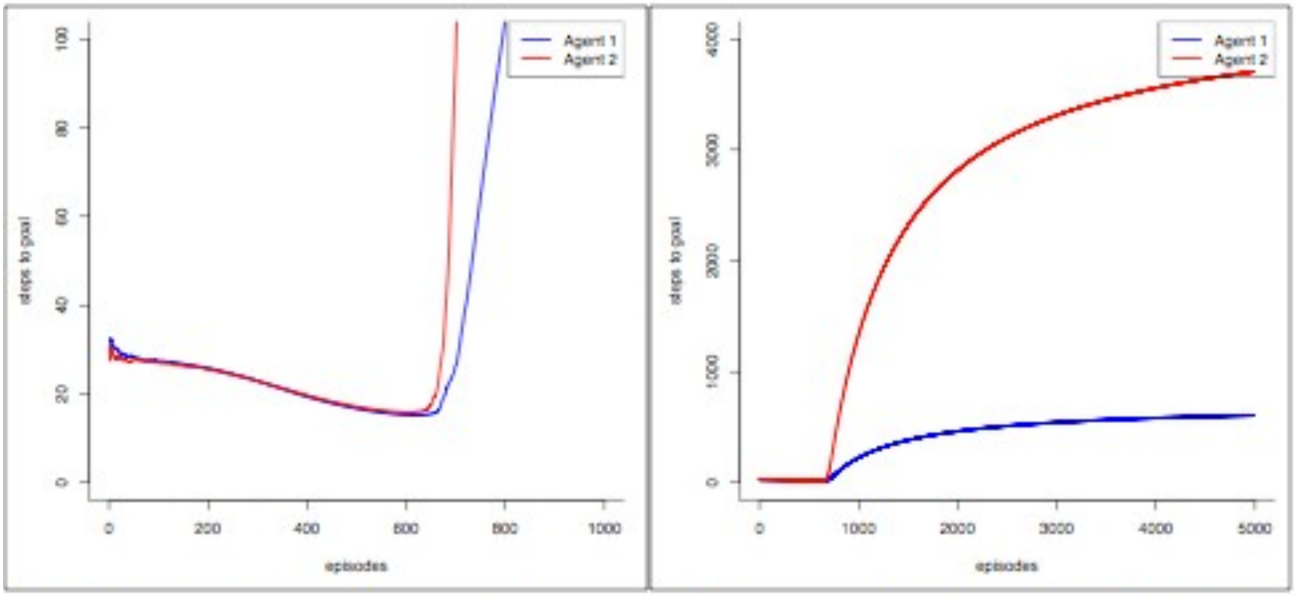
- Several issues arise when applying single agent RL techniques in multi-agent settings:
  - One vs. many learning agents?
  - Convergence? non-stationary, non-Markovian,...
  - Learning goal: e.g. maximize common reward vs. individual reward
  - Influence of action selection strategies and interactions
  - Credit assignment?
  - ...

Simple example



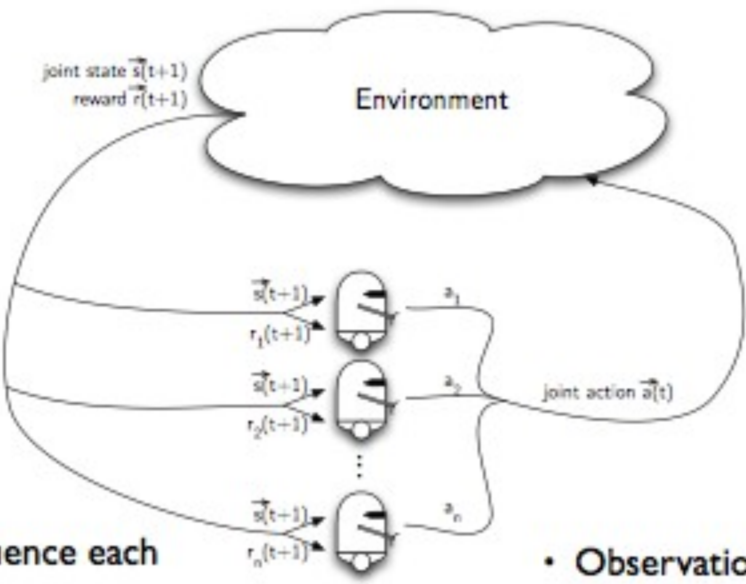
Boltzmann exploration





Agents need information on other agents to coordinate

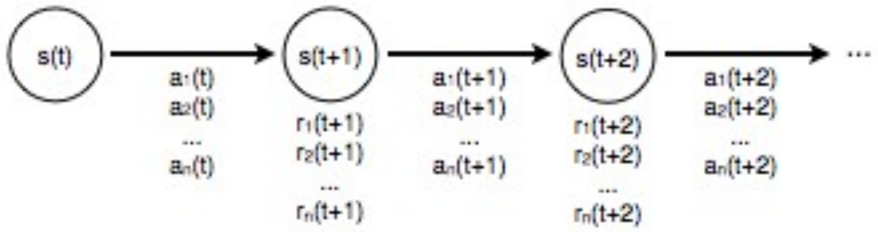
Multi-agent reinforcement learning



- Agents influence each other
- Possibly conflicting interests

- Observations
- Expensive communication

Markov Games



n	• the number of agents
$S = \{s^1, ..., s^n\}$	• a finite set of states
$A = A_1 \times ... \times A_n$	• with A_k the action set of agent k
$T : S \times A_1 \times ... \times A_n \times S \to [0,1]$	• the transition function
$R_k : S \times A_1 \times ... \times A_n \times S \to \mathbb{R}$	• the reward function of agent k

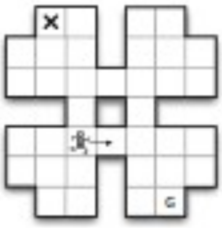
7

Learning in Markov Games

- Learning occurs in join state space (= all local information of all agents)
- Coordination mechanisms often require learning in joint action space
- Large information/communication requirements
- Exponential increases in problem size

8

Sparse interactions



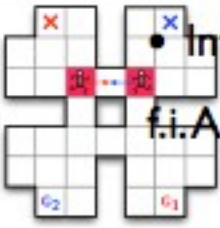
1 agent

Transitions & rewards are only dependent on 1 agent

2 agents

Far away and not interacting with each other

Transitions & rewards are independent of state/action of other agents



2 agents

Close to each other and interacting!!!

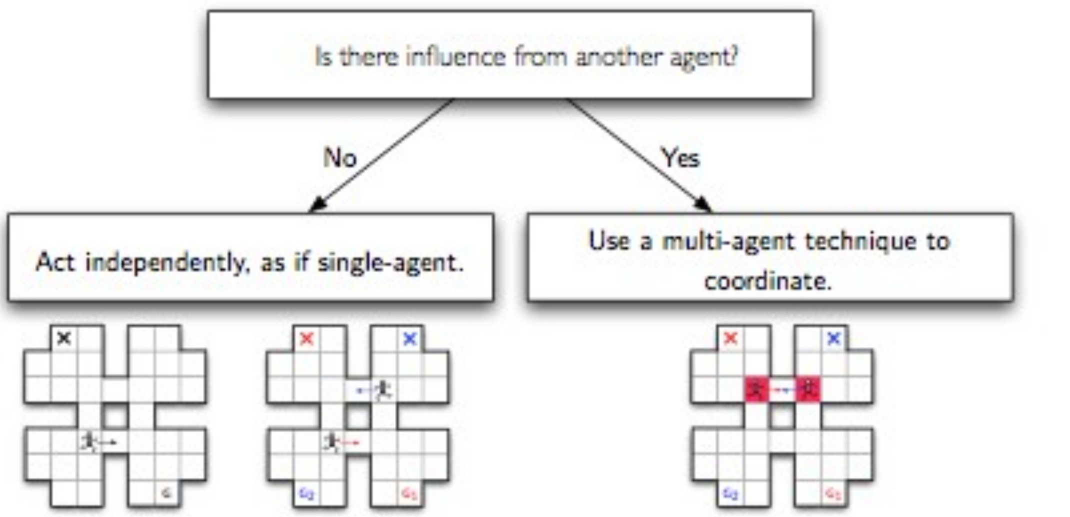
i.e. transitions & rewards are dependent

Assumptions:

- Agents can do something useful alone
- Interactions are sparse

f.i. Air traffic control, automated warehouses, ...

Intuition of sparse interactions



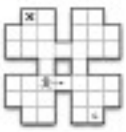
When should agents observe the state information of other agents to avoid coordination problems?

Modeling interactions

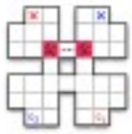
- Dynamics of the system are a Markov game
- Model sparse interactions as a DEC-SIMDP (Melo et al., 2010)

$$\Gamma = (\underbrace{M^k}, (\underbrace{M^{I,l}, S^{I,l}}))$$

MDP for each agent *k* in the absence of other agents (containing local states)

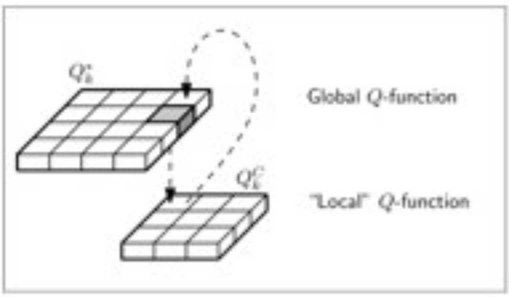


Team Markov game for the local interaction between K agents in L interaction states (containing system states)



Learning of Coordination

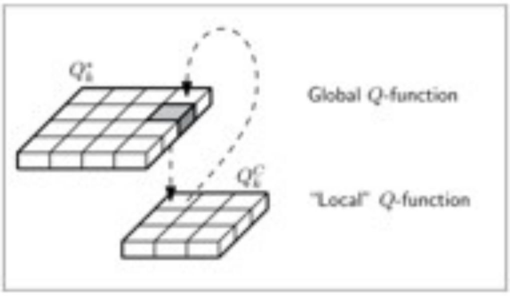
When to observe?



Melo & Veloso, 2009

Learning of Coordination

- Add Pseudo COORDINATE action
- External Active Perception
- Cost for coordination



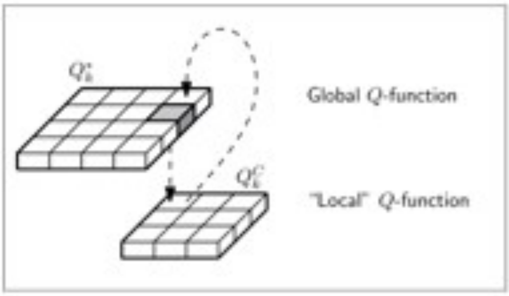
The algorithm

Algorithm 1 Learning algorithm for agent k

Initialize Q_k^* and Q_k^C;
Set $t = 0$;
while (FOREVER) **do**
  Choose $A_k(t)$ using π_e;
  if $A_k(t) =$ COORDINATE **then**
    if $ActivePercept =$ TRUE **then**
      $\hat{A}_k(t) = \pi_g(Q_k^C, X(t))$;
    else
      $\hat{A}_k(t) = \pi_g(Q_k^*, X_k(t))$;
    end if
    Sample $R_k(t)$ and $X_k(t+1)$;
    if $ActivePercept =$ TRUE **then**
      QLUpdate$(Q_k^C; X(t), \hat{A}_k(t), R_k(t), X_k(t+1), Q_k^*)$;
    end if
  else
    Sample $R_k(t)$ and $X_k(t+1)$;
  end if
  QLUpdate$(Q_k^*; X_k(t), A_k(t), R_k(t), X_k(t+1), Q_k^*)$;
  $t = t + 1$;
end while

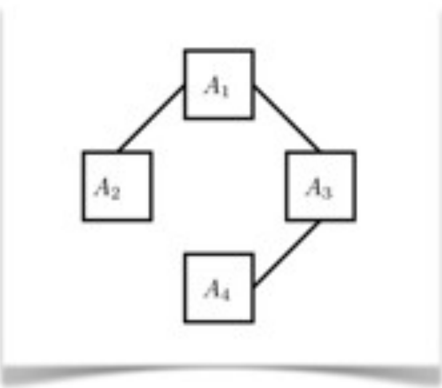
Utile Coordination

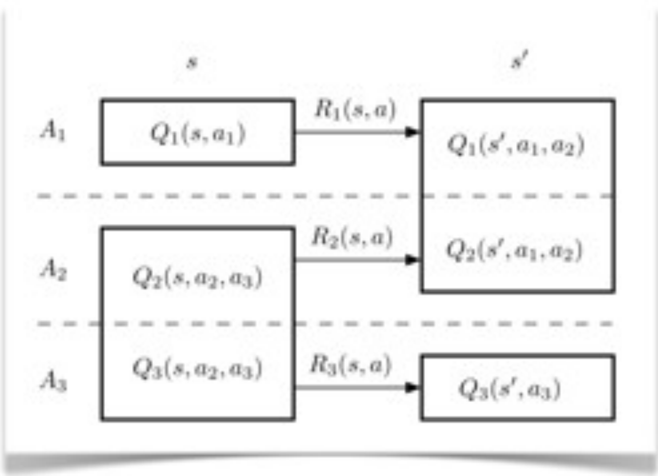
When to coordinate?



Kok & Vlassis, 2005

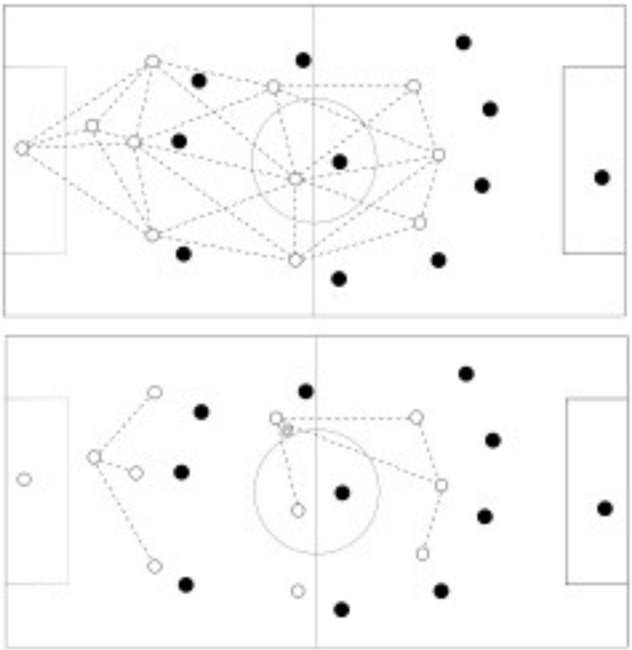
Coordination graphs





Coordination through variable elimination algorithm

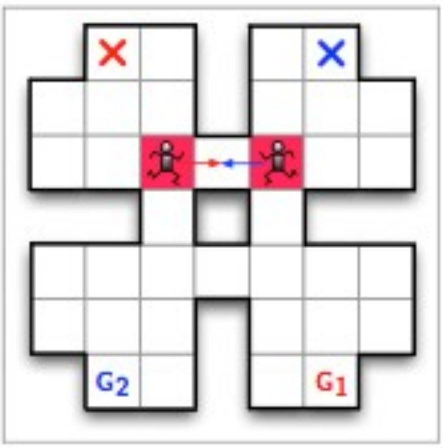
Example: Robosoccer



17

CQ-Learning

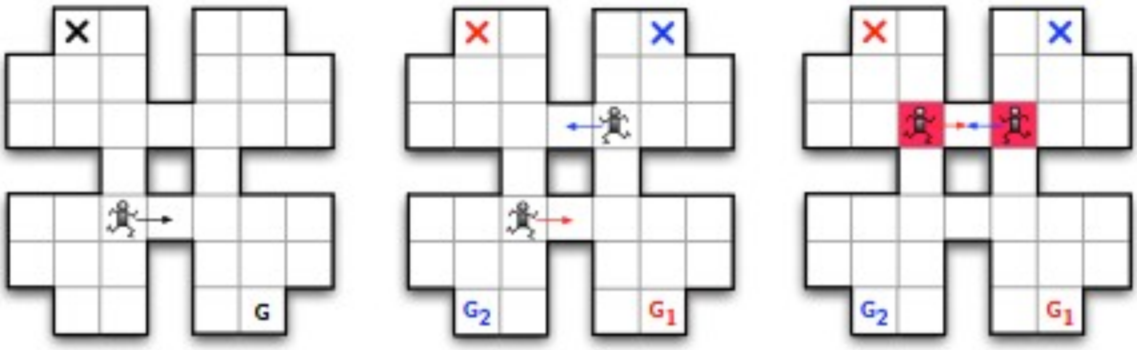
Who to observe when?



De Hauwere et al, 2010

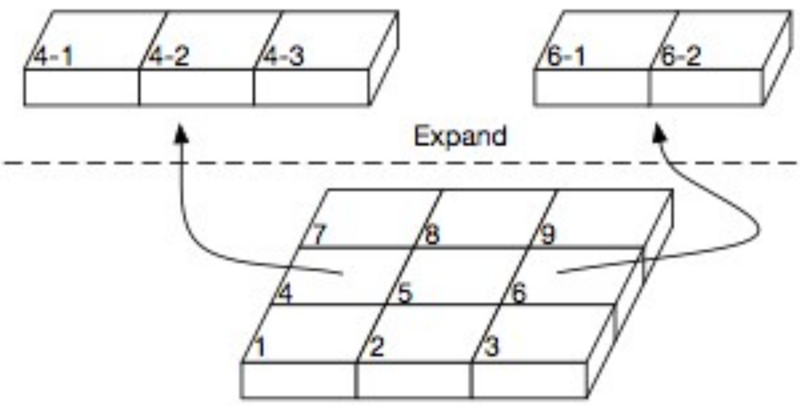
18

Problem setting

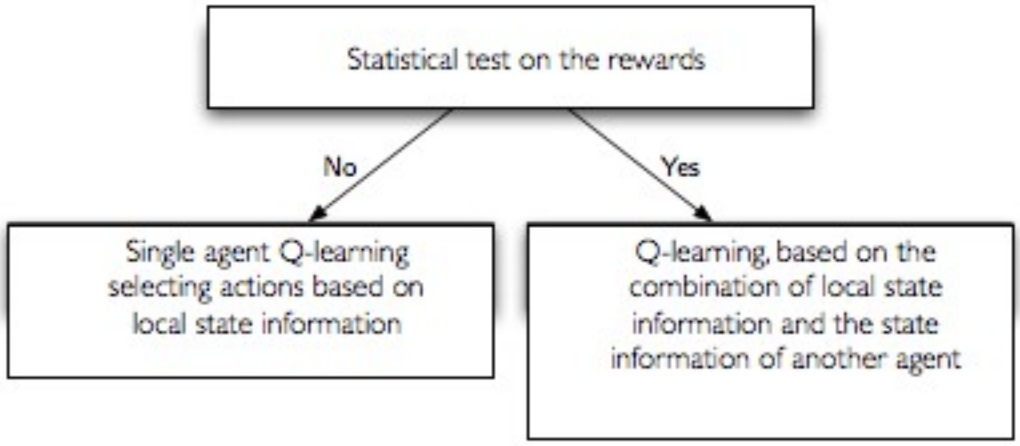


- Agents only interact where their policies interfere
- Locally adapt policy

Representation idea

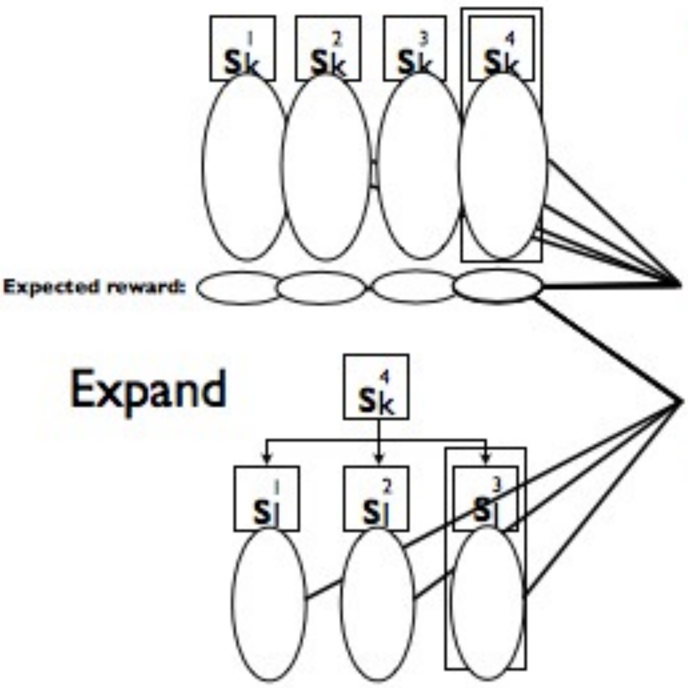


Solution method: *CQ-learning*



CQ-learninG

STATISTICAL TESTS

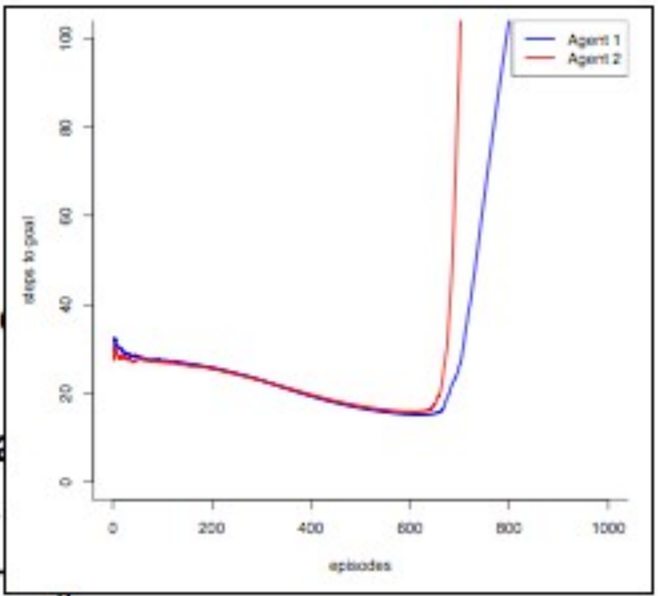


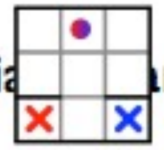
- Agents have been learning alone in the environment
- Agent k acts independently using only local state information (s_k^i) in a multi-agent environment
- Performs statistical test against a baseline
- Samples its rewards, based on the state information of other agents & performs the same test

$$s_k^4 \Rightarrow \langle s_k^4, s_l^3 \rangle$$

CQ-LEARNING

BASELINE FOR STATISTICAL TESTS



- Initia  rds (sli

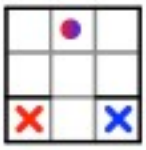
for a particular sta

○○○○○○○○○○○○○○

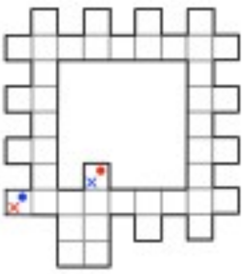
W_k^1 W_k^2

Compare W_k^1 against W_k^2

Experimental results (1)

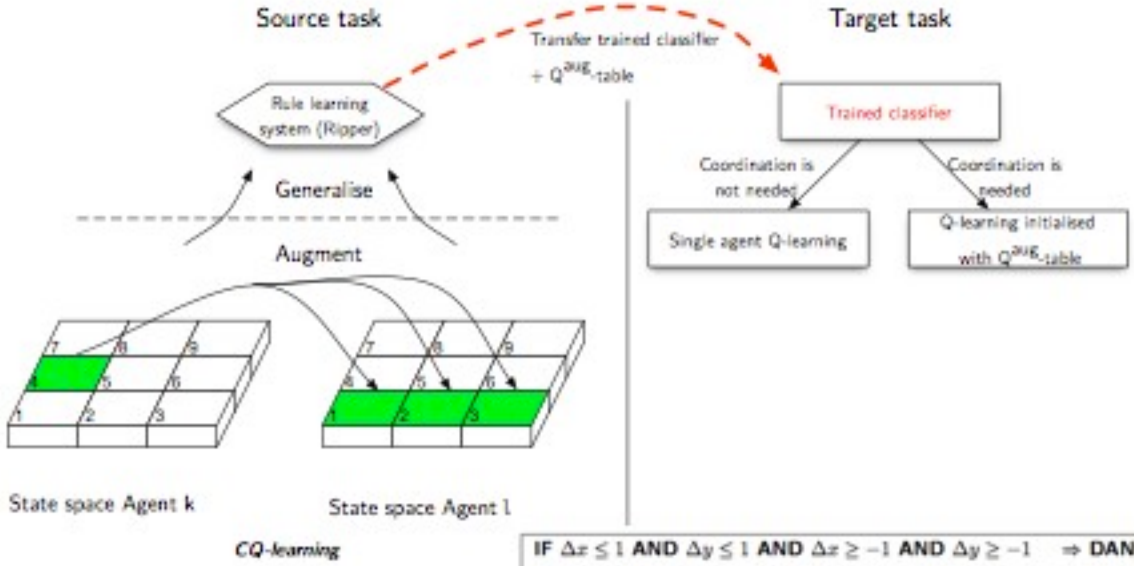


Env	Alg	#states	#actions	#coll	#steps
Grid_game_2	Indep	9	4	2.7	22.2 ± 17.9
	JS	81	4	0.1	4.0 ± 0.2
(min steps: 3)	JSA	81	16	0.0	4.7 ± 0.1
	LOC	9.9 ± 0.5	5	0.1	4.0 ± 0.4
	CQ	10 ± 0.0	4	0.0	**3.6 ± 0.3**
	CQ_NI	10.9 ± 2.0	4	0.1	4.0 ± 0.3



Env	Alg	#states	#actions	#coll	#steps
ISR	Indep	43	4	0.4	9.3 ± 44.8
	JS	1849	4	0.1	5.7 ± 1.6
(min steps: 4)	JSA	1849	16	0.0	7.6 ± 1.4
	LOC	51.3 ± 82.3	5	0.2	6.7 ± 7.5
	CQ	49.0 ± 2.3	4	0.1	**5.1 ± 0.7**
	CQ_NI	49.9 ± 7.8	4	0.1	6.0 ± 1.9

Transfer learning with CQ-learning

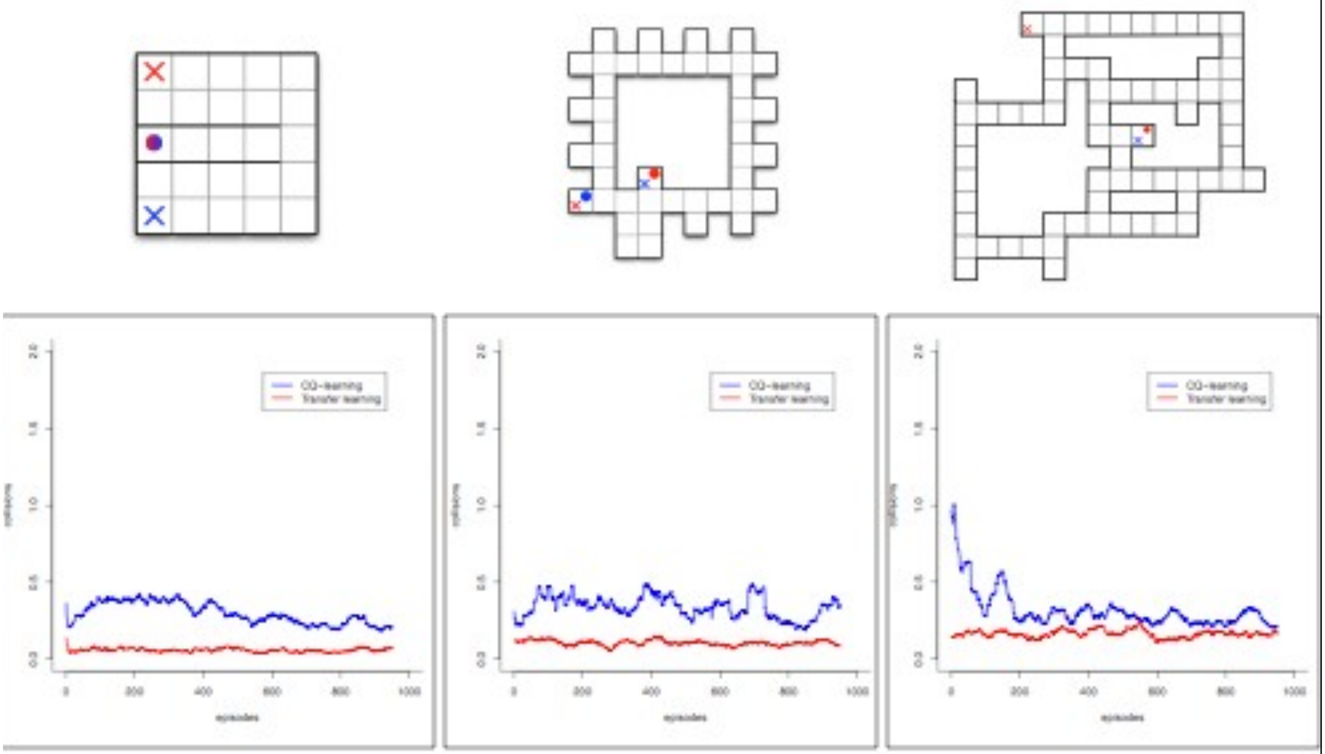


IF $\Delta x \leq 1$ AND $\Delta y \leq 1$ AND $\Delta x \geq -1$ AND $\Delta y \geq -1$ ⇒ DANGEROUS
IF $\Delta x \leq 0$ AND $\Delta x \geq 0$ AND $\Delta y \leq 2$ AND $\Delta y \geq -2$ ⇒ DANGEROUS
IF $\Delta y \leq 0$ AND $\Delta y \geq 0$ AND $\Delta x \geq -2$ AND $\Delta x \leq 2$ ⇒ DANGEROUS
ELSE ⇒ SAFE

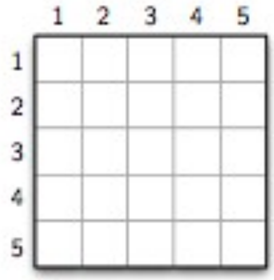
Vrancx et al, 2011

De Hauwere et al, 2011

Results (2)

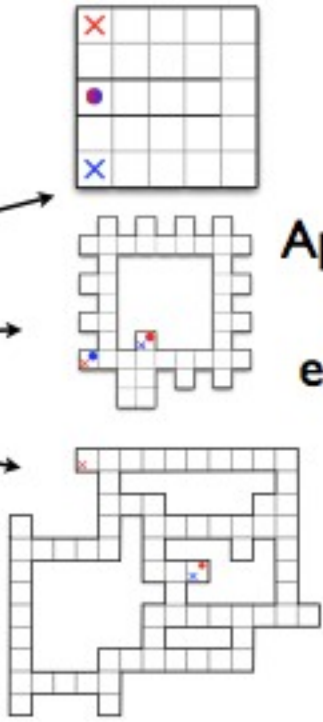


Transfer learning with CQ-learning (2)



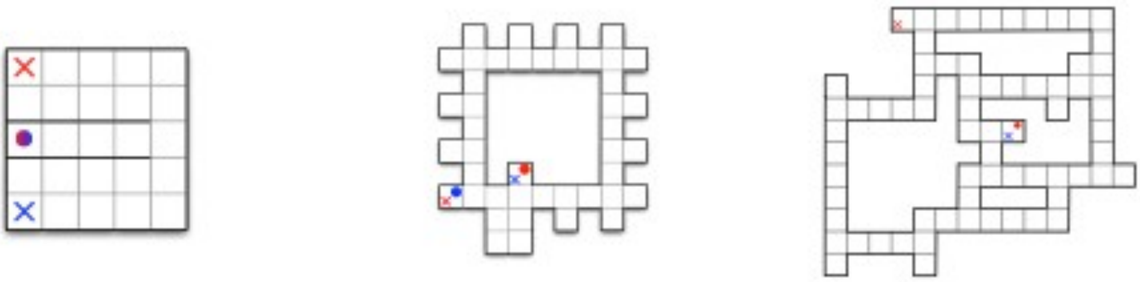
Learn to coordinate in simple environment

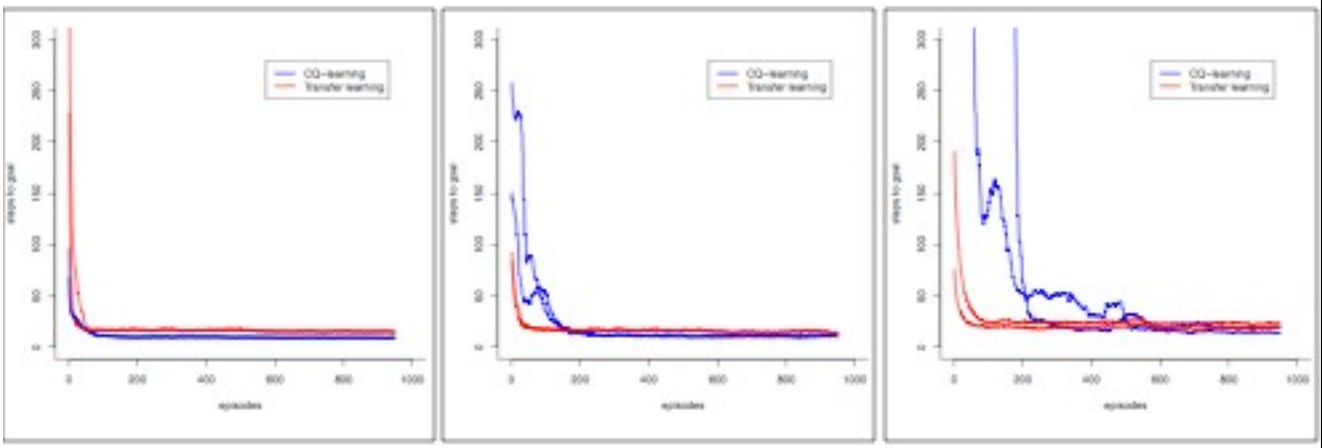
Transfer high level knowledge



Apply in more complex environment

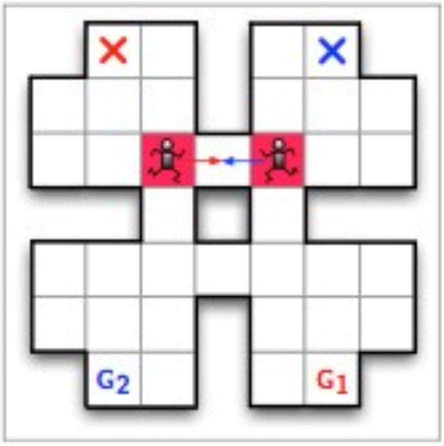
Results





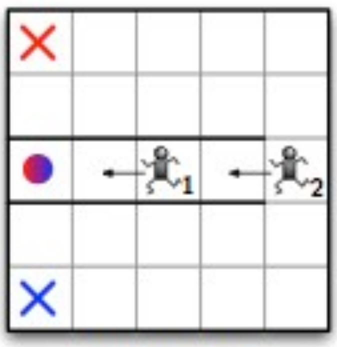
FCQ-Learning

Who to observe when?

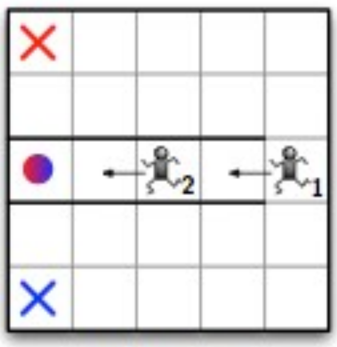


De Hauwere et al, 2011

Problem setting



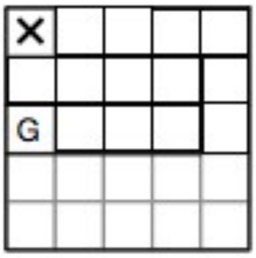
Reward: +20

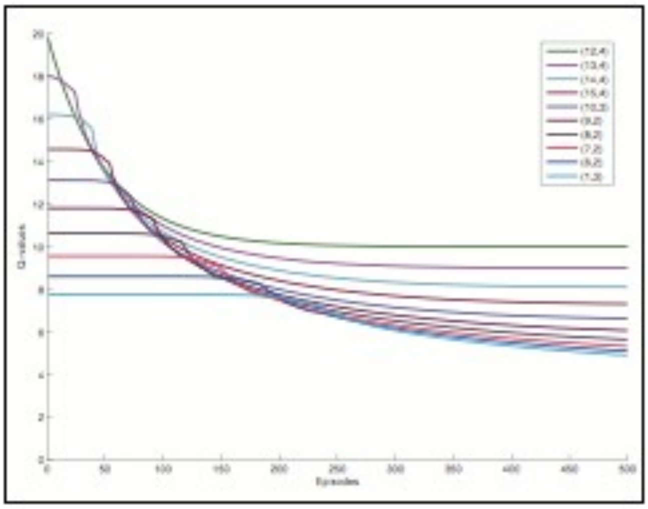


Reward: +10

- Reflected in immediate reward signal
- Too late to solve the problem

Detecting relevant states

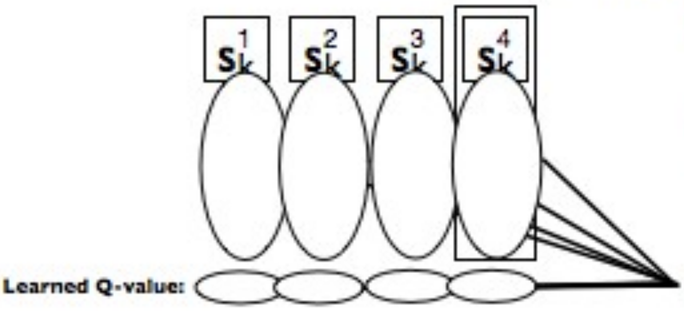




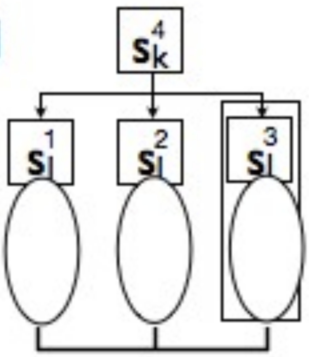
Changes in reward signal are reflected in the Q-values

FCQ-learning

statistical tests



Expand

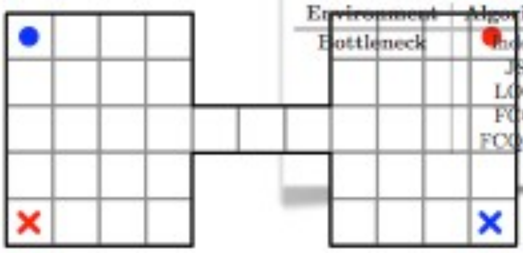


- Agent k has been learning alone, and its Q-values have converged
- Agent k acts independently using only local state information (s_k) in a multi-agent environment
- Performs statistical test against the single agent Q-values
- Samples rewards monte carlo and perform a comparison test to determine what information should be included

$$s_k^4 \Rightarrow \langle s_k^4, s_l^3 \rangle$$

Experimental results

Environment	Algorithm	#states	#actions	#collisions	#steps	reward
Grid_game_2	Indep	9	4	2.4 ± 0.0	22.7 ± 30.4	-24.3 ± 35.6
	JS	81	4	0.1 ± 0.0	$\mathbf{6.3 \pm 0.3}$	$\mathbf{18.2 \pm 0.6}$
	LOC	9.0 ± 0.0	5	1.8 ± 0.0	10.3 ± 2.7	-6.8 ± 8.0
	FCQ	19.4 ± 4.4	4	0.1 ± 0.0	8.1 ± 13.9	17.6 ± 3.7
	FCQ_NI	21.7 ± 3.1	4	0.1 ± 0.0	7.1 ± 6.9	17.9 ± 0.7



Environment	Algorithm	#states	#actions	#collisions	#steps	reward
Bottleneck	Indep	43	4	n.a.	n.a.	n.a.
	JS	1849	4	0.0 ± 0.0	23.3 ± 30.8	$13.1 \pm 36.$
	LOC	54.0 ± 0.8	5	1.7 ± 0.6	$167.2 \pm 19,345.1$	$-157.5 \pm 10,3$
	FCQ	124.5 ± 32.8	4	0.1 ± 0.0	$\mathbf{17.3 \pm 1.3}$	$\mathbf{16.6 \pm 0.4}$
	FCQ-NI	135.0 ± 88.7	4	0.2 ± 0.0	19.2 ± 5.6	15.4 ± 2.3

Conclusions

In multi-agent environments with sparse interactions, learning these interaction states improves the learning process

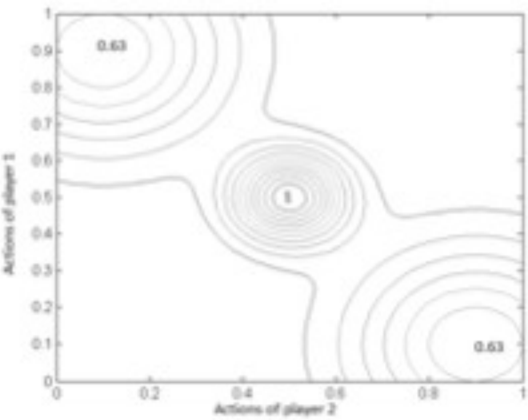
- Interaction states can be learned through increased penalties for miscoordination [Melo & Veloso, 2009]
- Interaction states can be identified using statistical tests on the reward signal (immediate + future) [De Hauwere et al, 2010 & 2011]
- Information about interaction states can be generalized and transferred between agents and environments [De Hauwere et al 2010, Vrancx et al 2010]

Multi-agent Reinforcement Learning in Continuous Action Games

CoMo

Vrije Universiteit Brussel

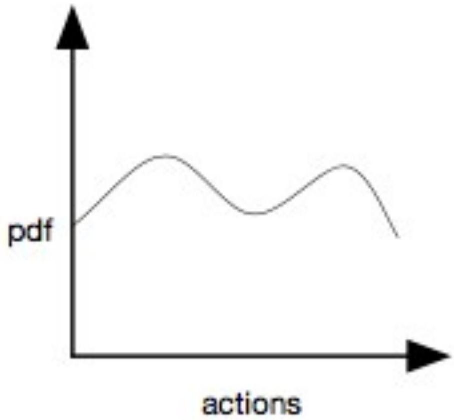
Learning in Continuous Action Games



Assumptions: reward function is continuous, action sets are compact

- Generalization of discrete normal form games
- Each agent now selects actions from continuous set
- Reward function is continuous function of all agents' actions

Continuous Action Reinforcement Learning Automata (CARLA)

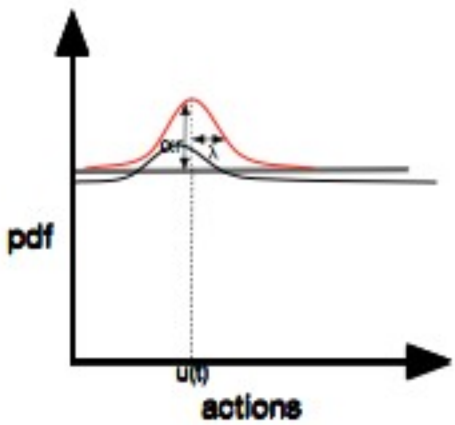


$$f_{k+1}(u) = \begin{cases} \eta_k \left(f_k(u) + \rho(u_k)\,\alpha e^{-\frac{1}{2}\left(\frac{u-u_k}{\lambda}\right)^2} \right) \\ 0 \end{cases}$$

Howell, 1997

- Extension of learning automata idea to continuous action spaces
- Now store continuous probability density distribution
- nonparametric pdf over actions
- parametric alternative: CALA (Santharam et al., 1994)

CARLA Update



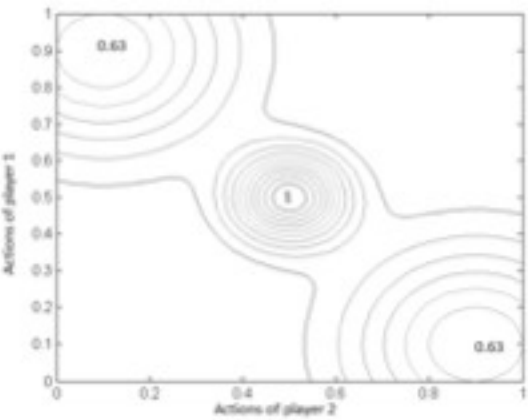
Select action,
receive corresponding
distribution

- Extension of reward-inaction principles
- Reinforce selected action by adding Gaussian Bell to distribution
- Amplitude (strength of reinforcement) is determined by magnitude of reward and learning rate α
- Width (generalization) is determined by spreading rate parameter λ

CARLA Results

- In single agent systems: Converges to optimal neighborhood, depending on spreading rate λ (Rodriguez et al, 2011)
- In (cooperative) games a set of CARLA will converge to a locally superior strategy (Rodriguez et al, 2012)
- More accurate convergence can be achieved by transforming rewards and adaptively tuning the spreading rate λ (Rodriguez et al, 2011)

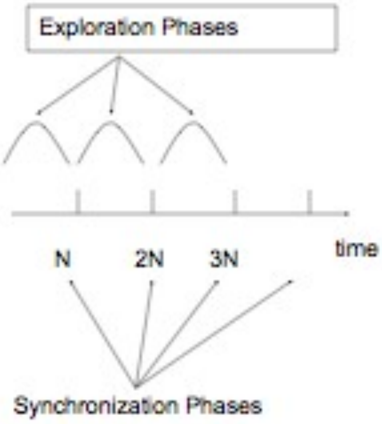
Coordinated Exploration in Continuous Action Games



- In games CARLA may get stuck in local optima
- A narrow basin of attraction can make the global optimum difficult to find
- Coordinated exploration can allows learners to efficiently explore the joint action space

Coordinated Exploration in discrete Games:

Exploring selfish reinforcement learners (**ESRL**)



Basic idea: 2 phases

Exploration: Be Selfish

- Independent learning
- Convergence to different NE and Pareto Optimal non-NE

Synchronization: Coordinate

Verbeeck, 2004

ESRL in discrete games

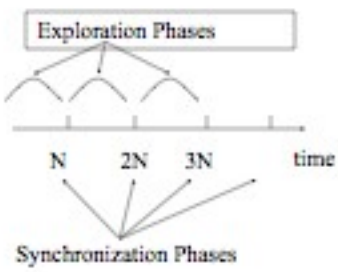
The Penalty Game

Player B

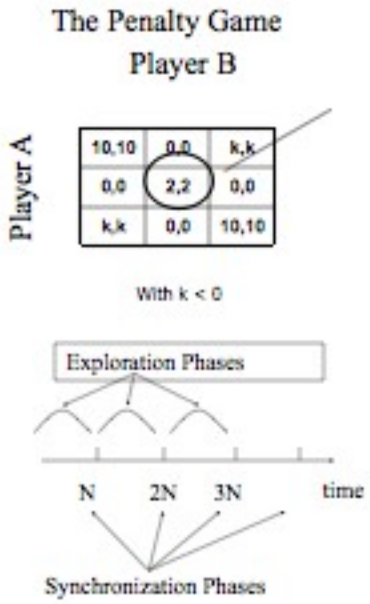
Player A

10,10	0,0	k,k
0,0	2,2	0,0
k,k	0,0	10,10

With $k < 0$

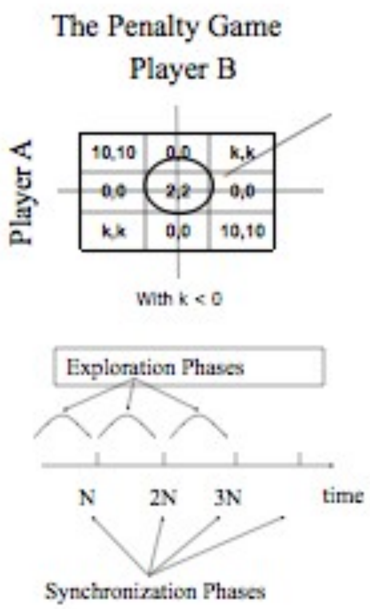


ESRL



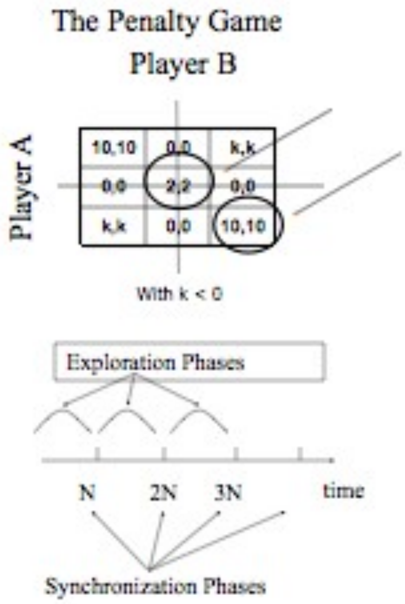
- Exploration
  - Use LRI → the agents converge to pure (Nash) joint action
- Synchronization
  - Update average payoff for action *a* converged to, optimistically
  - Exclude action *a* and explore again if empty *action* set → RESET
- If done, select BEST

ESRL



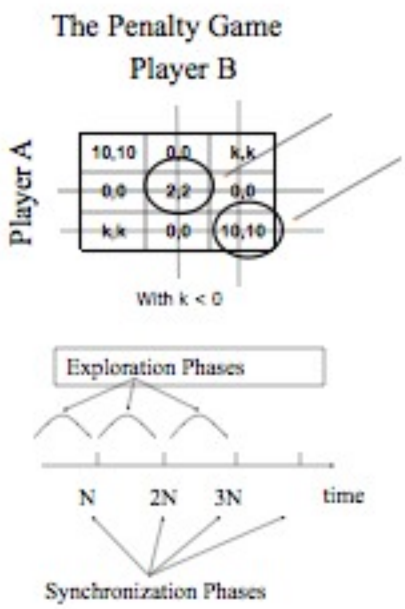
- Exploration
  - Use LRI → the agents converge to pure (Nash) joint action
- Synchronization
  - Update average payoff for action *a* converged to, optimistically
  - Exclude action *a* and explore again if empty *action* set → RESET
- If done, select BEST

ESRL



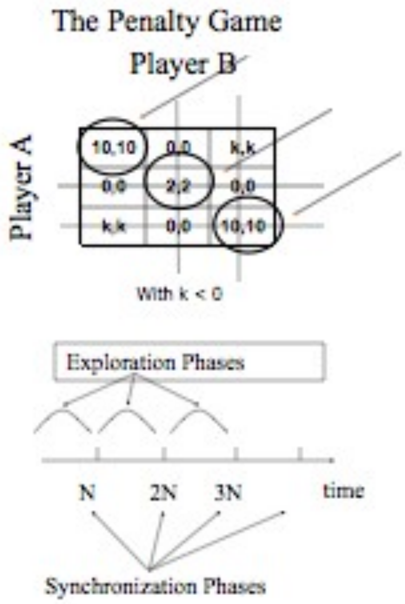
- Exploration
  - Use LRI → the agents converge to pure (Nash) joint action
- Synchronization
  - Update average payoff for action *a* converged to, optimistically
  - Exclude action *a* and explore again if empty *action* set → RESET
- If done, select BEST

ESRL



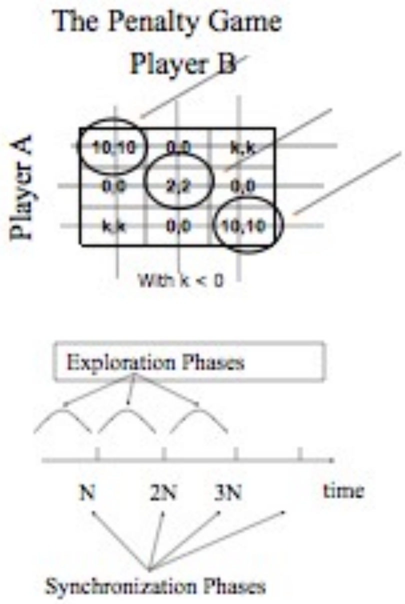
- Exploration
  - Use LRI → the agents converge to pure (Nash) joint action
- Synchronization
  - Update average payoff for action *a* converged to, optimistically
  - Exclude action *a* and explore again if empty *action* set → RESET
- If done, select BEST

ESRL



- Exploration
  - Use LRI → the agents converge to pure (Nash) joint action
- Synchronization
  - Update average payoff for action *a* converged to, optimistically
  - Exclude action *a* and explore again if empty *action* set → RESET
- If done, select BEST

ESRL



- Exploration
  - Use LRI → the agents converge to pure (Nash) joint action
- Synchronization
  - Update average payoff for action *a* converged to, optimistically
  - Exclude action *a* and explore again if empty *action* set → RESET
- If done, select BEST

ESRL

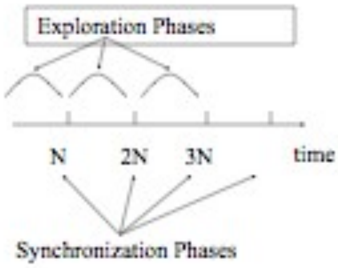
The Penalty Game

Player B

Player A

10,10	0,0	k,k
0,0	2,2	0,0
k,k	0,0	10,10

With k < 0



- Exploration
  - Use LRI → the agents converge to pure (Nash) joint action
- Synchronization
  - Update average payoff for action *a* converged to, optimistically
  - Exclude action *a* and explore again if empty *action* set → RESET
- If done, select BEST

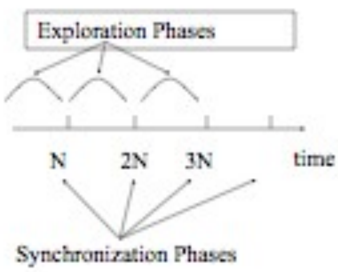
ESRL in conflicting interest games

Battle of the sexes

Player 2

Player 1

	B	S
B	2,1	0,0
S	0,0	1,2



- Exploration
  - Use LRI → the agents converge to pure (Nash) joint action
- Synchronization
  - Update average payoff for action *a* converged to, optimistically
  - Exclude action *a* and explore again if empty *action* set → RESET
- Keep alternating to ensure fair payoffs

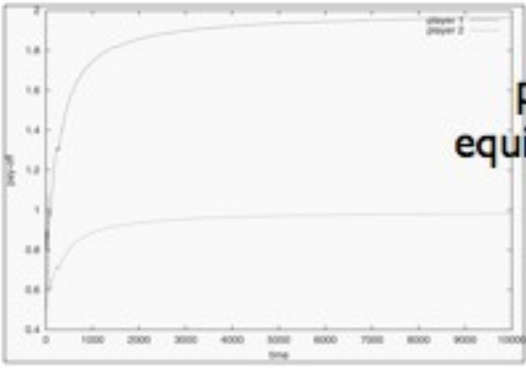
ESRL in conflicting interest games

Battle of the sexes

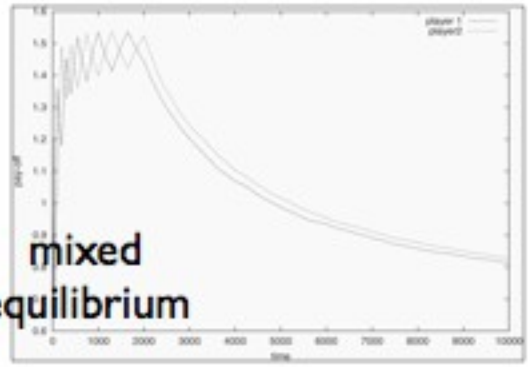
Player 2

Player 1

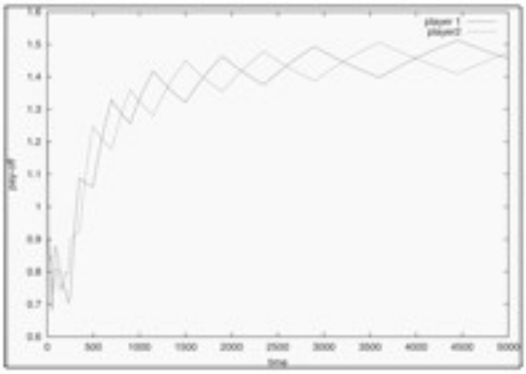
	B	S
B	2,1	0,0
S	0,0	1,2



pure equilibrium

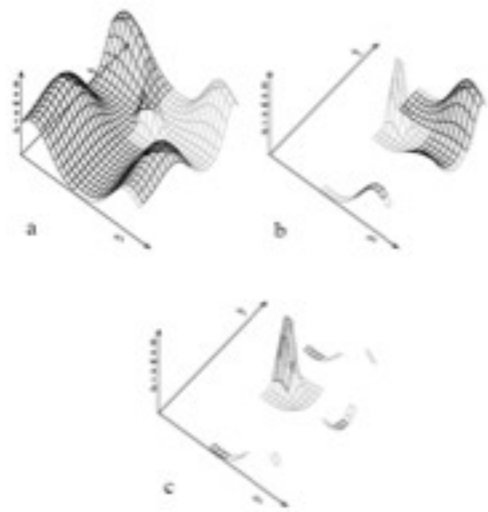


mixed equilibrium



ESRL

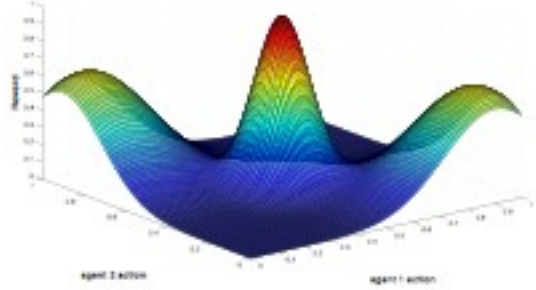
ESRL in Continuous Settings

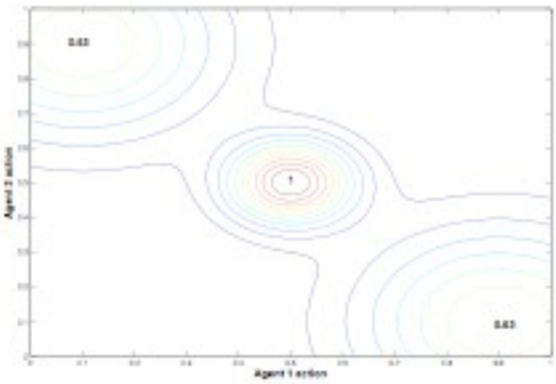


- Apply the same exploration / synchronization idea
- In continuous action spaces it makes no sense to exclude a single action
- We identify basin of attraction for learning outcomes, and eliminate entire region from the action space

Rodriguez et al., 2013a

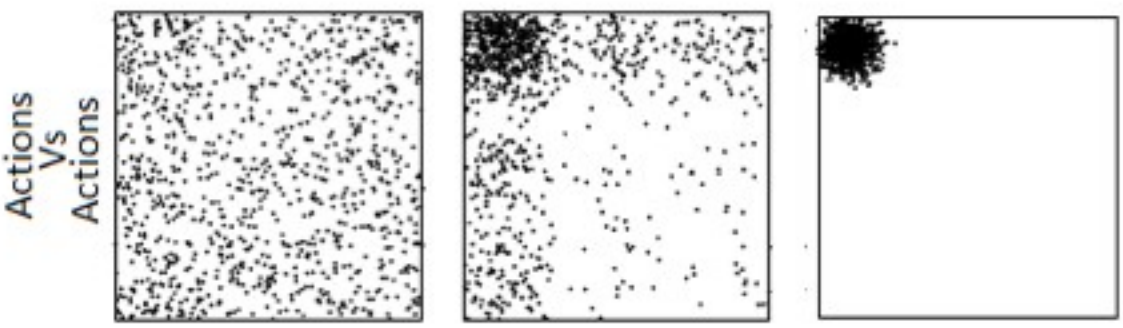
Example Game





- 2 player continuous action game
- 3 local optima
- global optimum has smallest basin of attraction

Identifying the Basin of Attraction

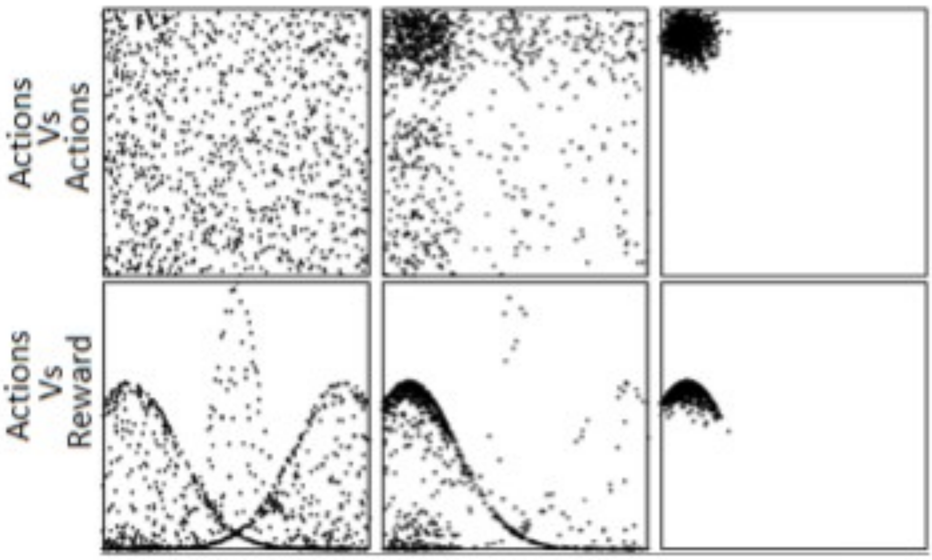


Initially learners explore randomly

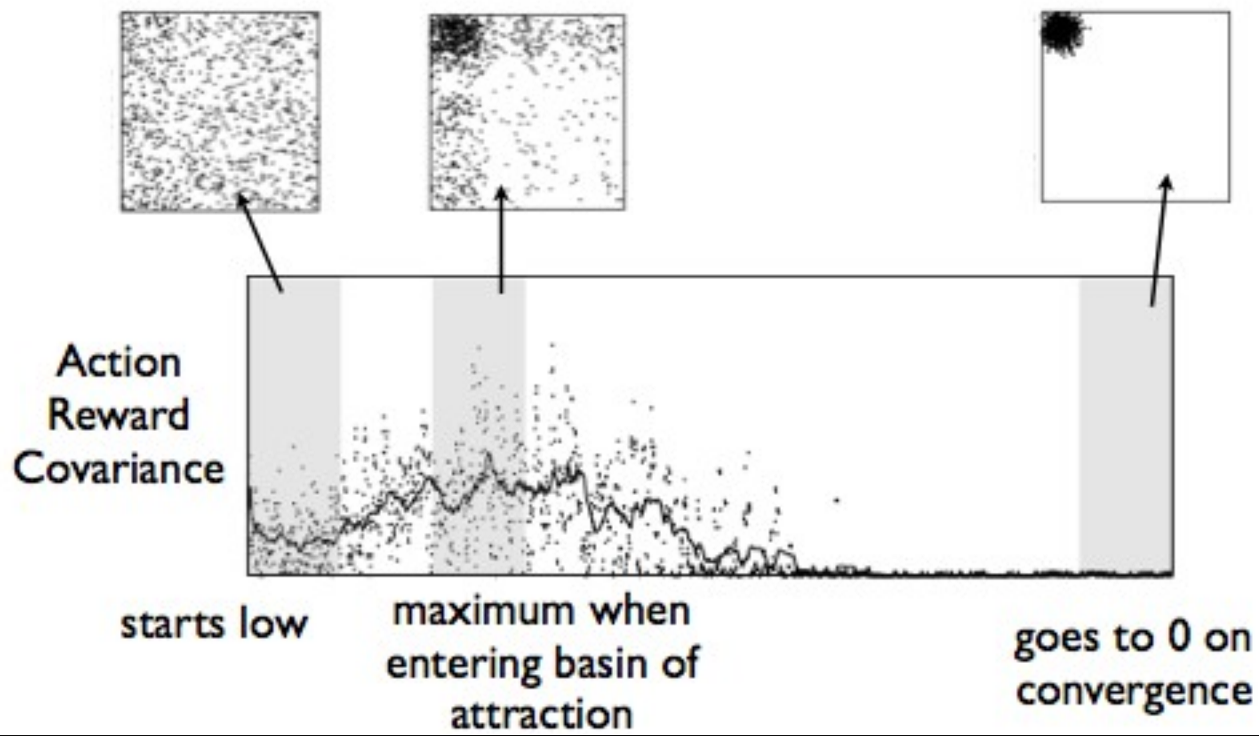
Learners enter basin of attraction

Learners converge to attractor

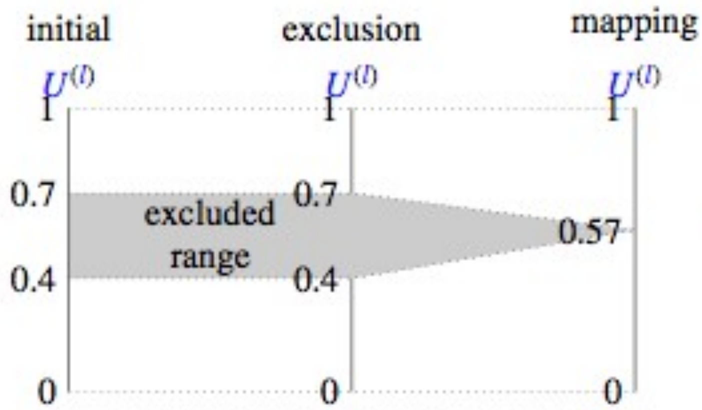
Rewards During Learning



Action - Reward Covariance

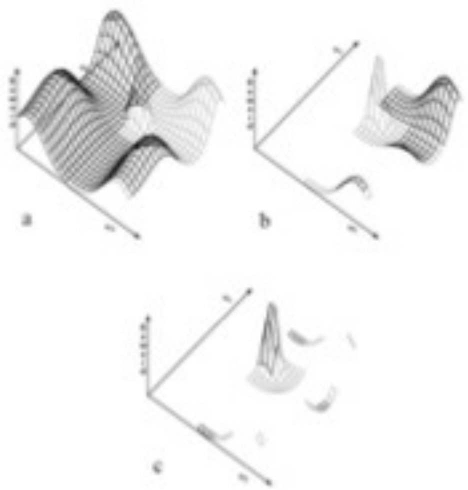


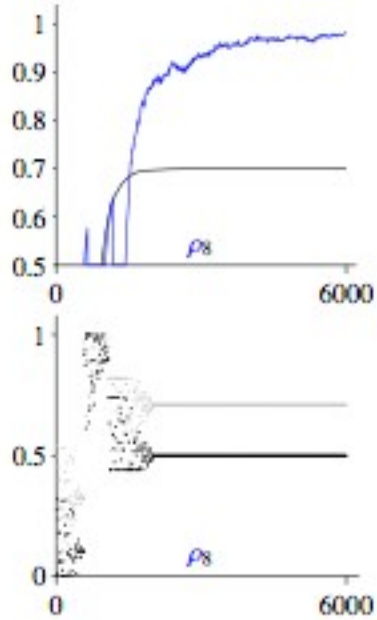
Eliminate Action Range



- Use Covariance to identify action range
- Delete range from CARLA PDF
- Renormalize PDF

Results





Rewards

Actions

Conclusions

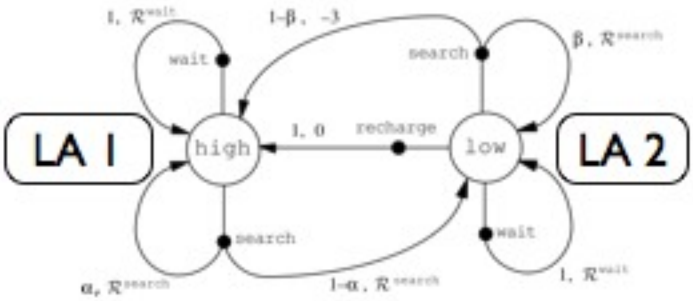
- Learning automata formalisms can also be used in continuous action games
- Discrete coordination mechanisms can be extended to continuous case

Multi-agent Reinforcement Learning in Continuous State Spaces

CoMo

Vrije Universiteit Brussel

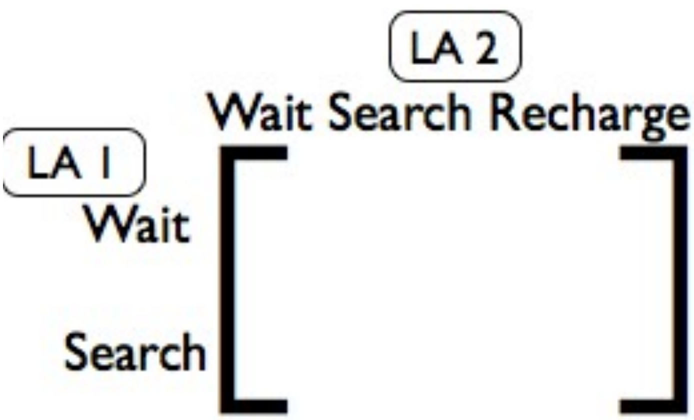
Interconnected Learning Automata (ILA)



- Actor-Critic architecture
- LA is assigned to each state
- LA is updated using estimate of accumulated reward (state value) under current policy

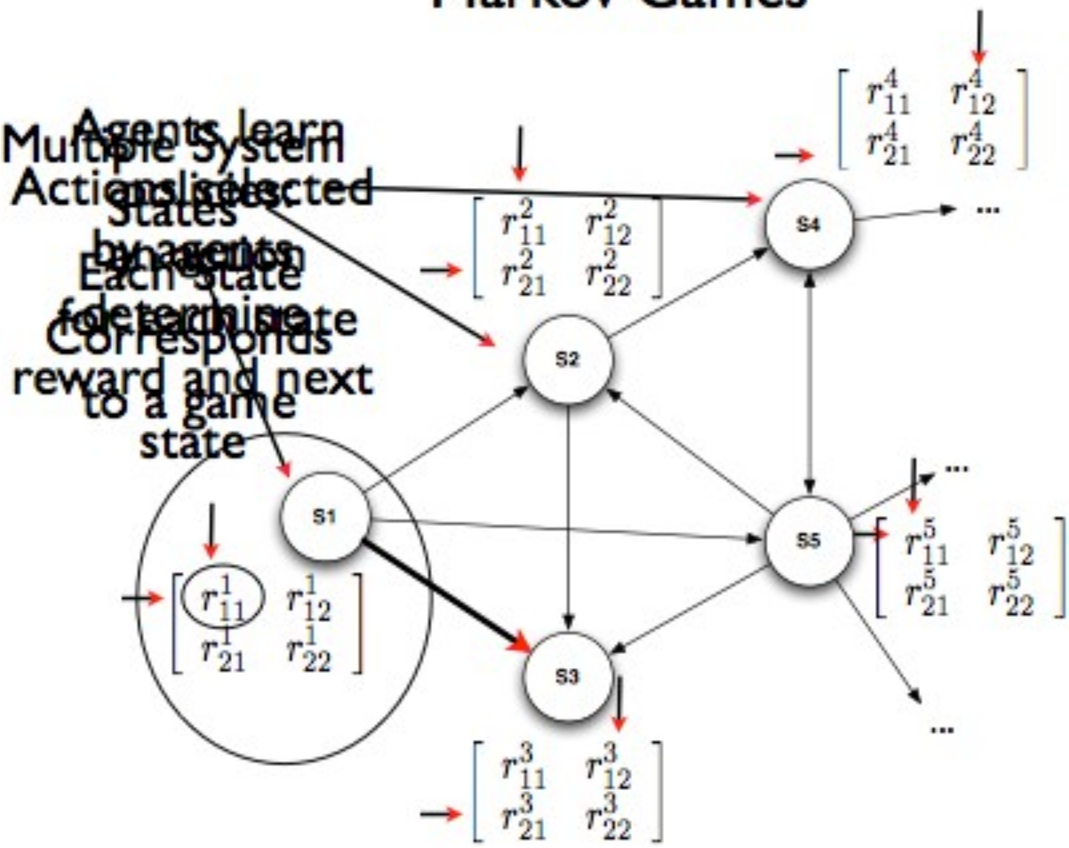
Witten, 1977
Wheeler & Narendra, 1986

ILA Analysis



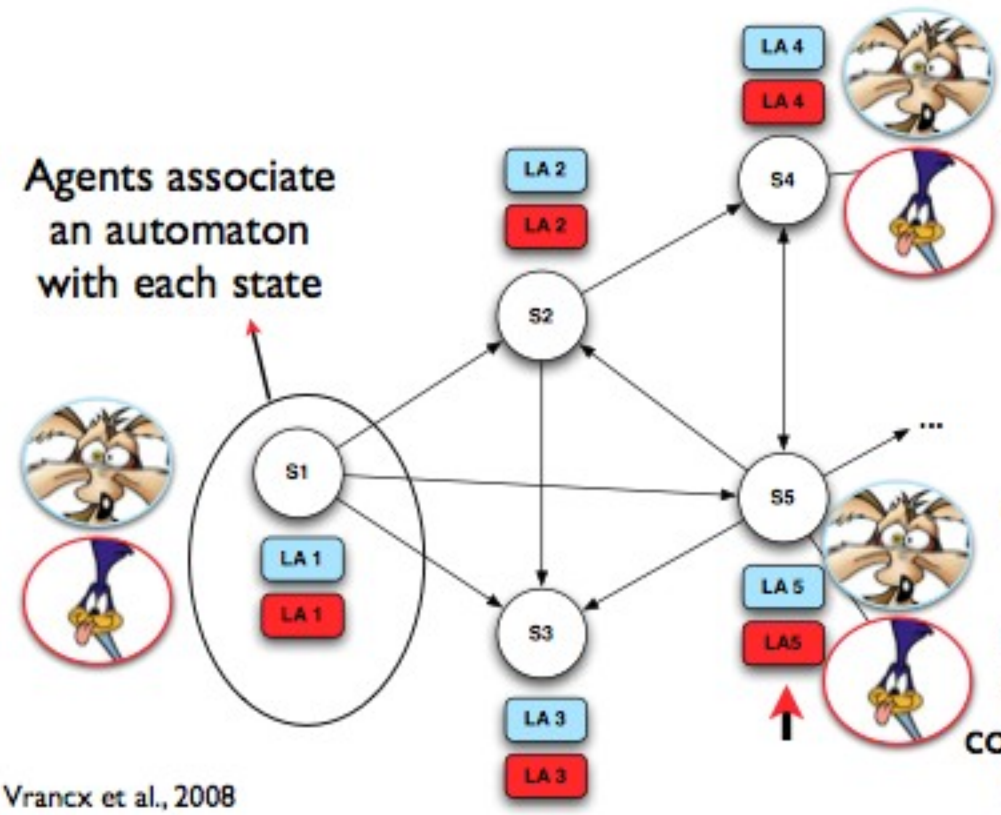
- Behavior of automata can be approximated by a game
- Equilibria of game represent optimal policies
- Game does not need to be explicitly calculated

Markov Games



MG-ILA Algorithm

Agents associate an automaton with each state

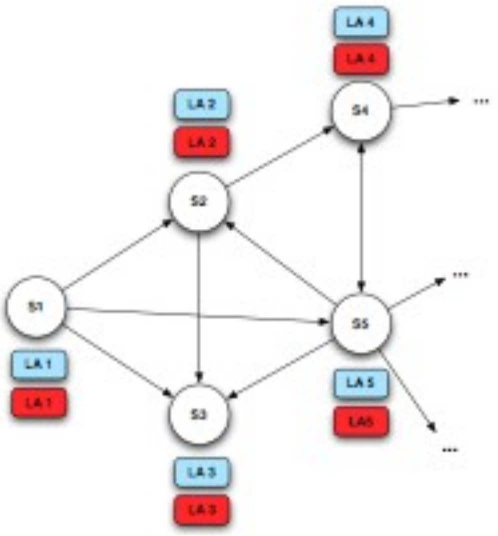


Agents keep track of rewards and state and update automata

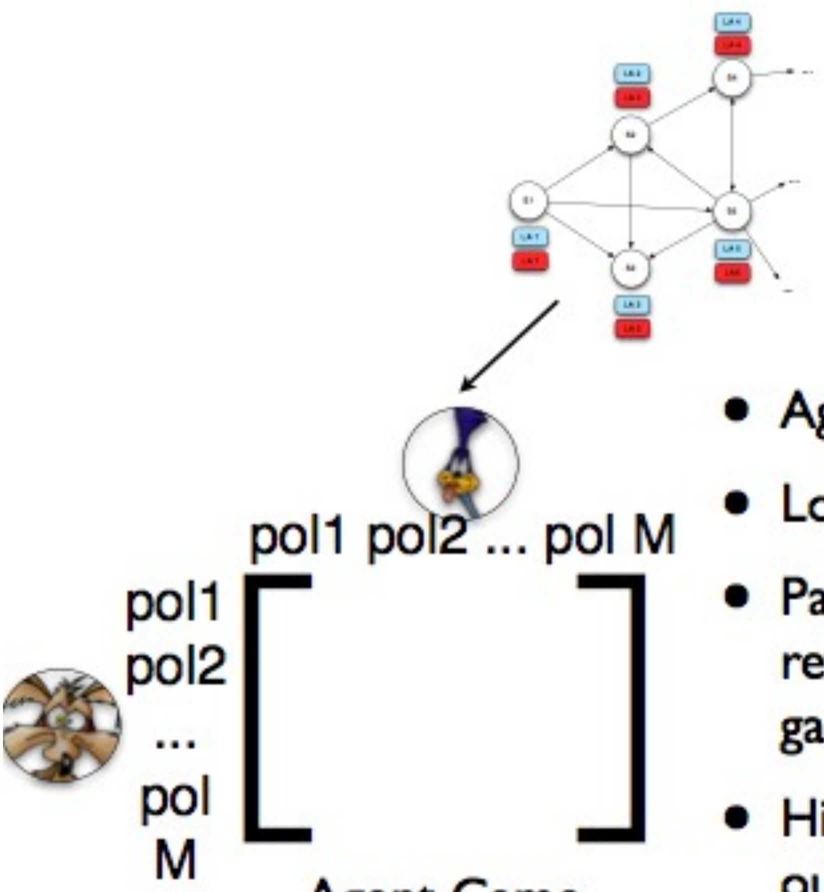
When the system is in a state, agents use corresponding automata

Vrancx et al., 2008

MG-ILA Analysis

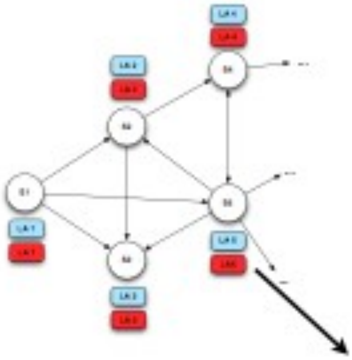


- Approximate learning behaviour by 2 games
- High level and low level view
- High level: agent interactions (Markov game)
- Low level: automata interactions
- Link both views

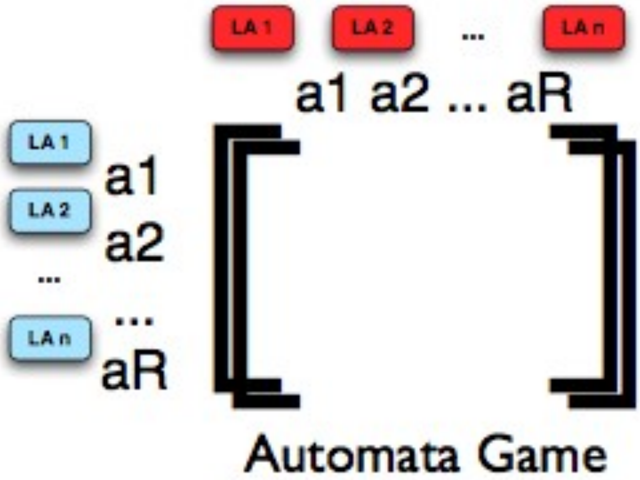


Agent Game

- Agents are players
- Look at agent Policies
- Payoff are expected rewards in Markov game
- High level view of outcome



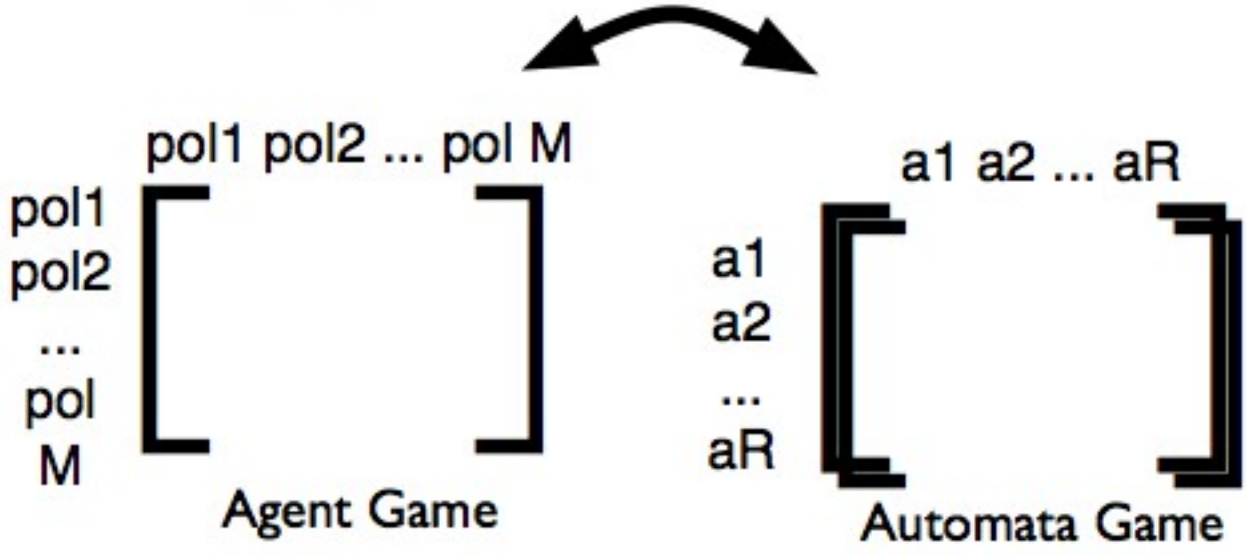
- Automata are players
- Each player selects action for 1 agent, 1 state
- Payoff are expected rewards in Markov game
- Low level view of interactions



Agents using automata find equilibrium between policies

Equilibria in both games correspond

Learning automata find equilibrium in automata game

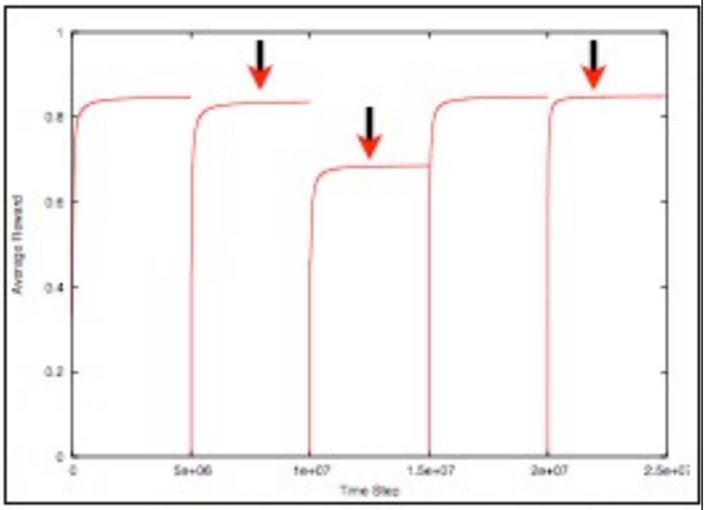


Coordinated Exploration

Coordination mechanisms like ESRL can be used to find global optimum / achieve fair payoffs

$$\begin{bmatrix} \mathbf{0.85} & 0.76 & 0.61 & 0.18 \\ 0.48 & 0.75 & 0.30 & 0.33 \\ 0.82 & \mathbf{0.84} & 0.27 & 0.32 \\ 0.43 & 0.30 & \mathbf{0.69} & 0.44 \\ 0.31 & 0.33 & 0.36 & 0.50 \\ 0.32 & 0.33 & 0.51 & \mathbf{0.54} \end{bmatrix}$$

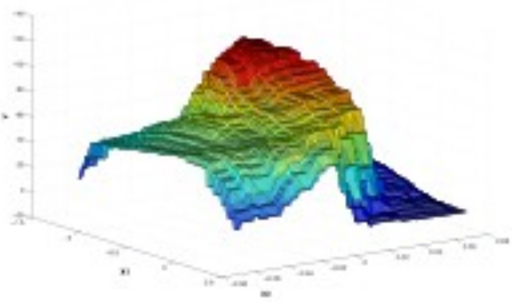
Sub-optimal Equilibria can occur



algorithm may converge to sub-optimal points

Vrancx et al., 2007

Learning in Continuous MDPs



- State and Action space are continuous
- Exact tabular representations are no longer possible
- Approximation is needed to represent policies and value functions
- Approximate TD-algorithms (Q-learning/ SARSA) typically discretize action set to find greedy policy

Linear Approximation

Value function approximation:

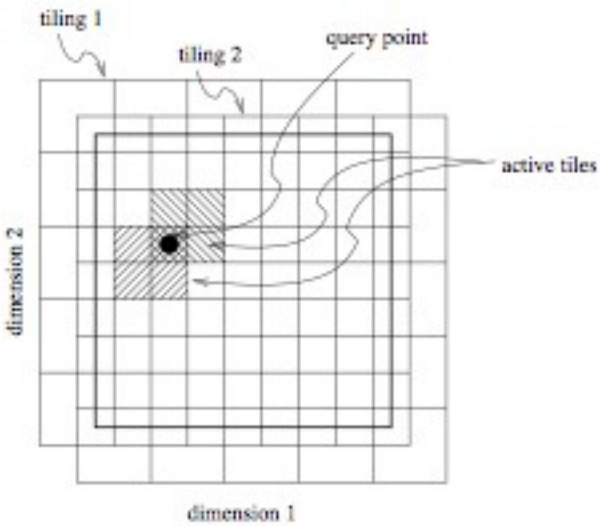
$V(x) = \phi(x)^T * \theta_V$

Policy approximation:

$\pi(x) = \phi(x)^T * \theta_u$

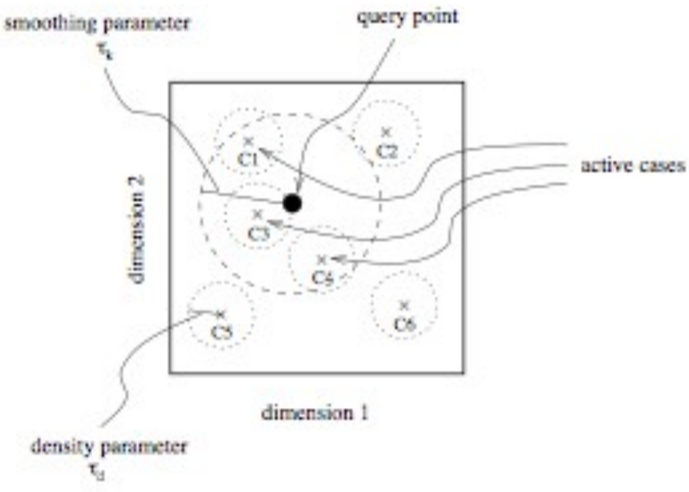
- $\phi(x)$ are basis functions / features of state x
- θ are learnt parameters
- approximation is linear in state features
- $\phi(x)$ can be non-linear functions of state variables

Tile-Coding



- State space is covered with overlapping grids
- $\phi(x)$ is binary vector with 1 value for each tile
- In each grid only the tile in which state x falls is active
- $\phi(x)$ is 1 for active tiles, 0 else

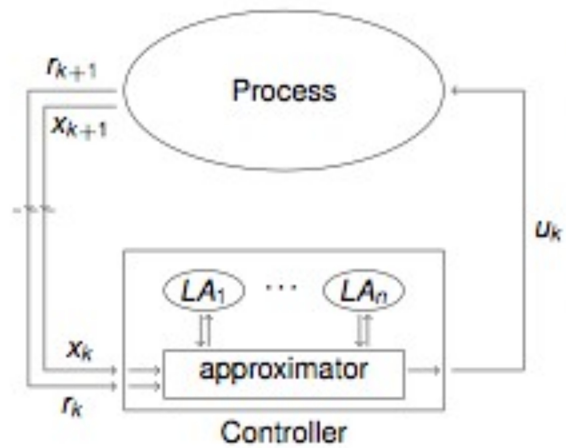
Kernel Based Approximation



$$K(d) = e^{(d^2/\tau_k^2)}$$

- Select sample states C1,C2,...,Cn
- For state x: select nearest neighbors based on distance d(x,Ci)
- ϕ(x)[i] is K(d(q,Ci)) if d(q,Ci) < τ_k, 0 else.
- K is kernel function (typically Gaussian)
- Can also be used instance based (add centers as needed)

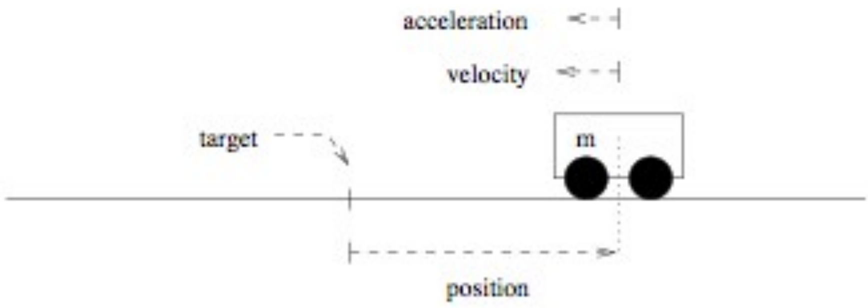
CARLA learning in Continuous MDPs



- Same idea as ILA algorithm
- Set of CARLA learn policy (actor)
- Critic learns value of current policy
- Approximator is necessary to represent policy & values
- Continuous Action Game between CARLA with values as rewards

Rodriguez et al., 2013b

Double Integrator

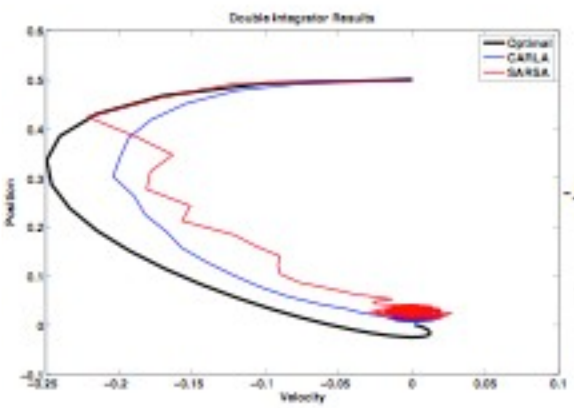


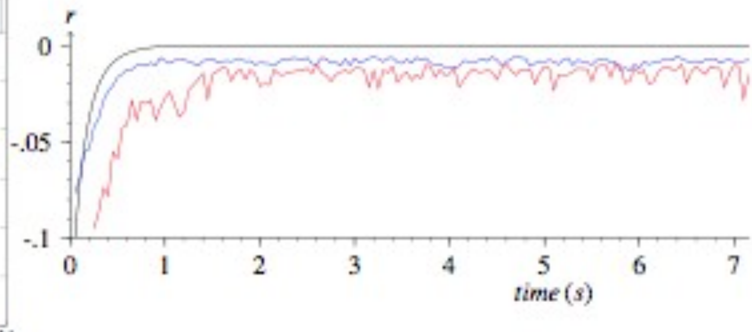
$x_t = [v_t; p_t]$ $u, v, p \in [-1, 1]$

$$\mathbf{x}_{t+1} = \begin{bmatrix} v_{t+1} \\ p_{t+1} \end{bmatrix} = \begin{bmatrix} v_t \\ p_t \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} v_t \\ p_t \end{bmatrix} \Delta t + \begin{bmatrix} 1 \\ 0 \end{bmatrix} [a] \Delta t = \mathbf{x}_t + A\mathbf{x}_t \Delta t + B\mathbf{u}_t \Delta t$$

75

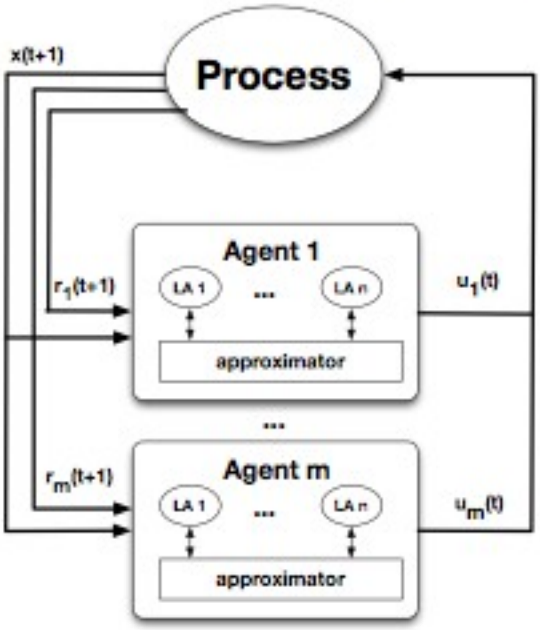
Results: CARLA + tile coding





76

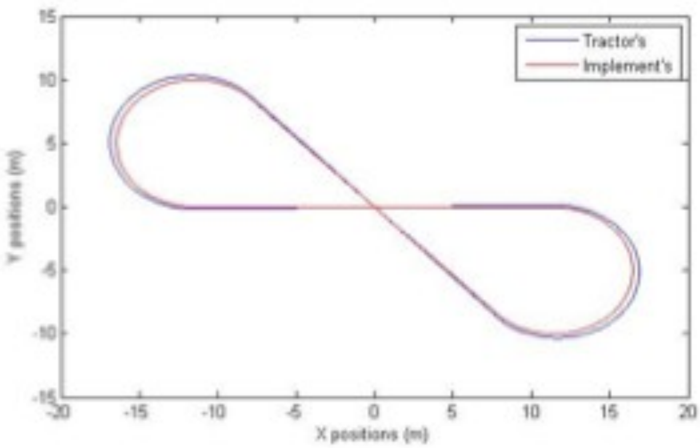
Continuous Markov Games



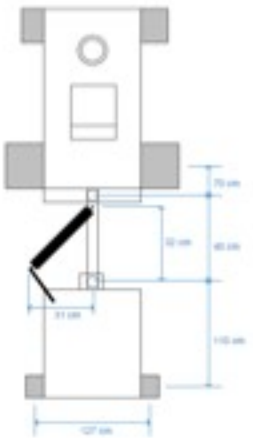
- Idea can also be extended to Continuous Markov Games
- Each agent now uses set of CARLA + approximator

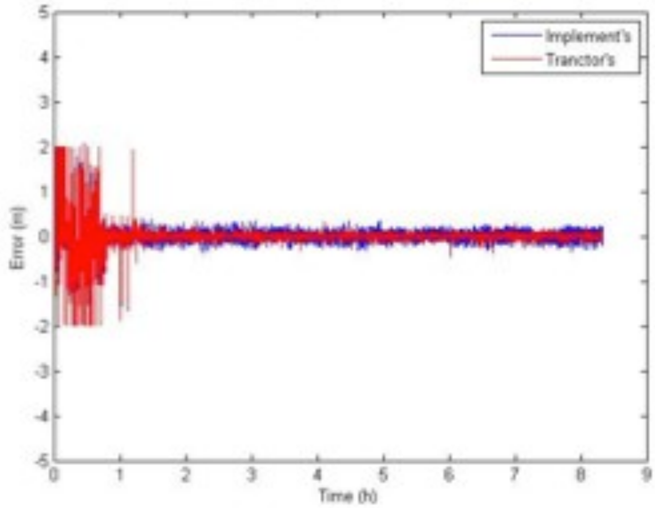
Rodriguez et al., 2013b

Example: Autonomous tractor



Results





Conclusions

- Learning automata can be used as basic building blocks for more complex multi-state, multi-agent algorithms
- Game theoretic methods can still be used to analyze algorithms
- Methods can also be extended to continuous state-action spaces

References

[Kok et al, 2005] J. Kok, P. 't Hoen, B. Bakker, and N. Vlassis. Utile coordination: Learning interdependencies among cooperative agents. In Proceedings of the IEEE Symposium on Computational Intelligence and Games (CIG05), pages 29–36, 2005.

[Melo & Veloso, 2009] F. Melo and M. Veloso. Learning of coordination: Exploiting sparse interactions in multiagent systems. In Proceedings of the 8th International Conference on Autonomous Agents and Multi-Agent Systems, pages 773–780, 2009.

[De Hauwere et al, 2009] Y-M. De Hauwere, P. Vrancx, A. Nowé, Multi-layer learning and knowledge transfer in MAS, In proceedings of the 7th European Workshop on Multi-Agent Systems, Ayia Napa, Cyprus, 2009.

[De Hauwere et al, 2010] Y-M. De Hauwere, P. Vrancx, A. Nowé, Learning Multi-Agent State Space Representations. In Proceedings of the 9th International Conference on Autonomous Agents and Multi-Agent Systems, pages 715–722, 2010.

[De Hauwere et al, 2011] Y-M. De Hauwere, P. Vrancx, A. Nowé, Solving sparse delayed coordination problems in multi-agent reinforcement learning, Chapter in Adaptive Agents and Multi-agent Systems V, Springer-Verlag, 2011

[Howell,1997] Howell, M.N., Frost, G.P., Gordon, T.J. & Wu, Q.H. Continuous action reinforcement learning applied to vehicle suspension control. Mechatron- ics. xvii, 2, 3, 23, 24, 95,1997

[Rodriguez,2011] A. Rodriguez, R. Grau, and A. Nowe. CONTINUOUS ACTION REINFORCEMENT LEARNING AUTOMATA. Performance and convergence. 3rd International Conference on Agents and Artificial Intelligence, pages 473–478. SciTePress, 2011.

[Rodriguez,2012] Rodriguez, A., Vrancx, P., Grau, R., & Nowé, A... Learning Approach to Coordinate Exploration with Limited Communication in Continuous Action Games. Proceedings ALA2012 Workshop (2012)

[Rodriguez,2013a] Rodriguez, A., Vrancx, P., Grau, R., & Nowé, A.. .A Reinforcement Learning Approach to Coordinate Exploration with Limited Communication in Continuous Action Games. Knowledge Engineering Review (in Press)

[Rodriguez,2013b] Rodriguez, A., Reinforcement Learning Approach to Coordinate Exploration with Limited Communication in Continuous Action Games. PhD thesis. Vrije Universiteit Brussel (2013)

[Santharam, 1994] Santharam, G and Sastry, PS and Thathachar, MAL. Continuous action set learning automata for stochastic Optimization. In: Journal of the Franklin Institute, 331 (5). pp. 607-628, 1994

[Verbeeck, 2004] Verbeeck, K. Coordinated Exploration in Multi-Agent Reiforcement Learning. PhD thesis, Vrije Universiteit Brussel, 2004

[Vrancx et al, 2008] Vrancx, P., Verbeeck, K., & Nowé, A.. . Decentralized Learning in Markov Games. (F. L. Lewis, Liu, D., & Lendaris, G. G.)IEEE Transactions on Systems, Man and Cybernetics (Part B: Cybernetics), 38, 976-81 (2008)

[Vrancx et al, 2007] Vrancx, P., Verbeeck, K., & Nowé, A.. . Optimal Convergence in Multi-Agent MDPs. Lecture Notes in Computer Science, Knowledge-Based Intelligent Information and Engineering Systems (KES 2007), 4694, (2007)

[Vrancx et al, 2011] Vrancx, P., De Hauwere, Y. - M., & Nowé, A.. . Transfer Learning for Multi-agent Coordination. In International Conference on Agents and Artificial Intelligence (ICAART). (2011)

[Wheeler & Narendra, 1986] Wheeler Jr., R. & Narendra, K. Decentralized learning in finite markov chains. IEEE Transactions on Automatic Control, 31, 519–526. 37, 75, 1986

[Witten, 1997] Witten, I.H .An adaptive optimal controller for discrete-time Markov environments. Information and Control, 34, 286–295. 37, 75, 1977