

Tutorial@EWRL 08: Multi-Automata Learning

Ann Nowé, Katja Verbeeck, Peter Vrancx

CoMo, Vrije Universiteit Brussel, Belgium

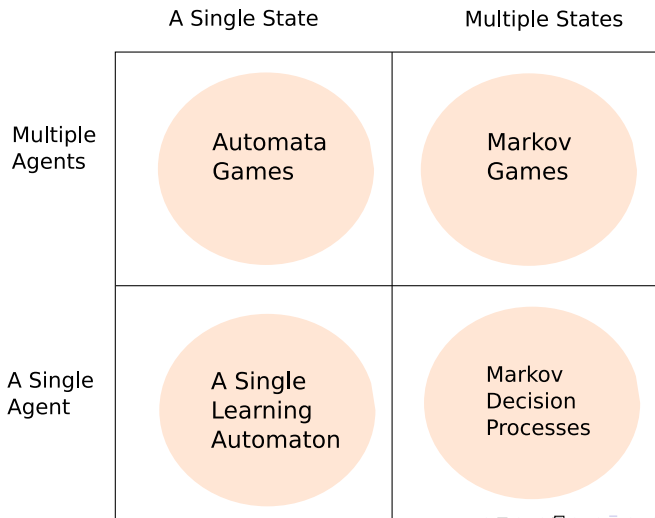
KaHo-Sint Lieven Gent, KUL, Belgium

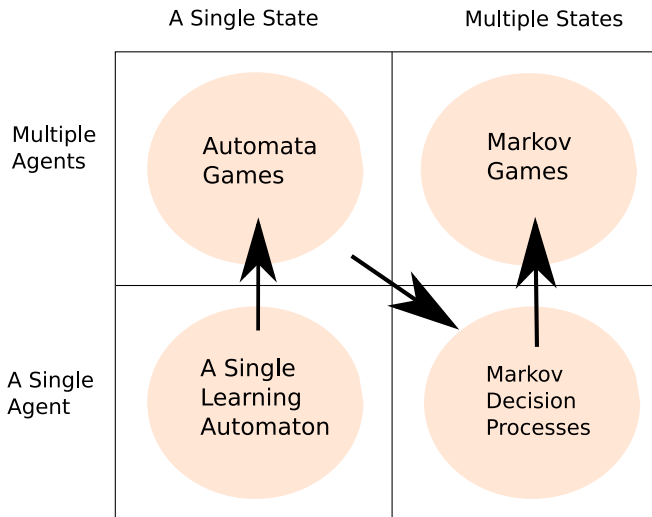
July 1st 2008

Tutorial Overview

- Part 1: An introduction to Learning Automata
- Part 2: Learning Automata Games
- Part 3: Decentralized Learning in MDP's
- Part 4: Decentralized Learning in Markov Games

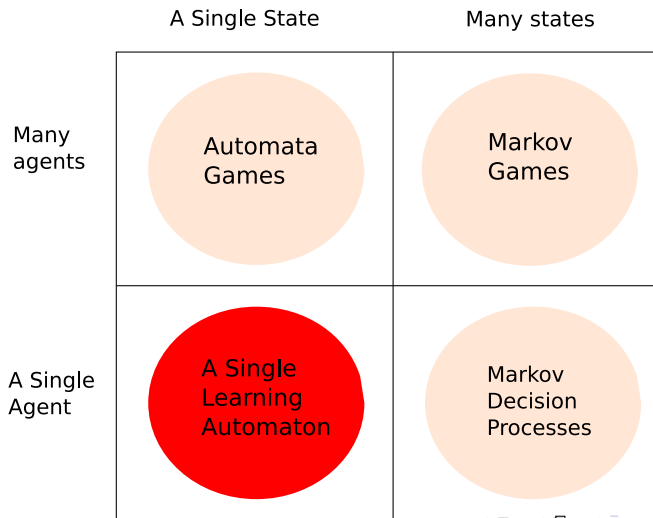
2 Dimensions: # states and # agents





- 1 Part 1
 - Definition
 - First Examples: Tsetlin Automata
 - Reward-Penalty Schemes
 - Generalizations
- 2 Part 2
 - Automata Games
 - Exploration vs Exploitation
- 3 Part 3
 - Decentralized learning in MDPs
 - The Link with Ant Algorithms
- 4 Part 4
 - Learning in Markov Games
 - MMDPs
 - Partial Observability
 - Applications
- 5 To Conclude

Part 1



A Qualitative Definition

(Narendra, Thathachar 1989)

*A fusion of the work of **psychologists** in modeling observed behavior (stimulus-response theory of Estes (1959)), the efforts of **statisticians** to model the choice of experiments based on past observations, the attempts of **operation researchers** to implement optimal strategies in the context of the 2-armed bandit problem and the endeavors of **systems theorists** to make rational decisions in random environments.*

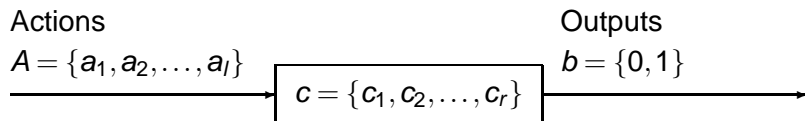
A Qualitative Definition

(Narendra, Thathachar 1989)

Simple decision Making Devices

*A finite number of **actions** can be performed in a **random** environment. When a specific action is performed the environment provides a random **response** which is either **favorable** or **unfavorable**. The objective in the design of the automaton is to determine how the choice of the action at any stage should be guided by past actions and responses. The important point to note here is that the decisions must be made with **little knowledge** concerning the nature of the environment.*

A Formal Definition: The Environment



$b = 0$ identifies a favorable response or success,
 $b = 1$ identifies a failure

$$Pr(b(n) = 1 | a(n) = a_i) = c_i \quad (i = 1, \dots, l)$$

Stationarity vs Non-stationarity

Definition

An environment is stationary when, corresponding to each action the probability of obtaining a favorable response is independent of time.

Stationarity vs Non-stationarity

Definition

An environment is stationary when, corresponding to each action the probability of obtaining a favorable response is independent of time.

Types of Non-stationarity:

Stationarity vs Non-stationarity

Definition

An environment is stationary when, corresponding to each action the probability of obtaining a favorable response is independent of time.

Types of Non-stationarity:

- 1 $c_i(n)$ is periodic

Stationarity vs Non-stationarity

Definition

An environment is stationary when, corresponding to each action the probability of obtaining a favorable response is independent of time.

Types of Non-stationarity:

- 1 $c_i(n)$ is periodic
- 2 $c_i(n)$ is slow-varying

Stationarity vs Non-stationarity

Definition

An environment is stationary when, corresponding to each action the probability of obtaining a favorable response is independent of time.

Types of Non-stationarity:

- 1 $c_i(n)$ is periodic
- 2 $c_i(n)$ is slow-varying
- 3 $c_i(n)$ is a random variable that depends on the actions of the automaton

Stationarity vs Non-stationarity

Definition

An environment is stationary when, corresponding to each action the probability of obtaining a favorable response is independent of time.

Types of Non-stationarity:

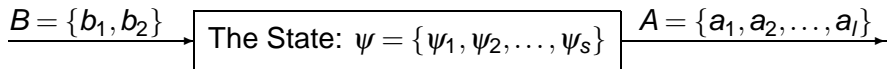
- 1 $c_i(n)$ is periodic
- 2 $c_i(n)$ is slow-varying
- 3 $c_i(n)$ is a random variable that depends on the actions of the automaton
- 4 multi-automata environment: changes in the actions chosen by other automata

A Formal Definition: The Automaton

Transition Function: $\mathcal{F} : \psi \times B \rightarrow \psi$

Input Set B

$$B = \{b_1, b_2\}$$



Output Set A

$$A = \{a_1, a_2, \dots, a_l\}$$

Output Function: $\mathcal{G} : \psi \rightarrow a$

a deterministic automaton: \mathcal{F} and \mathcal{G} are deterministic

a stochastic automaton: \mathcal{F} or \mathcal{G} are stochastic

State and Action Probabilities

$$\alpha_j(0) = Pr\{\psi(0) = \psi_j\}$$

$$\alpha_j(n) = Pr\{\psi(n) = \psi_j | b(0), \dots, b(n-1)\}$$

$$\alpha_j(1) = Pr\{\psi(1) = \psi_j | b(0)\}$$

$$= \sum_{i=1}^S Pr\{\psi(1) = \psi(j) | \psi(0) = \psi(i), b(0)\} Pr\{\psi(0) = \psi(i)\}$$

$$= \sum_{i=1}^S f_{ij}^{b(0)} \alpha_i(0)$$

and thus in vector form: $\alpha(1) = F^T(b(0))\alpha(0)$ or

$$\alpha(n) = F^T(b(n-1))F^T(b(n-2))\dots F^T(b(0))\alpha(0)$$

State and Action Probabilities

Using the action probability vector $p(n)$ with

$$p_i(n) = Pr\{a(n) = a_i | b(0), \dots, b(n-1)\}$$

we get:

$$p(n) = G^T \alpha(n)$$

State and Action Probabilities

Using the action probability vector $p(n)$ with

$$p_i(n) = Pr\{a(n) = a_i | b(0), \dots, b(n-1)\}$$

we get:

$$p(n) = G^T \alpha(n)$$

→ Given a deterministic input sequence, action probabilities can be calculated!

State and Action Probabilities

Using the action probability vector $p(n)$ with

$$p_i(n) = Pr\{a(n) = a_i | b(0), \dots, b(n-1)\}$$

we get:

$$p(n) = G^T \alpha(n)$$

→ Given a deterministic input sequence, action probabilities can be calculated!

→ Input sequences are not deterministic but random!

(Narendra, Thathachar 1989)

Consider the case where you have to choose between 2 restaurants-Simeone's and Delmonaco's. Let us assume that at stage $(n-1)$ you chose to go to Simeone's. If your decision at stage n , is to return to Simeone's if the food was good at stage $(n-1)$, and to switch to Delmonaco's if it was poor, then your decision corresponds to that of a **deterministic automaton**. If your decision is to go to the same restaurant if the food was good on the previous visit, but to toss a coin to choose which of the two restaurants to visit if it was not, then you are acting like a **fixed structure automaton**. Finally, if you update the probs of going to the two restaurants at every stage based on the outcome of the previous visit your decision rule corresponds to a **variable-structure automaton**.

Random inputs and Variable structure Automata

Alternative view:

The action probability vector describes a continuous state, discrete-time, Markov process on the r -dimensional simplex:

$$\{p \mid \sum_{i=1}^r p_i = 1; 0 \leq p_i \leq 1\}$$

and when $p(n+1)$ depends on $p(n)$ but not explicitly on n , $\{p(n)\}_{n \geq 0}$ is a homogenous Markov process.

Norms of Behavior

Average penalty :

$$\begin{aligned}M(n) &= E[b(n) = 1 | p(n)] = \sum_{i=1}^r [b(n) = 1 | a(n) = a_i] \\ &= \sum_{i=1}^r c_i p_i(n)\end{aligned}$$

For the pure-chance automaton $M(n) = M_0 = \frac{1}{n} \sum_{i=1}^r c_i$

A learning automaton is said to be:

Norms of Behavior

Average penalty :

$$\begin{aligned} M(n) &= E[b(n) = 1 | p(n)] = \sum_{i=1}^r [b(n) = 1 | a(n) = a_i] \\ &= \sum_{i=1}^r c_i p_i(n) \end{aligned}$$

For the pure-chance automaton $M(n) = M_0 = \frac{1}{n} \sum_{i=1}^r c_i$

A learning automaton is said to be:

- ① **expedient** if $\lim_{n \rightarrow \infty} E[M(n)] < M_0$

Norms of Behavior

Average penalty :

$$\begin{aligned} M(n) &= E[b(n) = 1 | p(n)] = \sum_{i=1}^r [b(n) = 1 | a(n) = a_i] \\ &= \sum_{i=1}^r c_i p_i(n) \end{aligned}$$

For the pure-chance automaton $M(n) = M_0 = \frac{1}{n} \sum_{i=1}^r c_i$

A learning automaton is said to be:

- 1 **expedient** if $\lim_{n \rightarrow \infty} E[M(n)] < M_0$
- 2 **optimal** if $\lim_{n \rightarrow \infty} E[M(n)] = \min_i c_i$

Norms of Behavior

Average penalty :

$$\begin{aligned} M(n) &= E[b(n) = 1 | p(n)] = \sum_{i=1}^r [b(n) = 1 | a(n) = a_i] \\ &= \sum_{i=1}^r c_i p_i(n) \end{aligned}$$

For the pure-chance automaton $M(n) = M_0 = \frac{1}{n} \sum_{i=1}^r c_i$

A learning automaton is said to be:

- 1 **expedient** if $\lim_{n \rightarrow \infty} E[M(n)] < M_0$
- 2 **optimal** if $\lim_{n \rightarrow \infty} E[M(n)] = \min_i c_i$
- 3 **ε -optimal** if $\lim_{n \rightarrow \infty} E[M(n)] < \min_i c_i + \varepsilon$

Norms of Behavior

Average penalty :

$$\begin{aligned} M(n) &= E[b(n) = 1 | p(n)] = \sum_{i=1}^r [b(n) = 1 | a(n) = a_i] \\ &= \sum_{i=1}^r c_i p_i(n) \end{aligned}$$

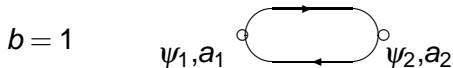
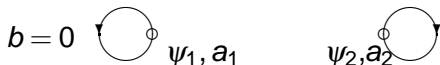
For the pure-chance automaton $M(n) = M_0 = \frac{1}{n} \sum_{i=1}^r c_i$

A learning automaton is said to be:

- 1 **expedient** if $\lim_{n \rightarrow \infty} E[M(n)] < M_0$
- 2 **optimal** if $\lim_{n \rightarrow \infty} E[M(n)] = \min_i c_i$
- 3 **ε -optimal** if $\lim_{n \rightarrow \infty} E[M(n)] < \min_i c_i + \varepsilon$
- 4 **absolutely expedient** if $E[M(n+1) | p(n)] < M(n)$

$L_{2,2}$, Tsetlin, 1961

Performs whatever action it was using earlier as long as the response is positive, but changes to the other action otherwise.



Behavior of $L_{2,2}$

The resulting Markov chain is **ergodic**, the limiting probabilities are:

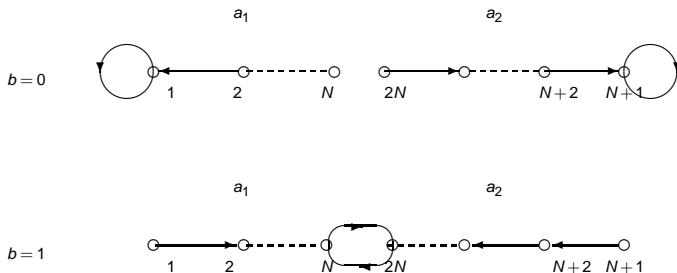
$$\alpha_1 = \frac{c_1}{c_1 + c_2}, \quad \alpha_2 = \frac{c_2}{c_1 + c_2}$$

and thus $L_{2,2}$ is **expedient** ($c_1 \neq c_2$),

$$M(L_{2,2}) = \frac{2c_1c_2}{c_1 + c_2} < M_0 = \frac{c_1 + c_2}{2}$$

Adding Memory, $L_{2N,2}$, Tsetlin

Keeps an account of the number of successes and failures and when the number of failures exceeds a number N , it switches the action.



If $\min_i \{c_i\} \leq \frac{1}{2}$ then $\lim_{N \rightarrow \infty} M(L_{2N,2}) = \min(c_1, c_2)$ (conditionally ϵ -optimal)

Variable Structure Automata

- 1 evolve action probabilities (or equiv state probs)

Variable Structure Automata

- 1 evolve action probabilities (or equiv state probs)
- 2 assume each state corresponds to one distinct action ((\mathcal{G}) is the identity mapping)

Variable Structure Automata

- 1 evolve action probabilities (or equiv state probs)
- 2 assume each state corresponds to one distinct action ((\mathcal{G}) is the identity mapping)
- 3 analyzing tool: discrete-time Markov processes $\{p(n)\}_n$

Variable Structure Automata

- 1 evolve action probabilities (or equiv state probs)
- 2 assume each state corresponds to one distinct action ((\mathcal{G}) is the identity mapping)
- 3 analyzing tool: discrete-time Markov processes $\{p(n)\}_n$
- 4 define a mapping T so that : $p(n+1) = T[p(n), a(n), b(n)]$

Variable Structure Automata

- 1 evolve action probabilities (or equiv state probs)
- 2 assume each state corresponds to one distinct action ((\mathcal{G}) is the identity mapping)
- 3 analyzing tool: discrete-time Markov processes $\{p(n)\}_n$
- 4 define a mapping T so that : $p(n+1) = T[p(n), a(n), b(n)]$
- 5 characteristics:
 - asymptotic behavior: expedient, (ϵ) -optimal
 - nature of the mapping: linear, nonlinear, hybrid
 - properties of the Markov process: ergodic, nonergodic

Reward-Penalty with binary feedback

$$b = 0 \begin{cases} p_i(t+1) = p_i(t) + \lambda_1(1 - p_i(t)) \\ \text{if action } i \text{ was taken at time step } t \\ p_j(t+1) = (1 - \lambda_1)p_j(t) \quad \forall j \neq i \end{cases}$$

$$b = 1 \begin{cases} p_i(t+1) = p_i(t) - \lambda_2 p_i(t) \\ \text{if action } i \text{ was taken at time step } t \\ p_j(t+1) = p_j(t) + \lambda_2[(I - 1)^{-1} - p_j(t)] \quad \forall j \neq i \end{cases}$$

$\lambda_1, \lambda_2 \in]0, 1[$ and I the number of actions. λ_1 = Reward parameter and λ_2 = penalty parameter

$\lambda_1 = \lambda_2$: **linear reward-penalty** (L_{R-P}), $\lambda_2 = 0$: **linear reward-inaction** (L_{R-I}),

$\lambda_2 \ll \lambda_1$ **linear reward- ϵ -penalty** ($L_{R-\epsilon P}$).

Reward-Penalty with continuous valued feedback

In the general setting of continuous values reward, the update is given by:

$$\left\{ \begin{array}{l} p_i(t+1) = p_i(t) + \lambda_1 r(t)(1 - p_i(t)) - \lambda_2(1 - r(t))p_i(t) \\ \quad \text{if action } i \text{ was taken at time step } t \\ \\ p_j(t+1) = p_j(t) - \lambda_1 r(t)p_j(t) + \lambda_2(1 - r(t))[(I - 1)^{-1} - p_j(t)] \\ \quad \forall j \neq i \end{array} \right.$$

Analytical Behavior of $(p(n))_n$

Analytical Behavior of $(p(n))_n$

- the process is **ergodic** (L_{R-P} and $L_{R-\varepsilon P}$)
 - $p(n)$ converges in distribution to a random variable p^* , whose distribution function is independent of the initial value $p(0)$.
 - Slow learning: study the stability properties of the equilibrium state of the associated ODE, and the limiting distribution is normal

Analytical Behavior of $(p(n))_n$

- the process is **ergodic** (L_{R-P} and $L_{R-\varepsilon P}$)
 - $p(n)$ converges in distribution to a random variable p^* , whose distribution function is independent of the initial value $p(0)$.
 - Slow learning: study the stability properties of the equilibrium state of the associated ODE, and the limiting distribution is normal
- the process has **absorbing states** (L_{R-I})
 - $p(n)$ converges with probability one to the set of unit vectors
 - Slow learning: prob. of convergence to the optimal action can be made as close to 1 as desired.
 - Trade-off between speed and level of accuracy.

Experimental Behavior

 L_{R-I}

λ_1	% of wrong convergence
0.9	22
0.5	12
0.3	5
0.1	0
0.01	0
0.001	0

Pursuit Algorithm

At each time step

- Estimate the expected reward for each action by keeping track of the average reward obtained so far for each action.
- Determine the current best action
- Update the probabilities of all actions :

$$p_i(n+1) = p_i(n) + \lambda(1 - p_i(n))$$

$$p_j(n+1) = p_j(n) - \lambda p_j(n)$$

Convergence speed is increased in case action set is small

Continuous actions

- initial action probability distribution $N(\mu(0), \sigma(0))$
- goal: converge to $N(x_0, 0)$ where x_0 maximizes $E[\beta_x|x]$.

Parameterized Learning Automata

(Thathachar & Phansalkar, 1995)

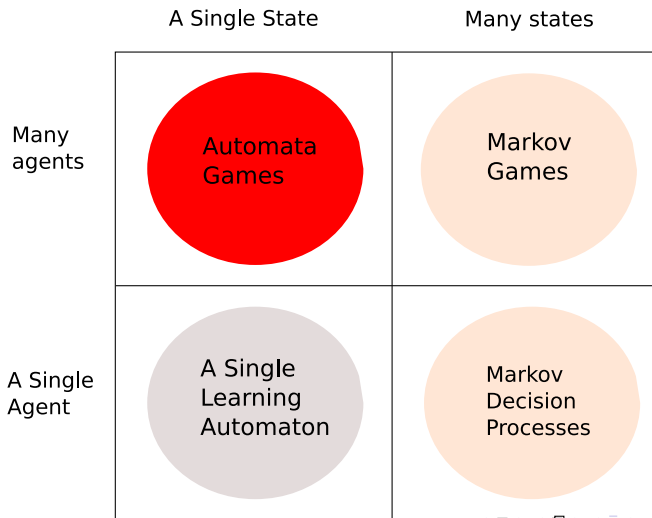
$$u_i(t+1) = u_i(t) + \lambda \beta(t) \frac{\delta \ln g}{\delta u_i}(\mathbf{u}(t), a(t)) + \lambda h'(u_i(t)) + \sqrt{\lambda} s_i(t)$$

$$g(\mathbf{u}(t), a_i) = \frac{e^{u_i(t)}}{\sum_j e^{u_j(t)}}$$

- Use parameter vector \mathbf{u} and probability generating function g , in stead of modifying probabilities directly.
- Based on REINFORCE (Williams, 1992).
- Function $h(x)$ bounds parameter values.
- Added noise $s_i(t)$ allows algorithm to escape local optima.
- Converge to global optimum in Team Games.

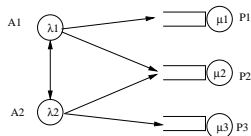
- 1 Part 1
 - Definition
 - First Examples: Tsetlin Automata
 - Reward-Penalty Schemes
 - Generalizations
- 2 Part 2
 - Automata Games
 - Exploration vs Exploitation
- 3 Part 3
 - Decentralized learning in MDPs
 - The Link with Ant Algorithms
- 4 Part 4
 - Learning in Markov Games
 - MMDPs
 - Partial Observability
 - Applications
- 5 To Conclude

Part 2



Single State, multi-agent problems

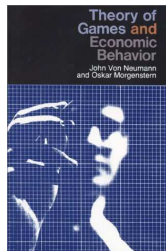
e.g. scheduling, distruted vs centralized load balancing, ...



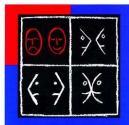
Formalisation

Normal form games from game theory (Osborne and Rubenstein, 1994)

Interactive decision-making, where the outcome for each participant or "player" depends on the actions of all.



A COURSE IN GAME THEORY



Common interest Games: a diagonal game

	#1	#2	#3
#1	1	0	0
#2	0	2	0
#3	0	0	3

The **guessing game** - a diagonal game with 2 players. The Nash equilibria are located on the diagonal of the normal form matrix.

More difficult common interest games

	a_1	a_2	a_3
a_1	11	-30	0
a_2	-30	7	6
a_3	0	0	5

	a_1	a_2	a_3
a_1	10	0	k
a_2	0	2	0
a_3	k	0	10

The **climbing game**. The Pareto optimal Nash equilibrium (a_1, a_1) is surrounded by heavy penalties.

The **penalty game**. Mis-coordination at the Pareto optimal Nash equilibria (a_1, a_1) and (a_3, a_3) is penalized with $k < 0$.

Conflicting interest games: the Battle of the Sexes

	<i>DVC</i>	<i>MI</i>
<i>DVC</i>	(2, 1)	(0, 0)
<i>MI</i>	(0, 0)	(1, 2)

Battle of the sexes game: a husband and wife want to spend their evening together, but they have conflicting interests. They independently choose between 2 movies. When both choose the same movie, a reward is given, according to their preference. This game has 2 pure and 1 mixed Nash equilibria.



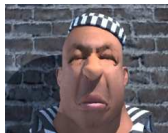
A Correlated Equilibrium

	<i>DVC</i>	<i>MI</i>
<i>DVC</i>	(2, 1)	(0, 0)
<i>MI</i>	(0, 0)	(1, 2)

What if there is a coin toss observable for both players?

	(DVC, DVC)	(DVC, MI)	(MI, DVC)	(MI, MI)
(DVC, DVC)	(2, 1)	(1, 0.5)	(1, 0.5)	(0, 0)
(DVC, MI)	(1, 0.5)	$(\frac{3}{2}, \frac{3}{2})$	(0, 0)	(0.5, 1)
(MI, DVC)	(0.5, 1)	(0, 0)	$(\frac{3}{2}, \frac{3}{2})$	(0.5, 1)
(MI, MI)	(0, 0)	(0.5, 1)	(0.5, 1)	(1, 2)

Conflicting interest Games: the Prisoners' Dilemma



	<i>Stay Silent</i>	<i>Betray</i>
<i>Stay Silent</i>	(1, 1)	(10, 0)
<i>Betray</i>	(0, 10)	(5, 5)



The Prisoner's Dilemma Game: 2 prisoners committed a crime together. The police have insufficient evidence for a conviction, and, having separated both prisoners, visit each of them to offer the same deal: if one testifies ("defects") for the prosecution against the other and the other remains silent, the betrayer goes free and the silent accomplice receives the full 10-year sentence. However, if both remain silent, both prisoners are sentenced to only one year in jail for a minor charge. If each betrays the other, each receives a five-year sentence. This game has one pure nash equilibrium which is the only solution that is not **Pareto optimal**.

Pure Conflicting interest games: Zero-sum games

	<i>head</i>	<i>tail</i>
<i>head</i>	$(1, -1)$	$(-1, 1)$
<i>tail</i>	$(-1, 1)$	$(1, -1)$

The Matching Pennies game: 2 children independently choose which side of a coin to show to the other. If they both show the same side, the first child wins, otherwise child 2 wins.

So what should the agents learn anyway?

So what should the agents learn anyway?

- Nash equilibria?

So what should the agents learn anyway?

- Nash equilibria?
- Pareto optimal solutions?

So what should the agents learn anyway?

- Nash equilibria?
- Pareto optimal solutions?
- Mixed Nash equilibria?

So what should the agents learn anyway?

- Nash equilibria?
- Pareto optimal solutions?
- Mixed Nash equilibria?
- Correlated equilibria?

So what should the agents learn anyway?

- Nash equilibria?
- Pareto optimal solutions?
- Mixed Nash equilibria?
- Correlated equilibria?
- Or a set of solutions ... ?

and what if ...

and what if ...

- Rewards are stochastic?

and what if ...

- Rewards are stochastic?
- Rewards come delayed?

and what if ...

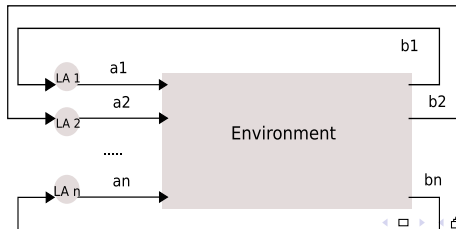
- Rewards are stochastic?
- Rewards come delayed?
- *In a multi-agent setting **coordinated exploration** is important !*

and what if ...

- Rewards are stochastic?
- Rewards come delayed?
- *In a multi-agent setting **coordinated exploration is important !***
- *Learning Automata have a **built-in exploration strategy which is very suited for MARL***

Learning Automata Games

A play $\vec{a}(t)$ of n automata is a set of strategies chosen by the automata at stage t , such that $a^j(t)$ is an element of the action set of the j th automaton. Correspondingly the outcome is now also a vector $\vec{b}(t) = (b^1(t) \dots b^n(t))$. At every time-step all automata update their probability distributions based on the responses of the environment. Each automaton participating in the game operates without information concerning the number of other participants, their strategies, actions or payoffs.



Important Result

Theorem

(Narendra and Wheeler, 1989) Players in an n -person non-zero sum game who use independently a reward-inaction update scheme with an arbitrarily small step size will always converge to a pure equilibrium point. If the game has a NE, the equilibrium point will be one of the NE.

Coordinated Exploration in Single State Games

Exploring Selfish Reinforcement Learning (ESRL, Verbeeck & Nowé 2004) Basic Idea: 2 phases

- 1 Exploration: Be Selfish
 - Independent learning
 - Local convergence to attractor point (NE, Pareto Optimal point, other)
- 2 Exploitation: Be Social
 - Exclusions: each can shrink the joint action space by excluding individual actions

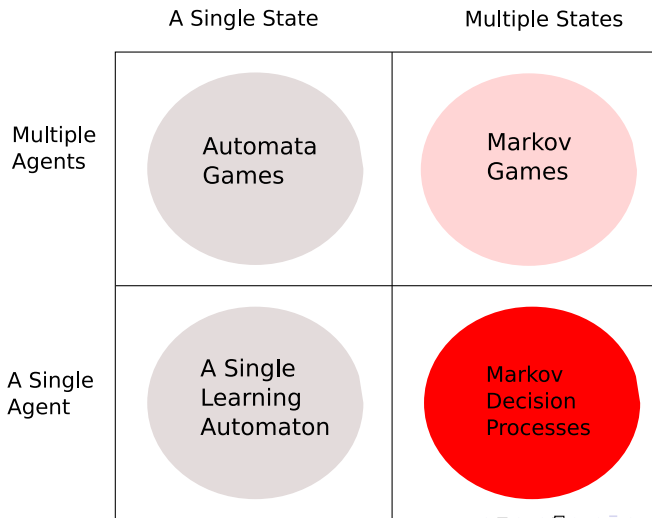
LA vs Boltzmann

MAS-RL: convergence is sensitive to the exploration strategy used by the agents.

- LAs: have built-in exploration strategy.
- Other approaches often use a Boltzman exploration strategy.

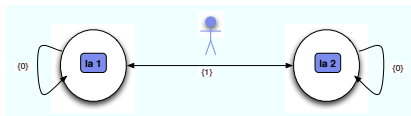
- 1 Part 1
 - Definition
 - First Examples: Tsetlin Automata
 - Reward-Penalty Schemes
 - Generalizations
- 2 Part 2
 - Automata Games
 - Exploration vs Exploitation
- 3 Part 3**
 - Decentralized learning in MDPs**
 - The Link with Ant Algorithms**
- 4 Part 4
 - Learning in Markov Games
 - MMDPs
 - Partial Observability
 - Applications
- 5 To Conclude

Part 3



Decentralized learning in MDPs (Wheeler & Narendra, 1986)

- Single agent learns policy in Markov Decision Problem
- One LA is associated with each state of the problem
- When agent reaches a state, state LA becomes active to select action.
- LA is updated when state is visited next time.



Agent tries to learn policy π to maximize the average reward over time:

$$J^\pi \equiv \lim_{l \rightarrow \infty} \frac{1}{l} E \left[\sum_{t=0}^{l-1} R^\pi(s(t), s(t+1)) \right]$$

Decentralized Learning in MDPs (2)

- LA are updated in a Monte Carlo fashion: feedback is given over whole episode once agent returns to state.
- Markov chain under each policy is assumed to be ergodic.
- Feedback given to LA in state s_i is calculated as follows:

$$\beta^i(t_i + 1) = \frac{\rho^i(t_i + 1)}{\eta^i(t_i + 1)}$$

- $\rho^i(t_i + 1)$ is the cumulative total reward generated for action a^i in state s_i
- $\eta^i(t_i + 1)$ the cumulative total time elapsed.

Decentralized Learning in MDPs (3)

Provided the Markov chain corresponding to each policy π is ergodic:

- the process will converge to a limiting stationary distribution over the states:

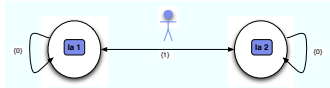
$$\alpha^\pi = \{\alpha^\pi(s_1), \dots, \alpha^\pi(s_N)\} \text{ with } \sum_i \alpha^\pi(s_i) = 1 \text{ and } \forall i: \alpha^\pi(s_i) > 0$$

- We can calculate the expected reward J^π associated with a policy π as follows:

$$J^\pi = \sum_{i=1}^N \alpha^\pi(s_i) \sum_{j=1}^N T(s_i, s_j, \pi(s_i)) R(s_i, s_j, \pi(s_i))$$

Approximating Automata Game

By viewing each policy as a play of automata we can approximate the MDP by a limiting automata game:



- Agent gets reward 1.0 for transitions $(s1,s2)$ and $(s2,s1)$, reward 0 else.
- Transitions are stochastic: 0.1 chance of performing other action

π	α^π
(0,0)	{0.5, 0.5}
(0,1)	{0.9, 0.1}
(1,0)	{0.1, 0.9}
(1,1)	{0.5, 0.5}

LA1/LA2	0	1
0	0.1	0.18
1	0.18	0.9

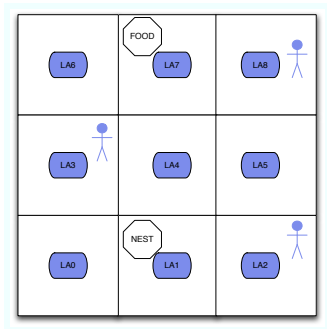
Wheeler and Narendra (1986) show that this game will only have optimal Nash equilibria, which can be found by using the L_{R-L} update scheme.

Ant Algorithms

- Algorithms that use a model of ant behavior as an optimization technique. e.g. Ant Colony Optimization
- A colony of agents walks around in graph of solution components.
- Each ant agent probabilistically creates a path that represents a possible solution.
- The amount of pheromones associated with a component, determines the probability of that component to be include in an agent's solution.
- Pheromones are updated in a monte carlo fashion, based on quality of the entire solution.

Modeling Ant Algorithms (Verbeeck & Nowé, 2002)

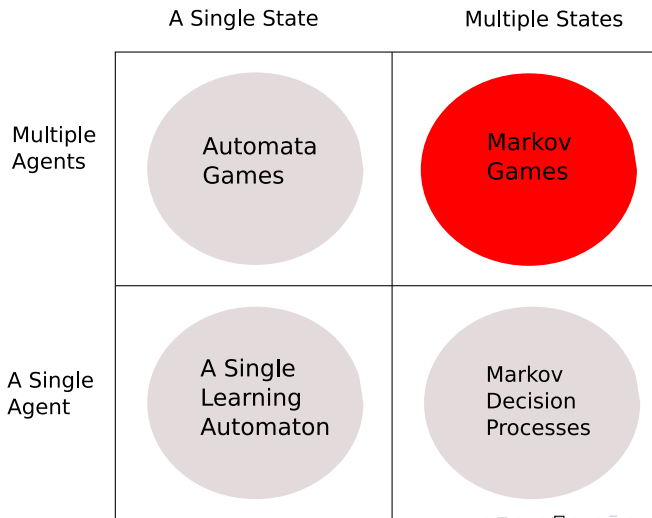
- Instead of pheromone vectors, put a learning automaton in each location.
- Multiple agents visit the states in parallel, causing multiple LA to be active at the same time.
- In each location agents use and update the same LA.



The learning system will still converge, provided that all agents have the same goal and individual agents do not affect each other's reward.

- 1 Part 1
 - Definition
 - First Examples: Tsetlin Automata
 - Reward-Penalty Schemes
 - Generalizations
- 2 Part 2
 - Automata Games
 - Exploration vs Exploitation
- 3 Part 3
 - Decentralized learning in MDPs
 - The Link with Ant Algorithms
- 4 Part 4**
 - Learning in Markov Games**
 - MMDPs**
 - Partial Observability**
 - Applications**
- 5 To Conclude

Part 4



Markov Games (a.k.a. Stochastic Games)

Extension of MDP to multi-agent setting:

- Set of states S , n agents, A_k^i is the action set for agent k in state $s_i \in S$.
- The joint action $\vec{a}^i = (a_1^i, \dots, a_n^i)$ selected in state s_i is the joint result of n agents individually choosing an action.
- Each agent k has individual reward function $R_k(s_i, s_j, \vec{a}^i)$
- State transition probabilities $T(s_i, s_j, \vec{a}^i)$ and rewards $R_k(s_i, s_j, \vec{a}^i)$ depend on joint agent action.
- Since agents can have different objectives, an equilibrium between agent policies is sought.

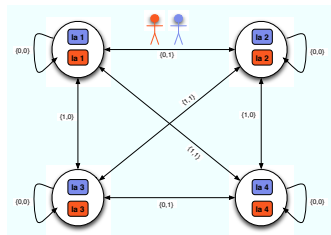
Markov Games (2)

Several special cases of Markov Games can be considered:

- Single player MG reduces to MDP
- Single state MG reduces to repeated game
- Single player, single states reduces to single LA with stationary environment.
- Team Markov games: all agents share the same reward function.

Decentralized Learning in Finite Markov Games

- Multiple agents traverse state space together.
- Each agent has its own LA in every action state.
- Each time step every agent activates LA in current state to select an action.
- Agents cannot observe actions or rewards of other agents.



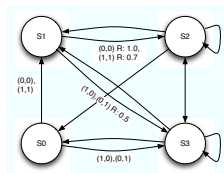
Decentralized Learning in Finite Markov Games (2)

- By using the agent's individual reward functions we can still calculate the expected reward for an agent k under a given joint policy π .

$$J_k^\pi = \sum_{i=1}^N \alpha^\pi(s_i) \sum_{j=1}^N T(s_i, s_j, \pi(s_i)) R_k(s_i, s_j, \pi(s_i))$$

- The markov game can then be approximated by 2 limiting games:
 - an **agent game** between n agents in which each agent has policies as actions.
 - an **automata game** between all $n \times N$ automata present in the states
- Both limiting games share the same pure Nash equilibria.

Limiting Games



2 agents and 2 action states \rightarrow 4 automata

agent 1 / agent 2	policies			
	(0,0)	(0,1)	(1,0)	(1,1)
(0,0)	0.2857	0.1667	0.1429	0.0833
(0,1)	0.1667	0.2	0.0833	0.1167
(1,0)	0.1429	0.0833	0.2857	0.1667
(1,1)	0.0833	0.1167	0.1667	0.2

$(LA_0^0, LA_1^0, LA_0^1, LA_1^1)$	$J(\alpha)$	$(LA_0^0, LA_1^0, LA_0^1, LA_1^1)$	J^π
(0,0,0,0)	0.2857	(1,0,0,0)	0.1667
(0,0,0,1)	0.1429	(1,0,0,1)	0.0833
(0,0,1,0)	0.1429	(1,0,1,0)	0.0833
(0,0,1,1)	0.2	(1,0,1,1)	0.1167
(0,1,0,0)	0.1667	(1,1,0,0)	0.2857
(0,1,0,1)	0.0833	(1,1,0,1)	0.1429
(0,1,1,0)	0.0833	(1,1,1,0)	0.1429
(0,1,1,1)	0.1167	(1,1,1,1)	0.2

Theorem (Vrancx et al., 2008)

The agent game and the automata game share the same pure Nash equilibria. Proof Outline:

- Agent game equilibria must always be automata game equilibria. When no agent alone can improve its payoff, no automaton can improve.
- Suppose the automata game contains equilibria not present in the agent game.
 - Then no single automaton can change its action to improve its payoff, but at least one agent can change its policy in 2 or more states to do better.
 - A single agent improving its policy reduces to the MDP case.
 - Proof of Wheeler and Narendra can be used to show that suboptimal policy cannot be equilibrium.

Limiting Games

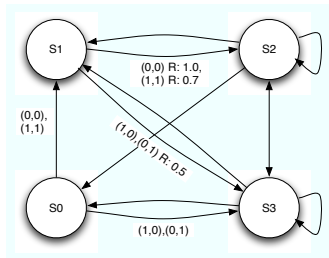
To summarize:

- Markov Games can be viewed at different levels: an agent game and an automata game.
- Both games share the same pure Nash Equilibria.
- If all automata use L_{R-I} scheme they will converge to a pure equilibrium between agent policies.
- The underlying games do not need to be known.

Team Markov Games (a.k.a. Multi-Agent MDPs)

- MDP with multiple agents: rewards and transitions depend on joint actions of all agents.
- Common Interest: All agents have the same reward function.
- Optimal joint policy, which maximizes all rewards still exists.
- Suboptimal equilibria can occur.
- L_{R-I} only guarantees local optimum.

MMDP



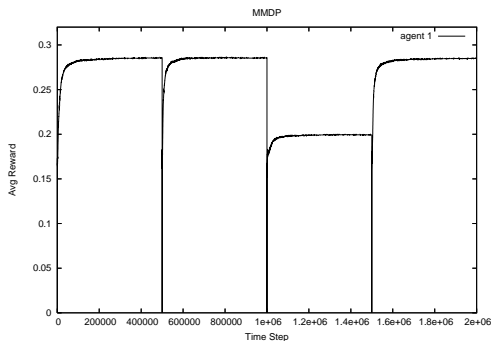
- 4 state, 2 agent MMDP
- Only 2 action states: s_0 and s_1
- Agents have 2 actions: 0 or 1
- Transitions leaving action states are deterministic
- Agents go from s_2 and s_3 to any other state with equal probability ($1/4$)
- Non-zero rewards are shown with transitions (R)

Limiting Games

- agent game: 2 players, 4 actions
- LA game: 4 players, 2 actions
- each agent play corresponds to LA play
- Equilibria are the same

policies agent 1 / agent 2		(0,0)	(0,1)	(1,0)	(1,1)
		(0,0)	0.2857	0.1667	0.1429
(0,1)	0.1667	0.2	0.0833	0.1167	
(1,0)	0.1429	0.0833	0.2857	0.1667	
(1,1)	0.0833	0.1167	0.1667	0.2	

$(LA_0^0, LA_1^0, LA_0^1, LA_1^1)$	J^π	$(LA_0^0, LA_1^0, LA_0^1, LA_1^1)$	$J(\pi)$
(0,0,0,0)	0.2857	(1,0,0,0)	0.1667
(0,0,0,1)	0.1429	(1,0,0,1)	0.0833
(0,0,1,0)	0.1429	(1,0,1,0)	0.0833
(0,0,1,1)	0.2	(1,0,1,1)	0.1167
(0,1,0,0)	0.1667	(1,1,0,0)	0.2857
(0,1,0,1)	0.0833	(1,1,0,1)	0.1429
(0,1,1,0)	0.0833	(1,1,1,0)	0.1429
(0,1,1,1)	0.1167	(1,1,1,1)	0.2

Experimental Results: L_{R-I} 

- Single typical run, LAs are reset every 5 million steps
- LA probabilities are initialized randomly
- Depending on initialization either convergence to optimal or suboptimal equilibrium

Parameterized Learning Automata

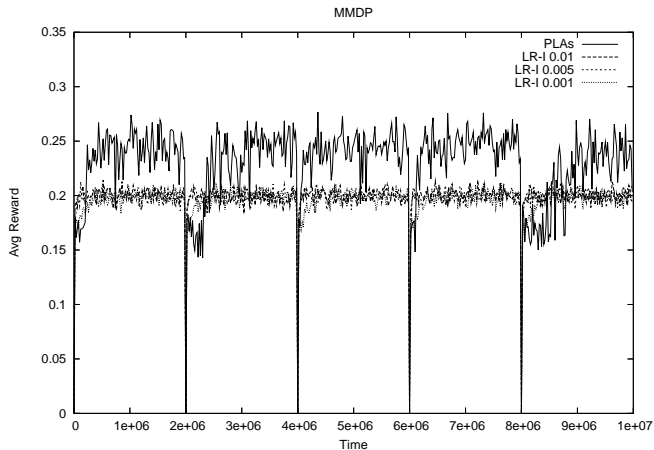
(Thathachar & Phansalkar, 1995)

$$u_i(t+1) = u_i(t) + \lambda b(t) \frac{\delta \ln g}{\delta u_i}(\vec{u}(t), a(t)) + \lambda h'(u_i(t)) + \sqrt{\lambda} f_i(t)$$

$$g(\vec{u}(t), a_i) = \frac{e^{u_i(t)}}{\sum_j e^{u_j(t)}}$$

- Use parameter vector \mathbf{u} and probability generating function g , in stead of modifying probabilities directly.
- Based on REINFORCE (Williams, 1992).
- Function $h(x)$ bounds parameter values.
- Added noise $f_i(t)$ allows algorithm to escape local optima.
- Converge to global optimum in Team Games.

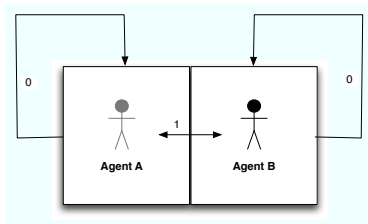
Experimental Results



Partially Observable Problems

- In all previous settings we assumed that agents can observe the entire state.
- But: this assumption is often not valid.
- System state often depends on unknown information about other agents.
- Can LAs still converge?

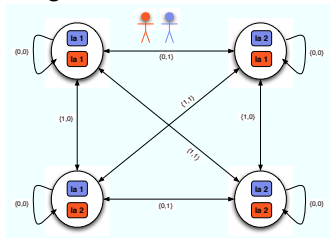
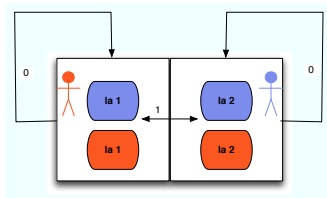
Grid World Problems



- Agents move around in grid
- Each agent can only observe its own location
- Agent rewards also depend on location and action of other agents.

Markov Game Mapping

We can map the problem to a Markov game by considering the joint location of the agents as a single state.



In the resulting Markov game the same LA is associated with multiple states, as the agent cannot distinguish between these states.

Example

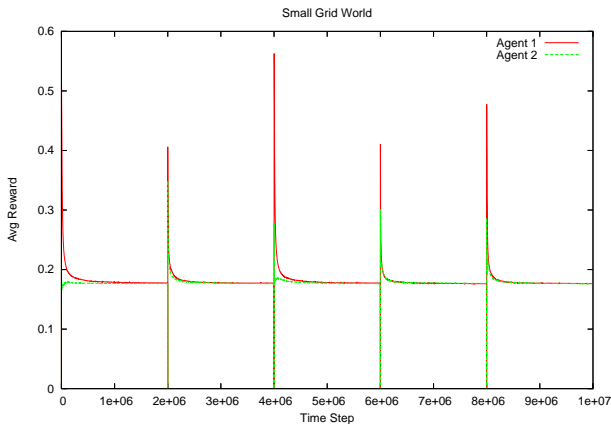
-Reward function based on joint location:

state	(R_{a1}, R_{a2})
{L1,L1}	(0.5,0.5)
{L1,L2}	(0.0,1.0)
{L2,L1}	(1.0,0.0)
{L2,L2}	(0.1,0.1)

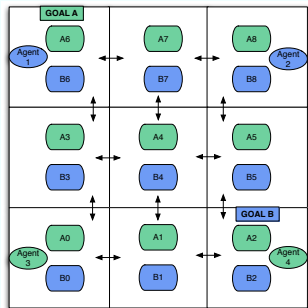
-Resulting Limiting Game:

		agent 2			
		{0,0}	{0,1}	{1,0}	{1,1}
agent 1	{0,0}	(0.4,0.4)	(0.68,0.28)	(0.12,0.52)	(0.4,0.4)
	{0,1}	(0.28,0.68)	(0.496,0.496)	(0.064,0.864)	(0.28,0.68)
	{1,0}	(0.52,0.12)	(0.864,0.064)	(0.176,0.176)	(0.52,0.12)
	{1,1}	(0.4,0.4)	(0.68,0.28)	(0.12,0.52)	(0.4,0.4)

Experimental Results



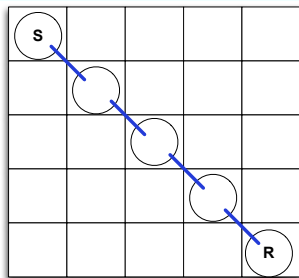
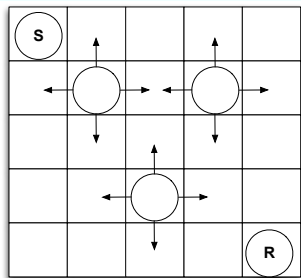
Multi-objective Ant Algorithms



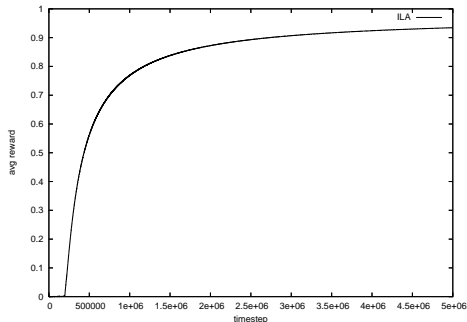
- Extension of ant algorithms: Multiple colonies of agents.
- Each colony has its own objective.
- In each location we put a LA for every agent objective.
- Agents that visit a location, activate only the LA corresponding to their objective.
- Agents of different colonies can influence each other's reward.

Application: ad hoc networking

A number of mobile nodes must form a link between a stationary source and sink node. Each node can only view its own location. Agents get reward 1 if a connection is established, 0 else. (Based on Chang et al., 2004)



Application: ad hoc networking (2)



- 3 agents and a 5×5 grid results in Markov game with 15625 states
- Since automata are associated with agent locations instead of states, only 75 LAs are used.

- 1 Part 1
 - Definition
 - First Examples: Tsetlin Automata
 - Reward-Penalty Schemes
 - Generalizations
- 2 Part 2
 - Automata Games
 - Exploration vs Exploitation
- 3 Part 3
 - Decentralized learning in MDPs
 - The Link with Ant Algorithms
- 4 Part 4
 - Learning in Markov Games
 - MMDPs
 - Partial Observability
 - Applications
- 5 To Conclude

Conclusions

LA are interesting building blocks to solve different type of RL problems

Challenges

- Different LA updates
- Equilibrium selection (multi-state)
- (State dependent) dynamic environments.
- Influence different state observations (POMDP setting)
- Heterogenous agents



References I



Y.-H. Chang, T. Ho, and L. P. Kaelbling.

All learning is local: Multi-agent learning in global reward games.

In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.



K. Narendra and M.A.L. Thathachar.

Learning Automata: An Introduction.

Prentice-Hall International, Inc, 1989.



M.A.L. Thathachar and V.V. Phansalkar.

Learning the global maximum with parameterized learning automata.

Neural Networks, IEEE Transactions on, 6(2):398–406, 1995.



M.A.L. Thathachar and Sastry P.S.

Networks of Learning Automata: Techniques for Online Stochastic Optimization.

Kluwer Academic Publishers, 2004.



K. Verbeek and A. Nowe.

Colonies of learning automata.

IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 32 (6):772 – 780, 2002.



K. Verbeek, A. Nowé, P. Vrancx, and M. Peeters.

Multi-automata learning.

In Cornelius Weber, Mark Elshaw, and Norbert Michael Mayer, editors, *Reinforcement Learning: Theory and Applications*, pages 167–185. I-Tech Education and Publishing, 2008.

References II



P. Vrancx, K. Verbeeck, and A. Nowé.

Analyzing stigmergetic algorithms through automata games.

Lecture Notes in Bioinformatics, Knowledge Discovery and Emergent Complexity in Bioinformatics, 4366:145–156, 2007.



P. Vrancx, K. Verbeeck, and A. Nowé.

Optimal convergence in multi-agent MDPs.

Lecture Notes in Computer Science, Knowledge-Based Intelligent Information and Engineering Systems (KES 2007), 4694:107–114, 2007.



P. Vrancx, K. Verbeeck, and A. Nowé.

Networks of learning automata and limiting games.

Lecture Notes in Artificial Intelligence, ALAMAS III, 4865:224–238, 2008.



P. Vrancx, K. Verbeeck, and A. Nowé.

Decentralized learning in markov games.

IEEE Transactions on Systems, Man and Cybernetics, Part B: Special Issue on Approximate Dynamic Programming and Reinforcement Learning for Control, 38(4), Appears August 2008.



R.M. Wheeler and K.S. Narendra.

Decentralized Learning in Finite Markov Chains.

IEEE Transactions on Automatic Control, AC-31:519 – 526, 1986.