Extended Replicator Dynamics as a Key to Reinforcement Learning in Multi-agent Systems

4 authors, including:

Karl Tuyls
DeepMind and University of Liverpool
296 PUBLICATIONS   **2,871** CITATIONS

SEE PROFILE

Ann Nowe
Vrije Universiteit Brussel
403 PUBLICATIONS   **2,781** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

OPTIMIZATION OF HEAVILY CONSTRAINED HYBRID-FLEXIBLE FLOWSHOP PROBLEMS USING A MULTI-AGENT REINFORCEMENT LEARNING APPROACH View project

Multi-instance learning View project

Extended Replicator Dynamics as a key to Reinforcement Learning in Multi-Agent Systems

Karl Tuyls *, Dries Heytens, Ann Nowe, and Bernard Manderick

Computational Modeling Lab
Department of Computer Science
Vrije Universiteit Brussel
Belgium
{ktuyls@,dheytens@,asnowe@info,bmanderi@}vub.ac.be

Abstract. Modeling learning agents in the context of Multi-agent Systems requires an adequate understanding of their dynamic behaviour. Evolutionary Game Theory provides a dynamics which describes how strategies evolve over time. Börgers et al. [1] and Tuyls et al. [11] have shown how classical Reinforcement Learning (RL) techniques such as Cross-learning and Q-learning relate to the Replicator Dynamics (RD). This provides a better understanding of the learning process. In this paper, we introduce an extension of the Replicator Dynamics from Evolutionary Game Theory. Based on this new dynamics, a Reinforcement Learning algorithm is developed that attains a stable Nash equilibrium for all types of games. Such an algorithm is lacking for the moment. This kind of dynamics opens an interesting perspective for introducing new Reinforcement Learning algorithms in multi-state games and Multi-Agent Systems.

1 Introduction

In this paper a new RL algorithm, based on the Replicator Dynamics (RD) from Evolutionary Game Theory (EGT), is introduced. Several authors have already noticed and proved that the RD can emerge from several RL schemes. Börgers et al. [1] have shown that the RD can emerge from Cross learning and the authors [10, 11] have shown that the RD emerge from Learning Automata and Boltzmann Q-learning. This emergence offers a lot of advantages. For instance, these evolutionary dynamics open a new perspective in understanding and fine tuning the learning process in games and more general in Multi-Agent Systems (MAS). Learning can be very time consuming, especially when you need to fine tune some parameters. As the experiments in Tuyls et al. illustrate [10, 11], plotting the direction field of the RD beforehand in one-state games gives information on how to initialize the learning agents so that they end up in the most interesting attractors. As these previous results show, convergence is not always guaranteed for some particular kind of games. For this reason we adapted the RD to

---

an extended evolutionary dynamics which describes the desired behaviour from the Reinforcement Learning (RL) agents. After this, the accompanying RL algorithm of these Extended Replicator Dynamics (ERD) will be developed.

The outline of the paper is as follows, in section 2 we elaborate on the RD from EGT and on the important connection with RL, more specifically the Cross learning model. Section 3 describes how the RD are extended to satisfy the need of convergence to certain attractors in certain games. After this we describe the new RL algorithm matching this ERD. Section 4 describes the experiments, confirming the results from section 3. Finally, we end with a conclusion.

2 Selection Dynamics and Cross Learning

In this section we elaborate on an important result of Börgers and Sarin [1]. They showed that in an appropriately constructed time limit, the Cross learning model converges to the Replicator Equations. Also it is shown [10, 11] how the RD emerge from Learning Automata and Boltzmann Q-learning. The perspective of evolutionary dynamics in reinforcement learning offers a lot of advantages. First, it becomes possible to understand RL in terms of evolutionary dynamics, i.e. selection and mutation mechanisms, second it allows to fine tune the learning process in advance. In this paper a new RL algorithm will be developed based on the replicator equations, which behaves as one desires. This means that the learning process is guaranteed to converge to a stable Nash equilibrium in all types of one-state games. In a first subsection we briefly explain the RD, as we will alter these dynamics in section 3. After this the Cross learning model will be explained, as this model will serve as a basis for a new RL algorithm in section 3.

2.1 The Replicator Equations

The basic concepts and techniques developed in EGT were initially formulated in the context of evolutionary biology [13, 9]. In this context, the strategies of all the players are genetically encoded (called genotype). Each genotype refers to a particular behavior which is used to calculate the payoff of the player. The payoff of each player's genotype is determined by the frequency of other player types in the environment.

One way in which EGT proceeds is by constructing a dynamic process in which the proportions of various strategies in a population evolve. Examining the expected value of this process gives an approximation which is called the RD. Simpy stated, an abstraction of an evolutionary process usually combines two basic elements: selection and mutation. Selection favors some varieties over others, while mutation provides variety in the population. RD highlights the role of selection, it describes how systems consisting of different strategies change over time. They are formalized as a system of differential equations. Each replicator (or genotype) represents one (pure) strategy. This strategy is inherited by all

the offspring of that replicator. The general form of a replicator dynamic is the following:

$$\frac{dx_i}{dt} = [(A\mathbf{x})_i - \mathbf{x} \cdot A\mathbf{x}]x_i \qquad (1)$$

In equation (1), x_i represents the density of strategy i in the population, A is the payoff matrix which describes the different payoff values each individual replicator receives when interacting with other replicators in the population. The state of the population ($\mathbf{x}$) can be described as a probability vector $\mathbf{x} = (x_1, x_2, ..., x_J)$ which expresses the different densities of all the different types of replicators in the population. Hence $(A\mathbf{x})_i$ is the payoff which replicator i receives in a population with state x and $\mathbf{x} \cdot A\mathbf{x}$ describes the average payoff in the population. The growth rate $\frac{\frac{dx_i}{dt}}{x_i}$ of the population share using strategy i equals the difference between the strategy's current payoff and the average payoff in the population. For further information we refer the reader to [13, 2].

In this paper the players are reinforcement learners. We consider a game to be played between the members of two different populations, each population representing one reinforcement learner. As a result, we need two systems of differential equations: one for the row player (P) and one for the column player (Q). This setup corresponds to a RD for asymmetric games. If $A = B^t$, equation (1) would again emerge.

This translates into the following replicator equations for the two populations:

$$\frac{dp_i}{dt} = [(A\mathbf{q})_i - \mathbf{p} \cdot A\mathbf{q}]p_i \qquad (2)$$

$$\frac{dq_i}{dt} = [(B\mathbf{p})_i - \mathbf{q} \cdot B\mathbf{p}]q_i \qquad (3)$$

As can be seen in equation (2) and (3), the growth rate of the types in each population is now determined by the composition of the other population. Note that, when calculating the rate of change using these systems of differential equations, two different payoff matrices (A and B) are used for the two different players.

2.2   The Cross Learning model

The cross learning model is a special case of the standard reinforcement learning model [1] . The model considers several agents playing the same normal form game repeatedly in discrete time. At each point in time, each player is characterized by a probability distribution over her strategy set which indicates how likely she is to play any of her strategies. At each time step (indexed by n) , a player chooses one of her strategies based on the probabilities which are related to each isolated strategy. As a result a player can be represented by a probability vector:

$$p(n) = (p_1(n), ..., p_r(n))$$

In case of a 2-player game with payoff matrix U, player k $(k \in 1, 2)$ gets payoff U_{ij}^k when player 1 chooses strategy i and player 2 chooses strategy j. Players do not observe each others' strategies and payoffs, they are uninformed players. After each stage they update their probability vector, according to,

$$p_i(n + 1) = U_{ij} + (1 - U_{ij})p_i(n) \qquad (4)$$

$$p_{i'}(n + 1) = (1 - U_{ij})p_{i'}(n) \qquad (5)$$

where $0 \leq U_{ij} \leq 1$. Equation (4) expresses how the probability of the selected strategy (i) is updated and equation (5) expresses how all the other strategies $i' \neq i$ are adjusted. The probability vector of $Q(n)$ is updated in an analogous manner. This entire system of equations defines a stochastic update process for the players $\{p^k(n)\}$. This process is called the "Cross learning process" in [1]. Börgers and Sarin showed that in an appropriately constructed continuous time limit, this model converges to the asymmetric, continuous time version of the replicator dynamics, see section 2.

3 Extending the Replicator Equations and the Cross learning model

The reasons for changing the RD and looking for a new dynamics become clear from [10, 11]. In one-state games it is impossible for Cross learning and Learning Automata to guarantee convergence to a stable Nash equilibrium in all types of games. In Boltzmann Q-learning a Nash equilibrium can be attained, but there is no guarantee for stability. If a dynamical system can be found that offers these guarantees, we can construct a reinforcement learning algorithm in an analogous manner with the same behaviour of this adapted dynamical system. This makes the approach of replicator equations very interesting and promising toward multi-state games and Multi-Agent Systems.

In the first subsection we will alter the traditional replicator equations in such a manner that in all classes of games the players will converge to a particular Nash equilibrium (see section 4). These new equations are referred to as the Extended Replicator Dynamics (ERD). In the second subsection we will present the accompanying learning algorithm of the changed dynamics based on the Cross learning model.

3.1 Developing an Extended Replicator Dynamics

When constructing an altered selection dynamics, we take the replicator dynamics and its interpretation as a starting point. In replicator dynamics, the probabilities a players has over its strategies are changed greedily with respect to payoff in the present. In this section a method is shown to change this probabilities over strategies not only with respect to payoff growth in the present but

also to payoff growth in the future. We call those players that act so as to optimize future payoff extended Cross learners and the class of dynamics associated extended dynamics.

There are of course different ways to build such extended players. The most obvious is to use a linear approximation of the evolution of fitness in time. This is the approach we use here.

For the ERD we compose the following equation f,

$$f(x) = RD(x) + (dRD(x)/dt) * \eta \qquad (6)$$

were RD(x) is,

$$\frac{dx_i}{dt} = [(A\mathbf{x})_i - \mathbf{x} \cdot A\mathbf{x}]x_i \qquad (7)$$

and η is the parameter that determines how far in the future we need to look.

The composition of equation 6 can best be understood as follows. When using the classical replicator equations (i.e. RD(x)), we act greedily toward payoff in the present. When adding our second term,

$$(dRD(x)/dt) * \eta \qquad (8)$$

we act greedily toward payoff in the future. From an analytical point of view, the second term gives actions that are winning fitness (whether its fitness is negative or positive) a positive push toward a higher chance of getting selected. On the other hand, actions that are losing fitness (again whether its fitness is negative or positive) are given a negative push toward a lower chance of getting selected. This extends the traditional replicator equations. The algorithm we used to calculate this dynamics can be found in algorithm 1. It will be referred to as the ERD algorithm since it extends the tradional $RD(x)$ with the future. This extended evolutionary dynamics succeeds in converging to a stable Nash Equilibrium (NE) in all 3 categories of 2*2 games. Experiments confirming this can be found in section 4.

In the next section the reinforcement learning algorithm based on these extended dynamics is developed.

3.2 Developing an Extended RL-algorithm

To develop a RL-algorithm based on the Extended Replicator Dynamics, we start from the result of Börgers and Sarin. They showed that in an appropriately constructed continuous time limit, this model converges to the asymmetric, continuous time version of the replicator dynamics [1]. Recall that we extended these dynamics with the following acceleration term,

$$(dRD(x)/dt) * \eta \qquad (9)$$

expressing that we act greedily toward payoff in the future. So for the part of the RD we can rely on the Cross algorithm of Börgers and Sarin. For the part

---
Algorithm 1 The algorithm used to calculate extended replicator dynamics.

Parameters:

> η How far in the future we look for calculating future fitness growth.
> $stepSize$ Determines how large steps we take in the updaterule
> $a_{1..r}$ The actions a available to the players p.
> $p(n) = (p_1(n), ..., p_r(n))$ The chances for p playing a_i.
> $RD_{a_i(n)}$ The replicator dynamics function of action a from player p at timestep n according to the current position of all player's strategies. This implicitly defines the game being played.

For all actions a_i.

1. Calculate an approximation to the replicator dynamics acceleration in the current position of all player's strategies in Δ.
   $acceleration_{a_i(n)} := RD_{a_i(n-1)} - RD_{a_i(n-2)}$;
2. Ensure payoff positivity. It turns our dynamics into one that is stable in a NE.
   if$(\neg(((RD_{a_i(n)} > 0) \wedge (RD_{a_i(n)} + \eta * acceleration_{a_i(n)} > 0))\vee$
   $((RD_{a_i(n)} < 0) \wedge (RD_{a_i(n)} + \eta * acceleration_{a_i(n)} < 0)))$
   $\{acceleration_{p_a} := 0;\}$
3. Adjust the strategy according to:
   $f_{a_i(n)} := stepSize * RD_{a_i(n)} + \eta * acceleration_{a_i(n)}$; f is the function we are approximating
   $p_i(n) := p_i(n) + f_{a_i(n)}$

---

of equation 9 we calculate an approximation in algorithm 2.

Step a of the algorithm is nothing more than the calculation of Cross Learning. Step b calculates the approximation of the ERD, where the *accel* variable contains the approximation of equation 9. Furthermore payoff positivity is ensured. Payoff positivity means that strategies that earn above (below) average have positive (negative) growth rates. Actually, with payoff positivity we ensure that there is stability in all Nash equilibria. Step c executes the update of the probabilities of the different actions and updates the acceleration term (or the payoff in the future). To finalize, the probabilities are normalized.

In the next section we will show experiments in all classes of games with this algorithm and it will become clear that it always converges to the Extended Replicator Dynamics.

4 Experiments

In this section we describe some experiments that illustrate the mathematical derivation of section 3. The experiments have been conducted with 2×2 games. The general payoff matrices, A for the first player and B for the second, are defined in table 1. The family of 2×2 games is usually classified in three subclasses, as follows[5],

---
Algorithm 2 An algorithm for RL based on extended replicator dynamics.

Parameters:

η How far in the future we look for calculating future fitness growth.

$stepSize$ Determines how large steps we take in the updaterule

$a_{1..r}$ The actions a available to the players p.

$p(n) = (p_1(n), ..., p_r(n))$ The chances for p playing a_i.

$\overrightarrow{accel}$ A variable containing an approximative current acceleration for each $p_i(n)$.

θ Learningrate for learning $\overrightarrow{accel}$.

$\overrightarrow{formerSpeed}$ A variable containing an approximative speed for each a_i.

$Env(\overrightarrow{a_i})_p$ The function returning the immediate payoff from the environment to player p when all players have acted according to $\overrightarrow{a_i}$. This implicitly defines the game being played.

Act_p The function defining the action selection method.

For all players p
    1. $\overrightarrow{actions}_p := Act_p$;

For all players p
    1. $\overrightarrow{rewards}_p := Env(\overrightarrow{actions})_p$

For all players p
    1. For all actions a_i
        (a) Calculate CrossLearning.
            if $(a_i == \overrightarrow{actions}_p)$
            then {
            $CrossLearning := \overrightarrow{rewards}_p * (1 - p_i(n))$; }
            else {
            $CrossLearning := -\overrightarrow{rewards}_p * p_i(n)$;}
        (b) Calculate extended RD approximation, making sure we retain payoff positive.
            if$(sign(CrossLearning) == sign(CrossLearning + \overrightarrow{accel}_{a_i}))$
            then $ExtendRDLearning := CrossLearning + \overrightarrow{accel}_{a_i}$;
            else $ExtendRDLearning := 0$;
        (c) Perform the updaterule and calculation of $\overrightarrow{accel}_{a_i}$.
            $p_i(n) := p_i(n) + stepSize * extendRDLearning$;
            $\overrightarrow{accel}_{a_i} := \overrightarrow{accel}_{a_i} + \theta * ((stepSize * CrossLearning - \overrightarrow{formerSpeed}_{a_i}) - \overrightarrow{accel}_{a_i})$;
            $\overrightarrow{formerSpeed}_{a_i} := stepSize * CrossLearning$; }
    2. Normalize the $p_i(n)$ so that their sum is 1.

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \quad B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$$

Table 1. The left matrix (A) defines the payoff for the row player, the right matrix (B) defines the payoff for the column player

Subclass 1: if $(a_{11} - a_{21})(a_{12} - a_{22}) > 0$ or $(b_{11} - b_{12})(b_{21} - b_{22}) > 0$, at least one of the 2 players has a dominant strategy, therefore there is just 1 strict equilibrium.

Subclass 2: if $(a_{11} - a_{21})(a_{12} - a_{22}) < 0$,$(b_{11} - b_{12})(b_{21} - b_{22}) < 0$, and $(a_{11} - a_{21})(b_{11} - b_{12}) > 0$, there are 2 pure equilibria and 1 mixed equilibrium.

Subclass 3: if $(a_{11} - a_{21})(a_{12} - a_{22}) < 0$,$(b_{11} - b_{12})(b_{21} - b_{22}) < 0$, and $(a_{11} - a_{21})(b_{11} - b_{12}) < 0$, there is just 1 mixed equilibrium.

The first subclass includes those type of games where each player has a dominant strategy, as for instance the prisoners dilemma. However it includes a larger collection of games since only 1 of the players needs to have a dominant strategy. In the second subclass none of the players has a dominated strategy. But both players receive the highest payoff by both playing their first or second strategy. This is expressed in the condition $(a_{11} - a_{21})(b_{11} - b_{12}) > 0$. The third subclass only differs from the second in the fact that the players do not receive their highest payoff by both playing the first or the second strategy. This is expressed by the condition $(a_{11} - a_{21})(b_{11} - b_{12}) < 0$. In the following three subsections we describe the results of the experiments conducted in each subclass. In all subclasses we used the following general settings,

- η is set at 300
- $stepsize$ is set to 0.003
- $theta$ is set to 0.0006

4.1 Category 1: Prisoners dilemma

In category 1 we considered the prisoners dilemma game [13, 3]. In this game both players have a dominant strategy, more precisely *defect*. The payoff matrices for this game are as follows,
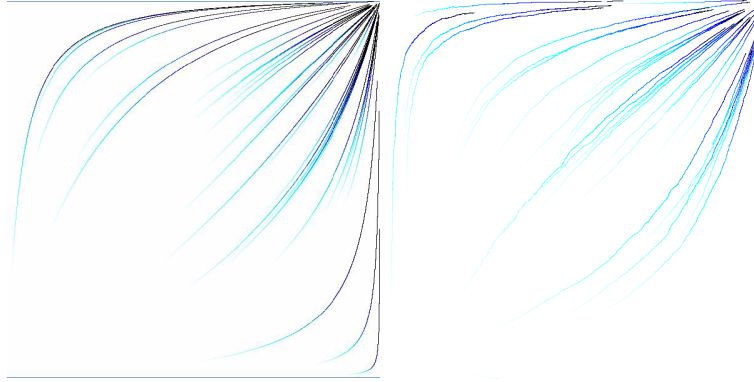
$$\begin{pmatrix} 1 & 5 \\ 0 & 3 \end{pmatrix} \text{ and } \begin{pmatrix} 1 & 0 \\ 5 & 3 \end{pmatrix}$$

In figure 1 the replicator dynamic of the game is plotted using the differential equations of 6.

More specifically, the figure on the left illustrates the direction field of the extended replicator dynamics and the figure on the right shows the learning process of algorithm 1. We plotted for both players the probability of choosing their first strategy (in this case defect). So, the first players probabilities are on the X-axis and the second players probabilities on the Y-axis. As starting points for the learning process we generated 50 random points. In every point a learning path starts and converges to the equilibrium at the point $(1, 1)$. As you can see all the sample paths of the reinforcement learning process approximate the paths of the RD.
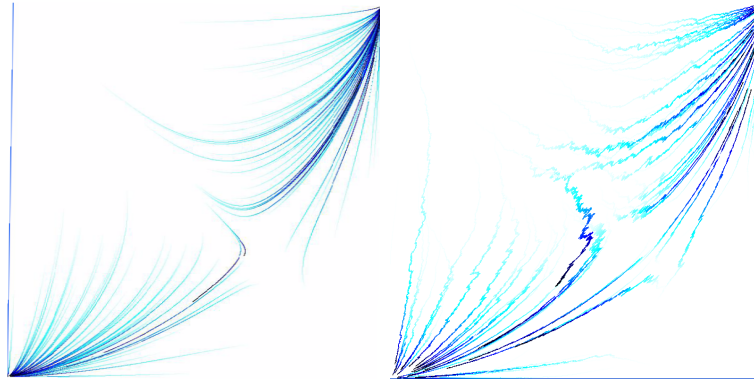
4.2 Category 2: Battle of the sexes

For the second game we considered the battle of the sexes game, defined by the following payoff matrices [13, 3]:

Fig. 1. *Left*: The direction field of the ERD of the prisoners game. *Right*: The paths induced by the learning process.

$$\begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix} \text{ and } \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}$$

Figure 2 demonstrates the results. On the left you see the direction field of this game, on the right the sample paths induced by the learning process. You can see 3 equilibria: two pure equilibra at $(0,0)$ and at $(1,1)$, and one mixed at $(2/3, 1/3)$. Now we have convergence to the 2 strict equilibria. The third equilibrium is very unstable as you can see in the direction field plot. This instability is the reason why it will not emerge from the learning process on the long run. Again we used a grid of random starting points.


Fig. 2. *Left*: The direction field of the RD of the battle of the sexes game. *Right*: The paths induced by the learning proces.

4.3 Category 3:

The third class consists of the games with a unique mixed equilibrium. We considered the following game,

$$\begin{pmatrix} 2 & 3 \\ 4 & 1 \end{pmatrix} \text{ and } \begin{pmatrix} 3 & 1 \\ 2 & 4 \end{pmatrix}$$

Typical for the traditional RD in this class of games is that the interior trajectories define closed orbits around the equilibrium point. Figure 3 illustrates this.
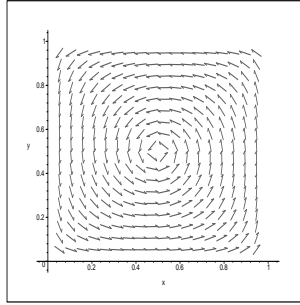

Fig. 3. The direction field plot of the RD for subclass 3

This type of game shows an important difference with our ERD and the matching learning algorithm. ERD and the matching learning algorithm will not circle but converge to the mixed Nash equilibrium. This is illustrated in figure 4. Moreover the equilibrium is stable, meaning that the learning process will not abandon it. The long-run learning dynamics are illustrated in the figure on the right. Again we used a grid of random starting points for the learning process.

5 Conclusion

In this paper it is shown that the RD from EGT are an adequate basis for reinforcement learning in games. This opens a new perspective on developing reinforcement learning algorithms for multi-state games and Multi-Agent Systems. More precisely we showed that with an extension of the traditional replicator equations (ERD), Nash equilibria can be attained in all kind of games. Based on this new dynamics we constructed a RL-algorithm that converges to this extended replicator dynamics. In [12] we showed that for the matter of one-state games Cross learning is the simplest learning model (over Learning Automata, Q-learning) and suffices to attain the same results as the other learning models. It turned out that the Cross model keeps things most simple in the sense of
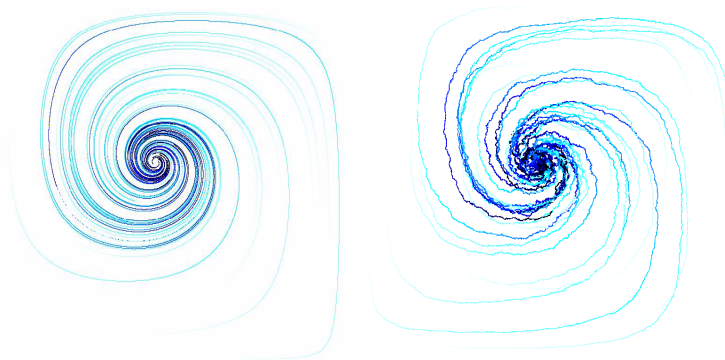
Fig. 4. *Left*: The direction field of the RD. *Right*: The paths induced by the learning process.

setting parameters and computational effort. The experiments confirmed that with the Cross model, the Nash equilibria can be reached in the most elegant way. Therefore this new algorithm, extending Cross Learning and guaranteeing a stable Nash equilibrium, is sufficient for any type of one-state game.

In a next phase, these results will be extended to multiple state games. Developing such algorithms will be based on Learning Automata and Q-learning, two possible techniques for multi-state games. In both techniques the connection with the Replicator Dynamics has been proved [10, 11].

References

1. Börgers, T., Sarin, R., Learning Through Reinforcement and Replicator Dynamics. Journal of Economic Theory, Volume 77, Number 1, November 1997.
2. Hofbauer, J., Sigmund, K., Evolutionary Games and Population Dynamics, Cambridge University Press, 1998.
3. Gintis, C.M., Game Theory Evolving. Princeton University Press, June 2000.
4. Narendra, K., Thathachar, M., Learning Automata: An Introduction. Prentice-Hall (1989).
5. Redondo, F.V., Game Theory and Economics, Cambridge University Press, (2001).
6. Schneider, T.D., Evolution of biological information. journal of NAR, volume 28, pages 2794 - 2799, 2000.
7. Stauffer, D., Life, Love and Death: Models of Biological Reproduction and Aging. Institute for Theoretical physics, Köln, Euroland, 1999.
8. Sutton, R.S., Barto, A.G. : Reinforcement Learning: An introduction. Cambridge, MA: MIT Press (1998).
9. L. Samuelson, Evolutionary Games and Equilibrium Selection, MIT Press, Cambridge, MA, 1997.
10. Tuyls, K., Lenaerts, T., Verbeeck, K., Maes, S. and Manderick, B, Towards a Relation Between Learning Agents and Evolutionary Dynamics. Proceedings of BNAIC 2002. KU Leuven, Belgium.

11. Tuyls, K., Verbeeck, K. and Lenaerts, T., A Selection-Mutation model for Q-learning in MAS. Accepted at AAMAS 2003. Melbourne, Australia.
12. Tuyls, K., Verbeeck, K., and Maes, S. On a Dynamical Analysis of Reinforcement Learning in Games: Emergence of Occam's Razor. Accepted at CEEMAS 2003.
13. Weibull, J.W., Evolutionary Game Theory, MIT Press, (1996).