Auton Agent Multi-Agent Syst (2007) 14:239–269
DOI 10.1007/s10458-006-9007-0

Exploring selfish reinforcement learning in repeated games with stochastic rewards

Katja Verbeeck · Ann Nowé · Johan Parent · Karl Tuyls

Published online: 10 November 2006

Abstract In this paper we introduce a new multi-agent reinforcement learning algorithm, called exploring selfish reinforcement learning (ESRL). ESRL allows agents to reach optimal solutions in repeated non-zero sum games with stochastic rewards, by using coordinated exploration. First, two ESRL algorithms for respectively common interest and conflicting interest games are presented. Both ESRL algorithms are based on the same idea, i.e. an agent explores by temporarily excluding some of the local actions from its private action space, to give the team of agents the opportunity to look for better solutions in a reduced joint action space. In a latter stage these two algorithms are transformed into one generic algorithm which does not assume that the type of the game is known in advance. ESRL is able to find the Pareto optimal solution in common interest games without communication. In conflicting interest games ESRL only needs limited communication to learn a fair periodical policy, resulting in a good overall policy. Important to know is that ESRL agents are independent in the sense that they only use their own action choices and rewards to base their decisions on, that ESRL agents are flexible in learning different solution concepts and they can handle both stochastic, possible delayed rewards and asynchronous action selection. A real-life experiment, i.e. adaptive load-balancing of parallel applications is added.

K. Verbeeck (✉)
Computational Modeling Lab (COMO), Vrije Universiteit Brussel, Brussels, Belgium
e-mail: kaverbee@vub.ac.be

A. Nowé
Computational Modeling Lab (COMO), Vrije Universiteit Brussel, Brussels, Belgium
e-mail: asnowe@info.vub.ac.be

J. Parent
Computational Modeling Lab (COMO), Vrije Universiteit Brussel, Brussels, Belgium
e-mail: jparent@info.vub.ac.be

K. Tuyls
Institute for Knowledge and Agent Technology (IKAT), University of Maastricht, The Netherlands,
e-mail: k.tuyls@micc.unimaas.nl

Keywords Multi-agent reinforcement learning · Learning automata · Non-zero sum games

1 Introduction

Exploring selfish reinforcement learning (ESRL) is a new approach to multi-agent reinforcement learning (MARL). It originates from the theory of learning automata (LA), an early version of reinforcement learning (RL), which has its roots in psychology [15]. The collective behavior of LA is one of the first examples of MARL that have been studied. A learning automaton describes the internal state of an agent as a probability distribution over actions. These probabilities are adjusted based on the success or failure of the actions taken. The work of [15] analytically treats learning not only in the single automaton case, but also in the case of hierarchies and distributed interconnections. Of special interest to MARL research is the work done within the framework of learning automata games.

Learning automata games were developed for learning repeated normal form games, well known in game theory [18,6]. A central solution concept for these games, is that of a *Nash equilibrium*. Learning automata games with suitable reinforcement update schemes were proved to converge to one of the Nash equilibria in repeated games, [20,15]. However, global optimality is not guaranteed. As will be discussed later, the same criticism holds for most work in current MARL research.

One important problem is that games can have multiple Nash equilibria and thus the question arises which equilibrium the agents should learn. Another related issue, is how the rewards are distributed between the agents. In some systems, the overall performance can only be as good as that of the worst performing agent. As such the agents have to learn to equalize the average reward they receive. A Nash equilibrium is not always fair in the sense that the agents' rewards or payoffs are not necessarily divided equally among them. Even worse, a Nash equilibrium can be dominated by a Pareto optimal solution which gives a higher payoff to all agents. So, in games with multiple equilibria, typically the agents have preferences toward different Nash equilibria.

In order to deal with the observations stated above, ESRL approaches not only the set of Nash equilibria, but also the set of Pareto Optimal solutions of the game. As such, the solution that is most suitable for the problem at hand can be selected. For example, an appropriate solution could be to learn what we call a *periodical policy*. In a periodical policy, agents alternate between periods in which they play different pure Nash equilibria, each equilibrium is preferred by one of the agents. As such, a fair solution can be obtained for all agents in a game with conflicting interests. Note that we use the word *fair* in the sense that the solution will distribute the rewards equally among all agents. We call a solution *optimally fair* when there is no other solution that is also fair for the agents but gives the agents more reward on average. Periodical policies were first introduced in [17].

The key feature of ESRL agents is that they use a form of coordinated exploration to reach the optimal solution in a game. ESRL is able to find attractors of the single stage game in the same way a game of learning automata does. But, as soon as an attractor is visited, learning must continue in a subaction space from which that attractor is removed. As such the team of agents gets the opportunity to look for possibly better solutions in a reduced space. So, ESRL searches in the joint action space of

the agents for attractors as efficiently as possible by shrinking it temporarily during different periods of play.

ESRL learning is organized in phases. Phases in which agents act independently and behave as selfish optimizers are alternated with phases in which agents are social, and act so as to optimize the group objective. The former phases are called *exploration phases*, while the latter phases are called *synchronization phases*. During these synchronization phases, attractors are removed from the joint action space by letting each agent remove one action from its private action space. For these synchronization phases, limited communication may be necessary, however sometimes a basic signal suffices. The name *exploring selfish reinforcement learning* stresses that during the exploration phases the agents behave as selfish reinforcement learners.

ESRL agents experience games either as a pure common interest game or as a conflicting interest game. The expected rewards given in the game matrices we consider here, represent the mean of a binomial distribution. Extensions to other continuous values distributions will discussed at the end. A nice property of ESRL agents is that they do not need to know in advance the type of the game. Also, no further restrictions are imposed on the form of the game, for instance pure Nash equilibria may or may not exist. In a common interest game, ESRL is able to find one of the Pareto optimal solutions of the game. In a conflicting interest game, we show that ESRL agents learn optimal fair, possibly periodical policies [17,26]. Important to know is that ESRL agents are independent in the sense that they only use their own action choices and rewards to base their decisions on, that ESRL agents are flexible in learning different solution concepts and they can handle both stochastic, possible delayed rewards and asynchronous action selection. In [26] a job scheduling experiment is solved by conflicting interest ESRL agents. In this paper, we describe the problem of adaptive load-balancing parallel applications, handled by ESRL agents as a common interest game.

This paper is organized as follows. In the next section game theoretic terminology is introduced. We continue with a short overview of learning automata theory in Sect. 3. Next, ESRL in respectively stochastic common interest (Sect. 4) and conflicting interest games (Sect. 5) is discussed. Testbed games are described and the behavior of ESRL is analyzed. In Sect. 6, ESRL is generalized to stochastic non-zero sum games. In Sect. 7, the robustness of ESRL to delayed rewards and asynchronous action selection is illustrated with the problem of adaptive load-balancing parallel applications. Finally the last section discusses the work presented and summarizes related literature.

2 Game theoretic background

In this section we introduce game theoretic terminology of strategic normal form games, see [18] for a detailed overview.

Assume a collection of n agents where each agent i has an individual finite set of actions A_i. The number of actions in A_i is denoted by $|A_i|$. The agents repeatedly play a single stage game in which each agent i independently selects an individual action a from its private action set A_i. The combination of actions of all agents at any time-step, constitute a joint action or action profile $\vec{a}$ from the joint action set $\mathbb{A} = A_1 \times \cdots \times A_n$. A joint action $\vec{a}$ is thus a vector in the joint action space $\mathbb{A}$, with components $a^i \in A_i, i : 1 \ldots n$.

	a_{21}	a_{22}
a_{11}	(1,−1)	(−1,1)
a_{12}	(−1,1)	(1,−1)

Fig. 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

With each joint action $\vec{a} \in \mathbb{A}$ and agent i a distribution over possible rewards is associated, i.e. $r_i : \mathbb{A} \rightarrow IR$ denotes agent i's expected payoff or expected reward function. The payoff function is often called the utility function of the agent, it represents the preference relation each agent has on the set of action profiles. The tuple $(n, \mathbb{A}, r_{1...n})$ defines a single stage strategic game, also called a normal form game.

Usually 2 player strategic games are presented in a matrix, an example is given in Fig. 1 and following. A row player, i.e. agent 1 and column player, i.e. agent 2 are assumed. The rows and columns represent the actions for respectively the row and column player. In the matrix the (expected) rewards for every joint action can be found; the first(second) number in the table cell is the (expected) reward for the row(column) player.

The agents are said to be in a Nash equilibrium, when it is the case that no agent can increase its own reward by changing its strategy when all the other agents stick to their equilibrium strategy. So, there is no incentive for the agents to play another strategy than their Nash equilibrium strategy. Nash proved the following important existence result [16]:

Theorem 1 *Every strategic game, which has finitely many actions has at least one mixed Nash equilibrium.*

Despite the existence theorem above, describing an optimal solution for strategic games is not always easy. For instance an equilibrium point is not necessarily unique and when more than one equilibrium point exists they do not necessarily give the same utility to the players. Even more, an equilibrium point does not take into account the relation between individual and group rationality.

Pareto optimality is a concept introduced to account for group rationality. An outcome of a game is said to be Pareto optimal if there exists no other outcome, for which all players simultaneously perform better. The classical example of a situation where individually rationality leads to inferior results for each agent, is the Prisonner's Dilemma game, see Fig. 2. This game has just one Nash equilibrium which is pure i.e. joint action (a_{12}, a_{22}). However, it is the only pure strategy that is not Pareto optimal. Joint action (a_{11}, a_{21}) is obviously superior; it is the only strategy which Pareto dominates the Nash equilibrium, however it is not a Nash equilibrium itself.

A special situation occurs when the individual utility of the agents coincides with the joint utility of the group. These games are called common interest games. In common interest games the agents' rewards are drawn from the same distribution. As a consequence, at least one Pareto optimal Nash equilibrium exists in common interest games and the agents' learning objective is to coordinate on one of them. All identical payoff games, i.e. games that always give identical payoffs to all the agents, are common interest games. Figure 3 shows two examples of identical payoff games, that are non-trivial from a coordination point of view.

	a_{21}	a_{22}
a_{11}	(5,5)	(0,10)
a_{12}	(10,0)	(1,1)

Fig. 2 The Prisoner's Dilemma Game: 2 prisoners committed a crime together. They can either confess their crime (i.e. play the first action) or deny it (i.e. play the second action). When only one prisoner confesses, he takes all the blame for the crime and receives no reward, while the other one gets the maximum reward of 10. When they both confess a reward of 5 is received, otherwise they only get 1

	a_{21}	a_{22}	a_{23}
a_{11}	11	−30	0
a_{12}	−30	7	6
a_{13}	0	0	5

	a_{21}	a_{22}	a_{23}
a_{11}	10	0	k
a_{12}	0	2	0
a_{13}	k	0	10

Fig. 3 Left: The climbing game, an identical payoff game from [4]. The Pareto optimal Nash equilibrium (a_{11}, a_{21}) is surrounded by heavy penalties. Right: The penalty game, an identical payoff game from [4]. Mis-coordination at the Pareto optimal Nash equilibria (a_{11}, a_{21}) and (a_{13}, a_{23}) is penalized with k

	a_{21}	a_{22}
a_{11}	$(2,1)$	$(0,0)$
a_{12}	$(0,0)$	$(1,2)$

Fig. 4 The Bach/Stravinsky game: 2 friends want to spend their evening together, but they have conflicting interests. They independently choose between a Bach concert or a Stravinsky concert. When both choose the same concert, a reward is given, according to their preference

In contrast to the common interest games mentioned above, unique rational solutions are difficult to define for general n-person non-zero-sum games. For instance, the Bach/Stravinsky game[1] given in Fig. 4 has 3 Nash equilibria. There are two pure Nash equilibria, i.e. joint actions (a_{11}, a_{21}) and (a_{12}, a_{22}). Both of them are Pareto optimal as they represent conflicting objectives for the two agents. There is also a unique mixed Nash equilibrium, i.e. strategy $((2/3, 1/3), (1/3, 2/3))$ where the first agent plays action a_{11} with probability 2/3 and action a_{12} with probability 1/3, while agent 2 plays action a_{21} with probability 1/3 and action a_{22} with probability 2/3. If we compute the expected payoffs for the mixed Nash equilibrium, we get a payoff of 2/3 for both agents. This is definitely fair to the agents as they get both the same payoff, however it is less than what they would get from either of the pure Nash equilibria. So the question arises, what is a rational solution for this game. In sect. 4 we introduce a periodical policy as an optimal fair solution for conflicting interest games like the Bach/Stravinsky game. In the periodical policy both equilibria are played in turns.

[1] Also known as the battle of the sexes game.

3 Learning automata

ESRL agents are simple reinforcement learners, more in particular they use a learning automata update scheme. In this section we give an overview of learning automata theory, in which we focus on automata games.

A learning automaton formalizes a general stochastic system in terms of states, actions, state or action probabilities and environment responses, see [15,22]. A learning automaton is a precursor of a policy iteration type of reinforcement learning algorithm [21] and has some roots in psychology and operations research. The design objective of an automaton is to guide the action selection at any stage by past actions and environment responses, so that some overall performance function is improved. At each stage the automaton chooses a specific action from its finite action set and the environment provides a random response.

In its current form, a learning automaton updates the probabilities of its various actions on the basis of the information the environment provides. Action probabilities are updated at every stage using a reinforcement scheme T.

Definition 1 A stochastic learning automaton is a quadruple $\{A, \beta, p, T\}$ for which A is the action set of the automaton, β is a random variable in the interval $[0, 1]$ and denotes the environment response, p is the action probability vector of the automaton and T denotes the update scheme.

A linear update scheme that behaves well in a wide range of settings[2] is the linear reward-inaction scheme, denoted by $(L_{\text{R-I}})$. The philosophy of this scheme is essentially to increase the probability of an action when it results in a success and to ignore it when the response is a failure. The update scheme is given by:

Definition 2 The linear reward-inaction update scheme for binary environment responses updates the action probabilities as follows: given that action a_i is chosen at time t then if action a_i was successful:

$$\begin{aligned} p_i(t+1) &= p_i(t) + \alpha(1 - p_i(t)) \\ &\quad \text{for action } a_i \\ p_j(t+1) &= p_j(t) - \alpha p_j(t) \\ &\quad \text{for all actions } a_j \neq a_i \end{aligned}$$

Otherwise when action a_i failed, nothing happens.

The constant α is called the reward or step size parameter and belongs to the interval $[0, 1]$. The $(L_{\text{R-I}})$ scheme as given here is called the P-model version, meaning that the environment response β is binary, i.e. the action was a success or a failure. In stationary environments $(p(t))_{t>0}$ is a discrete-time homogeneous Markov process and convergence results for $(L_{\text{R-I}})$ are guaranteed [15]. Despite the fact that multi-automata settings are non-stationary, the $(L_{\text{R-I}})$ scheme is still appropriate in learning automata games.

[2] $(L_{\text{R-I}})$ is what is called absolutely expedient and ϵ optimal in all stationary random environments. This means respectively that the expected average penalty for a given action probability is strictly monotonically decreasing with n and that the expected average penalty can be brought arbitrarily close to its minimum value.

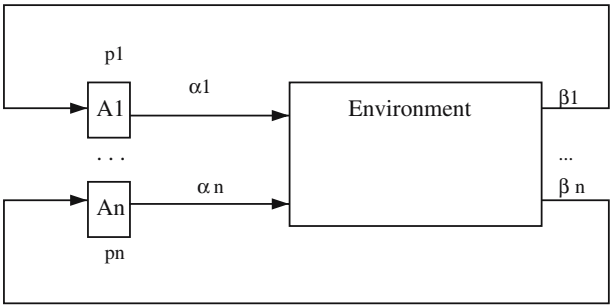

Fig. 5 Automata game formulation

3.1 Learning automata games

Automata games were introduced to see if automata could be interconnected in useful ways so as to exhibit group behavior that is attractive for either modeling or controlling complex systems.

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 jth automaton. Correspondingly the outcome is now also a vector $\beta(t) = (\beta^1(t) \cdots \beta^n(t))$. 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 (See Fig. 5).

The following results were tested and proved in [15]: In identical payoff games as well as some non-zero-sum games it is shown that when the automata use a ($L_{\text{R-I}}$) scheme the overall performance improves monotonically. Moreover if the identical payoff game is such that a unique, pure Nash equilibrium point exists, convergence is guaranteed. In cases were the game matrix has more than one pure equilibrium the ($L_{\text{R-I}}$) scheme will converge to one of the Nash equilibria. Which equilibrium is reached depends on the initial conditions.

In [24] the dynamics of several reinforcement learning algorithms (including the ($L_{\text{R-I}}$) scheme) is studied by mapping the algorithms into the replicator dynamics from evolutionary game theory, [19]. Paths with different starting points induced by the learning process of the ($L_{\text{R-I}}$) scheme are visualized in Fig. 6 for respectively the Bach/Stravinsky game of Fig. 4 and the matching pennies game of Fig. 1. The agents' probabilities of playing the first action are plotted against each other. In the first game two pure Nash equilibria exist, and the ($L_{\text{R-I}}$) agents are able to find them. Which equilibrium is reached depends on the initialization of the agent's action probabilities. The last game has only one mixed equilibrium, in this case the agents converge to a pure joint action, which is not Nash. In general for the ($L_{\text{R-I}}$) scheme, the following can be proved [25].

Lemma 1 *With probability* 1 *a team of* $L_{\text{R-I}}$ *learning automata will always converge to a limit in which all players play some pure strategy.*

An important issue here is the dependence of the convergence on the size of the step-size α. As shown in the following sections learning automata games of ($L_{\text{R-I}}$) schemes will form the basis for ESRL.

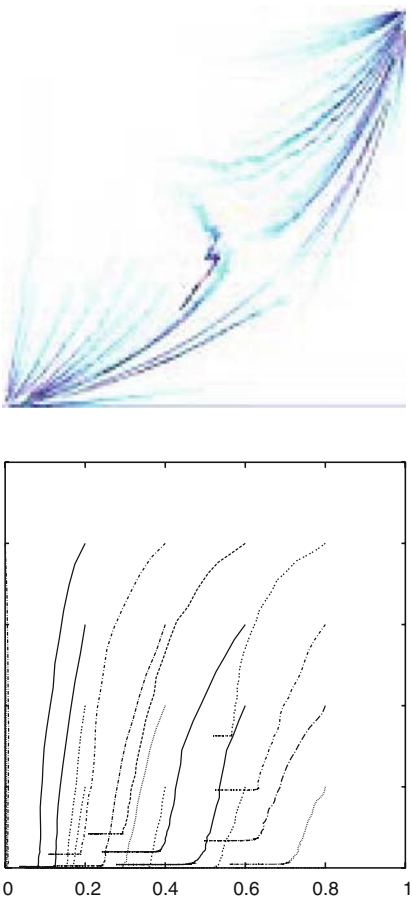

Fig. 6 Sample paths induced by the (L_{R-I}) learning automata scheme for the Bach/Stravinsky game of Fig. 4 (top). Sample paths induced by the (L_{R-I}) learning automata scheme for the matching pennies game of Fig. 1 (Bottom). The *x*-axis represents the probability with which the row player plays its first action, while the *y*-axis represents the probability with which the column player plays it first action

4 ESRL in common interest games

In this section ESRL is introduced for stochastic common interest games. The objective of learning in these games is easily defined, since a Pareto optimal Nash equilibrium in pure strategies always exists. We will focus on identical payoff games, however the approach can be applied to more general common interest games.

The main characteristic of ESRL agents is the way they coordinate their exploration. Phases in which agents act independently and behave as selfish optimizers are alternated by phases in which agents are social and act so as to optimize the group objective. The first type of phase is called an exploration phase. In this phase, an agent is nothing more than a learning automaton or a policy hill-climber. Convergence results of automata games assure us that the agents converge to a pure joint Nash equilibrium in each exploration phase. The second type of phase is called a synchronization phase. In this phase, the solution converged to in the previous exploration phase is evaluated and removed from the joint action space. In this way new attractors can be explored. The removal is achieved by letting each agent exclude its private action, that is involved in the solution found in the previous exploration phase, from its private action space. Agents alternate between exploration and

	a_{11}		
	a_{31}	a_{32}	a_{33}
a_{21}	0.9	0.1	0.1
a_{22}	0.1	0.1	0.1
a_{23}	0.1	0.1	0.1

	a_{12}		
	a_{31}	a_{32}	a_{33}
a_{21}	0.1	0.1	0.1
a_{22}	0.1	0.6	0.1
a_{23}	0.1	0.1	0.1

	a_{13}		
	a_{31}	a_{32}	a_{33}
a_{21}	0.1	0.1	0.1
a_{22}	0.1	0.1	0.1
a_{23}	0.1	0.1	0.4

Fig. 7 The guessing game – a diagonal game with 3 players. The Nash equilibria are located on the diagonal of the normal form matrix. The game is given in stochastic form; only expected rewards are given. With each joint action an expected reward is associated, which represents the probability of success (reward = 1) for that action

synchronization phases to efficiently search a shrinking joint action space in order to find the Pareto optimal Nash equilibrium.

We distinguish two ESRL versions depending on the type of the game. In the first type, the removal of a Nash equilibrium does not cut out other Nash equilibria from the joint action space. These games will be referred to as diagonal games. In the second ESRL version, this diagonal property is not assumed, as such the technique will need random restarts. As will be shown, both ESRL variants only assume independent agents; no agent communication is necessary.

4.1 ESRL in diagonal games

Consider the following diagonal property:

Definition 3 A strategic game is called diagonal if for all pure Nash equilibria the following condition holds: if $\vec{a} \in (A_1 \times A_2 \times \cdots \times A_n)$ is a pure Nash equilibrium, then the other pure Nash equilibria should be located in the action sub-space $(A_1 \backslash \{a^1\} \times A_2 \backslash \{a^2\} \times \cdots \times A_n \backslash \{a^n\})$

This means that the n agents can safely exclude their action involved in the current Nash equilibrium, without losing other Nash equilibria in the reduced joint action space. A 3-player example is given in Fig. 7. The pure Nash equilibria are located on the diagonal of the matrix. For all diagonal games a suitable permutation of each players' actions exists so that the pure Nash equilibria are situated on the diagonal of the game.[3] Note, however, that the actual location of the Nash equilibria is not important for the learning algorithm, so this permutation need not be applied.

The ESRL algorithm for diagonal games alternates between phases of independent learning, called exploration phases and synchronization phases. During the exploration phase the agents use the $L_{\text{R-I}}$ learning automata scheme of Eq. 2. Since a team of $L_{\text{R-I}}$ agents will always converge to a pure strategy; see Lemma 1, each agents' action probability vector $\vec{p}$ will converge to a pure action, i.e. there exists an action a_i in each agents' private action space for which $p_i \approx 1$. The pseudo code of the exploration phase is given in Algorithm 1. The agent keeps on selecting actions and updating its action probabilities until a predefined, sufficiently large number N of time steps have passed. During this period of time, the agents should be able to approach a pure joint

[3] To see this, take a joint action which constitutes a Nash equilibrium. For each agent, take the private action that is part of this Nash equilibrium and make it action 1. If still present, take another Nash equilibrium and again, for each agent take the agent's private action that belongs to this second Nash equilibrium and make it action 2. Since the game is diagonal, no action can be part of more than one Nash equilibrium, so this renumbering of actions is possible. Continue this process until all Nash equilibria are considered. Number the rest of the agents' private actions randomly. All Nash equilibria will now be of the form $(a_i, a_i, \ldots, a_i)$.

action and learn its expected reward, so this number N should be chosen carefully.[4] The *WINDOW* parameter is used for calculating a sliding average $average(t)$ of the reward the agent collects. At the end of the exploration phase this number approaches the expected reward of the pure joint action reached.

Algorithm 1 Exploration phase for ESRL agent j in common interest games

Initialize
time step $t \Leftarrow 0$,
average payoff $average(t) \Leftarrow 0$ and
for all actions i that are not excluded:
initialize action probability $p_i(t)$:
uniformly in case of diagonal games
randomly in case of general games
repeat
$t := t + 1$
choose action a_i from A_j probabilistically using p_t
execute action a_i, observe immediate reward r in $\{0, 1\}$
if $r = 1$ (action a_i was successful) **then**
update action probabilities $p(t)$ as follows: $L_{\text{R-I}}$

$$p_i(t) \Leftarrow p_i(t-1) + \alpha(1 - p_i(t-1)) \quad \text{for action } a_i$$
$$p_{i'}(t) \Leftarrow p_{i'}(t-1) - \alpha p_{i'}(t-1) \quad \text{for all actions } a_{i'} \text{ in } A_j \text{ with: } a_{i'} \neq a_i$$

end if
set $average(t) \Leftarrow \frac{t-1}{WINDOW} average(t-1) + \frac{r}{WINDOW}$
until $t = N$

When the first exploration phase is finished, the synchronization phase starts, see Algorithm 2. If a_i is the private action agent j has converged to, the value $q(a_i)$, which represents the quality of action a_i, is assigned the average payoff $average(N)$ that was collected during the previous exploration phase. Next this value is compared to the value of the temporarily excluded action, which holds the best action *best* seen so far.

Initially, no action is excluded and *best* is set to the first action that is selected. A second exploration phase starts in the reduced action space during which the agents converge to a new pure Nash equilibrium. Again a synchronization phase follows. The action to which agent j converged after the second cycle can now be compared with action *best*. The one with the highest value is again temporarily excluded and becomes action *best*, the inferior action is permanently excluded.

Let us assume that the agents are symmetric, that is they have the same number of actions l in their private action space.[5] Then, this idea is repeated until all actions of all agents' private action space have been excluded. At the end the action that is part of the best Nash equilibrium is the one that is temporarily excluded, so this one is made available for each agent.

In terms of speed, this means that when the individual action sets A_j have l elements, exactly l alternations between exploration and synchronization should be run

[4] For now, the number of iterations done before convergence is chosen in advance and is thus a constant. We will discuss later how agents can find out by themselves that they have converged.

[5] This assumption can be relaxed.

before the optimal solution is found. Note that learning as well as excluding actions happens completely independently. Only synchronization is needed so that the agents can perform the exclusion of their actions at the same time.[6]

Algorithm 2 Synchronization phase for ESRL agent j in diagonal games

if $T \leq |A_j|$ (T = number of exploration phases played) **then**
  $T \Leftarrow T + 1$
  get action a_i in A_j for which the action probability $p_i \approx 1$ (Lemma 1)
  set action value: $q(a_i) \Leftarrow average(N)$
  if temporarily excluded action *best* exists and $q(a_i) > q(best)$ **then**
    exclude action *best* permanently : $A_j \Leftarrow A_j \backslash \{best\}$ and
    temporarily exclude action a_i : $A_j \Leftarrow A_j \backslash \{a_i\}$, $best \Leftarrow a_i$
  else if temporarily excluded action *best* exists and
  $q(a_i) \leq q(best)$ **then**
    exclude action a_i permanently : $A_j \Leftarrow A_j \backslash \{a_i\}$
  else if temporarily excluded action *best* does not exists **then**
    temporarily exclude action a_i : $A_j \Leftarrow A_j \backslash \{a_i\}$, $best \Leftarrow a_i$
  end if
  if all actions are excluded **then**
    free temporarily excluded action *best* : $A_j \Leftarrow \{best\}$
  end if
end if

4.2 ESRL with random restarts

Of course in non-diagonal common interest games, a lot of Nash equilibria may be cut out, because in these games it is possible that an agents' private action belongs to more than one equilibrium point. So, when the agents have only one action left in their private action space, there is no guarantee that the Pareto optimal joint action was encountered.

Therefore, for general common interest games, the ESRL synchronization phase, of which the pseudo code is given in Algorithm 3, is adopted. Now, actions can only be temporarily excluded. When the action space has reduced to a singleton, the agent restores it, so that the exploration/synchronization alternation can be repeated. Agents take random restarts after each synchronization phase—this means that the agents put themselves at a random place in the joint action space by initializing their action probabilities randomly. In the exploration phase of Algorithm 1, $p(t)$ is now initialized as a random vector in $[0, 1]^k$ so that $\Sigma_{i=1}^{k} p_i(t) = 1$. Here, k is the number of non-excluded actions in the agent j's action space.

Because of the restarts, it is possible for agent j to reach the same private action a_i more than once. Therefore, action values $q(a_i)$ (q_i for short) are updated optimistically, i.e. updates for a_i only happen when a better $average(N)$ for a_i is reached during the learning process. The best action seen so far, referred to as *best*, is remembered. After a predefined number M of exploration/synchronization phases each agent selects action *best* from its private action space. As a result the agents coordinate on the first best joint action visited.

[6] This assumption can be relaxed for asynchronous games, see Sect. 7.

Algorithm 3 Synchronization phase for ESRL agent j in common interest games

if $T \leq M$ (M = total number of exploration phases to be played) **then**
  $T \leftarrow T + 1$
  get action a_i from A_j for which the action probability $p_i \approx 1$
  update action values vector q as follows:

$$q_i(T) \Leftarrow \max\{q_i(T-1), average(N)\}$$
$$q_{i'}(T) \Leftarrow q_{i'}(T-1) \quad \text{for all actions } a_{i'} \text{ in } A_j \text{ with: } a_{i'} \neq a_i$$

  if $q_i(T) > q_{best}(T)$ **then**
    set $best \Leftarrow a_i$ {keep the best action until now in the parameter $best$ and its value in q_{best}}
  end if
  if $|A_j| > 1$ **then**
    temporarily exclude action $a_i : A_j \Leftarrow A_j \backslash \{a_i\}$
  else
    restore the original action set: $A_j \Leftarrow \{a_1, \ldots, a_l\}$ with $l = |A_j|$
  end if
else if $T = M$ **then**
  set $A_j \Leftarrow \{best\}$
end if

4.3 Analytical results

In this section we analyze the above algorithms analytically. We rely on the result of Lemma 1, which states that with α small, for any $\delta > 0, \epsilon > 0$ there exists $T = T(\delta, \epsilon) \in \mathbb{N}$ such that for all $t \geq T$ we have that $\vec{p}(t)$ is ϵ-close to an absorbing state vector $\vec{p}$ with a probability of at least $1 - \delta$. Assume for simplicity reasons, that all players have an equal number of actions l in their private action space. Then we can state the following result:

Theorem 2 *Given $\epsilon > 0$ and $\delta > 0$, ESRL in diagonal games is able to let the agents approach each pure Nash equilibria ϵ-close with a probability bigger then $1 - \delta$ in maximally l exploration phases.*

Proof In a diagonal game, an agents' private action can belong to a single pure Nash equilibrium only. This means that there are maximally l pure Nash equilibria present in the game. Furthermore a pure Nash equilibrium of the original game will still be a Nash equilibrium of the reduced game which results after all agents excluded one action from their private action space. This can be seen as follows; by excluding joint actions from the game matrix, the number of constraints in the definition of a Nash equilibrium [18] only reduces. As such, a pure Nash equilibrium of the original game, will still be a Nash equilibrium in the sub-game after the exclusions.

During each exploration phase i, the $L_{\text{R-I}}$ learning automata of the agents approach a pure joint action ϵ close with a probability at least $1 - \delta$ in $T_i(\epsilon, \delta)$ time steps, see Lemma 1. When the current joint action space has pure Nash equilibrium points, one of those will be approached, see Sect. 3.1.

Putting this together means that during each exploration phase the agents approach a different pure joint action. Moreover, every pure Nash equilibrium of the original game will be reached in one of the l exploration phases. □

As the agents agree on which joint action they visited is the best one, they can jointly select it. When the length N of each exploration phase, is so that the agents are

able to find each pure Nash equilibria and approach the equilibria's expected rewards sufficiently close, then the best joint action visited is the Pareto Optimal one. To guarantee this, we take N bigger than each of the $T_i(\epsilon, \delta)$. For sampling reasons a constant period of time could be added to N so that the agents play the equilibrium for some period and the expected reward is approached more accurately. As will be explained later, it is fairly easy to let the agents decide on the length of this exploration period themselves.

To analyze ESRL learning in general common interest games, we rely on the convergence result of the Q-learning update rule in a single agent setting [21]. The Q-learning iteration rule is an update rule for state-action pairs. In a single stage game[7] this update rule for joint actions $\vec{a}$ becomes:

$$\begin{aligned} Q_{t+1}(\vec{a}) &= Q_t(\vec{a}) \text{ if } \vec{a} \text{ was not selected at time } t \\ Q_{t+1}(\vec{a}) &= (1-\alpha)Q_t(\vec{a}) + \alpha\, r(\vec{a}) \text{ otherwise.} \end{aligned} \tag{1}$$

Formulating the Q-learning update rule for joint actions, is as if one single super-agent whose action set consists of all joint actions is learning the optimal joint action. As such the convergence assumptions for Q-learning hold, considered that every joint action is visited infinitely often [23]. In the following lemma, the link between q_j value updates of Algorithm 3 and Q-value updates for joint actions is established.

Assume a time line T which counts the number of exploration/synchronization phases played by ESRL Algorithms 1 and 3, and assume *Visited* to be the set of joint actions the agents converged to after T exploration/synchronization phases. Then we can prove that the value q_i ESRL agent j has learned for action a_i after T exploration phases equals the Q-value that the super-agent should have learned for the best joint action $\vec{a}$ that has action a_i as its ith component and that was visited during the previous $t = T \times N$ time steps.

Theorem 3 *Given the action-selection process of ESRL, for every index T, agent j and private action a_i in action space A_j:*

$$q_i(T) = \max_{\vec{a} \in \textit{Visited with } a^j = a_i} Q_t(\vec{a}) \;\; \textit{where} \;\; t = T \times N \tag{2}$$

Proof By induction on T:

- T = 0: initially all action values are initialized to 0, as such the induction holds for the initial values.
- $T-1 \rightsquigarrow T$: Assume that the following induction hypothesis holds for index $T-1$, agent j and private action a_i in action space A_j:

$$q_i(T-1) = max_{\vec{a} \in \textit{Visited with } a^j = a_i} Q_t(\vec{a}) \text{ where } t = (T-1) \times N$$

The induction step which is left to prove, is that the above equation also holds for T.

Select agent j and assume that after exploration phase T, the agents converge to joint action $\vec{a}^T = (a_1^T, \ldots, a_n^T)$.

1. For all private actions a_i in action space A_j for which $a_i \neq a_j^T$ the induction step is trivially true since neither the value update scheme of Algorithm 3 nor the Q-learning update scheme in Eq. 1 perform an update in this case.

[7] As all rewards must have an equal impact on the Q-values in a single stage game, there is no need for a discount factor γ.

2. $a_i = a_j^T$ then because of the induction hypothesis we have:

$$\begin{aligned} q_i(T) &= \max\{q_i(T-1), \text{average}(N)\} \\ &= \max\{max_{\vec{a} \in \text{Visited with } a^j = a_i} Q_{(T-1)N}(\vec{a}), \text{average}(N)\} \\ &= \max_{\vec{a} \in \text{Visited} \cup \{a^T\} \text{ with } a^j = a_i} Q_{TN}(\vec{a}) \end{aligned}$$

□

Again, when N is so that $Q_{TN}(\vec{a}) \approx Q(\vec{a})$, i.e., the length of an exploration phase is long enough so that the agents can approximate the expected value of $\vec{a}$ sufficiently closely and also when M is large enough so that the set *Nash* of all Nash equilibria in the game is included in *Visited*, i.e., $Nash \subset Visited$, then the convergence results of Q-learning (see [23]) can be used to conclude that the agents can select a solution with an expected reward very close to that of the Pareto Optimal Nash equilibrium. Recall that N can be chosen by the agent itself and depends on when the agents thinks it's action probability vector has converged. M will depend on the learning time available to the agents. When time is up, the agents can select the best joint action visited up-till then.

4.4 Illustrations

Figure 8 shows the working of ESRL for the diagonal game of Fig. 7. The plot only gives information on one run of the algorithm. Agents can possibly find different Nash equilibria during the different phases, therefore average results do not make sense here. In the guessing game the 3 pure Nash equilibria are located on the diagonal. These are all found during the three exploration phases of ESRL (Fig. 8). The length of each exploration or learning phase is fixed here on 2000 time steps. From time-step 6000 on, the optimal joint Nash equilibrium is played, the agents' private action spaces are reduced to a singleton.

In Fig. 9 average results are given for ESRL agents with random restarts learning in joint action spaces of different sizes. Random common interest games, for which the expected payoffs for the different joint actions are uncorrelated random numbers between 0 and 1 are generated. ESRL assures that an important proportion of the Nash equilibria are visited. Almost 100% in small joint action spaces and still about 94% in the larger ones. It may seem strange that although the optimal Nash equilibria is visited, the players sometimes do not recognize it. This is a consequence of the fact that rewards were stochastic and when payoffs are very close to each other, the sampling technique used here by the agents is not able to distinguish them. So actually when convergence is not reached, agents still find extremely good solutions which are almost as good as the optimal one. We tested this by relaxing convergence conditions in the last two experiments of Fig. 9. We considered the agents to be converged when the difference in payoff of the joint action they recognized as the best and the true optimal one was no more than 0.0001 and 0.001, respectively. This increases the success rate considerably as can be seen in Fig. 9.

5 ESRL in conflicting interest games

In this section, we introduce ESRL for stochastic conflicting interest games. For now, we assume that conflicting interest means that each agent prefers another pure Nash

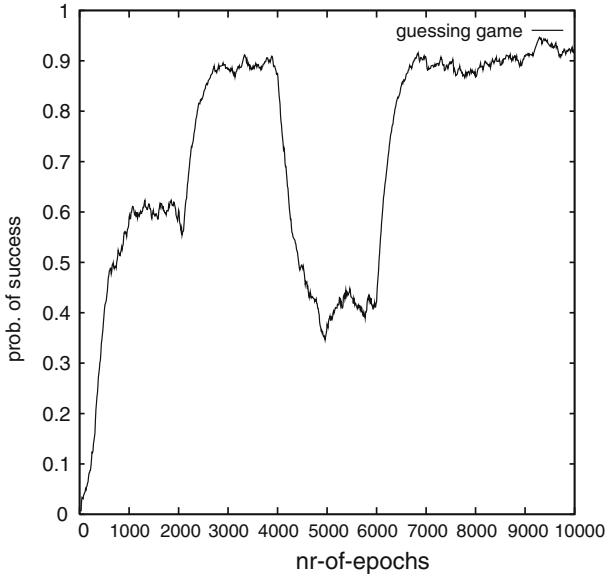

Fig. 8 The reward accumulated by diagonal ESRL agents in a typical run of the guessing game, given in Fig. 7. The step-size parameter $\alpha = 0.05$, the length of the exploration phases is 2000 time steps. Three exploration/synchronization phases took place. The agents respectively converge to Nash equilibrium (a_{12}, a_{22}, a_{32}) yielding an average payoff of 0.6, Nash equilibrium (a_{11}, a_{21}, a_{31}) yielding an average payoff of 0.9 and finally Nash equilibrium (a_{13}, a_{23}, a_{33}), yielding the best average payoff of 0.4. From time-step 6000 on, the agents play the Pareto optimal joint action

#agents / #actions	N	run time	% of optimal Nash equilibrium found	% of diff. Nash equilibria visited
2/3	2000	2.10^4	97	99.67
2/5	2000	10^5	95	100
3/3	2000	10^5	96	98.38
3/5	3000	25.10^4	87	93.67
5/3	2000	5.10^5	78	94.51
5/5	3000	6.10^6	36	94.39
5/5	3000	6.10^6	68(0.0001)	94.39
5/5	3000	6.10^6	98(0.001)	94.39

Fig. 9 Average results for common interest ESRL over 100 iterations. $\alpha = 0.005$

equilibrium. As such, a single common optimal solution concept for these games does not exist. Therefore, we propose periodical policies as optimal fair solution concepts that the agents should learn in this type of game. In a periodical policy the agents alternate between periods of time in which different pure Nash equilibria are played.

An example of a conflicting interest game is the Bach/Stravinsky game given earlier in Fig. 4. The game has 2 pure Nash equilibria which reflect the conflicting interest of the agents, and one fair mixed Nash equilibrium. The mixed Nash equilibrium is strategy $((2/3, 1/3), (1/3, 2/3))$, i.e. the first agent plays action a_{11} with probability $2/3$ and action a_{12} with probability $1/3$, while agent 2 plays action a_{21} with probability $1/3$ and action a_{22} with probability $2/3$. This mixed Nash equilibrium gives both agents an average payoff of $2/3$, which is fair but which is also less than what they would get from either of the pure Nash equilibria. A fair periodical policy would be to play both pure

Nash equilibria in turns, i.e. joint action (a_{11}, a_{21}) and (a_{12}, a_{22}). As a consequence, the average payoff for both agents will be 1.5.

The main idea of exploring selfish reinforcement learning agents is that agents exclude actions from their private action space so as to further explore a reduced joint action space. This idea is here used to let a group of agents learn a periodical policy. However, the agents may not have the same incentives to do this as in common interest games. As a result, we have to assume that agents will behave socially. Our ideas behind the social behavior of agents were motivated by the Homo Egualis society from sociology [6]. In a Homo Egualis society, agents do not only care about their own payoff, but also about how it compares to the payoff of others. A Homo Egualis agent is displeased when it receives a lower payoff than the other group members, but it is also willing to share some of its own payoff with the others when it is winning. Very limited communication between the agents will be needed to implement the Homo Egualis idea, i.e. the comparison of payoff to decide which agent is best off. For the rest, agents remain independent.

5.1 Learning a periodical policy with ESRL

The exploration phase and synchronization phase for ESRL in conflicting interest games is given in Algorithms 4 and 5. The exploration phase is more or less the same than the one given in Algorithm 1. However, here we present a version with real-valued rewards instead of binary rewards. The Q-learning update rule for stateless environments is used to calculate the average reward of private actions. For each action a_i in agent j's action space, a Q-value v_i is associated. These values are normalized into action probabilities p_i for the action selection process. Agent j now calculates a global average reward *average* for the whole learning process.

Algorithm 4 Exploration phase for ESRL agent j in conflicting interest games

Initialize
time step $t \Leftarrow 0$,
for all actions i that are not excluded :
initialize action values $v_i(t)$ and action probabilities $p_i(t)$
(assume $l = |A_j|$ and $k \leq l$ is the number of not excluded actions)
repeat
$t := t + 1$
choose action a_i in A_j probabilistically using $p(t-1) = (p_1(t-1), \ldots, p_l(t-1))$
take action a_i, observe immediate reward r in $[0, 1]$
with a given learning rate α, update the action value as follows:

$$\begin{aligned} v_i(t) &\Leftarrow (1-\alpha)v_i(t-1) + \alpha r \\ &\quad \text{for action } a_i \\ v_{i'}(t) &\Leftarrow v_{i'}(t-1) - \alpha v_{i'}(t-1) \\ &\quad \text{for all actions } a_{i'} \text{ in } A_j \text{ with : } a_{i'} \neq a_i \end{aligned}$$

normalize the action values $v_i(t)$ in probabilities $p_i(t)$
set the global average: $average \Leftarrow \frac{t-1}{t} average + \frac{r}{t}$
until $t = N$

In ESRL the best performing agent gives the other agents the chance to increase their payoff. All agents broadcast their global average *average* as well as the average

payoff $v_i(N)$ that was collected for the private action a_i to which the agent converged during the previous exploration phase. The agent who experienced the highest cumulative payoff and the highest average payoff during the last exploration phase, will exclude the action it is currently playing so as to give the other agents the opportunity to converge to a more rewarding situation. So after finding one of the Nash equilibria, all the joint actions containing the action of the best performing agent are temporarily removed from the joint action space. A new exploration phase is started in which the resulting action subspace is explored. During this second exploration phase the group will reach another Nash equilibrium and one of the other agents might have caught up with the former best agent. At this point, communication takes place and the currently "best" agent can be identified. Note that not only the reward accumulated in the previous exploration phase is used to select the best performing player, also the cumulative payoff or global average is needed to see if an agent needs more time to catch up with the others. Only the agent that is recognized as the best in the above sense, has action(s) excluded in its private action space. If some other agent becomes the best, the previous best agent can use its full original action space again.

In this ESRL version, the phases given by Algorithms 4 and 5 should alternate until the end of the learning process. As a result agents play a periodical policy, they alternate between the different Nash equilibria of the game.

Algorithm 5 Synchronization phase for ESRL agent j in conflicting interest games

$T \leftarrow T+1$
get action a_i in A_j for which the action probability $p_i((T-1)N) \approx 1$
broadcast action value $v^j = v_i((T-1)N)$ and global $average^j = average$ to all other agents
receive action value v^b and global average $average^b$ for all agents b
if $v^j > v^b$ and $average^j > average^b$ for all b **then**
  temporarily exclude action $a_i : A_j \Leftarrow A_j \backslash \{a_i\}$
else
  restore the original action set: $A_j \Leftarrow \{a_1, \ldots, a_l\}$ with $l = |A_j|$
end if

In Fig. 10 the evolution of the average reward received in the Bach/Stravinsky game of Fig. 4 is plotted. When the players use the classical Q-learning algorithm [21], they find for both games an unfair pure Nash equilibria as is shown in Fig. 10 (left). One of the players seems to win the game, while the other one performs sub-optimally. Fig. 10 (middle) shows the payoffs of the players when both players play the mixed Nash equilibrium strategy. Finally in Fig. 10 (right) the average payoff for the ESRL agents is given. The periodical policy they find is an alternation of the two pure Nash equilibria so that the agents' payoffs are equalized. The average payoff the ESRL agents receive is maximally fair. This is 1.5 for the standard Bach/Stravinsky game, see Fig. 10 (right).

Off course it will not always be possible to exactly equalize the payoffs of all agents. For instance in a game where the first agent always receives a better payoff than the second agent, playing a periodical policy will not equalize the payoffs. Even more, in some conflicting interest games a periodical policy may be outperformed by a Pareto optimal action which is not Nash. See for instance the game of Fig. 11. A periodical policy would switch between the conflicting Nash equilibria (a_{11}, a_{21}) and (a_{13}, a_{23}) yielding an average payoff of 4, while the Pareto optimal joint action (a_{12}, a_{22}) yields for both agents an average payoff of 7. So the agents should try to

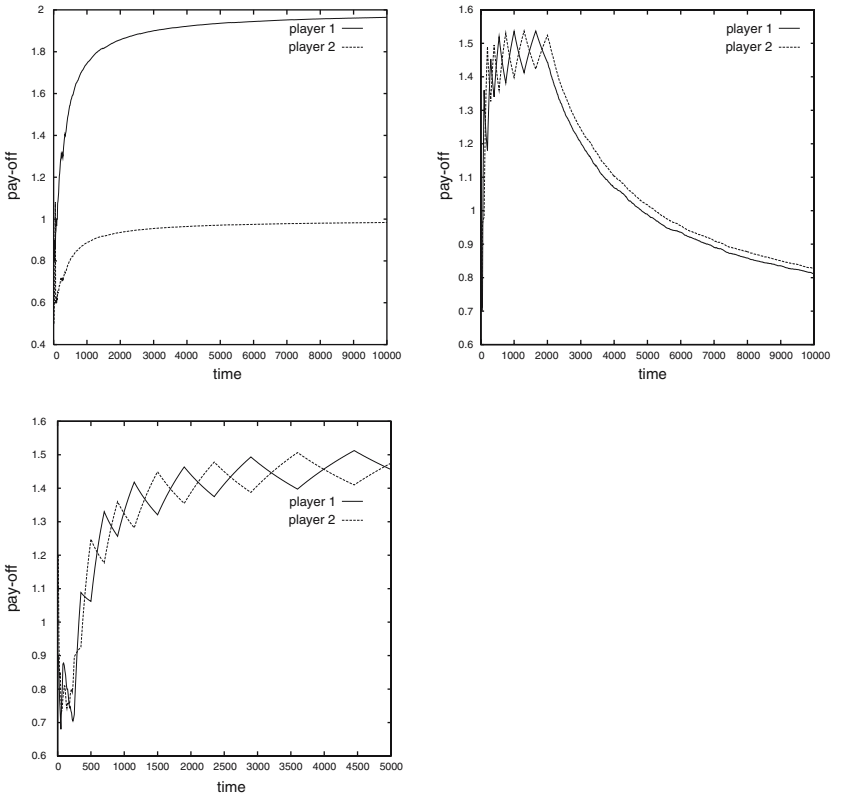

Fig. 10 Learning in the Bach/Stravinsky game of Fig. 4. Left: the average payoff for 2 Q-learners converging to one of the Nash equilibria, Middle: the average payoff for 2 agents playing the mixed Nash equilibrium, Right: the average payoff for 2 Homo Egualis ESRL agents playing a periodical policy

	a_{21}	a_{22}	a_{23}
a_{11}	$(7,1)$	$(0,0)$	$(0,0)$
a_{12}	$(0,0)$	$(7,7)$	$(0,10)$
a_{13}	$(0,0)$	$(10,0)$	$(1,7)$

Fig. 11 A 2 player with 2 conflicting Nash equilibria, (a_{11}, a_{21}) and (a_{13}, a_{23}) and a Pareto optimal joint action (a_{12}, a_{22}), which is not Nash

learn, when periodical policies are interesting, i.e. optimal and fair. In the next section ESRL agents are faced with games of which they do not know the type in advance. The goal of learning is to find an optimal fair solution, which may or may not be periodical.

6 Learning in non-zero sum games

In this section we generalize ESRL to general non-zero sum games, of which the agents do not know the type in advance, i.e. pure common interest or conflicting interest. In this case, the objective of the ESRL agents is to learn a good fair solution, which can

be a periodical policy when conflicting interests are present and which should be the Pareto Optimal solution when the agents' interests coincide.

6.1 The general synchronization phase

When the agents use the ESRL algorithm for learning in common-interest games, i.e. Algorithm 1 combined with Algorithm 3 they will know at the end of exploration whether the game they are playing is pure common interest or not. Indeed, when exploitation begins the agent will notice after a while if its performance achieves the same level that was reached during learning. If it does, the game must be common interest and the agents coordinate on the same joint action that the agents recognized as the best during learning. If not, clearly the agents did not coordinate on the same solution and thus either conflicting interests exists or the Pareto optimal solution is not found yet. Anyhow, in the latter situation exploration should start over again, because for now the agents only have individual information on the joint actions they visited, which is not enough to find an optimal fair (periodical) policy for all agents. So, preferably after every exploration phase the agents should broadcast their performance as in the synchronization phase of Algorithm 5, so that at the end of all exploration/synchronization phases, a fair possible (periodical) policy can be formed. The idea is to combine both previous versions of the synchronization phase. Then we can use random restarts to move more quickly through the joint action space, together with broadcasts to share the preference the agents have on the joint actions converged to. The generalized synchronization phase is given in Algorithm 6.

Each agent keeps the payoff information from the previous exploration phase in a history list, called *hist*. An element of *hist* consists of the private action the agent converged to and a payoff vector, containing the payoff received by all the agents during the previous phase. So, each agent stores its own private action combined with a payoff vector $\vec{v}$. This vector is the same for all agents. Important is that the list *hist* keeps the original order of its elements. At the end of all phases, the Pareto optimal elements of list *hist* are filtered. Since this operation only depends on the payoff vectors, each agent filters the same elements from its list *hist*. With these remaining elements the agent can now build a policy. Beforehand, the system designer can specify what this policy should look like. He can set the policy of each agent to action $a_{\min}$, for which the associated payoff vector $\vec{v}$ is most fair in the sense that the variation in payoff between the players, denoted by s is minimal for $\vec{v}$. Since the agents have the same payoff vectors in the same order in their list, they can independently set this policy that allows them to select the best, fair joint action they encountered together during learning. A periodical policy can also be built; for instance include in each agent's policy those actions, whose associated payoff vectors have a deviation which is close to the deviation of the most fair payoff vector in the history list. This closeness can be set using parameter δ. Other policies with all elements of the history list can also be defined. As such, different agents' objectives can be specified at the beginning of learning.

A practical improvement that was added, is that the length of the exploration phase is no longer preset. Instead, the agents decide for themselves when their action probabilities have converged sufficiently and when this is the case, they broadcast their payoff. In practice this happens when an action exists, for which the probability of being selected is more than a predefined number $0 \ll \zeta < 1$. The synchronization phase starts when the agent received the broadcast information from all other agents. However, not implemented here, statistical tests could be added to help the

Algorithm 6 Synchronization phase for ESRL agent j in non-zero sum games. The agents' objective is to find the optimal fair solution

if $T \leq M$ (M = total number of exploration phases to be played) **then**
  $T \Leftarrow T+1$
  get action a_i in A_j for which $p_i((T-1)N) \approx 1$
  broadcast action value $v^j = v_i((T-1)N)$ to all other agents
  receive action value v^b of all agents b
  update history *hist* with action (a_i) and payoff vector $\vec{v} = (v^1, \ldots, v^n)$, i.e.:

$$hist(T) \Leftarrow hist(T-1) \cup (a_i, \vec{v})$$

  if $|A_j| > 1$ **then**
    temporarily exclude action $a_i : A_j \Leftarrow A_j \backslash \{a_i\}$
  else
    restore the original action set: $A_j \Leftarrow \{a_1, \ldots, a_l\}$ with $l = |A_j|$
  end if
else if $T = M$ **then**
  collect Pareto optimal actions from history: $PO(hist)$
  compute for every $\vec{v} \in PO(hist)$, deviation s i.e.

$$s(\vec{v}) = \frac{\sum_j^n (v^j - \frac{\sum_j^n v^j}{n})^2}{n-1}$$

  set policy : take $\delta \geq 0$, set a_{min} :

$$a_{min} \Leftarrow \{a_i | (a_i, \vec{v}) \in PO(hist) \text{ and } s(\vec{v}) \text{ is minimal}\}$$

  if $\delta = 0$ **then**

$$policy \Leftarrow a_{min}$$

  else
    make policy periodical:

$$policy \Leftarrow \{a_i | (a_i, \vec{v}) \in PO(hist) \text{ and } s(\vec{v}) \leq s(\vec{v}_{min}) + \delta\}$$

  end if
end if

agents to decide when convergence has happened and when the average reward is approximated accurately. An example test is given in [3].

The general ESRL algorithm is demonstrated on the testbed games of Fig. 12. Important to notice is that the agents do not know the type of game beforehand. The results for a typical run of the games can be found in Fig. 13 (left) and (right), respectively. For both games the Pareto optimal equilibrium is found. The agents find all equilibria and at the end the set of Pareto optimal solutions drawn from their history is a singleton. The time steps needed before the optimum is reached, is comparable with the earlier versions of ESRL. Off course, just as before, when different equilibria exist with payoffs very close to each other, it can be difficult for the agents to recognize the genuine best one. In this case the agents may form a periodical policy or identify the other joint action as the best. However when this happens, all agents will decide to do so and the overall payoff information gathered will still be very close to the optimal.

	a_{21}	a_{22}	a_{23}
a_{11}	1.0	0.0	0.73
a_{12}	0.0	0.9	0.88
a_{13}	0.73	0.73	0.85

	a_{21}	a_{22}	a_{23}
a_{11}	1.0	0.83	0.0
a_{12}	0.83	0.86	0.83
a_{13}	0.0	0.83	1.0

Fig. 12 Stochastic versions of the climbing game and penalty game ($k = -50$). With each joint action an expected reward is associated that represents the probability of success for that action

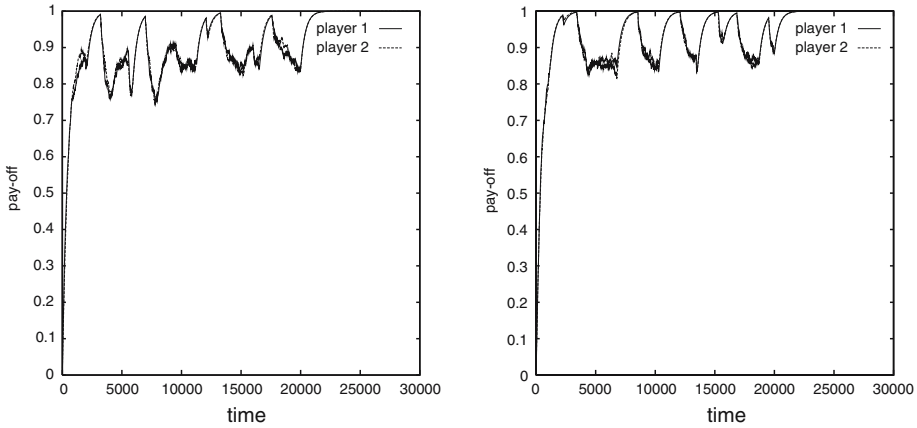

Fig. 13 Generalized ESRL learning in common interest games. Left: Average payoff in the climbing game of Fig. 12. Right: Average payoff in the penalty game of Fig. 12. Step size parameter $\alpha = 0.05$, while $\delta = 0$. The agents decide for themselves when an exploration phase can be ended

The behavior of generalized ESRL in conflicting interest games is illustrated by the results of Fig. 14. On the left side the average payoff for a typical run of the stochastic variant of the Bach/Stravinsky game is given, the right side gives the average payoff for a typical run of the stochastic version of the Prisoner's dilemma game. The stochastic version of both games we used, is given in Fig. 15. Again, the generalized ESRL behaves optimally. In the first game, a periodical policy consisting of the two pure Nash equilibria is found. In the second game, the best Pareto optimal Nash equilibrium is found.

How well the agents approximate the expected rewards of the joint actions influences the behavior of the algorithm. For instance, in the Bach/Stravinsky game, the deviation s of the 2 pure Nash equilibria could be seen as being different by the agents because of the stochasticity. As such the agents may decide to choose one of them instead of agreeing on a periodical policy. In this case it is a good idea to relax the assumption that only the action with the minimal deviation in the players' payoff is chosen. Instead all the actions for which the deviation of the associated payoff vector is close to minimal could be added to the policy.

Figure 16 gives results for randomly generated non-zero sum games with a varying number of agents n and actions l. The expected payoffs for the different joint actions are uncorrelated random numbers between 0 and 1. Results are averaged over 50 different games, each run 10 times. For these randomly generated games, as can be seen from Column 2 and 3 the number of Nash equilibria only slightly increases with enlarging joint action space, while the number of Pareto optimal joint actions increases rapidly. Columns 6 and 8 give average information on the optimal

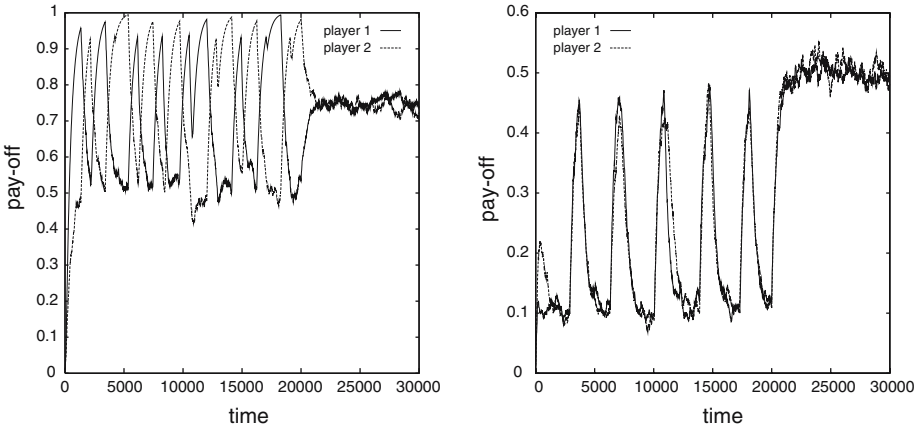

Fig. 14 Generalized ESRL learning in conflicting interest games. Left: Average payoff in a stochastic version of the Bach/Stravinsky game. Right: Average payoff in a stochastic version of the prisoner's dilemma. Step size parameter $\alpha = 0.05$ and $\delta = 0$. The agents end the exploration phase when their action probabilities have converged

	a_{21}	a_{22}
a_{11}	$(0.5, 0.5)$	$(0.0, 1.0)$
a_{12}	$(0.1, 0.0)$	$(0.1, 0.1)$

,

	a_{21}	a_{22}
a_{11}	$(1.0, 0.5)$	$(0.0, 0.0)$
a_{12}	$(0.0, 0.0)$	$(0.5, 1.0)$

Fig. 15 Stochastic versions of the Prisoner's Dilemma Game (left) and the Bach/Stravinsky game (right). With each joint action an expected reward is associated, which represents the probability of success (reward 1) for that action

solution, i.e., the Pareto Optimal solution present in the game, with minimal deviation s between the agents' payoff. How well the agents play the game can be found in Column 7 and 9. In all cases, the average best payoff found after learning (Column 7) and the corresponding deviation between the agents for this solution (Column 9) is very close to the optimal results given in Columns 6 and 8, respectively. So, even for large joint action spaces, the agents are able to find interesting Pareto optimal solutions, for which the payoff is fairly distributed between the agents in a moderate number of exploration/synchronization phases. For instance in the 5-player, 5-actions games on average only 124.88 phases were played, while in the 7-player, 3-actions games only 175.9 phases were played on average. A typical run for a randomly generated 5-agent game, with each agents having 5 actions (and thus a joint actions space of 3125 joint actions) is given in Fig. 17. In the left experiment of Fig. 17, total learning time, i.e. the time the agents use for playing exploration/synchronization is set to 80,000 time steps, whereas for the experiment on the right of Fig. 17, learning time is set to 200,000 time steps. In the first experiment with limited exploration time, a reasonable fair solution was found. By letting the agents play more exploration/synchronization pairs, the average performance of all the agents increases. In fact in the experiment on the right the agents play the optimal solution, as defined by the given objective.

1	2	3	4	5	6	7	8	9
n/l	#NE	#PO	% of NE found	% of PO found	mean of opt. sol.	mean of sol. found	s of opt. sol.	s of sol. found
3/3	0.88	8.54	92	42	0.78	0.76	0.013	0.015
4/4	0.92	43.44	59.92	22	0.84	0.81	0.005	0.006
5/5	1.08	270.14	36.66	8.25	0.87	0.82	0.003	0.004
7/3	2	703	57.5	10.57	0.77	0.73	0.005	0.009

Fig. 16 General ESRL agents in random generated non-zero sum games. $\alpha_{reward} = 0.005$, total learning time is maximal = 400.000 time steps for the large joint action spaces, $\delta = 0$ and the agents decide for themselves when their action probabilities have converged and thus when their exploration phase can be ended. The percentage of convergence to the optimal solution is 35% for the 5-player, 5-action experiment and 50% for the 7-player, 3-action experiment

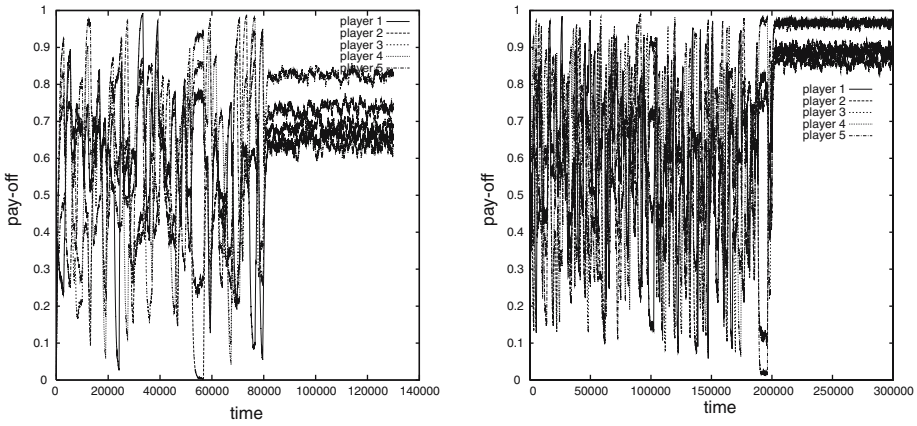

Fig. 17 Average payoff for generalized ESRL agents in a randomly generated game with 5 agents, each having 5 actions. Left: exploitation begins after 80,000 time steps. Right: exploitation begins after 200,000 time steps. Step size parameter $\alpha = 0.05$ and $\delta = 0$. The agents decide for themselves when an exploration phase can be ended

7 Adaptive load-balancing of parallel applications

In this section, the robustness of ESRL to delayed rewards and asynchronous action selection is illustrated with the problem of adaptive load-balancing parallel applications.

Load balancing is crucial for parallel applications since it ensures a good use of the capacity of the parallel processing units. Here we look at applications that put high demands on the parallel interconnect between nodes in terms of throughput. Examples of such applications are compression applications which both process important amounts of data and require a lot of computations. Data intensive applications usually require a lot of communication and are therefore dreaded for most parallel architectures. The problem is exacerbated when working with heterogeneous parallel hardware.

A traditional architecture to use, is the master–slave software architecture, illustrated in Fig. 18. In this architecture the slaves (one per processing unit) repeatedly request a chunk of a certain size from the master. As soon as the data has been processed

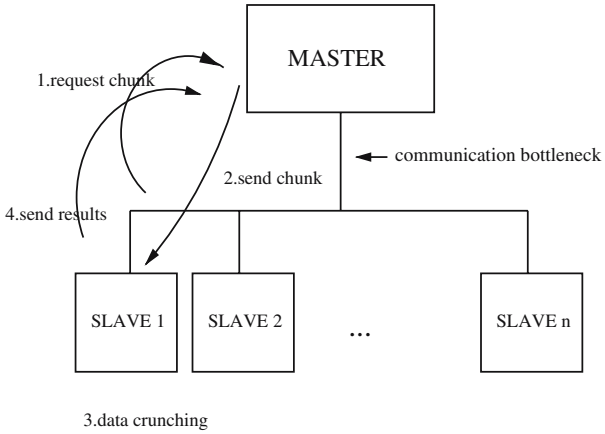

Fig. 18 Master/Slave computation model

the result is transferred to the master and the slave sends a new request. This classical approach is called job farming. This scheme has the advantage of being both simple and efficient. Indeed, in the case of heterogeneous hardware the load (amount of processed data) will depend on the processing speed of the different processing units. Faster processing units will more frequently request data and thus be able to process more data.

The bottleneck of data intensive applications with a master–slave architecture is the link connecting the slaves to the master. In the presented experiments all the slaves share a single link to their master. In an optimal load balancing setting, master and slave processors never become idle. This means that the processors are either communicating or computing. The master is able to serve the slaves constantly and all processors work at 100% efficiency. One can say that in this situation, the communication bandwidth of the master equals the computation power of the slaves. In all other cases, there will be either a clear communication or a computation bottleneck, what makes adaptive load balancing unnecessary. For instance, when the total computation power of the slaves is lower than the master's communication bandwidth, the master will be able to serve the slaves constantly and the slaves will work at full efficiency. However, the master has non-zero idle time and as such more slaves could be added to increase the speedup. This situation is depicted in Fig. 19. When the total computation power of the slaves increases, the master will get request at a high rate and the communication becomes the bottleneck. The slaves will block each other because they are waiting to be served and the total efficiency of the system drops. Figure 20 shows this latter situation.

In the following experiments we will only consider the settings in which computational power and communication matches. We will assume that the slave processors are heterogeneous and as such workload assignment and synchronization becomes necessary. The equilibrium point represents an ideal load distribution, meaning that the master can serve the slaves constantly and keep them fully busy.

7.1 Learning to request data using ESRL

The problem defined above can be viewed as a single stage common interest coordination problem. The actions of the agents consists of different chunk sizes the slave can request from the master. A joint action of this game is therefore a combination

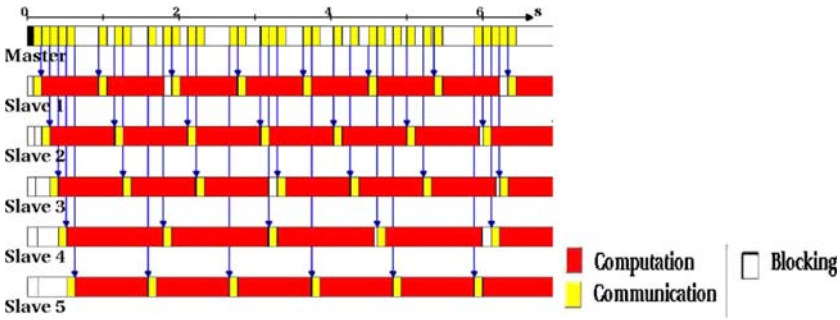

Fig. 19 Load Balancing with a clear computation bottleneck. Execution profile and overhead distribution

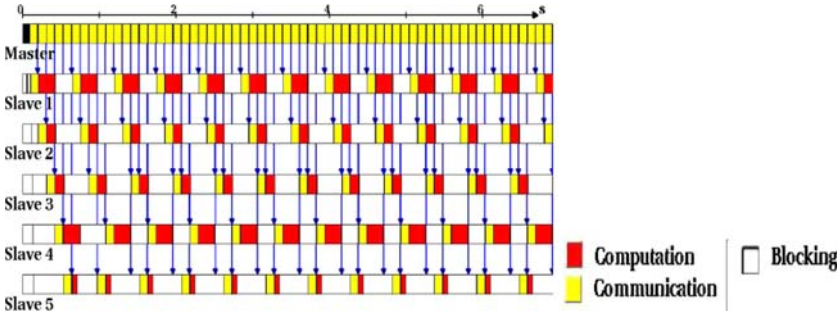

Fig. 20 Load Balancing with a clear communication bottleneck. Execution profile and overhead distribution

of chunk sizes. So instead of having all slaves asking the same chunk size of data to the master as in the job farming scheme, in our experiment ESRL slaves will learn to request a suitable chunk size, so as to minimize the total idle time of the master and the blocking times for the slaves. This in turn ensures a reduced total computation time of the parallel application. Because the actions of the agents are discretized (only a finite number of chunk sizes is allowed, because ESRL is based on finite action learning automata), it is possible that the load balancing equilibrium, i.e. the optimal load distribution that gives zero idle time for the master and the slaves is not a joint action of the game. However, with a given discretization the ESRL agents are able to find the best combination that is present in the game.

During one exploration phase, each slave converges to a particular chunk size. The immediate feedback r provided to the slaves while learning is the inverse blocking time, which is the time a slave has to wait before the master acknowledges its request for data. The expected reward sent back by the master to all ESRL slaves at the end of the exploration phase is the total computation time needed for the data that was processed in parallel during this phase. This reward coincides with the value of *average* used in Algorithm 1 and is the same for all slaves.

During the synchronization phase, *average* is used to optimistically update the action value of the action converged to. All agents then exclude this action from their private action space and a new exploration phase can be played in a reduced joint

action space. At the end of all exploration/synchronization phases, the best combination found is exploited.

Note that the slaves use a real-numbered reward version of ESRL Algorithms 1 and 3. Note also that since their is already communication between the master and the slaves, no extra communication overhead is created when the average reward is send back at the end of each exploration phase. There is also no communication necessary between the slaves. As such ESRL can easily be implemented without introducing new overheads in the parallel system.

This model has been simulated using the PVM [5] message-passing library to experiment with the different dimensions of the problem. The application has been designed not to perform any real computation, but instead it replaces the computation and communication phases by delays with equivalent duration. The global goal is minimizing the total computation time. A typical experiment consists of computation, communication and blocking phases. The data size to be processed is set to 10 GB, the communication speed is 10 MB/s and the possible chunk sizes (the agents' actions) that can be requested by the slaves are 1, 2 or 3 MB. For the ESRL algorithm, the number of time steps during one exploration phase is set to 100 s, in total 10 exploration/synchronization phases are used. The hardware heterogeneity of the setup is simulated so that there is a match between the system's communication bandwidth and total processing power. Figure 21 shows the time course of 3 ESRL slaves after learning. One can observe that the slaves have learned to distribute the requests nicely and hence use the link with the master efficiently. Slave 3 which is the fastest processor, with lowest granularity, is served constantly and has no idle time. Slave 2 has some idle time, however it is the slowest processor.

As shown in Fig. 22 we have run several experiments in different sized joint action spaces. The 4th column gives the absolute gain in total computation time for the ESRL agents/slaves compared to the total computation time for the classical farming approach. The 5th column gives the gain in total computation time, which is maximal possible for the given setting. The total computation time can only decrease when the idle time of the master is decreased. No gain is possible on the total communication time of the data. The last column gives the absolute gain compared to the gain that was maximally possible. In all experiments, considerable improvements were made. The ESRL agents are able to find settings in which the faster slave is blocked less frequently than the others.

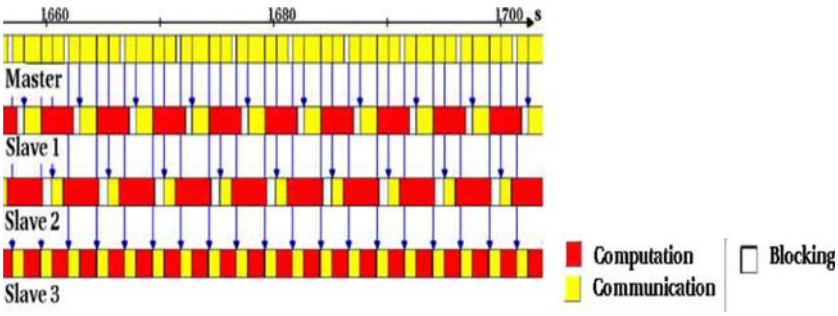

Fig. 21 Execution profile of the exploitation phase of a typical experiment with 3 slaves

#slaves	#actions	#joint actions	% abs. gain	% max gain	% rel. gain
2	3	9	8.40	13.54	62.00
2	4	16	8.32	13.54	60.99
2	5	32	6.64	13.54	49.02
3	3	27	4.69	14.14	33.17
3	5	125	4.33	14.14	30.65
5	3	243	1.93	13.13	10.60
5	5	3125	1.61	13.12	12.24
7	3	2187	2.40	12.20	16.91
7	5	78125	3.47	14.20	24.42

Fig. 22 Average gain in total computation time for ESRL agents compared to static load balancing (farming). The data size to be processed is set to 10 GB, the communication speed is 10 MB/s and the possible chunk sizes (the agents' actions) that can be requested by the slaves are 1, 2 or 3 MB. The number of time steps (in seconds) during one exploration phase is set to 100 s, in total 10 exploration/synchronization phases are used. For the last three experiments the data size was changed from 10 to 30 GB, the number of time steps N was increased from 100 to 200 time steps and the number M of exploration phases played was set to 20

Fig. 23 Average gain in master idle time and total blocking time of the slaves for ESRL agents compared to static load balancing (farming). The same parameter settings are used as in Fig. 22

#slaves	#actions	% idle gain	% blocking gain
2	3	55.6	50.7
2	4	54.3	51.3
2	5	47.7	38.7
3	3	30.3	20.7
3	5	26.8	20.7
5	3	4.0	−3.4
5	5	0.98	3.20
7	3	9.70	3.00
7	5	11.6	3.00

However, Fig. 22 shows that in the larger joint action spaces the performance gain drops. By adjusting the parameters of the ESRL algorithm in the last three experiments the gain in performance is again larger. The data size was changed from 10 to 30 GB, the number of time steps N used for the exploration phase was increased from 100 to 200 time steps (in seconds here) and the number M of exploration phases played, was set to 20.

The reduced computing time can be explained by a reduction of the idle time of both the master and the slave processors. Figure 23 gives an overview of the gain in blocking overhead for the same experiments as above. The idle time given, concerns the idle time of the master, the blocking time represents the cumulated blocking time of all the slaves.

These results prove that ESRL is robust enough to be used in asynchronous real-life experiments.

8 Discussion

The ESRL algorithm uses coordinated exploration to reach optimal solutions in repeated stochastic non-zero-sum games. As optimal solutions are not always described by a single solution concept such as a Nash equilibrium or a Pareto optimal joint action, ESRL learning leaves room for imposing different objectives on the agents.

Different interesting efforts to augment standard RL techniques to the multi-agent case can be found in MARL literature. They can be classified according to the type of games they solve and the type of agents they use.

8.1 Related work

One of the first theoretical results of reinforcement learning in games, was achieved by Littman [12] . A Q-learning algorithm, called minimax-Q was developed for learning in zero-sum two-player games. The minimax of the Q-values were used in the update of the value function. Afterward, convergence results were attained for minimax-Q in [14].

For common interest games, and in particular identical payoff games different approaches exist. In [11] a convergence proof for independent optimistic Q-learners in coordination games is given, however the algorithm is not suited for games with stochastic rewards. In [4] joint action learners are introduced. Joint action learners explicitly take the other agents into account, by keeping beliefs on their action selection, based on procedures known as fictitious play in game theory. Assumed is that the actions of the others can be observed, though this is not sufficient to guarantee convergence to optimal equilibria.

In [3] simple joint action learning algorithms are proposed as baselines for any future research on joint-action learning of coordination in cooperative MAS. The algorithms studied in [3] use fairly simple action selection strategies that are still guaranteed to convergence to the optimal action choice. As the general ESRL algorithm of this paper also uses communication one could wonder why not use simple joint action learners instead. The advantage of ESRL is that communication is only necessary when attractor points are reached at the end of an exploration phase. As such communication only occurs for the interesting joint actions and not for all joint actions taken, as is the case for joint action learners. This will make ESRL much more suitable to implement in a distributed setting. Moreover, the amount of communication will grow exponentially with the number of agents and actions for joint action learners.

Several optimistic heuristic methods are proposed to guide action selection to optimal equilibria. In [9] the Q-value of an action in the Boltzmann exploration strategy, is changed by an heuristic value, taking into account how frequently an action produces its maximum corresponding reward. This heuristic, which is called FMQ is suited for pure independent agents. It performs well in games with deterministic rewards. However, in games with stochastic rewards the heuristic may perform poorly. Therefore the authors propose the use of commitment sequences in [10] to allow for stochastic rewards. In a way these commitment sequences are very similar to the exploration phases of ESRL learning, however the former approach strongly depends on synchronous action-selection, whereas ESRL is suited for asynchronous real-life applications as shown in Sect. 7.

In [2] model-based algorithms are proposed for coordination in stochastic common-interest games under imperfect monitoring. The learning algorithm is proved to have a polynomial-time convergence to a near-optimal value of the game. The main difference with ESRL is that the games considered in [2] have deterministic rewards taken from the interval of real numbers $[0, R_{\max}]$. The transition function between the stage games is stochastic, whereas in the work presented here, the reward function itself is stochastic, which makes ESRL robust to use in real-life experiments. The underlying idea of [2] is for the agents to find a common order over joint

actions, so that these can all be jointly executed. Since payoffs are identical for all agents, all agents build the same model and in fact what happens is a distributed coordinated execution of a single-agent reinforcement learning algorithm, which is R-MAX in the case of [2]. In contrast, the agents applying the ESRL algorithm do not have to execute all joint actions together, they only need to find the interesting attractor points of the game, to which they will be driven by the learning automata. This makes ESRL more scalable. Besides this, ESRL is also applicable for conflicting interest games.

As opposed to learning in non-zero sum games and common interest games, the problem of learning in general-sum games is not well understood. Several algorithms are proposed, however as most work focuses on learning Nash equilibria as a central solution concept, a general treatment does not exists. Hu and Welmann [8], extended minimax-Q to general sum games. Their algorithm called Nash-Q developed for joint action learners, uses a Nash equilibrium computation rule in the update of the Q-values. However, only a limited type of games satisfy the assumptions necessary for convergence to be guaranteed. The Friend-or-Foe Q-learning algorithm (FFQ) of [13] solves the same classes of games, however it doesn't require Q-functions for opposing players to be optimized. Instead FFQ agents require to be told whether an opposing agent is a friend or a foe. In the first case, FFQ results in the agents maximizing their own payoff and in the latter case the agents update their Q-values as in minimax-Q. Only when a coordination equilibrium or an adversarial equilibrium[8] exist, convergence to a Nash equilibrium is guaranteed.

We propose the use of periodical policies as optimal fair solutions in games of conflicting interest like the Bach/Stravinsky game of Fig. 4. In [7], Correlated Q-learning (CE-Q) is proposed. Correlated Q-learning generalizes Nash-Q and FFQ and is empirically demonstrated to converge to correlated equilibria. A correlated equilibrium, a concept introduced by Aumann [1] is a generalization of a Nash equilibrium where each player may receive a private recommendation signal before the games is played. These signals do not affect the payoff, but the players may base their choices on the signals received. A correlated equilibrium results if each player realizes that the best he can do is to follow the recommendation, provided that all other players will do this too. In the same way as a periodical policy, a correlated equilibrium can achieve higher rewards than a mixed Nash equilibrium by avoiding undesirable joint actions such as (*Bach*, *Stravinsky*) or (*Stravinsky*, *Bach*) in the Bach/Stravinsky game of Fig. 4. However, to learn a correlated equilibrium in [7] Q-tables should be shared among the agents and thus all agents' actions and rewards must be fully observable. In contrast, a periodical policy can be learned decentralized using ESRL which only relies on a simple synchronization mechanism, whereas for correlated equilibria a central planner is necessary.

8.2 Concluding remarks

Compared to the techniques reported in literature, ESRL agents are unique on different levels. For a start, ESRL is a technique for independent agents learning in a non-zero sum game with stochastic rewards. No beliefs are maintained about the actions of others, which makes the algorithm simple and easy to use also in more than 2 player games. Moreover no assumptions are made on the observability of the other agents' payoff or actions, which makes it possible to use ESRL in distributed systems with

[8] An adversarial equilibrium is a Nash equilibrium with the extra property that no agent is hurt by any change in strategy by the other agents.

limited communication resources, an example is given in Sect. 7. Finally, most existing learning techniques assume the type of the game is known in advance. Assumed is that optimal equilibria do exist and that the goal of learning is to reach one of those equilibria. ESRL on the other hand approaches the set of Nash equilibria and Pareto optimal joint actions, so that different objectives of learning can be specified in advance.

We assumed that all agents have the same number of actions in their private action space. If this would not be the case, ESRL can still be applied. However, to make sure that eventually the agents search their original joint action space again, the agent with the smallest number of actions should send a broadcast message to the others when he has no more actions left. One important future direction for ESRL is the extension from repeated games to Markov or stochastic games.

In summary, ESRL overcomes the assumptions most other approaches in multi-agent reinforcement learning make on the structure of the game, on the knowledge of the agents and also on the solution concepts to be learned. Our periodical policies form promising compromises for agents in games where one can not pursue pure coordination nor pure competition.

References

1. Aumann, R. (1974). Subjectivity and correlation in randomized strategies. *Journal of Mathematical Economics, 1*, 67–96.
2. Brafman, R., & Tennenholtz, M. (2003). Learning to coordinate efficiently: A model-based approach. *Journal on Artificial Intelligence Research (JAIR), 19*, 11–23.
3. Carpenter, M., & Kudenko, D. (2004). Baselines for joint-action reinforcement learning of coordination in cooperative multi-agent systems. In *Proceedings of the 4th symposium on adaptive agents and multi-agent systems, (AISB04) Society for the study of Artificial Intelligence and Simulation of Behaviour* (pp. 10–19).
4. Claus, C., & Boutilier, C. (1998). The dynamics of reinforcement learning in cooperative multi-agent systems. In *Proceedings of the 15th national conference on artificial intelligence* (pp. 746–752).
5. Geist, A., & Beguelin, A. (1994). *PVM: Parallel virtual machine*. MIT Press.
6. Gintis, H. (2000). *Game theory evolving: A problem-centered introduction to modeling strategic behavior*. Princeton, New Jersey: Princeton University Press.
7. Greenwald, A., & Hall, K. (2003). Correlated q-learning. In *Proceedings of the twentieth international conference on machine learning* (pp. 242–249).
8. Hu, J., & Wellman, M. (2003). Nash q-learning for general-sum stochastic games. *Journal of Machine Learning Research, 4*, 1039–1069.
9. Kapetanakis, S., & Kudenko, D. (2002). Reinforcement learning of coordination in cooperative multi-agent systems. In *Proceedings of the 18th national conference on artificial intelligence* (pp. 326–331).
10. Kapetanakis, S., Kudenko, D., & Strens, M. (2003). Learning to coordinate using commitment sequences in cooperative multi-agent systems. In *Proceedings of the 3rd symposium on adaptive agents and multi-agent systems, (AISB03) Society for the study of Artificial Intelligence and Simulation of Behaviour*.
11. Lauer, M., & Riedmiller, M. (2000). An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *Proceedings of the 17th international conference on machine learning* (pp. 535–542).
12. Littman, M. (1994). Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th international conference on machine learning* (pp. 322–328).
13. Littman, M. (2001). Friend-or-foe q-learning in general-sum games. In *Proceedings of the 18th international conference on machine learning* (pp. 157–163).
14. Littman, M., & Szepesvári, C. (1996). A generalized reinforcement-learning model: Convergence and applications. In *Proceedings of the 13th international conference on machine learning* (pp. 310–318).
15. Narendra, K., & Thathachar, M. (1989). *Learning automata: An introduction*. Prentice-Hall International, Inc.

16. Nash, J. (1950). Equilibrium points in n-person games. *Proceedings of the national academy of siences 36*, 48–49.
17. Nowé, A., Parent, J., & Verbeeck, K. (2001). Social agents playing a periodical policy. In *Proceedings of the 12th European conference on machine learning* pp. 382–393. Freiburg, Germany: Springer-Verlag LNAI2168.
18. Osborne, J., & Rubinstein, A. (1994). *A course in game theory*. Cambridge, MA: MIT Press.
19. Samuelson, L. (1997). *Evolutionary games and equilibrium selection*. Cambridge, MA: MIT Press.
20. Sastry, P., Phansalkar, V., & Thathachar, M. (1994). Decentralized learning of nash equilibria in multi-person stochastic games with incomplete information. *IEEE Transactions on Systems, Man, and Cybernetics, 24*(5), 769–777.
21. Sutton, R., & Barto, A. (1998). *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press.
22. Thathachar, M., & Sastry, P. (2002). Varieties of learning automata: An overview. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 32*(6), 711–722.
23. Tsitsiklis, J. (1994). Asynchronous stochastic approximation and q-learning. *Machine Learning, 16*, 185–202.
24. Tuyls, K. (2004). *Multiagent reinforcement learning: A game theoretic approach.* PhD Thesis, Computational Modeling Lab, Vrije Universiteit Brussel, Belgium.
25. Verbeeck, K. (2004). *Coordinated exploration in multi-agent reinforcement learning*. PhD Thesis, Computational Modeling Lab, Vrije Universiteit Brussel, Belgium.
26. Verbeeck, K., Nowé, A., & Parent, J. (2002). Homo egualis reinforcement learning agents for load balancing. In *Proceedings of the 1st NASA workshop on radical agent concepts*, pp. 81–91. Springer-Verlag LNAI 2564.