Improving Cooperation among Self-Interested Reinforcement Learning Agents

Andrea Bonarini, Alessandro Lazaric, Enrique Munoz de Cote, and Marcello Restelli

Politecnico di Milano, Department of Electronics and Information,
piazza Leonardo da Vinci 32, I-20133 Milan, Italy
{bonarini,lazaric,munoz,restelli}@elet.polimi.it

Abstract. In many Multi-Agent Systems (MAS), agents (even if self-interested) need to cooperate in order to maximize their own utilities. Repeated play in social dilemmas (e.g., the iterated prisoner's dilemma) is a challenging problem for learning algorithms, since the Nash Equilibrium (NE) solution, pursued by most of the existing algorithms, is often inappropriate in these settings, while Pareto efficient (PE) solutions guarantee a better outcome for each agent. In this paper we propose two principles (*Change or Learn Fast* and *Change and Keep*) aimed at improving cooperation among Q-learning agents in self-play. Using a n-player and m-action version of the iterated prisoner's dilemma, we show how a best-response learning algorithm, such as Q-learning, improved as proposed, can achieve better cooperative solutions in a shorter time.

1 Introduction

In this paper we mainly focus on problems of cooperation and coordination between *self-interested agents*, where no explicit communication is possible and agents can only perceive the actions taken by the other agents at the previous time instant. Self-interested agents choose actions in order to maximize their own reward signal disregarding the benefit or loss –in terms of utility– to the other agents. In many multi-agents problems, even in not competitive settings, conflicting interests may arise. In these scenarios it is needed for the agents to find a compromise in order to maximize their own utilities. For this reason we focus on an extension of the popular prisoner's dilemma in which the Nash Equilibrium solution is collectively dominated by Pareto efficient solutions. In multi-agent learning systems, where one agent's actions may strongly influence the behavior of the whole system, normal learning techniques, like Q-learning, may exhibit cyclic and unpredicted behaviors [11] because of the intrinsic non-stationary characteristics typical of problems with multiple cooperating agents. Therefore, we propose two different variants of the joint-action Q-learning algorithm in order to deal with such non-stationarity and improve both its performance and learning speed, while maintaining its characteristics of best-responser against stationary opponents.

In the following section we review the main multi-agent learning approaches and algorithms. In Section 3 we give some basic definitions about multi-agent repeated matrix games, and we report the Q-learning algorithm that uses the last joint action taken by agents as state. In Section 4, we propose two principles to foster cooperation, and present how the Q-learning algorithm should be modified to follow them. Section 5 presents comparative results in a multi-agent social dilemma, showing how the use of both the proposed principles improves cooperation among Q-learning agents. In Section 6 we discuss some related works, while Section 7 contains conclusions and suggestions for future research.

2 Learning in Multi-agent Systems

Multi-agent learning (MAL) deals with the interaction between multiple complete agents perceiving, reasoning, and acting in a common dynamical environment. Reinforcement Learning (RL) algorithms have been widely adopted to cope with multi-agent problems. One of the most popular solutions is Q-learning, an on-line model-free RL algorithm that directly learns optimal policies in Markov Decision Processes (MDPs). Although Q-learning is inherently single agent, it has also been used –with some success– in multi-agent cooperative games [18], [15], [14]. Most of these works use Q-learning methods applied without much modification, thus considering other agents as part of the environment. This approach is sound when the other agents are playing stationary policies making the environment an MDP for the learning agent; otherwise, problems arise since the environment is treated as non-stationary while other agents are learning and changing their policies.

Most of the present multiagent learning techniques focus on learning one shot game theoretic rest points (e.g., Nash Equilibria). Littman was the first to explicitly account for others on 2-action, 2-player zero-sum games; in [9] he introduced the Minimax-Q algorithm to calculate the minimax solution that guarantees convergence to the unique NE in fully competitive games. Its counterpart for two-player general-sum games is Nash-Q [7], which is basically an extension of Minimax-Q pursuing a NE in this more general type of games, thus providing a generalization for either cooperative or competitive situations. Littman's Friend-or-Foe-Q (FFQ) [10] is another extension of Minimax-Q that classifies every other agent as "friend" or "foe" in the attempt of converging to a coordination or adversarial equilibria in presence of multiple equilibria. A further equilibrium learning technique is a generalization of Nash-Q that uses a broader class of equilibria called Correlated Equilibria (CE-Q) [6]. Könönen's approach [8] uses the Stackelberg equilibrium for making learning agents with different information to converge to asymmetric equilibrium points. The complexity of finding an equilibrium in repeated and sequential games has been studied by Littman and Stone in [12], in which they use a kind of punishment so that mutual cooperation becomes a *best response* strategy.

Our main motivation in this work is focused on situations where agents have to achieve an agreement to receive maximum reward; this coordination / cooper-

ation problem has, in the last decade, attracted much interest. Fully cooperative solutions were the first steps towards the study of cooperation in agent societies. M. Tan [18] showed the possibility of using communication as a general way of improving learning rates for cooperative agents . Claus and Boutilier [3] presented a relationship between single-agent Q-learning and multiagent Q-learning called *joint-action learning* (JALs) in fully cooperative repeated games (team matrix games). Their results suggested that new exploration heuristics may help convergence in complex games. Furthermore, they showed that having much information available does not necessarily imply performing better.

Several proposals [4, 5] have been put forth sustaining the fact that NE is an undesirable outcome in many repeated stage games (like social dilemmas). On the other hand, they propose algorithms that attempt to reach a Pareto efficient solution. The idea is to let the agents *collectively* find the joint strategy that will give the maximum reward under an infinite horizon of interactions. Since these approaches are more closely related to ours, a more detailed description and comparison is reported in Section 6.

3 Repeated Matrix Games

A matrix game is a tuple $\langle \mathcal{N}, \{\mathcal{A}_i\}_{i\in\mathcal{N}}, \{R_i\}_{i\in\mathcal{N}}\rangle$, where $\mathcal{N}$ is a collection of n agents, $\mathcal{A}_i$ is the set of actions available to agent i, and R_i is its payoff matrix. The i-th agent, simultaneously with the other agents, chooses an action from its own action set $\mathcal{A}_i$ and, on the basis of the actions performed by all the agents, receives a payoff r_i according to its payoff function R_i. Let $\mathbf{a}^t = [a_1^t, a_2^t, ..., a_n^t]$ be the joint action executed at iteration t, where the i-th agent has taken action a_i^t, and $\mathbf{a}_{-i}^t$ is the joint action of all the players except player i. In a repeated matrix (or normal form) game the agents repeatedly play the same matrix game for an undefined number of iterations.

In this paper we focus on games where, at any stage, any agent can observe the actions chosen by other agents, but knows neither the intentions of others, nor the reward functions. However, although matrix games do not have state, in repeated matrix games, learning agents may benefit from knowing the history of joint actions [14]. Therefore, we do not refer to the single state Q-learning as in [15], but we adopt a particular version where Q-learning states are represented by the previous state game joint action ($\mathbf{a}^{t-1}$), as in [5, 1]. Thus, an agent's experience at any stage game is characterized not only by its own action and payoff but also from all the actions actually executed by the agents in the environment $\langle a_i, \mathbf{a}_{-i}, r_i\rangle$. The main steps of Q-learning are reported in Algorithm 1.

The outcome of this process is to let the MAS collectively learn equilibrium points that payoff dominate the best response dynamics of normal Q-learners. The main problem of pairing best-response agents (i.e. Q-learners) in a repeated game is that they may end up in cyclic or suboptimal behaviors. The reason for this is that the parallel learning processes of all the agents cause the environment to be non-stationary, thus preventing agents from predicting the correct outcome of their actions. In the next section we show how to overcome these

Algorithm 1 Q-learning

Let α be a learning rate
initialize $Q(\mathbf{a}, a_i), \forall \mathbf{a} \in \mathcal{A}, a_i \in \mathcal{A}_i$
choose a random action a_i^0
execute a_i^0
read the joint action $\mathbf{a}^0$
$t \leftarrow 1$
for all steps **do**
  choose action a_i^t according to exploration strategy
  execute a_i^t and get the payoff r_i^t
  read the joint action $\mathbf{a}^t$
  $Q(\mathbf{a}^{t-1}, a_i^t) \leftarrow (1-\alpha)Q(\mathbf{a}^{t-1}, a_i^t) + \alpha \cdot \left(r_i^t + \gamma \cdot arg\max_{a_i} Q(\mathbf{a}^t, a_i)\right)$
  $t \leftarrow t+1$
end for

inconveniences by the definition of two different learning principles that try to lower the effects of the non-stationarity of the environment.

4 Making Q-learning More Cooperative

In this section we introduce two variants of the Q-learning algorithm, aimed at improving cooperation, both in terms of performance and of learning speed, among self-interested non-communicating Q-learning agents.

4.1 CoLF Principle

The CoLF (Change or Learn Fast) principle is inspired by the work of Bowling and Veloso [2], where a variable learning rate is considered. In particular, they have proposed the WoLF (Win or Learn Fast) principle, which, applied to rational learning algorithms, can make them convergent[1]. According to this principle, the agent must learn quickly while losing and slowly while winning. Although the WoLF variable learning rate has a convergent effect in many stochastic games, when applied in self-play to social dilemmas it converges to the Nash Equilibrium, thus failing to find a Pareto efficient solution ([16, 5]).

To foster cooperation, we propose to modify the learning rate of the algorithm according to the CoLF principle: if the payoff achieved by an agent is unexpectedly changing, then learn slowly, otherwise learn quickly. This principle aids in cooperation by giving less importance to "unexpected" payoffs (i.e. payoffs that are quite different from those achieved recently in the same state by the same action), probably generated by non-stationary causes like exploration activity or normal learning dynamics of the other agents, while allowing to speed up learning when the most of the agents are playing near-stationary strategies.

[1] This was proved only in self-play for two-person, two-action, iterated general-sum games

Algorithm 2 COLF – Change Or Learn Fast

Let $\alpha_{NS} > \alpha_S$, and λ be learning rates
$P(\mathbf{a}, a_i) \leftarrow S(\mathbf{a}, a_i) \leftarrow 0, \forall \mathbf{a} \in \mathcal{A}, a_i \in \mathcal{A}_i$
$Q(\mathbf{a}, a_i) \leftarrow \frac{r_{max}}{1-\gamma}, \forall \mathbf{a} \in \mathcal{A}, a_i \in \mathcal{A}_i$
choose a random action a_i^0
execute a_i^0
read the joint action $\mathbf{a}^0$
$t \leftarrow 1$
for all steps **do**
  choose action a_i^t according to exploration strategy
  execute a_i^t and get the payoff r_i^t
  read the joint action $\mathbf{a}^t$
  $\Delta r_i^t \leftarrow |r_i^t - P(\mathbf{a}^{t-1}, a_i^t)|$
  if $\Delta r_i^t > S(\mathbf{a}^{t-1}, a_i^t)$ **then**
    $\alpha \leftarrow \alpha_{NS}$
  else
    $\alpha \leftarrow \alpha_S$
  end if
  $Q(\mathbf{a}^{t-1}, a_i^t) \leftarrow (1-\alpha) Q(\mathbf{a}^{t-1}, a_i^t) + \alpha \cdot \left(r_i^t + \gamma \cdot arg\max_{a_i} Q(\mathbf{a}^t, a_i)\right)$
  $S(\mathbf{a}^{t-1}, a_i^t) \leftarrow (1-\lambda) S(\mathbf{a}^{t-1}, a_i^t) + \lambda \cdot \Delta r_i^t$
  $P(\mathbf{a}^{t-1}, a_i^t) \leftarrow (1-\lambda) P(\mathbf{a}^{t-1}, a_i^t) + \lambda \cdot r_i^t$
  $t \leftarrow t + 1$
end for

In Algorithm 2 we have reported how the Q-learning algorithm changes with the introduction of the CoLF principle. For each pair $\langle joint\_action, action \rangle$, besides the Q-value, the algorithm needs to store and update also the P and S-values. The P-values are exponential averages of the collected payoffs with weight factor λ, while the S-values are exponential averages of the absolute differences between the current payoff and the respective P-value. The algorithm requires two learning rates, one when the system shows rapidly varying payoffs (α_{NS}) and one when agents are playing near-stationary policies (α_S), with $\alpha_{NS} < \alpha_S$. The choice of which learning rate must be used to update the Q-value associated to the pair $\langle \mathbf{a}^{t-1}, a_i^t \rangle$ depends on whether the absolute difference between the current payoff and the respective P-value is greater than the respective S-value. In fact, since P-values are an estimation of the expected payoffs and S-values of their variability, if the current payoff is near to the expected one, the environment is supposed to be nearly stationary and Q-values can be updated with a high learning rate (α_S). On the other hand, when the current payoff is highly different with respect to its average P, it is better for the agent to use a low learning rate (α_{NS}) in order to reduce the effect of non-stationarity on the update phase.

4.2 Change&Keep Principle

The Change&Keep (CK) principle is based on the following observation: when an agent, due to either learning or exploration, decides to choose a dif-

Algorithm 3 CK – Change&Keep

status $\leftarrow$ *update*
Let α be a learning rate
$Q(\mathbf{a}, a_i) \leftarrow \frac{r_{max}}{1-\gamma}, \forall \mathbf{a} \in \mathcal{A}, a_i \in \mathcal{A}_i$
choose a random action a_i^0
execute a_i^0
read the joint action $\mathbf{a}^0$
$t \leftarrow 1$
for all steps **do**
  if status = *update* **then**
    choose action a_i^t according to exploration strategy
  else
    $a_i^t \leftarrow a_i^{t-1}$
  end if
  execute a_i^t and get the payoff r_i^t
  read the joint action $\mathbf{a}^t$
  if status = *update* **then**
    if $a_i^t \neq a_i^{t-1}$ **then**
      status $\leftarrow$ *keep*
      $\mathbf{a}^{upd} \leftarrow \mathbf{a}^{t-1}$
    else
      status $\leftarrow$ *update*
      $Q(\mathbf{a}^{t-1}, a_i^t) \leftarrow (1-\alpha)Q(\mathbf{a}^{t-1}, a_i^t) + \alpha \cdot \left(r_i^t + \gamma \cdot arg\max_{a_i} Q(\mathbf{a}^t, a_i)\right)$
    end if
  else
    status $\leftarrow$ *update*
    $Q(\mathbf{a}^{upd}, a_i^t) \leftarrow (1-\alpha)Q(\mathbf{a}^{upd}, a_i^t) + \alpha \cdot \left(r_i^t + \gamma \cdot arg\max_{a_i} Q(\mathbf{a}^t, a_i)\right)$
  end if
  $t \leftarrow t+1$
end for

ferent action, it typically collects an uninformative payoff. In fact, these (non-stationary) changes cannot be foreseen by other agents, and the related payoffs may be misleading, thus negatively affecting cooperation.

The idea of the CK principle is to discard the payoff received in correspondence to a change in the action selection (thus suspending the Q-value update), repeat the same action, and use the corresponding payoff for performing the suspended update. In this way the agent gives time for the other agents to react to its new action, thus using a more informative payoff for the update of its Q-table. Figure 1 illustrates the two-states finite-state machine for CK. The agent starts in state s_C. While the agent selects the same action, it stays in s_C where it performs update and action selection according to its exploration strategy. When the selected action changes, the agent passes in state s_K without performing the update and without observing the action of the other agents. In s_K the agent does not perform the action selection, but it simply repeats its last

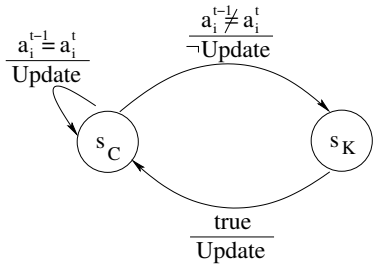

Fig. 1. CK finite-state machine. For each state transition are reported both the trigger condition (above the line) and whether the update phase occurs or not (below the line). In state s_C the agent performs the usual action selection, while in state s_K it repeats the previous action.

action, updates the Q-table and comes back to state s_C. Algorithm 3 shows how the Q-learning algorithm can be combined with the CK principle.

5 Experimental Results

In this section, we compare the performance (in self-play) of Q-learning with those obtained by our variants (CK, CoLF, and CK-CoLF, obtained by combining both the principles together) in the MASD (Multi-Agent Social Dilemma) game [16], an extended version of the prisoner's dilemma with n-players and m-actions that preserves the same structure (one Nash equilibrium and a dominated cooperative strategy). In the MASD game, N agents hold M resource units each. At each iteration, the i-th agent must choose how many of its M units will be allocated for a group goal G, while the remaining will be used for a self-interested goal S_i. Let a_i be the amount contributed by agent i towards goal G, and $\mathbf{a} = [a_1, ..., a_N]$ the joint action. The utility of agent i given the joint action is $P_i(\mathbf{a}) = \frac{\left[\frac{1}{N}\sum_{j=1}^{N} a_j\right] - k a_i}{M(1-k)}$, where $k \in \left(\frac{1}{N}; 1\right)$ is a constant that indicates how much each agent estimates its contribution towards the selfish goal. The payoff function is such that when all the agents put M units in the group goal, each agent is rewarded with 1. On the other hand, if nobody puts units in the group goal, a payoff of 0 is produced. If each agent adopts a random strategy the expected average payoff is 0.5.

In order to choose the parameters for our experiments, we studied their effects on a medium size MASD game with three agents and four actions ($N = 3, M = 3$). We have experimentally verified that high discount factors allow to achieve better performances, but with a lower learning speed. For what concerns exploration, as suggested in [17], the use of a relaxation search, obtained by setting the initial values of the Q-table to high values, considerably improves the results in this kind of repeated matrix games. Furthermore, we add an ϵ-greedy exploration with the aim of reducing the probability of being trapped

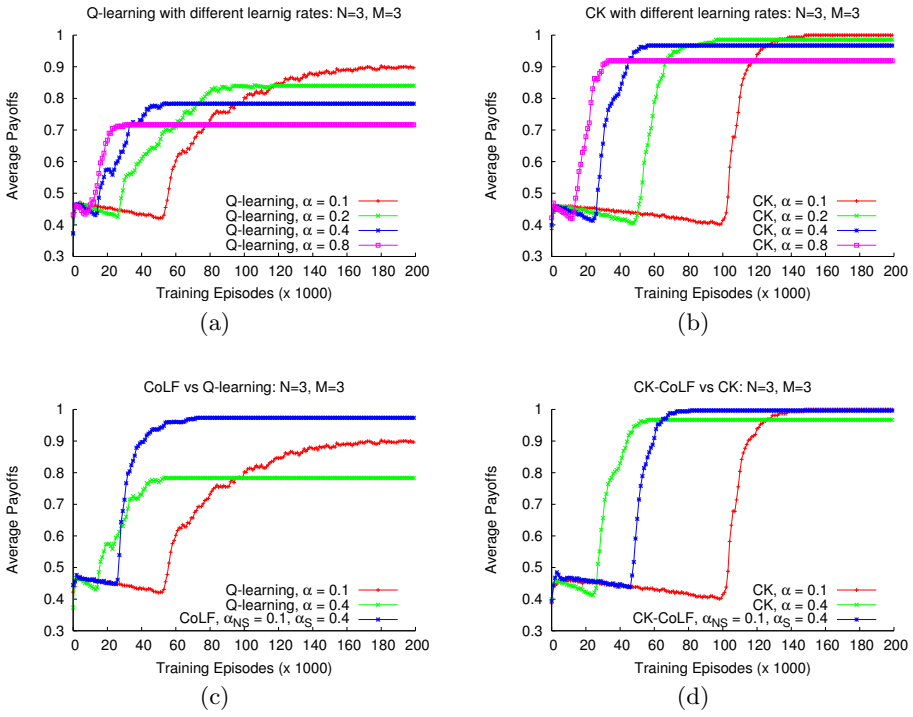

Fig. 2. Plots showing the effect of different learning rates in the MASD problem with 3 agents, 4 actions, and $k = 2/3$. Each plot shows the moving average of the payoffs obtained by agents in self-play over time. The results are averaged over 100 trials.

in a local maximum. On the basis of these considerations, in the experiments we are presenting here, we have used the following parametrization: $\gamma = 0.95$, $Q(\mathbf{a}, a_i)_{init} = \left(\frac{r_{max}}{1-\gamma}\right)$, and $\epsilon = \max(0.2 - 0.00006t, 0)$. For what concerns the weight factor λ, used in the exponential averages of the CoLF principle (see Section 4.1), we found the best results by putting $\lambda = 0.1$.

In Figure 2 we report some interesting results about learning rates. Figure 2(a) shows average payoffs obtained with different initial values of the learning rate α[2]. As already pointed out in [5], the use of low learning rates in Q-learning allows to get higher payoffs, even if the time required to reach a cooperative solution considerably increases. For comparison, in Figure 2(b) we report the results of CK with the same learning rates. As we can see, the learning rate variation has the same qualitative effect on both the algorithms, even if the performance of CK are less affected by high learning rates. For what concerns the

[2] In all the experiments the initial learning rate α_i is decreased in the following way: $\alpha^t = \frac{\alpha_i}{(1+0.0001 \cdot n(\mathbf{a}^{t-1}, a_i^t)}$, where $n(\mathbf{a}^{t-1}, a_i^t))$ is the number of times that action a_i has been taken after the joint action $\mathbf{a}^{t-1}$.

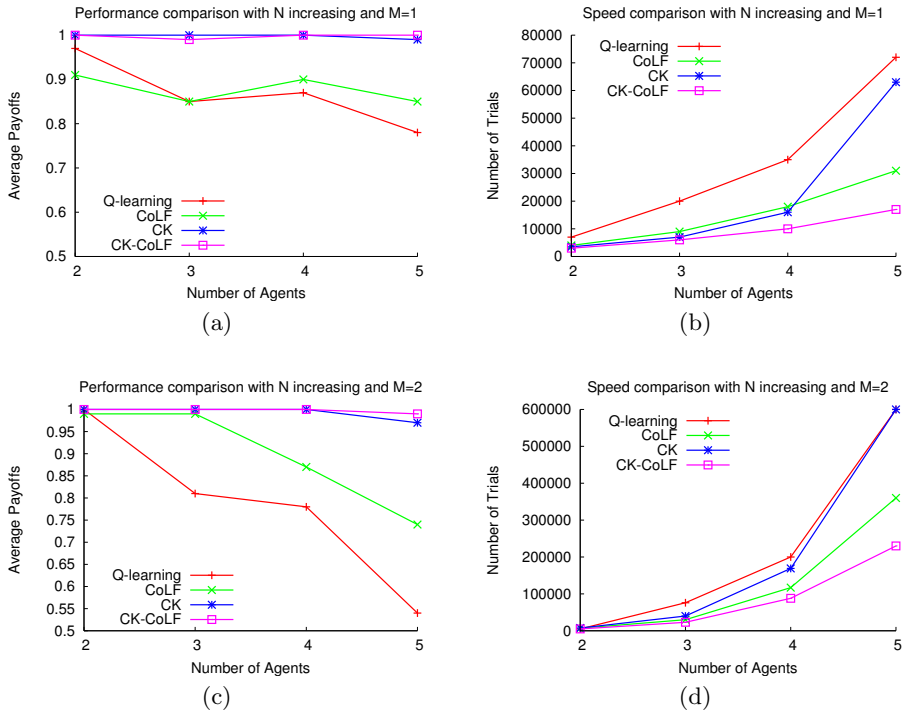

Fig. 3. Plots comparing performances and learning speeds of Q-learning, CoLF, CK, and CK-CoLF in MASD problems with multiple agents and actions. Plots on the left show the average payoff obtained by agents in self-play for a fixed number of actions as the number of agents increases. On the right we have reported the number of trials required to learn a constant joint policy. The results are averaged over 100 trials.

learning speed, CK performs poorly (less than the random policy, i.e. 0.5) for many steps, but its learning curve is steeper than that of Q-learning; the result is that both the algorithms (when using the same learning rate) reach a cooperative solution nearly at the same time, with the difference that CK outperforms Q-learning for every α, and actually reaches the PE solution for $\alpha = 0.1$. For what regards CoLF, we have chosen 0.1 for the non-stationary learning rate α_{NS}, while we set the stationary learning rate to $\alpha_S = 4 \cdot \alpha_{NS}$. Figure 2(c) shows the comparison between CoLF, Q-learning with learning rate α_S, and Q-learning with learning rate α_{NS}. As we can see, using a variable learning rate according to the CoLF principle, it is possible to achieve higher payoffs than those obtained by Q-learning with a single learning rate. In fact, the algorithm succeeded in exploiting both the low learning rate for what concerns the payoff performance and the high learning rate in terms of learning speed, that is comparable to the one obtained by Q-learning with high learning rate. Similarly, in Figure 2(d) we

can see how CoLF improves the learning speed of CK, while still reaching the PE solution.

Figure 3 shows comparative results among Q-learning, CoLF, CK, and CK-CoLF on MASDs characterized by a different number of agents and actions[3]. In Figure 3(a) are reported the performances of the four algorithms for MASDs with two actions ($M = 1$) as the number of agents increases. Unsurprisingly, Q-learning performs quite well in the two-player two-action game, but the average payoff decreases quite quickly with a few more agents. CoLF performs slightly worse than Q-learning for $N = 2$, while its performance degrades more gracefully as N increases. CK and CK-CoLF are able to converge, almost in any case, to mutual cooperation. Besides average payoff, the algorithms have been evaluated on the basis of the number of learning trials required to reach a stable cooperative solution. Figure 3(b) displays the average learning times of the four algorithms: Q-learning is the slowest of the pool, while CK is slightly quicker. Algorithms that use the CoLF principle learn to cooperate much faster (since they benefit from using higher learning rates); in particular, CK-CoLF turns out to be the fastest algorithm. Similar considerations hold (even with more evidence) for the results on problems with three actions reported in Figures 3(c) and 3(d). However, it is worth to notice that the learning times do not allow the application of the algorithms to problems with tens of agents and actions.

From the graphs in Figure 3, it results that, using both Ck and CoLF principles, Q-learning strongly improves its cooperative capabilities in self-play.

6 Related Works

Many works in literature investigated the possibility to reach Pareto efficient solutions. In particular, we relate to the approaches based on an exhaustive search over the *joint action space* until reaching an outcome that *satisfies* the agent's *aspiration level*. A series of work, leaded by Goodrich [5, 17, 16, 4], in n-players, m-actions social dilemmas use a "floating" aspiration level to avoid an agent to get *unsatisfied* against non cooperative players (reaching a NE) and reach near Pareto efficient outcomes in self-play, without knowing neither the structure of the game nor the actions of the other players [4]. As opposed to us, their algorithm requires to use a random policy for (sufficiently big) n rounds and then change to a deterministic *satisficing* policy. They replaced the NE perspective with *Nash bargaining*, a perspective more suited for repeated interactions. Comparing our results to those obtained by [16], our modified versions of Q-learning achieves a better performance, even if their approach does not rely the knowledge of the other agents' actions. Macy [13] uses a similar aspiration based approach but only for 2-player, 2-action symmetric social dilemmas that, as in [4], needs not neither the structure of the game nor the actions of the other players.

In [1] Banerjee proves convergence to Pareto-optimality using conditional probabilities over a joint action learners (CJALs) for 2-action, 2-player dilem-

[3] Experiments for Q-learning and CK use 0.1 as learning rate, while CoLF and CK-CoLF use 0.1-0.4

mas. Convergence is guaranteed iff random exploration for N rounds is used, and then each agent adopts a *greedy* (or bounded $\epsilon - greedy$) policy. Littman and Stone uses the weakness of best-response strategies and developed a "leader" algorithm to teach best-response agents to play Pareto efficient solutions. On the other hand, [5] proposes M-Qubed, an algorithm that provably satisfies a *security property* so agents do not get less than the *minimax* value of the game, and in several two player games empirically displays (in self play) a *compromise/cooperate property* in hope of promoting cooperation.

7 Conclusions and Future Research

In this paper, we have proposed two heuristic principles to improve cooperation among self-interested RL agents in repeated general-sum games. The CoLF principle is mainly concerned with non-stationarity caused by the other agents; it uses a variable learning rate that takes low values when the agent gathers unexpected payoffs and a higher one when the environment is supposed to be near-stationary. On the other hand, the CK principle deals with non-stationarity induced in the MAS by the behavior of the agent itself; since actions selected following a non-stationary learning policy may not be foreseen by other agents, whenever an agent decides to change its action, according to the CK principle, it should not perform the update, but it should repeat the same action and update the related value with the latter payoff. The experiments carried out in the MASD framework show that the proposed principles applied to a best-response learning algorithm (Q-learning) largely improve its cooperation capabilities in self-play, both in terms of performance and learning speed.

This work puts the bases for several future studies. It will be useful to investigate theoretical properties of the proposed heuristics, such as convergence properties or the Cooperate/Compromise property [5]. For what concerns the experimental activity, it will be interesting to study the effect of CK and CoLF outside of self-play, in order to see if they are able to make also other learning algorithms to reach a cooperative solution. Furthermore, it will be instructive to verify whether the good results obtained in an extended version of the iterated prisoner's dilemma may be replicated in other social dilemmas such as chicken, Shapley's game, and tricky game.

Another important future research direction is about the exploration strategy. Even if initializing the Q-table with high values allows to perform an extensive exploration of the search space, in problems with many agents and many actions, it leads to large learning times (notice the number of learning steps in Fig. 3(b) and 3(c)). On the other hand, a simple ϵ-greedy exploration may not be enough in problems with many agents and actions. The identification of smarter learning strategies (e.g. the one proposed in [5]) is a key factor for increasing the learning speed.

References

1. Dipyaman Banerjee and Sandip Sen. Reaching pareto optimality in prisoner's dilemma using conditional joint action learning. In *Working Notes of the AAAI-05 Workshop on Multiagent Learning*, to appear.
2. Michael Bowling and Manuela Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136:215–250, 2002.
3. Caroline Claus and Craig Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the 6th Conference On Artificial Intelligence (AAAI-98) and of the 11th Conference On Innovative Applications of Artificial Intelligence (IAAI-98)*, pages 746–752. AAAI Press, 1998.
4. Jacob W. Crandall and Michael A. Goodrich. Learning ε-pareto efficient solutions with minimal knowledge requirements using satisficing. In *Proceedings of the 19th National Conference On Artificial Intelligence*. AAAI Press, 2004.
5. Jacob W. Crandall and Michael A. Goodrich. Learning to compete, compromise, and cooperate in repeated general-sum games. In *Proc. of ICML 2005*, to appear.
6. Amy Greenwald, Keith Hall, and Roberto Serrano. Correlated-q learning. In *Proceedings of NIPS Workshop On Multiagent Learning*, 2002.
7. Junling Hu and Michael P. Wellman. Multiagent reinforcement learning: Theoretical framework and an algorithm. In *Proceedings of ICML*, 1998.
8. Ville Könönen. Asymmetric multiagent reinforcement learning. In *Proceedings of International Conference On Intelligent Agent Technology (IAT 03)*. IEEE Computer Society, 2003.
9. Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of ICML*, pages 157–163, 1994.
10. Michael L. Littman. Friend-or-foe q-learning in general-sum games. In *Proceedings of ICML*, pages 322–328, 2001.
11. Michael L. Littman and Peter Stone. Leading best-response strategies in repeated games. In *Seventeenth Annual International Joint Conference On Artificial Intelligence Workshop On Economic Agents, Models and Mechanisms*, 2001.
12. Michael L. Littman and Peter Stone. A polynomial-time nash equilibrium algorithm for repeated games. *Decision Support Systems*, 39:55–66, 2005.
13. Michael W. Macy and Andreas Flache. Learning dynamics in social dilemmas. *Proceedings of the National Academy of Sciences*, 72(9):29–36, 2002.
14. Tuomas W. Sandholm and Robert H. Crites. Multiagent reinforcement learning in the iterated prisoner's dilemma. *Biosystems*, 37(1–2):147–146, 1995. Special Issue on the Prisoner's Dilemma.
15. Sandip Sen, M. Sekaran, and J. Hale. Learning to coordinate without sharing information. In *Proceedings of the 12th National Conference On Artificial Intelligence*, volume 1, pages 426–431, Seattle, Washington, 1994. AAAI Press.
16. Jeffrey L. Stimpson and Michael A. Goodrich. Learning to cooperate in a social dilemma: A satisficing approach to bargaining. In *Proceedings of ICML*, 2003.
17. Jeffrey L. Stimpson, Michael A. Goodrich, and Lawrence C. Walters. Satisficing and learning cooperation in the prisoner's dilemma. In *Proc. of IJCAI 2001*, 2001.
18. Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of ICML*, pages 330–337, 1993.