Learning to cooperate in multi-agent social dilemmas

Conference Paper · May 2006

DOI: 10.1145/1160633.1160770 · Source: DBLP

3 authors:

Enrique Munoz de Cote
Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE)
74 PUBLICATIONS **965** CITATIONS
SEE PROFILE

Alessandro Lazaric
National Institute for Research in Computer Science and Control
124 PUBLICATIONS **2,279** CITATIONS
SEE PROFILE

Marcello Restelli
Politecnico di Milano
210 PUBLICATIONS **2,452** CITATIONS
SEE PROFILE

Learning to Cooperate in Multi-Agent Social Dilemmas

Enrique Munoz de Cote
Politecnico di Milano
Department of Electronics and Information
piazza Leonardo da Vinci 32, I-20133 Milan, Italy
munoz@elet.polimi.it

Alessandro Lazaric
Politecnico di Milano
Department of Electronics and Information
piazza Leonardo da Vinci 32, I-20133 Milan, Italy
lazaric@elet.polimi.it

Marcello Restelli
Politecnico di Milano
Department of Electronics and Information
piazza Leonardo da Vinci 32, I-20133 Milan, Italy
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. Most of the multi-agent learning algorithms focus on one-shot games, whose rational optimal solution is a Nash Equilibrium. Many times, these solutions are no longer optimal in repeated interactions, where, in the long run, other more profitable (Pareto Efficient) equilibrium points emerge (e.g., in the iterated Prisoner's Dilemma). The goal of this work is to improve existing rational Reinforcement Learning (RL) algorithms, that typically learn the one-shot Nash Equilibrium solution, using design principles that foster the reaching of the Pareto Efficient equilibrium. In this paper we propose two principles (*Change or Learn Fast* and *Change and Keep*) aimed at improving cooperation among Q-learning (a popular RL algorithm) agents in self-play. Using MASD (Multi-Agent Social Dilemma), an n-player and m-action version of the iterated prisoner's dilemma, we show how a best-response learning algorithm, such as Q-learning, modified as proposed, can achieve better cooperative solutions in a shorter time. To test the robustness of the proposed approaches, we present an analysis of their sensitiveness to several learning parameters and some variants of MASD.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems

1. INTRODUCTION

In this paper we mainly focus on problems of cooperation and coordination between an open community of *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 are utility maximizers, meaning that they 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 where agents *interact repeatedly*, conflicting interests may arise; this may happen even in not competitive settings. Under many of these scenarios, self-interested agents will settle to –competitive– equilibrium points (e.g. Nash equilibrium) that are most likely sub-optimal. Thus, it is needed for the agents to find a compromise to reach other more profitable equilibrium points 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.

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 RL algorithm that does not need a model of the environment –i.e., the state transition probabilities– and learns optimal policies in Markov Decision Processes (MDPs) by directly interacting with the environment. Although Q-learning is inherently single agent, it has been used – and with some success– as the first (naive) approach in a multi-agent setting (cooperative games) [15, 12, 11]. 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 Q-learning treats the environment as stationary, while in fact, other agents are learning, changing their policies and making the environment non-stationary.

Most of the present multi-agent learning techniques focus on learning one shot game theoretic rest points (e.g., Nash, Correlated, Stackelberg equilibrium) [8, 7, 9, 6, 16]. Our main motivation in this work is focused on situations where agents have to achieve an agreement to receive maximum reward; this coordination / cooperation problem has, in the last decade, attracted much interest. Fully cooperative solutions were the first steps towards the study of cooperation in agent societies [15]. This cooperation among individuals was the motivation for Claus and Boutilier to study the relationship between single-agent Q-learning and multi-agent Q-learning in a fully cooperative repeated game (team matrix game) setting [3], they called this kinship *joint-action learning* (JALs). Which is a Q-learning extension with the joint action vector that relate the vector of joint actions with marginal probabilities. Their results suggest that new exploration heuristics should help convergence in complex games.

Several proposals [14, 4, 5, 10] have been put forth sustaining the fact that NE is an undesirable outcome in many repeated stage games (like social dilemmas); thus 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 and then settle an agreement to that joint strategy, therefore, reaching an outcome that *satisfies* the agent's *aspiration level*. They replaced the NE perspective with *Nash bargaining*, a perspective more suited for repeated interactions. As test bed , they introduced the Multi-Agent Social Dilemma (MASD), an n-players, m-actions social dilemma that captures the essence of the prisoner's dilemma game [14]. Throughout the remaining of this work, we use this test bed to test the performance of our algorithms.

In Section 2 we give some basic definitions about multi-agent repeated matrix games, and we report a Q-learning algorithm that uses the last joint action taken by agents as state. In Section 3, we propose two principles to foster cooperation, and present how the Q-learning algorithm should be modified to achieve efficient solutions. Section 4 presents comparative results in a multi-agent social dilemma, showing how the use of both the proposed principles improves cooperation among Q-learning agents, while Section 5 contains conclusions and suggestions for future research.

2. 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 [11]. Therefore, we do not refer to the single state Q-learning as in [12], 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.

Q-learning, in its multi-agent form is a best-response algorithm. Best-response algorithms attempt to learn a best-response to the other players current policies $BR_i(\pi^{-i})$. This mean that it will learn to play deterministic policies (pure strategies). This is no restriction for us given that the domain we are interested in has a unique NE in pure strategies. Furthermore, most general-sum games have equilibrium points in pure strategies. In zero-sum games this is not the case so an agent that learns a deterministic strategies could get exploited by a clever opponent.

The desired outcome of the 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 sub-optimal 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 inconveniences by the definition of two different learning principles that try to lower the effects of the non-stationarity of the environment.

3. 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.

3.1 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. According to this principle, the agent must learn quickly while losing and slowly while winning. WoLF introduces a variable learning rate, which has a convergent effect in many stochastic games[1]. However, when applied in self-play to social dilemmas it converges to the Nash Equilibrium, thus failing to find a Pareto efficient solution [13, 5].

To foster cooperation, we propose to modify the learning rate of a Q-learning 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.

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

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

Algorithm 1 Q-learning

```
Let $\alpha$ 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 $\mathbf{a}^t$
  $Q(\mathbf{a}^{t-1}, a_i^t) \leftarrow (1-\alpha)Q(\mathbf{a}^{t-1}, a_i^t) + \alpha \cdot \big(r_i^t + \gamma \cdot \max_{a_i} Q(\mathbf{a}^t, a_i)\big)$
  $t \leftarrow t + 1$
end for
```

Algorithm 2 COLF – Change Or Learn Fast

```
Let $\alpha_S > \alpha_{NS}$, and $\lambda$ 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
  $Q(\mathbf{a}^{t-1}, a_i^t) \leftarrow (1-\alpha)Q(\mathbf{a}^{t-1}, a_i^t) + \alpha \cdot \big(r_i^t + \gamma \cdot \max_{a_i} Q(\mathbf{a}^t, a_i)\big)$
  $S(\mathbf{a}^{t-1}, a_i^t) \leftarrow (1-\lambda)S(\mathbf{a}^{t-1}, a_i^t) + \lambda \cdot \Delta r_i^t$
  $P(\mathbf{a}^{t-1}, a_i^t) \leftarrow (1-\lambda)P(\mathbf{a}^{t-1}, a_i^t) + \lambda \cdot r_i^t$
  $t \leftarrow t + 1$
end for
```

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.

Even if the CoLF principle, like WoLF, uses a variable learning rate, the implications are quite dissimilar. WoLF was introduced as a modification to Infinite Gradient Ascent (IGA) to make it convergent in the cases where it was not. The resulting algorithm, WoLF-IGA, belongs to the *actor-critic* framework, where agents map value functions to strategy space in order to update their strategies. The variable learning rate in WoLF-IGA, is used in the strategy update face, while we use a variable learning rate in the Q-table update. Furthermore, WoLF-IGA's criterion for choosing among learning rates is based on determining if a player is winning or loosing: each player selects a Nash equilibrium and compares the expected payoff of playing this NE with their current strategy expected payoff. We do not require that each agent computes a NE (which can be a demanding requirement) since CoLF selects the learning rate according to a simple computation on the variability of payoffs.

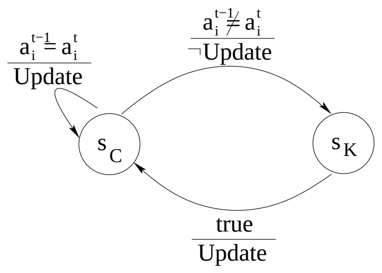
3.2 Change&Keep Principle

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

Algorithm 3 CK – Change&Keep

```
status ← $update$
Let $\alpha$ 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 ← $keep$
      $\mathbf{a}^{upd} \leftarrow \mathbf{a}^{t-1}$
    else
      status ← $update$
      $Q(\mathbf{a}^{t-1}, a_i^t) \leftarrow (1-\alpha)Q(\mathbf{a}^{t-1}, a_i^t) + \alpha \cdot \left(r_i^t + \gamma \cdot \max_{a_i} Q(\mathbf{a}^t, a_i)\right)$
    end if
  else
    status ← $update$
    $Q(\mathbf{a}^{upd}, a_i^t) \leftarrow (1-\alpha)Q(\mathbf{a}^{upd}, a_i^t) + \alpha \cdot \left(r_i^t + \gamma \cdot \max_{a_i} Q(\mathbf{a}^t, a_i)\right)$
  end if
  $t \leftarrow t+1$
end for
```



$\frac{a_i^{t-1} = a_i^t}{\text{Update}}$
$\frac{a_i^{t-1} \neq a_i^t}{\neg\text{Update}}$
s_C
s_K
$\frac{\text{true}}{\text{Update}}$

Figure 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, 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 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.

4. 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 [13], 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 ($N = 3, M = 3$). For what concerns exploration, as suggested in [14], 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 in a local maximum. We have experimentally verified (see Figure 2) that high discount factors allow to achieve better performances, but with a lower learning speed. In fact, using 0.8 as discount factor, Q-learning and CoLF are not able to achieve a cooperative solution, which is obtained by adding the CK principle. On the other hand, all the algorithms achieve a cooperative solution, but they require much longer time to converge. We have also tested the performances of the four algorithms in different versions of the MASD problem obtained by varying the k factor. For higher values of k the agents become more selfish and the reaching of the cooperative solution is more difficult. Figure 3 shows the performances of the four algorithms with different values of k. None of the four algorithms is able to cooperate with $k = 0.9$ since the selfish behavior is highly fostered. The problem becomes easier and easier as k decreases. On the basis of these considerations, in the following experiments we are presenting, 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 3.1), we found the best results by putting $\lambda = 0.1$.

In Figure 4 we report some interesting results about learning rates. Figure 4(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 4(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 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 4(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 4(d) we can see how CoLF improves the learning speed of CK, while still reaching the PE solution.

Figure 5 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 5(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 5(c) 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 5(b) and 5(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 5, it results that, using both Ck and CoLF principles, Q-learning strongly improves its cooperative capabilities in self-play.

5. DISCUSSION AND FUTURE WORKS

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 symmetric (chicken game) or asymmetric (tricky game) social dilemmas.

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

[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 $\mathbf{a}^{t-1}$.

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

actions, it leads to large learning times (notice the number of learning steps in Fig. 5(c) and 5(b)). 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, in combination with a convenient exploration strategy is a key factor for increasing the learning speed.

6. ADDITIONAL AUTHORS

Additional authors: Andrea Bonarini (Politecnico di Milano, email: `bonarini@elet.polimi.it`).

7. REFERENCES

[1] D. Banerjee and S. 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] M. Bowling and M. Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136:215–250, 2002.

[3] C. Claus and C. 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] J. W. Crandall and M. 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] J. W. Crandall and M. A. Goodrich. Learning to compete, compromise, and cooperate in repeated general-sum games. In *Proc. of ICML 2005*, to appear.

[6] A. Greenwald, K. Hall, and R. Serrano. Correlated-q learning. In *Proceedings of NIPS Workshop On Multiagent Learning*, 2002.

[7] J. Hu and M. P. Wellman. Multiagent reinforcement learning: Theoretical framework and an algorithm. In *Proceedings of ICML*, 1998.

[8] M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of ICML*, pages 157–163, 1994.

[9] M. L. Littman. Friend-or-foe q-learning in general-sum games. In *Proceedings of ICML*, pages 322–328, 2001.

[10] M. W. Macy and A. Flache. Learning dynamics in social dilemmas. *Proceedings of the National Academy of Sciences*, 72(9):29–36, 2002.

[11] T. W. Sandholm and R. H. Crites. Multiagent reinforcement learning in the iterated prisoner's dilemma. *Biosystems*, pages 147–146, 1995.

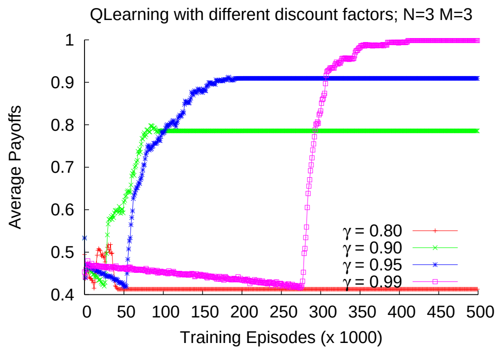
[12] S. Sen, M. Sekaran, and J. Hale. Learning to coordinate without sharing information. In *Proceedings of the 12th National Conference On Artificial Intelligence*, volume 1, pages 426–431, Seattle, Washington, 1994. AAAI Press.

[13] J. L. Stimpson and M. A. Goodrich. Learning to cooperate in a social dilemma: A satisficing approach to bargaining. In *Proceedings of ICML*, 2003.

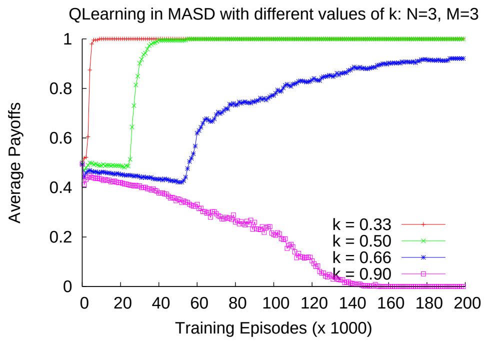
[14] J. L. Stimpson, M. A. Goodrich, and L. C. Walters. Satisficing and learning cooperation in the prisoner's dilemma. In *Proc. of IJCAI 2001*, 2001.

[15] M. Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of ICML*, pages 330–337, 1993.

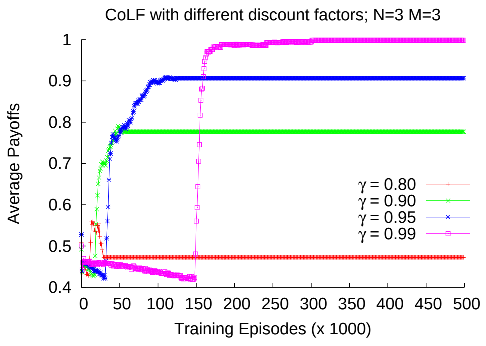
[16] M. Veloso and M. Bowling. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136(2):215–250, 2002.



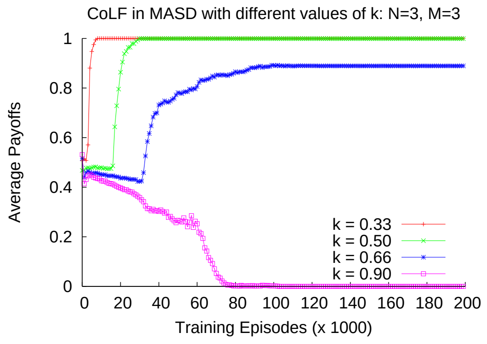
(a)



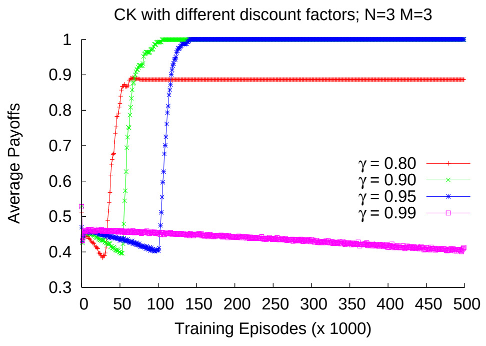
(a)



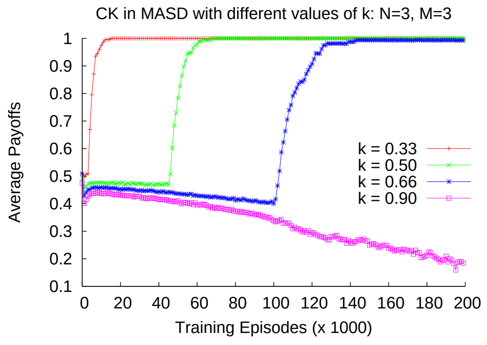
(b)



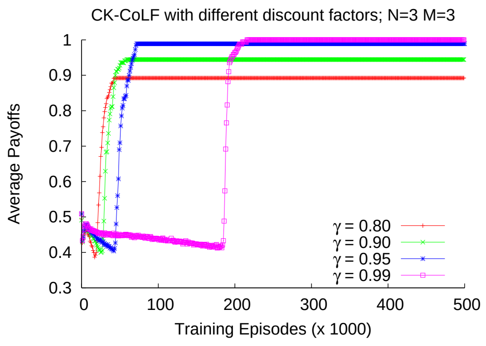
(b)



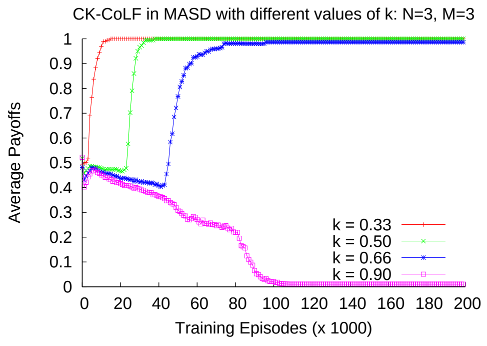
(c)



(c)



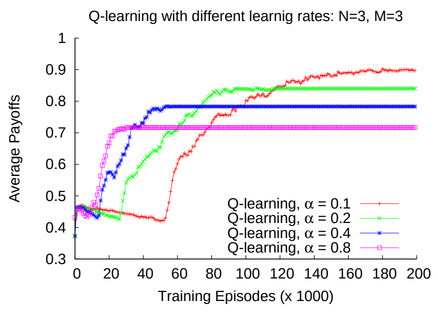
(d)



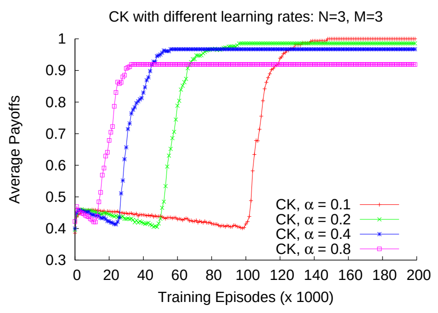
(d)

Figure 2: Plots comparing performances and learning speeds of Q-learning, CoLF, CK, and CK-CoLF in MASD problems with different discount factors.

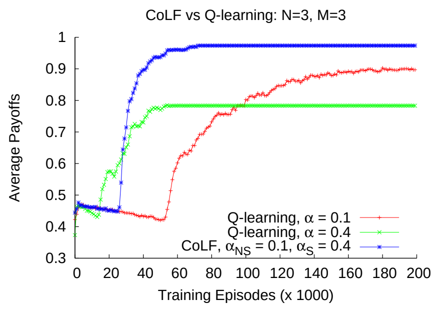
Figure 3: Plots comparing performances and learning speeds of Q-learning, CoLF, CK, and CK-CoLF in MASD problems characterized by different values of k. High values of k lead to more selfish problems.



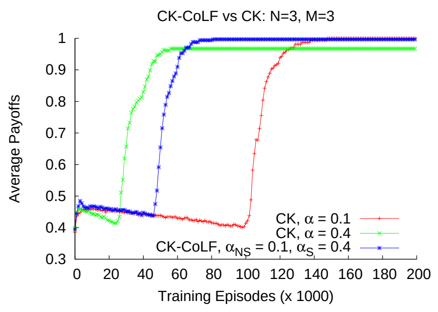
(a)



(b)

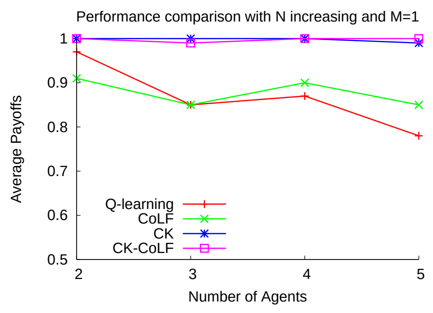


(c)

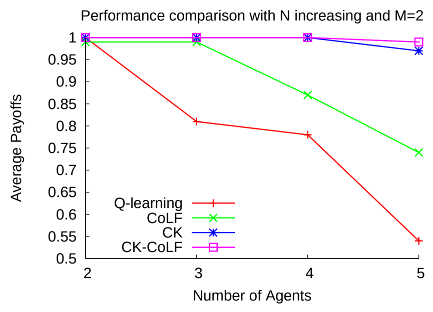


(d)

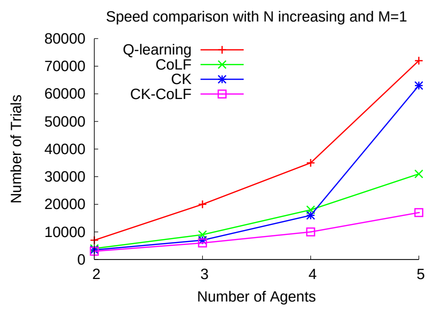
Figure 4: Plots showing the effect of different learning rates in the MASD problem with 3 agents, 4 actions, and $k = 2/3$. Each plot shows the moving average of the payoffs obtained by agents in self-play over time. The results are averaged over 100 trials.



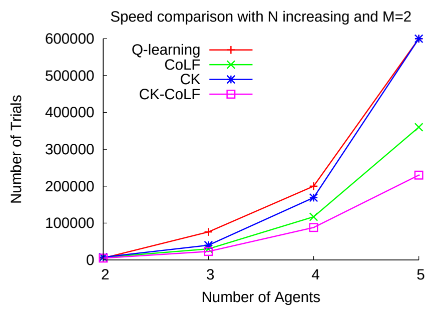
(a)



(b)



(c)



(d)

Figure 5: Plots comparing performances and learning speeds of Q-learning, CoLF, CK, and CK-CoLF in MASD problems with multiple agents and actions. Plots 5(a) and 5(b) show the average payoff obtained by agents in self-play for a fixed number of actions as the number of agents increases. In plots 5(c) and 5(d) we have reported the number of trials required to learn a constant joint policy. The results are averaged over 100 trials.