Multiagent Reinforcement Learning in the Iterated Prisoner's Dilemma: Fast Cooperation through Evolved Payoffs

Vassilis Vassiliades and Chris Christodoulou

***Abstract*— In this paper, we investigate the importance of rewards in Multiagent Reinforcement Learning in the context of the Iterated Prisoner's Dilemma. We use an evolutionary algorithm to evolve valid payoff structures with the aim of encouraging mutual cooperation. An exhaustive analysis is performed by investigating the effect of: i) the lower and upper bounds of the search space of the payoff values, ii) the reward sign, iii) the population size, and iv) the mutation operators used. Our results indicate that valid structures that encourage cooperation can quickly be obtained, while their analysis shows that: i) they should contain a mixture of positive and negative values and ii) the magnitude of the positive values should be much smaller than the magnitude of the negative values.**

I. Introduction

Multiagent Reinforcement Learning (MARL) research has recently attracted a serious amount of scientific work. The main problem of MARL is that the presence of multiple learning agents creates a non-stationary environment, therefore, for a system to perform well, the agents might need to base their decisions on a history of joint past actions and on how they would like to influence future ones. In MARL there could be different kinds of situations: fully competitive or adversarial (which could be modelled with zero-sum games), fully cooperative or coordinative (which could be modelled with team games), and a mixture of both (which could be modelled with general-sum games). As different issues arise in each situation, researchers have developed algorithms with a variety of research goals.

The current study lies in the "prescriptive non-cooperative" agenda [1], i.e., we are interested in effective techniques that result in high rewards for the agents. More specifically, we investigate cooperation between self-seeking reward agents in a non-cooperative setting. This situation is modelled with the Iterated Prisoner's Dilemma (IPD) which is a general-sum game. Although the cooperative outcome is a valid equilibrium of the IPD, our study does not aim to assess the strength of the learning algorithms to attain equilibria of the game or best responses to any given strategy; instead, we focus on mutual cooperation and investigate whether it can be achieved by simple reinforcement learning (RL) agents and enhanced by evolving its payoff matrix. The rationale behind this is that we would like to motivate the agents into cooperation by making them perceive the payoff values differently, i.e., as rewards and penalties. Therefore in this paper, the evolved payoff values represent reinforcement signals generated inside the agent and not external/environmental stimuli. The mapping between external and internal reinforcements is ignored for simplicity.

This work was supported by the University of Cyprus under an Internal Research Project grant.

Vassilis Vassiliades and Chris Christodoulou are with the Department of Computer Science, University of Cyprus, 75 Kallipoleos Avenue, P.O. Box 20537, 1678 Nicosia, Cyprus (e-mails: {v.vassiliades, cchrist}@cs.ucy.ac.cy).

In its standard one-shot version, the Prisoner's Dilemma (PD) [2] is a game summarized by the payoff matrix of Table I. There are two players, Row and Column. Each player has the choice of either to "Cooperate"(C) or "Defect" (D). For each pair of choices, the payoffs are displayed in the respective cell of the payoff matrix of Table I. These values were used in various IPD tournaments (see [3] and [4]) and to the best of our knowledge are the ones most commonly studied. In game theoretical terms, where rational players are assumed, DD is the only Nash equilibrium outcome [5] (i.e., a state in which no player can benefit by changing his/her strategy while the other players keep theirs unchanged), whereas the cooperative (CC) outcome satisfies Pareto optimality [6] (i.e., a state in which it is impossible to increase the gains of one player without increasing the losses of other players). The "dilemma" faced by the players in any valid payoff structure is that, whatever the other player does, each one of them is better off by defecting than cooperating. The outcome obtained when both players defect however is worse for each one of them than the outcome they would have obtained if both had cooperated. In the IPD, an extra rule $R > (T + S)/2$ guarantees that the players are not collectively better off by having each player alternate between C and D, thus keeping the CC outcome Pareto optimal.

TABLE I

Payoff matrix of the Prisoner's Dilemma game with the most commonly studied payoff values. Payoff for the Row player is shown first. R is the "reward" for mutual cooperation. P is the "punishment" for mutual defection. T is the "temptation" for unilateral defection and S is the "sucker's" payoff for unilateral cooperation. The only condition imposed to the payoffs is that they should be ordered such that $T > R > P > S$.

	Cooperate (C)	Defect (D)
Cooperate (C)	$R(=3), R(=3)$	$S(=0), T(=5)$
Defect (D)	$T(=5), S(=0)$	$P(=1), P(=1)$

As pointed out in [7] "perfectly predicting the environment is not enough to guarantee good performance", because the performance depends partly on properties of the environment. In our case, we believe that the property of the environment which plays a significant role in the CC outcome is the

reward function, since it specifies the type and strength of the reinforcement the agents receive. Therefore, we introduce a method that evolves the payoff values of the IPD while satisfying its constraints, in order for simple RL algorithms to rapidly reach the CC outcome.

The remainder of the paper is organised as follows. In Section II, we present some related work. Section III describes our methodology, while the results are given in Section IV. Finally, in Section V, we briefly discuss some issues related to this work and summarise the conclusions of this paper.

II. Related Work

Sandholm and Crites [8] investigated MARL in the IPD by representing the state-action estimates (Q-function) inside lookup tables and simple recurrent neural networks and showed that lookup tables lead to better results. However, the payoff values they used were a scaled down version of the commonly studied values indicated in Table I. They report results where CC occurred 50% of the time in the final rounds of the game, after running the simulation for 62.5 million rounds. Our previous work with MARL in the IPD [9], [10] showed that it is possible to train spiking neural networks to reach mutual cooperation. In this case however, a mixture of both positive and negative payoff values is necessary, as the learning algorithms used [11], [12] work with positive and negative reinforcements extracted from the payoff matrix and directly applied to the synapses. This mixture was found to be beneficial for non-spiking neural networks as well as lookup tables.

Reward shaping is a technique that introduces imaginary rewards to a reinforcement-learning algorithm, during the learning process, as an addition to the actual reward from the environment, with the purpose of helping the agent learn a desirable behaviour more efficiently [13], [14]. Buffet et al. [15] proposed a shaping methodology for automatically designing multiagent systems. Their approach was based on progressively growing: i) the complexity of the task, so that agents would incrementally learn harder and harder tasks, and ii) the multiagent system, by cloning agents that were trained in a simple environment to a more complex one. They tested their approach on simulated environments where the agents had to coordinate in order to reach a common goal and showed that incremental learning results in more efficient learning of complex tasks. Babes et al. [16] presented a technique they called "social reward shaping" that initialises the Q-function of a Q-learning agent [17] with values derived from an analysis of a mutually beneficial subgame perfect equilibrium of the IPD. By doing so, they effectively encouraged the agents to converge more quickly to mutual cooperation; this was achieved because the initialisation of the Q-function has been shown to be equivalent to adding shaping rewards during the learning process [18].

Agents that rely only on pre-designed reward functions in order to be trained might not truly be called adaptive and autonomous, because they can only cope with environment types to which these functions apply. Different approaches addressing this issue do exist (e.g., [19], [20]). Snel and Hayes [21] investigated the evolution of valence systems (i.e., systems that evaluate positive and negative nature of events) in an environment largely based on Ackley and Littman's artificial life world [19]. They compared the performance of motivational systems that are based on internal drive levels versus systems that are based purely on external sensory input and showed that the performance of the former is significantly better than the performance of the latter. In a recent work, Singh et al. [22] introduced a framework for reward that complements existing RL theory by placing it in an evolutionary context. They demonstrated the emergence of reward functions which capture regularities across environments, as well as the emergence of reward function properties that do not directly reflect the fitness function.

In the case of the IPD, there have been some studies that examined the impact of varying the payoff values. Johnson et al. [23] investigated the reason why the PD has hardly been found in nature. They argued that the assumption of a fixed payoff matrix for each player is not realistic due to variations between individuals on the payoff matrix. They examined the effect of: i) adding normally distributed random errors to the payoff values and ii) the spacing between payoffs. They showed that frequent violations of the payoff structure occur when the interval between payoffs is low. Chong and Yao [24] introduced a co-evolutionary framework where each strategy has its own self-adaptive payoff matrix. The adaptation of each payoff matrix is done by an update rule that provides a form of reinforcement feedback between strategy behaviours and payoff values. By relaxing the restriction of a fixed and symmetric payoff matrix, they showed how different update rules affected the payoff values and subsequently the levels of cooperation in the population. Rezaei and Kirley [25] investigated cooperation in the spatial PD game. They provided each agent with its own payoff matrix which was affected by attributes such as the agent's age and experience level. They showed that time-varying non-symmetric payoff values promote cooperation in this version of the game.

III. Methodology

As we mentioned in the Introduction, the purpose of this work is to evolve the payoff values of the IPD (i.e., T, R, P, S), in order to find more appropriate rewards for the agents, which would motivate them to cooperate. A preliminary investigation was performed with the payoff values shown in Table I, and 2 Q-agents using ϵ-greedy exploration $(\epsilon = 0.1)$ [26], while the game simulation ran for 60000 rounds and 30 trials. The results showed that the agents do not engage in mutual cooperation. The following sections describe the MARL simulation, a method that generates an initial population of valid solutions for the IPD, two mutation operators that alter the individuals but do not change their validity, and the evolutionary algorithm used.

A. Multiagent Reinforcement Learning

The environment is set to be the IPD and the agents implement the Q-learning algorithm. The step size parameter, α, as well as the discount factor, γ, are empirically set to 0.9. We use lookup tables and not function approximators (such as neural networks) to represent the value function, as our previous work showed that lookup tables yield better results in these settings [9], [10]. This might be because the state-action space is simple, so there is no need to generalise from previously experienced states to unseen ones. At every round each agent chooses either to Cooperate (C) or Defect (D), and is provided with incomplete information. In particular, each agent receives only the state of the environment, i.e., the actions of both itself and its opponent in the previous round, as well as the payoff associated with its action, but not the payoff associated with the opponent's action. Both agents need to learn a policy that maximises their long-term rewards and the only way to achieve that is by learning to cooperate. A Boltzmann exploration schedule is utilised, as it gives a good balance between exploration and exploitation, as well as fast convergence. More specifically, an action a_i is selected from state *s* with probability $p(a_i)$ given by Equation 1:

$$p(a_i) = \frac{e^{Q(s,a_i)/t}}{\sum_{a \in \{C,D\}} e^{Q(s,a)/t}} \quad (1)$$

where the temperature *t* is given by t = 1 + 10 $\times$ 0.99^n, with *n* being the number of games played so far. The constants 1, 10 and 0.99 are chosen empirically. More specifically, 10 and 0.99 are chosen because we want to move from exploration to exploitation in less than 1000 rounds, while the constant 1 is added to restrict the temperature from falling below that value[1]. The number of rounds is set to 1000 and the percentages of all outcomes (CC, CD, DC, DD) are averaged over 30 trials.

B. Generating Valid Solutions for the IPD

When generating the initial population, we need to ensure that all individuals consist of payoff values that fall inside the rules of the game. If not, then we need to assign a penalty term on the fitness function in order to be fair with all individuals. However, by evolving the payoff values with this approach, the final solutions might still not satisfy the IPD rules. Moreover, it might be difficult to construct a penalty function for this purpose. One way to generate valid individuals is to randomly assign the T, R, P, S values and reject solutions that are invalid or use a repair method to ensure that they will satisfy the rules. Another way is to incrementally construct and repair the values by utilising domain knowledge (i.e., the rules of the game) at every step. We use the latter approach and more specifically, we randomly generate the values of each individual, all uniformly distributed within the bounds specified below, in the following sequence:

1) $T' \in (b_{min}, b_{max})$
2) $S' \in (b_{min}, T')$
3) $P' \in (S', T')$
4) $R' \in (\max(P', (T' + S')/2), T')$

where the primed payoff values (T', R', P', S') are the newly generated values. The lower bound of R' is $\max(P', (T' + S')/2)$ because we know from the rules of the game that R' should be greater than both P' and $(T' + S')/2$, and P' could either be less or greater than $(T' + S')/2$.

The payoffs need to be generated in this order to ensure that each individual will satisfy the constraints of the game. The initial lower bound, b_{min}, and initial upper bound, b_{max}, are the bounds of the values within which the initial population is generated. We also set some absolute lower and upper bounds, $B_{min} = -50$ and $B_{max} = 50$, in order to restrict the search space. The values -50 and 50 are chosen empirically[2]. In Section IV, we present some experiments where we vary the bounds $[B_{min}, B_{max}]$, but never go beyond [-50,50]. In addition, in some experiments we set the bounds of the initial population $[b_{min}, b_{max})$ either to $[B_{min}, B_{max})$ (which may change according to the experiment), or $[0, 1)$. When initialising the values of the population in the range $[B_{min}, B_{max})$ we effectively help evolution to converge more quickly, since the individuals span the entire search space and not just one region of it (such as $[0, 1)$). The reason why we have chosen to sometimes initialise the population to the range $[0, 1)$ is to show that evolution manages to find very good solutions even though the initial population is not spread in the entire search space. We also conducted experiments where all individuals in the population were initialised to the values of Table I and the results were statistically the same with the results obtained from any other initialisation we tested, therefore we do not report them in this paper.

C. Mutation Operators

In order to introduce variation into the population, two mutation operators are implemented, OP_1 and OP_2. OP_1 changes the magnitude and relative distance between the payoffs by selecting uniformly distributed values for each payoff T, R, P and S, within the following bounds:

- $T' \in (R, \min(2R - S, B_{max}))$
- $R' \in (\max(P, (T + S)/2), T)$
- $P' \in (S, R)$
- $S' \in (B_{min}, \min(P, 2R - T))$

These bounds were calculated from the rules of the game. For example, when considering the value T', we know from the rule $R > (T+S)/2$ that $T < 2R-S$. Since $2R-S$ could be less or greater than the upper bound of the search space, B_{max}, the upper bound of T' should be $\min(2R-S, B_{max})$.

[1]If we let the temperature fall below 1 and use high reward values (such as 50) the term $Q(s, a_i)/t$ in Equation 1 becomes very large causing unnecessary difficulties in calculating the exponential.

[2]Values for B_{max} greater than 50 cause the term $Q(s, a_i)/t$ in Equation 1 to become very large causing unnecessary difficulties in calculating the exponential.

In contrast with OP_1, OP_2 keeps the magnitude and relative distance between the payoffs the same, but it effectively shifts them as follows. A random value is generated from the normal distribution $N(0,1)$ and is added on all the payoff values. If the new payoff values go out of bounds (i.e., $S' < B_{min}$ or $T' > B_{max}$) this is repeated. It is important to note that if the parent payoff values satisfy the rules of the game, this shifting operator generates a valid offspring solution.

D. Evolving payoff values for the IPD

When deciding which fitness function to use, we need to base our decision on a function that is more informative in terms of our goal, which is the fast convergence to the cooperative outcome (CC). We could design our fitness function to be the accumulated payoff of the system (i.e., the sum of the accumulated payoffs of the two agents), since in the IPD the CC outcome results in the highest payoff of the system. However, if we wanted to compare the performance of the system when using payoff matrices that have only negative values, as opposed to when using payoff matrices that have only positive values, this fitness function becomes problematic. The reason is that when using negative payoff values, the fitness value will be negative, whereas when using positive payoff values, the fitness value will be positive. Therefore we need a fitness function that is independent of the payoff and dependent only on the outcome of the game. Such function is the one we are using where the fitness of an individual f_i is calculated as shown in Equation 2:

$$f_i = \overline{CC}_i - \overline{CD}_i - \overline{DC}_i - \overline{DD}_i \quad (2)$$

where i is the individual (i.e., payoff matrix) being evaluated, and $\overline{CC}_i$, $\overline{CD}_i$, $\overline{DC}_i$, and $\overline{DD}_i$ are the average percentages of the corresponding outcomes at the end of the rounds. By trying to maximise this function, we effectively find solutions that maximise the cooperative outcome and minimise all other outcomes. Therefore, a value of 1 means that the game simulation consists of only the CC outcome, a value of 0 means that the CC outcome occurs 50% of the time and a value of -1 means that the CC outcome never occurred in the simulation. A $(\mu + \lambda)$-truncation selection is used which means:

1) μ is the population size and $\lambda = 2 * \mu$, i.e., for each parent individual we generate 2 offspring (one from each mutation operator),
2) we rank all individuals (parents μ and offspring λ) based on their fitness and
3) the best μ individuals from parents and offspring are selected as the new parents for the next generation.

The number of evolution trials is set to 30, the population size to 10 and evolution is repeated for 50 generations. The pseudocode is illustrated in Fig. 1.

```
1  foreach evolution trial do
2      generate and evaluate initial population (P);
3      foreach generation do
4          foreach Individual i ∈ P do
5              Offspring1[i] = mutationOP1(P[i]);
6              Offspring2[i] = mutationOP2(P[i]);
7              evaluate(Offspring1[i]);
8              evaluate(Offspring2[i]);
9          end
10         P = truncSelection(P, Offspring1, Offspring2);
11     end
12 end
```

Fig. 1. Pseudocode of the evolutionary algorithm. Each mutation operator generates one offspring for each member of the population and the best individuals from both parents and offspring are selected for the next generation.

IV. Results

A. Evolution of payoffs

In this first experiment we evolve the payoffs in the range $B_{min} = -50$ and $B_{max} = 50$. The bounds of the values of the initial population are set to $b_{min} = 0$ and $b_{max} = 1$. Fig. 2 illustrates how the probability of mutual cooperation (CC) between the agents changes with the game rounds, when using the initial payoff values and the payoff values found by evolution. The evolved values are the following: $T = 6.37$, $R = 3.01$, $P = -41.04$ and $S = -44.05$. It is clear that the probability of CC when the evolved values are used becomes almost equal to 1 at the end of the simulation, which means that the agents start cooperating from very early in the game. It is interesting to observe that while P and S become close to B_{min}, T and R do not become close to B_{max}. This may suggest that when "winning" (i.e., receiving payoffs T and R), the agents should accumulate small amounts of reward, whereas when "losing" (i.e., receiving payoffs P and S), the agents need to lose much, in order to understand that such behaviour is detrimental.

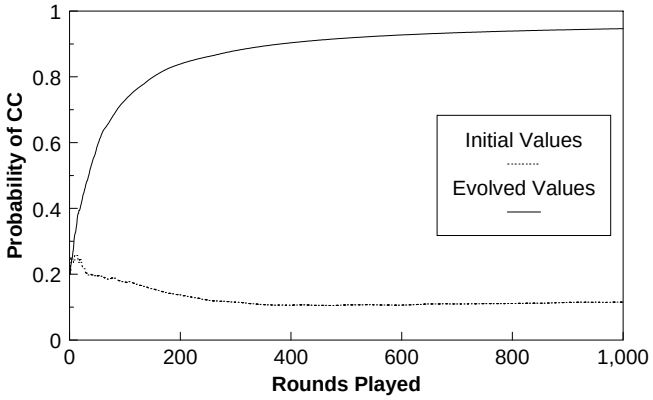

Fig. 2. Probability of mutual cooperation (CC) over time when the Q-agents learn using the initial payoff values (*dotted line*) and the evolved payoff values (*solid line*). The evolved values make the agents reach mutual cooperation very early in the game, whereas with the initial values the agents do not learn to cooperate in 1000 rounds, as the probability of CC reaches only 0.12.

B. Effect of the lower/upper bounds

In the second experiment we investigate how the absolute lower and upper bounds (i.e., parameters B_{min} and B_{max}) affect the solutions, with the purpose being to check whether the magnitude of rewards is important. More specifically, we check 6 cases for the range $[B_{min}, B_{max}]$: $[-1, +1]$, $[-2, +2]$, $[-4, +4]$, $[-8, +8]$, $[-16, +16]$ and $[-32, +32]$. The range of the values of the initial population $[b_{min}, b_{max})$ is set to $[0, 1)$. Table II shows a comparison between these cases based on the fitness of the best individual found by evolution. It is clear that as the range of the values increases so does the fitness. Therefore, by allowing higher magnitudes of evolved rewards/payoffs the system reaches mutual cooperation more often. If the range of the initial population $[b_{min}, b_{max})$ is set to $[B_{min}, B_{max})$ in each case, the results are approximately the same; the only difference is that in the latter case, individuals with high fitness are found earlier, since their values span the entire search space instead of just one region (i.e., $[0, 1)$).

TABLE II

FITNESS VALUES OF THE EVOLVED SOLUTIONS IN THE BOUNDS SPECIFIED. AS THE RANGE OF THE PAYOFFS INCREASES, SO DOES THE FITNESS. THE FITNESS OF THE PAYOFF VALUES OF TABLE I IS SHOWN FOR COMPARISON IN THE FIRST ROW OF THE TABLE

Configuration	Fitness
$T = 5, R = 3, P = 1, S = 0$	-0.769
$[-1, +1]$	0.045
$[-2, +2]$	0.478
$[-4, +4]$	0.557
$[-8, +8]$	0.618
$[-16, +16]$	0.775
$[-32, +32]$	0.910

C. Effect of the reward sign

The third experiment deals with the sign of the rewards. The bounds of the payoffs are kept in a range equal to 5, which is the range specified by the values of the initial case (i.e., the values shown in Table I). More specifically, we check whether evolved solutions that contain all positive values, all negative values, or a mixture of both positive and negative values give better results than the initial case. We test 8 configurations by varying the bounds $[B_{min}, B_{max}]$ from $[-6, -1]$ to $[1, 6]$ incrementing them by 1 in every configuration. The range of the values of the initial population $[b_{min}, b_{max})$ is set to $[B_{min}, B_{max})$ in each configuration. Table III compares these configurations based on the average fitness of the best individual found by evolution.

We observe that when the values of the bounds are negative, the fitness values are greater than 0.5 which means that mutual cooperation occurred more than 75% of the time. The fitness values rise slightly further with the introduction of small positive values; however, as the "penalty" (i.e., negative values) is reduced by moving towards bounds with only positive values, the percentage of mutual cooperation decreases. The highest fitness and thus percentage of mutual cooperation is observed for the range $[-3, 2]$ (as shown in bold).

TABLE III

FITNESS VALUES OF THE EVOLVED SOLUTIONS IN THE BOUNDS SPECIFIED. HIGHEST FITNESS IS OBSERVED FOR THE RANGE $[-3, 2]$ (SHOWN IN BOLD). THE FITNESS OF THE PAYOFF VALUES OF TABLE I IS SHOWN FOR COMPARISON IN THE FIRST ROW OF THE TABLE

Configuration	Fitness
$T = 5, R = 3, P = 1, S = 0$	-0.769
$[-6, -1]$	0.523
$[-5, 0]$	0.525
$[-4, +1]$	0.547
$[-3, +2]$	**0.548**
$[-2, +3]$	0.492
$[-1, +4]$	0.437
$[0, +5]$	0.326
$[+1, +6]$	0.187

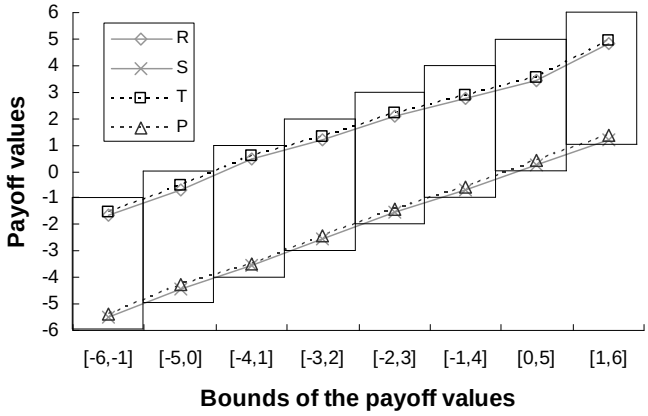

Fig. 3. Evolved payoffs for every range configuration. Boxes indicate the bounds of the corresponding configuration. The following pattern is observed: the values P and S become close to the lower bound of the range specified, while the values T and R become close to the upper bound when the range "includes more" negative values, and seem to be moving away from the upper bound when the range "includes more" positive values.

It is worth noting that for the range $[0, +5]$, in contrast with the payoff values of Table I which have the same range, the evolved payoffs have a positive fitness. The reason why this happens is illustrated more clearly in Fig. 3. The values of the evolved matrices from all configurations appear to follow a pattern. The P and S values become close together and closer to the lower bound, while the values T and R become close together, but closer to the upper bound especially when the values of the bounds are negative. As the values of the bounds increase, T and R seem to be moving away from the upper bound.

D. Effect of the population size

In this experiment, the effect of the population size (p) on the quality of solutions is examined. In particular, population sizes of 1, 2, 4, 6, 8, 10 and 12 are investigated. The bounds of the search space B_{min} and B_{max} are set to -50 and 50 respectively, as described above, and the range of the initial population, $[b_{min}, b_{max})$, is set to [0,1). The results

are depicted in Fig. 4. In Fig. 4 the value of the fitness is written next to the bars that show the payoff values.

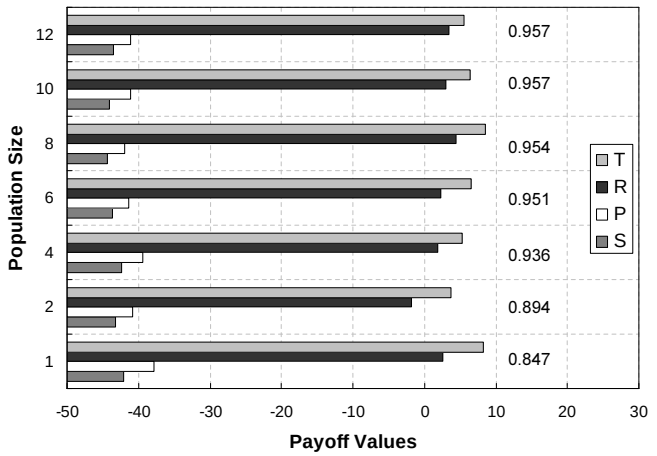

Fig. 4. Evolved payoff values with their fitness (written next to the bars) for different population sizes. Larger population sizes result in better solutions, while good solutions are also found with small population sizes. There is no significant improvement in the fitness with population sizes above 10. While the values P and S are closer to the lower bound, the values T and R are less than 10.

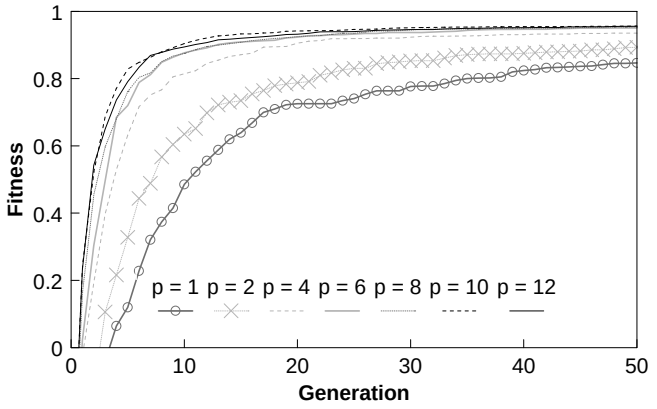

Fig. 5. Fitness values of the best solution found for different population sizes (p) over the generations. Good solutions are obtained in less than 10 generations for all populations.

We observe that as the population size increases, evolution finds better solutions; however, for $p = 10$ and $p = 12$ the fitness is approximately the same suggesting that there is no significant improvement with population sizes above 10. Good solutions are found even when there is only 1 individual in the population. This shows that the problem of evolving the payoff values is solved easily. We observe that in fitter solutions, the values T and R are closer together, while the P and S values are closer together and closer to the lower bound. This is better illustrated when comparing the solution found when $p = 1$, with the solution found when $p = 12$. It is interesting to note that for all population sizes, while the values P and S are closer to the lower bound, the values T and R are less than 10. The effect of the population size is also depicted in Fig. 5, where the fitness of the best individual for different population sizes, is shown over the generations.

E. Effect of the mutation operators

Finally, we examine the role of the mutation operators throughout the evolutionary process and how they affect the solutions. In order to do so, we mark each offspring, whenever it is generated, by a value that indicates which mutation operator created it (OP1 or OP2). After ranking the population in order to select the best elite individuals, we count how many individuals were generated by each mutation operator. At the beginning of each generation, this mark is reset for every member of the population. These counters are averaged over 30 evolution trials, as in all other results. Fig. 6 shows the average number of mutated solutions over the generations with each mutation operator. We notice that OP1 is mostly used in the beginning, thus, we could say that it mostly contributes to the exploration of the search space, while OP2 is used both in the beginning and afterwards, thus explores and fine-tunes the solutions. Towards the end of evolution, the average number of mutated individuals is less than 1, since all members of the population have converged and the quality of solutions cannot be improved any further. It is worth noting that these results are consistent with all population sizes.

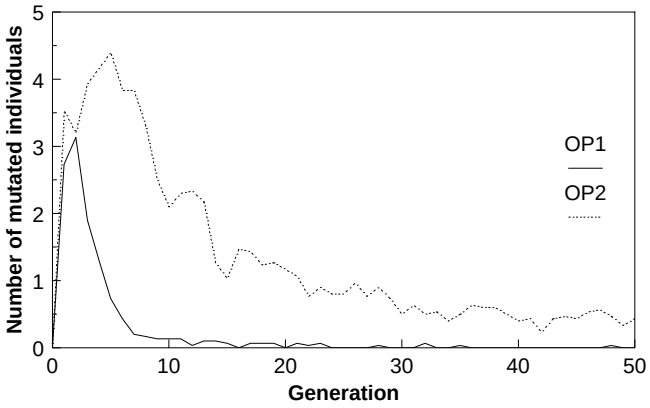

Fig. 6. Average number of mutated individuals over the generations with each mutation operator (OP1 and OP2). OP1 (*solid line*) is activated more in the beginning of evolution, therefore it mostly contributes to the exploration of the search space, while OP2 (*dotted line*) is activated both in the beginning and afterwards, thus explores and fine-tunes the solutions. Towards the end of evolution, the average number of mutated individuals is less than 1, since all members of the population have converged and the quality of solutions can not be improved any further.

V. Discussion and Conclusions

In this paper, we used an evolutionary algorithm to find more appropriate payoffs for agents trained with Q-learning in the IPD. It is important to note that Q-learning is an algorithm proposed for single-agent environments. While in the field of MARL a lot of algorithms were developed that fulfill certain criteria (see [27] for a comprehensive survey of MARL), in this paper we show that even "naive" Q-learners are able to behave efficiently in multiagent settings when their reward function is evolved.

Our approach does not evolve the payoff values while the agents learn; instead, as biological evolution hard-wires primary rewards in animals due to their reproductive success, our algorithm searches for fixed internal rewards for the agents without changing their goal, since the evolved solutions have a valid payoff structure. While the distinction between internal rewards and external sensations is considered ([20], [22]), we ignore the mapping between them for simplicity, and leaving it for future work.

We performed an exhaustive analysis by investigating the effect of: i) the lower and upper bounds of the search space of the payoff values, ii) the reward sign, iii) the population size, and iv) the mutation operators used. The evolved solutions suggest that mutual cooperation between the RL agents is enhanced when: i) T and R are positive, whereas P and S are negative, ii) T and R are close together, while P and S are close together as well, and iii) the magnitude of the negative payoffs P and S is high, whereas the magnitude of the positive payoffs T and R is low. These results show that evolution finds a way to create agents that are motivated to cooperate, since it effectively creates a sense of reward and penalty, thereby "pointing at" the goal (i.e., mutual cooperation).

The evolved payoffs could also suggest that when an agent is "losing", the penalty (i.e., negative reinforcement) should be much higher than the reward (i.e., positive reinforcement) it accumulates when it is "winning". This might indicate a similarity between this method and the WoLF (Win or Learn Fast) principle [28] in MARL, where a smaller learning rate is used when the agent is winning and a much higher one is used when the agent is losing. Certainly, one cannot compare these two approaches, since algorithms that use the WoLF principle work online and very differently than the method we presented. Our goal was not to design another MARL algorithm, but to investigate the impact of evolving the payoff values in the IPD.

It has to be noted that this method works for the configurations presented in this paper. For example, it is unclear how it will perform for different learning agents. Moreover, it was specifically designed for the IPD. How can we adapt it for every 2-player, 2-action game, or for games with an arbitrary number of players and actions? What happens when the game has continuous payoffs or continuous actions? How can it be generalised for stochastic games or for real-world environments? Is there a way to adaptively transform the reward functions of the agents, in order to switch between different sets of collective behaviours? These questions open a broad avenue of future directions, which could ultimately lead to truly adaptive and autonomous multiagent systems.

REFERENCES

[1] Y. Shoham, R. Powers, and T. Grenager, "If multi-agent learning is the answer, what is the question?" *Artificial Intelligence*, vol. 171, no. 7, pp. 365–377, 2007.

[2] A. Rappoport and A. M. Chammah, *Prisoner's dilemma: a study in conflict and cooperation*. Ann Arbor, MI: University of Michigan Press, 1965.

[3] R. M. Axelrod, *The Evolution of Cooperation*. New York: Basic Books, 1984.

[4] G. Kendall, X. Yao, and S. Y. Chong, *The Iterated Prisoners' Dilemma: 20 Years on*, ser. Advances in Natural Computation - Vol. 4. Singapore: World Scientific, 2007.

[5] J. F. Nash, "Equilibrium Points in N-Person Games," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 36, no. 1, pp. 48–49, 1950.

[6] V. Pareto, *Manuale di economia politica*. Milan:Societa Editrice, 1906.

[7] M. Zinkevich, A. R. Greenwald, and M. L. Littman, "A hierarchy of prescriptive goals for multiagent learning," *Artificial Intelligence*, vol. 171, no. 7, pp. 440–447, 2007.

[8] T. W. Sandholm and R. H. Crites, "Multiagent Reinforcement Learning in the Iterated Prisoner's Dilemma," *Biosystems*, vol. 37, no. 1-2, pp. 147–166, 1996.

[9] V. Vassiliades, A. Cleanthous, and C. Christodoulou, "Multiagent Reinforcement Learning with Spiking and Non-Spiking Agents in the Iterated Prisoner's Dilemma," in *Proceedings of the 19th International Conference on Artificial Neural Networks, Part I (ICANN '09)*, ser. Lecture Notes in Computer Science, C. Alippi, M. M. Polycarpou, C. Panayiotou, and G. Ellinas, Eds., vol. 5768. Springer, 2009, pp. 737–746.

[10] ——, "Multiagent Reinforcement Learning: Spiking and Non-Spiking Agents in the Iterated Prisoner's Dilemma," *IEEE Transactions on Neural Networks*, (submitted).

[11] H. S. Seung, "Learning in Spiking Neural Networks by Reinforcement of Stochastic Synaptic Transmission," *Neuron*, vol. 40, no. 6, pp. 1063–1073, 2003.

[12] R. V. Florian, "Reinforcement Learning Through Modulation of Spike-Timing-Dependent Synaptic Plasticity," *Neural Computation*, vol. 19, no. 6, pp. 1468–1502, 2007.

[13] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Proceedings of the 16th International Conference on Machine Learning*, Bled, Slovenia, 1999, pp. 278–287.

[14] J. Randløv, "Shaping in reinforcement learning by changing the physics of the problem," in *Proceedings of the 17th International Conference on Machine Learning (ICML'00)*, P. Langley, Ed., 2000, pp. 767–774.

[15] O. Buffet, A. Dutech, and F. Charpillet, "Shaping multi-agent systems with gradient reinforcement learning," *Autonomous Agents and Multi-Agent Systems*, vol. 15, no. 2, pp. 197–220, 2007.

[16] M. Babes, E. M. de Cote, and M. L. Littman, "Social reward shaping in the prisoner's dilemma," in *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems - Vol.3 (AAMAS'08)*, L. Padgham, D. C. Parkes, J. Müller, and S. Parsons, Eds., Richland, SC, 2008, pp. 1389–1392.

[17] C. Watkins, "Learning from delayed rewards," Ph.D. dissertation, University of Cambridge, 1989.

[18] E. Wiewiora, "Potential-based shaping and Q-value initialization are equivalent," *Journal of Artificial Intelligence Research*, vol. 19, no. 1, pp. 205–208, 2003.

[19] D. E. Ackley and M. L. Littman, "Interactions Between Learning and Evolution," in *Proceedings of the 2nd Conference on Artificial Life*, C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, Eds. Addison-Wesley, 1991, pp. 487–509.

[20] S. Singh, A. G. Barto, and N. Chentanez, "Intrinsically motivated reinforcement learning," in *Advances in Neural Information Processing Systems 17*, L. K. Saul, Y. Weiss, and L. Bottou, Eds. Cambridge, MA: MIT Press, 2005, pp. 1281–1288.

[21] M. Snel and G. M. Hayes, "Evolution of valence systems in an unstable environment," in *Proceedings of the 10th international conference on Simulation of Adaptive Behavior (SAB'08)*, ser. Lecture Notes in Computer Science, M. Asada, J. C. T. Hallam, J.-A. Meyer, and J. Tani, Eds., vol. 5040. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 12–21.

[22] S. P. Singh, R. L. Lewis, and A. G. Barto, "Where do rewards come from?" in *Proceedings of the Annual Conference of the Cognitive Science Society*, 2009, pp. 2601–2606.

[23] D. D. P. Johnson, P. Stopka, and J. Bell, "Individual variation evades the Prisoner's Dilemma," *BMC Evolutionary Biology*, vol. 2, p. 15, 2002.

[24] S. Y. Chong and X. Yao, "Self-adapting payoff matrices in repeated interactions," in *Proceedings of the 2006 IEEE Symposium on Compu-*

tational Intelligence and Games (CIG'06), S. J. Louis and G. Kendall, Eds., 2006, pp. 103–110.

[25] G. Rezaei and M. Kirley, "The effects of time-varying rewards on the evolution of cooperation," *Evolutionary Intelligence - Special issue on simulated evolution and learning*, vol. 2, pp. 207–218, 2009.

[26] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.

[27] L. Buşoniu, R. Babuška, and B. De Schutter, "A Comprehensive Survey of Multiagent Reinforcement Learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, vol. 38, no. 2, pp. 156–172, 2008.

[28] M. H. Bowling and M. M. Veloso, "Rational and Convergent Learning in Stochastic Games," in *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI '01)*, B. Nebel, Ed. Morgan Kaufmann, 2001, pp. 1021–1026.