Multi-Agent Model-Based Reinforcement Learning Experiments in the Pursuit Evasion Game

Bruno Bouzy and Marc Métivier

CRIP5 Université Paris Descartes
45, rue des saints-pères
75006 Paris, France
{bouzy,metivier}@math-info.univ-paris5.fr

Abstract. This paper describes multi-agent learning experiments performed on tactical sequences of the pursuit evasion game on very small grids. It underlines the performance difference between a centralized approach and a distributed approach when using Rmax, a model-based reinforcement learning algorithm. The prey's goal is to go out of the grid and the predators' goal is to kill the prey. The prey may learn or not. The predators learn in two ways: in the centralized approach they are part of one single learning agent, and in the distributed approach, each predator is a learning agent in itself. Every agents learn to accomplish its goal by using Rmax. Our results compare the centralized approach with the distributed approach. Future works mainly include scaling up to larger boards using model-free algorithms, and exploring partial observability of agents.

1 Introduction

Multi-Agent Learning (MAL) is a relatively new field with an increasing number of papers [19]. MAL mainly differs from single-agent learning in that the agent's environment is not stationary because other agents are learning as well. The aim of this paper is to assess the performance of a centralized approach against the performance of a MAL approach in a domain sufficiently simple to make the comparison possible, and sufficiently complex to draw meaningful conclusions. Similar comparisons already exist in coordination matrix games [4] but not in Stochastic Games (SG). Since the Pursuit Evasion Game (PEG) is a classical application in multi-agent systems, and a SG, we have chosen the PEG as testbed. We present the centralized approach and the distributed approach for the predators of the PEG, and make them work with Rmax [3], a model-based reinforcement learning algorithm.

The remainder of this document is as follows. Section 2 presents the current state of MAL. Section 3 lists the variants of the PEG, underlines the works performed on this game in the MAL perspective, and defines the variant used in our experiments. Section 4 presents the two approaches, and details the settings used

to adapt them to Rmax and to the PEG problem. Section 5 highlights the experiments performed with on-line learning agents using Rmax. Section 6 discusses the results. Section 7 concludes, and gives a roadmap for future researches.

2 Multi-agent learning

Multi-Agent Learning (MAL) inherits results from Game Theory (GT) and Reinforcement Learning (RL) [9], and is now rich of various results. When MAL interest came up in the nineties, GT already existed for a long time before, and MAL had to re-use existing GT concepts, appropriate for learning processes of several agents. The most important concept coming from GT is Nash Equilibrium (NE) [14]. In a NE, each player strategy is the best response to other player strategies. Consequently, in a NE, no player wants to change his strategy, and the equilibrium remains. However, although a NE corresponds to a relevant feature of learning algorithms, learning algorithms are not the most efficient method to determine NE, and a large computational literature exists to find them [13].

Although matrix games capture a lot of features of MAL such as cooperation and competition, realistic MAL applications cannot be modeled with them only. SG [17] are more powerful. They extend the single agent Markov Decision Process (MDP), classically used in RL, to include multiple agents whose actions all impact the resulting rewards and state. To this extent, SG are also called Multi-agent MDP (MMDP) [2]. A SG, or MMDP, is a tuple $< n, S, A_{1...n}, T, R_{1...n} >$ where n is the number of agents, S is the set of states, A_i is the set of actions available to agent i (and A is the joint action space $A_1 \times \cdots \times A_n$), T is the transition function $S \times A \times S \rightarrow [0, 1]$, and R_i is a reward function for the ith agent $S \times A \rightarrow \mathbb{R}$.

To address such problems, RL distinguishes two main categories: model-based algorithms or model-free algorithms. Model-based algorithms are very greedy in terms of memory usage because transitions between states are explicitly stored. They use a dynamic programming tool, such as Value Iteration (VI), to update V and Q values optimally regarding the model. Rmax by Brafman and Tennenholtz [3] is a model-based algorithm. Model-free algorithms are more tractable because they only update the values of the currently visited state, and do not store the returns and the transition counts. Examples of model-free algorithms are TD learning [21], or Q-learning [23].

One of the first work dealing with MAL is Tan's work [22]. Examples of first theoretical works were Minimax-Q [12] and Nash-Q [8]. In these works, the question was how to extend Q-learning to MAL by using the NE concept. Some surveys have been published [20, 16, 7] and several agendas have been set [19]: the computational agenda, the normative agenda, the descriptive agenda, the prescriptive agenda (cooperative or not).

3 Pursuit Evasion Game

The Pursuit Evasion Game (PEG) has been used many times to assess learning algorithms, or has been studied in itself. First subsection mentions the various features of PEG. Second subsection presents related works. Third subsection defines the features of the PEG used in the experiments.

3.1 The various features of the game

In all PEG variants, preys and predators are situated on a grid. At each top, the preys and the predators choose an action and move on the grid in respect of the grid rules. The grid can be either four-connected, six-connected or eight-connected. According to the considered variant, the predators' goal is either to kill the preys by occupying the same cell at the same time, or to capture the preys by occupying all the neighbouring cells. Those two goals are similar, but the killing goal is simpler and more basic than the capturing goal. The game can be played with any size of grid. Previous works use size going up to 100x100.

An important feature of the PEG is whether the prey has the explicit goal of going outside the grid, i.e. evade, or not. When the prey cannot quit the grid, its implicit goal is to delay the time of its death, and the goal of predators is to shorten it. In such case, the performance is assessed with a number: the time to kill the prey on average. In this variant the game is called the Pursuit Game. When the prey can escape on the edges, the outcome of a prey's become explicit and binary - killed or escaped - and the PEG name is sound with this definition.

The actions are simultaneous for every agents. They can be staying on the same cell (inaction), or moving to a neighbouring cell. The rewards obtained by the predators can be individual: the killing predator receives the reward, and the other predators receive nothing. The reward can be given to the team of predators. The grid may be either closed by edges or toroidal. Two predators may share the same cell or not. In such case, environmental rules has to be defined: if two predators are choosing the same cell, both may remain on their initial cell. When two predators are allowed to share the same cell, the two predators may become one predator only, or remain two individuals. Besides, obstacles may be part of the grid.

Another set of features regards the strategies used by agents. In most cases, the agents use a stochastic strategy which can be either fixed or learnt. Generally, in most papers, the prey uses a fixed strategy, and the predators are learning. The learning ability can be associated either to individuals, using RL in their lifetime, or associated to an evolving population of non-learning agents.

3.2 Related work

The original work about PEG was done in the context of distributed artificial intelligence by Brenda and al [1] in 1985. In this work, the prey moves randomly, and the predators use various levels of explicit cooperation and communication. In 1992, Levy and Rosenschein propose a game-theoretic view of the pursuit

problem [11]. Richard Korf shows a simple solution to some pursuit games [10]. Instead of using complex multi-agent cooperative schemes, he advises to use two distance-based heuristics: decrease the distance to the prey, and increase the distance between the predators. The first one is sound when the prey is far, the second one can help to kill the prey when she is near the predators. While it is true that these heuristics lower the importance of complex multi-agent communication mechanisms, it is not proven that they actually work when the predators are near the prey ready to find the last killing move. An early reference of PEG and MAL is the work by Tan [22].

The prey predator game has also been studied in the offline context of evolutionary computation: how populations of preys and predators evolve over generations when sharing the same environment. Nishimura and Ikegami model the way predators and preys interact, and they launch experiments to produce these population behaviours [15]. Denzinger and Fuchs describe experiments in learning prototypical situations for variants of the pursuit game [5]. Heynes and Sen describe evolving behavioral strategies [6]. Yong and Miikulainen show how evolving neural networks are successful in the pursuit game [24]. In the continuous case, Sheppard and Salzberg describe a teaching strategy for memory-based control [18].

3.3 Our PEG definition

In previous works, efficiency of the predators situated very close to the prey was not underlined. An exception is [6] which mentions learning preys or smart preys, and which highlights the difficulty of predators to kill a prey going straightforward. Previous works measure the average number of steps to capture or to kill a random prey on large grids. But what would happen if the prey was intelligent? When the predators are situated at distance one or two from an intelligent prey on a large grid, their actions become crucial. If they succeed to kill her, they achieve their goal and receive their reward. However, if they fail, the consequences in term of average number of steps is dramatic: the intelligent prey is not circled anymore, and the predators have to spend many steps to approach the prey again, and obtain another chance to kill her. When the prey is random, and sometimes slower than the predators, failing to kill her is not so dramatic: the predators have simply to follow her again, and wait the time she will go backward.

In our work, we intend to assess the ability of the predators to kill the prey when close to her. Therefore, we explicitly include the evasion goal. When the prey reaches the outside of the grid, its goal is achieved, and the predators cannot reach their goal anymore. In addition, we give the prey the ability to learn. We choose 3x3 grids to permit the use of an optimal model-based method (Rmax) in a completely observable context. Previous works used large sizes, but they did not use a model-based RL algorithm.

To sum up, the prey is killed when a predator is situated on her cell. The reward is +1 for every agents when the prey is killed, and -1 when the prey reached the outside. There is a maximal number of steps for an episode. When

the number of steps is reached without death or evasion, the reward is 0. The prey and the predators have 9 actions: inaction plus the 8 directions. The actions are simultaneous. The prey minimizes its rewards, and the set of predators maximizes them.

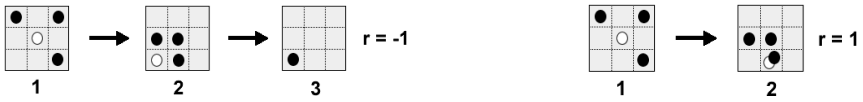

Fig. 1. Two episodes.

Figure 1 shows two simple episodes. A black disc denotes a predator, and a white disc denotes the prey. On the left hand episode of figure 1, the prey goes to the bottom-left cell at step 1, then reaches the outside at step 2 yielding a -1 return. On the right hand episode, the prey goes to the bottom centre cell, also reached by a predator. Thus, at step 1, the prey is killed, and the return is +1.

4 Two approaches: *SetP* and *SumP*

One SG or several MDPs can be modelled. First, the PEG can be modelled as a single SG, i.e. a MMDP called $< n_{all}, S_{all}, A_{all}, T_{all}, R_{all} >$. The agents are the prey and the predators; the positions of agents define the state; the elementary actions of agents are the 9 actions defined in subsection 3.3; the joint action is made up with elementary actions; the transition function is trivial because for each state-action pair there is a unique following state.

Second, one MDP can be modelled for each relevant entity of the PEG. In the MDP for the prey called $< S_{prey}, A_{prey}, T_{prey}, R_{prey} >$, the agent is the prey; the positions of the prey and the predators define the state; the actions are the 9 actions defined in subsection 3.3; the transition function corresponds to the policies of the predators. In the MDP for the set of predators called $SetP =< S_{set}, A_{set}, T_{set}, R_{set} >$, the agent is the set of predators; the positions of the prey and the predators define the state; the actions are the joint actions of the predators; the transition function corresponds to the prey's policy. In the MDP for one particular predator called $< S_{pred}, A_{pred}, T_{pred}, R_{pred} >$, the agent is this particular predator; the positions of the prey, the particular predator, and the other predators define the state; the actions are the 9 actions defined in subsection 3.3; the transition function corresponds to the combination of the the prey's policy with the policies of the other predators. In every cases, the reward function corresponds to the definition of subsection 3.3.

In the centralized approach, there are two agents: the prey with her MDP, and $SetP$ with its MDP. In the distributed approach, there are several agents: the prey with her MDP, the predators with their MDPs, and $SumP$ made up with the predators.

$SetP$ and $SumP$ share the same goal: given a state, they both select an action for the set of predators to play the PEG. $SetP$ and $SumP$ differ in their decision process. While $SetP$ simply chooses one action among its set of actions, $SumP$ asks each predator to choose an elementary action, and $SumP$ joins them into a joint action. In the background of [4], the predators of $SumP$ correspond to ILs, but $SetP$ does not correspond to a JAL since a JAL is part of a distributed approach. With such representation, we have $|S_{prey}| = |S_{set}| = |S_{pred}| = 9^9$ and $|A_{prey}| = |A_{pred}| = 9$ and $|A_{set}| = 9^8$. These bounds are too high to launch experiments in this representation.

4.1 The bitmap representation

These bounds can be largely reduced by considering that all the predators have identical properties. We can represent the predators with a bitmap of their positions, which is a number between 0 and 511. For instance, on problem A4 of Figure 3, the set of predators corresponds to $1 + 4 + 64 + 256 = 325$. A state in S_{set} is represented by the position of the prey, and the bitmap of the set of predators. Thus, we have $|S_{set}| = 9 \times 512$. For a given predator of $SumP$, a state is represented by the position of the prey, its own position, and the bitmap of the other predators. We have $|S_{pred}| = 9 \times 9 \times 512$. The great size reduction brought about by the bitmap representation allows the experiments. However, the downside is the loss of fairness in the number of states managed by $SetP$ or an agent of $SumP$. We have to take this remark into account when we shall interpret the experiments' results. In the bitmap representation, the cells cannot contain more than one predator. Therefore, when several predators go to the same cell, they merge into one predator irreversibly. This does not lose generality because predators learn not to merge.

4.2 The actions of agents

The actions of the prey and the predators are clearly defined in subsection 3.3. However, the actions of $SetP$ need to be defined carefully. On the one hand, considering an action of $SetP$ as a joint action made up with elementary actions, an action of $SetP$ must be sound with its elementary actions. For instance, in state a of figure 2, $SetP$ can act to move into states b, c, d, e and f, but not to state g. State e is reachable because the predators turn like a ring. State f is reachable because all the predators go to the centre and merge. State g is not reachable because the upper right predator cannot reach any black disc on state g. On the other hand, checking online whether an action is sound or not for $SetP$ costs too much time. Consequently, the actions of the $SetP$ MDP must be built and checked offline. Considering that many actions lead to the same following state, it is useful to denote a $SetP$ action by its following state. For instance, going from state a to state e, is possible in two ways: the predators turning clockwise or counterclockwise. For this reason, we also name a $SetP$ action a transition. In conclusion, we have $|A_{set}| = 512$, and the experiments can be launched.

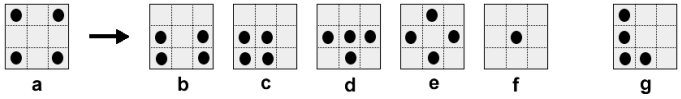

Fig. 2. Legal and illegal joint actions for SetP

4.3 Rmax in practice

The underlying idea of Rmax is very simple. The Rmax agent uses a Q table and a Rmax state table, and acts optimally according its Q table. A Rmax state contains all transition counts to all the possible following states with the corresponding returns. There is a fictitious state to which every Rmax state is linked to by a transition with a return Rmax, and transition count 1. For every actual Rmax states, the initial returns are initialized with the Rmax value as well, and the transition counts set to 0. The simulations are launched. When a transition is traversed, its count is incremented, and its return updated. Theoretically, after each Rmax state update, VI is called and the optimal policy computed regarding the current values of returns and transition counts. It means that the Rmax agent acts optimally regarding its fictitious model.

In practice, we overcame three downsides of Rmax. The first downside is memory usage of agent x which is in $|S_x| \times |A_x| \times |T_x|$. Here, $|T_x|$ denotes the number of transitions starting from a state-action pair of agent x. PEG being deterministic, we have $|T_x| = |A_y|$ where y represents the set of agents different from x. We have $|T_{prey}| = |A_{set}|$, $|T_{set}| = |A_{prey}|$, and $|T_{pred}| = |A_{prey}| \times |A_{set}|$. Thus $|S_{set}| \times |A_{set}| \times |T_{set}| = 9 \times 512 \times 512 \times 9 = 25,000,000$, which can be stored in today personal computer memory. $|S_{pred}| \times |A_{pred}| \times |T_{pred}| = 9 \times 512 \times 9 \times 9 \times 512 \times 9 = 2,500,000,000$, which can hardly be stored. Consequently, we limited the number of predators to 3, which leads to $|A_{set}| = 130$. With this limitation, $|S_{pred}| \times |A_{pred}| \times |T_{pred}| = 9 \times 130 \times 9 \times 9 \times 130 \times 9 = 150,000,000$ which can be contained in memory. In order to minimize the memory used by all the agents in the $SumP$ case, the predators share a single Q table and a single Rmax state table. This choice induces a kind of communication between predators which will be discussed in section 6.

The second downside of Rmax concerns its convergence time. In theory, Rmax garantees a polynomial time to reach near-optimal behaviour. In practice, in the beginning of a run, Rmax favours exploration, and not exploitation, thus the mean value of the episodes' returns is strongly biased at the beginning. Only when good policies are found, many subsequent episodes are used to make the mean value sound with these good policies.

The third downside of Rmax is mixed policy management. In theory, Rmax is supposed to launch VI after each step, and the optimal action can change from one step to the next one, which implicitly brings about a mixed behaviour to Rmax. In practice, the computational cost of VI is high. If VI was called after each step, the execution of the simulations would be too slow. Consequently, VI can be launched after a given number of steps only. Therefore, between two VI

runs, Rmax uses the same policy, which is harmful against an opponent more reactive.

5 Experiments

This section describes experiments in which all the agents are acting and learning online in the environment. Subsection 5.1 presents some basic experiments to test our PEG configurations with $SetP$. Then, subsection 5.2 defines new problems allowing to compare the two approaches $SetP$ and $SumP$. Then, subsection 5.3 shows the relative results obtained by $SetP$ and $SumP$ when playing against a prey on these new problems.

5.1 A first basic experiment with *SetP*

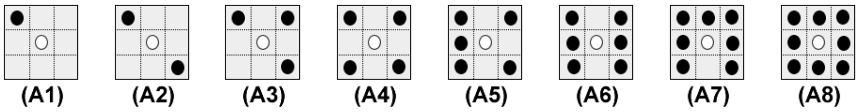

Fig. 3. The first set of test problems.

In this experiment, we set up eight problems, with a number of predators going from one up to eight, shown by figure 3. On these problems, we expect to observe the following behaviours. On the first three problems, the prey has a trivial optimal strategy to reach the outside. On problem A8, against a random prey, the set of predators has the optimal strategy to stay. Consequently, a prey may learn to stay as well. On this set of problems, the prey has a chance to escape decreasing in the number of predators.

	A1	A2	A3	A4	A5	A6	A7	A8
Rewards	-0.34	0.04	0.34	0.55	0.71	0.81	0.91	1.0

Table 1. Average rewards obtained by SetP in problems A1 to A8

Table 1 shows average rewards obtained by $SetP$ on problems A1 to A8. Because of memory usage discussed in subsection 4.3, $SumP$ could not be launched.

5.2 Setting up new problems

Subsection 4.3 mentioned the necessity to limit the number of predators to 3, which importantly reduced the interest of problems defined in subsection 5.1. Therefore, we added a new feature to the problems: a cell has a boolean value

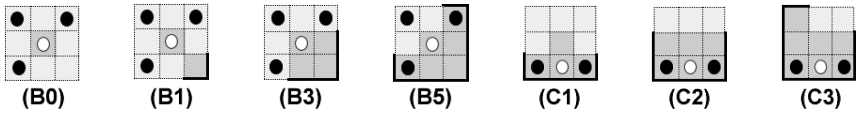

Fig. 4. The second set of test problems.

indicating wether the outside of the grid can be reached from this cell. We defined the problems of figure 4.

Problem B1 looks like problem A3 or B0, but the lower right cell is not an access to the outside (denoted by black edges, and a darker background). Problem B3 is the same as problem B1, with two supplementary edge cells without access to the outside. Problem B5 is the same as problem B3, with two supplementary corner cells without access to the outside. Similarly, problems C1, C2, C3 are identical in the agents' positions, but they differ in their access to the outside. Although created for reducing memory usage, this set contains interesting problems: instead of episodes with length two for an optimal prey to reach outside, these new problems may have episodes with length three. For non optimal agents, episodes can be longer.

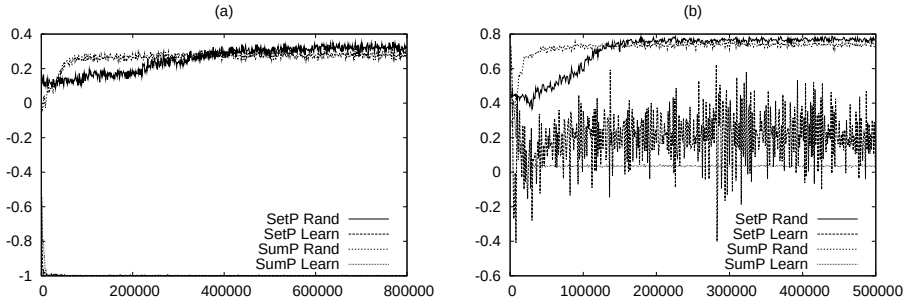

Fig. 5. Average returns obtained on problems B0(a) and B3(b).

5.3 Results

We launched experiments that consist of $1,000,000$ episodes. Each episode lasts from 2 up to 10 steps. It terminates either when the prey is killed (return +1), escaped (return -1) or maximal number of steps reached (return 0). The result of an experiment on a problem is the mean value of all the episodes starting on this problem. The predators are used either in the $SetP$ mode or in the $SumP$ mode. They learn by using Rmax. The prey is either random or learning with Rmax. We defined pVI, the episode period at which VI was called, and we experimentally set $pVI = 100$. There is no discount rate between states in Rmax. The value of

$Rmax$ was set to +1 for the predators, and to -1 for the prey (it should be named $Rmin$). For each problem, we provide the results in a figure showing the curves. On each figure, there are four curves corresponding to the four cases, $SetP$ or $SumP$, random or learning prey. Each curve represents an average computed on three experiments launched with different seeds. The average returns given in the curves descriptions are computed on the last $10,000$ episodes. We used single-core 2.8 GHz computer with 1 Gb of RAM. One experiment with $1,000,000$ episodes lasts about 10 minutes for $SetP$ and 100 minutes for $SumP$.

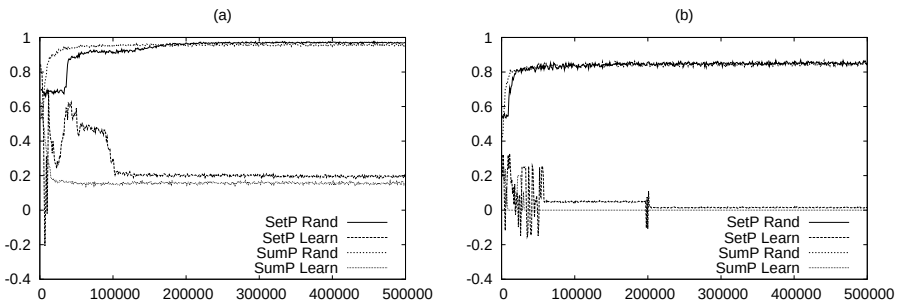

Fig. 6. Average returns obtained on problems B5(a) and C1(b).

Results on problems B0 B3 B5 Figure 5a shows the four curves corresponding to problem B0. In this experiment, both $SetP$ and $SumP$ reach the -1 result against a learning prey, which was expected. Against a random prey, they both learn to kill it with a medium success rate (0.3) only. During the first episodes, $SetP$ is better than $SumP$. Then, after $10,000$ episodes, $SumP$ becomes better than $SetP$ until episode $300,000$. Finally, after $1,000,000$ episodes, we observed that $SetP$ reaches a better rate (0.34) than $SumP$ (0.31). It is worth noting that the success rates between the single agent and the multi-agent approaches are rather similar.

Fig. 7. Some positions following B5, C1 and C3.

Figure 5b shows the four curves corresponding to problem B3. Against a random prey, $SumP$ starts better than $SetP$, but after $150,000$ episodes, $SetP$ becomes slightly better. Finally, after $1,000,000$ episodes, we observed that $SetP$

reaches a better rate (0.77) than $SumP$ (0.73). Against a learning prey, $SetP$ is better than $SumP$ but the variance is high. Besides, results obtained on problem B1 are similar to those obtained on problem B3.

Figure 6a shows the four curves corresponding to problem B5. Against a random prey, $SumP$ starts better than $SetP$ again, but after $200,000$ episodes, $SetP$ becomes slightly better. Finally, after $1,000,000$ episodes, we observed that $SetP$ reaches a better rate (0.99) than $SumP$ (0.96). Against a learning prey, between episode $50,000$ and $100,000$, $SetP$ is clearly better than $SumP$. But after episode $100,000$, the prey learns that inaction is better for her, and the average returns decrease down to 0.2. This kind of effect results from simultaneous competitive learning. Agents are learning given the other agents' behaviours. On problem B5, we observe learning stages: given $SetP$ is random at the beginning, the prey learns her sequence of actions. Then, given the prey performs her sequence, $SetP$ learns to do his sequence, and so on. Finally, the prey intends to move to the closed corner, and the three predators intend to move to the three cells giving access to the outside, leading to states similar to state h of figure 7.

Results on problems C1, C2, C3 Figure 6b shows the four curves corresponding to problem C1. Against a random prey, $SumP$ and $SetP$ reach the same rate (0.82). Against a learning prey, the average return roughly equals 0. Basically, after a sufficient number of episodes (about $100,000$), the predators and the prey learn a common tactical sequence generally leading to the maximal number of steps. Thus, the predators and the prey receive the 0 return. Actually, after the first step of problem C1, the agents come into state i of figure 7, and stay in it until the end of the episode. We saw that, if the prey leaves state i, then its expected return increases to a strictly positive value. Consequently, the prey, that aims at minimizing its returns, stays in this state. Similarly, if $SumP$ and $SetP$ move out from state i, then they decrease their expected returns to a strictly negative value, not interesting for them. Therefore, they stay in this state as well.

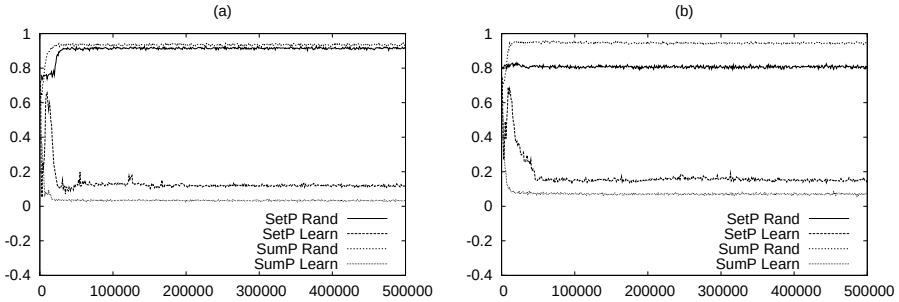

Fig. 8. Average returns obtained on problems C2(a) and C3(b).

This effect is very dependent to the way we give the returns to the system. To avoid such blocked situation, when the maximal number of steps is reached, instead of giving a 0 return, we could give different penalizing returns to the prey (+1), and to the predators (-1). This way, the agents would not compromise, and would fight to reach their goal at any condition.

Figure 8a shows the four curves corresponding to problem C2. Against a random prey, $SumP$ and $SetP$ reach the same rate (0.92). Against a learning prey, the average return is close to 0, but different to 0. Figure 8b shows the four curves corresponding to problem C3. Against a random prey, $SumP$ reaches a better rate (0.92) than $SetP$ (0.81), which is significant. This simple problem is important in that it illustrates the fact that a multi-agent system can be better than a centralized approach. The reason why $SumP$ is better than $SetP$ is hard to find out because this superiority lies in a significant but slight difference in a success rate. We looked at the sequence played by $SumP$ and $SetP$ against the learning prey. First, we saw that $SumP$ found out very quickly a working sequence consisting in going into state k of figure 7 at the first step, then oscillating between state j and state k. Such behaviour is simple. It surely prevents the prey from accessing to the outside, and hopefully expects to kill the random prey. Second, we saw that $SetP$ found out three important states: states j, k, and l of figure 7. As $SumP$ did at the first step, $SetP$ goes into one of these states. But, afterward, instead of oscillating between those states, $SetP$ still explores other states which are not important. $SumP$ found out very quickly a correct sequence, and $SetP$ is still exploring states. It is worth noting that problem C3 is not symmetrical, which can help the two independent agents to find their first move: going to state k, and not state j. Besides, against a learning prey, $SetP$ reaches a better rate than $SumP$.

6 Discussion

To sum up the results, we found out that, in most problems against a random prey, $SetP$ obtained slightly better rates than $SumP$ did. However, we saw specific problems like problem C3, in which $SumP$ is significantly better than $SetP$. We observed that against a random prey, the problems on which $SumP$ works better than $SetP$ are those where the number of predators is high compared to the number of outside accesses. Besides, against a learning prey, we did not see $SumP$ better than $SetP$ on our set of problems.

In the field of PEG, our work is new mainly for three reasons. First, it involves Rmax, a model-based method known as optimal in a completely observable context. Second, it focusses on tactical problems of PEG when the prey is close to the predators. Third, except in [6], previous works in PEG considered the prey as random and without the explicit goal of escaping. Our work shows results obtained by predators against a learning prey that has the explicit goal of escaping to the outside, which highly complicates the task of the predators.

An important question in the MAL background is to explicitly say to which extent our predators of $SumP$ are cooperative or not according to our imple-

mentation of Rmax. On the one hand, a predator of $SumP$ is non cooperative in that each predator has a state depending on himself: his position, the position of the set of other predators, and the prey position. Two distinct predators have two distinct states. The predator decision process lies in his Q values indexed by this information and its action. A predator chooses its action independently. On the other hand, as mentioned by subsection 4.3, for memory size consideration, the predators share the same Q table, and the same Rmax state table. Thus, the absence of communication between predators is not garanteed. However, a predator situated on a specific cell updates the Q values and Rmax states corresponding to this cell in particular, and another predator situated on another cell updates the Q values and Rmax states corresponding to this other cell. In other words, although sharing the same tables, the predators update different parts of the tables, and independence is mostly preserved. In addition to the memory reduction upside, sharing tables also speeds up the process of filling them: they are updated in parallel. In the context of exploration, this remark is crucial. To fairly assess $SumP$ and garantee it is really and undoubtfully non cooperative we should define one Q table and one Rmax state table per predator.

7 Conclusion

In this paper we assessed a MAL approach against a single-agent approach in the PEG. In previous works, PEG was generally used with very large grids, but in the current work, we focussed on 3x3 grids to measure the ability of predators to actually kill a prey already close to them, which is new in the PEG field. We also enabled the prey to escape and learn, which was not really done before. We performed online experiments with Rmax, a model-based reinforcement learning algorithm. We defined $SetP$ the centralized approach and $SumP$ the multi-agent system made up with predators taken as individuals. Online experiments were launched to assess the performances of $SetP$, and $SumP$, learning to act against a prey either random or learning. On most problems, we observe that $SetP$ gave better results than $SumP$, but $SumP$ surpassed $SetP$ on some other specific problems.

Future works consist in, first launching online experiments on problems A4 to A8 including more predators to test cooperation more deeply, second studying competition between the prey and the predators, third comparing the current model-based approach using Rmax with a model-free approach using Q-learning. Finally, we want to scale up to larger grids with partially observing predators which could apply the results presented here to efficiently kill the prey when close to the predators.

References

1. M. Benda, V. Jagannanthan, and R. Dodhiawalla. An optimal cooperation of knowledge sources. Technical report, Boeing AI Center, August 1985.

2. C. Boutilier. Planning, learning and coordination in multiagent decision processes. In *Theoretical Aspects of Rationality and Knowledge*, pages 195–201, 1996.
3. R. Brafman and M. Tennenholtz. R-max - a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3:213–231, 2002.
4. C. Claus and C. Boutilier. The dynamics of reinforcement learning in cooperative multi-agent systems. In *AAAI*, pages 746–752, 1998.
5. J. Denzinger and M. Fuchs. Experiments in learning prototypical situations for variants of the pursuit game. In *Proceedings of the Second ICMAS conference*, pages 48–55, Kyoto, Japan, August 1996.
6. T. Haynes and S. Sen. Evolving behavioral strategies in predators and prey. *Adaptation and Learning in Multi-Agent Systems*, 1996.
7. P. J. Hoen, K. Tuyls, L. Panait, S. Luke, and J. A. L. Poutré. An overview of cooperative and competitive multiagent learning. In K. Tuyls, P. J. Hoen, K. Verbeeck, and S. Sen, editors, *Learning and Adaption in Multi-Agent Systems, First International Workshop, LAMAS 2005, Revised Selected Papers*, volume 3898 of *Lecture Notes in Computer Science*, pages 1–46. Springer, 2006.
8. J. Hu and M. Wellman. Multi-agent reinforcement learning: theoretical framework and an algorithm. In *ICML*, pages 242–250, 1998.
9. L. Kaelbling, M. Littman, and A. Moore. reinforcement learning: a survey. *Journal of Artificial Intelligence Research*, 4, 1996.
10. R. E. Korf. A simple solution to pursuit games. In *Working Papers of the Eleventh International Workshop on DAI*, pages 195–213, Geneva, Switzerland, 1992.
11. R. Levy and J. S. Rosenschein. A game theoretic approach to distributed artificial intelligence and the pursuit problem (abstract). *SIGOIS Bull.*, 13(3):11, 1992.
12. M. Littman. Markov games as a framework for multi-agent reinforcement learning. In *ICML*, pages 157–163, 1994.
13. R. McKelvey and A. McLennan. Computation of equilibria in finite games. *Handbook of Computational Economics*, 1996.
14. J. Nash. Non cooperative games. *Annals of Mathematics*, 54:286–295, 1951.
15. S. I. Nishimura and T. Ikegami. Emergence of collective strategies in a prey-predator game model. *Artificial Life*, 3(1):243–260, July 1997.
16. L. Panait and S. Luke. Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 11(3):387–434, 2005.
17. L. Shapley. Stochastic games. *Proceedings of National Academy of Science*, 39:1095–1100, 1953.
18. J. W. Sheppard and S. L. Salzberg. A teaching strategy for memory-based control. *Artificial Intelligence Review, Special Issue on Lazy Learning*, 11:343–370, 1997.
19. Y. Shoham, R. Powers, and T. Grenager. If multi-agent learning is the answer, what is the question? *Artificial Intelligence*, 171:365–377, 2007.
20. P. Stone and M. M. Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, 2000.
21. R. Sutton. Learning to predict by the method of temporal differences. *Machine Learning*, 3:9–44, 1988.
22. M. Tan. Multi-agent reinforcement learning: independent vs cooperative agents. In *ICML*, pages 330–337, 1993.
23. C. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.
24. C. H. Yong and R. Miikkulainen. Cooperative co-evolution of multi-agent systems. Technical report, Department of Computer Science, University of Texas, July 2001.