



# Shaping Multi-Agent Systems with Gradient Reinforcement Learning

Olivier Buffet, Alain Dutech, François Charpillet

► **To cite this version:**

Olivier Buffet, Alain Dutech, François Charpillet. Shaping Multi-Agent Systems with Gradient Reinforcement Learning. Autonomous Agents and Multi-Agent Systems, Springer Verlag, 2007, 15 (2), pp.197–220. .

**HAL Id: inria-00118983**

**<https://hal.inria.fr/inria-00118983>**

Submitted on 8 Dec 2006

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

---

# Shaping Multi-Agent Systems with Gradient Reinforcement Learning

Olivier Buffet<sup>1\*</sup> – Alain Dutech<sup>\*\*</sup> – François Charpillet<sup>\*\*</sup>

\* LAAS/CNRS - Groupe RIS  
Avenue du Colonel Roche  
31077 Toulouse CEDEX 4 / France  
olivier.bufferet@laas.fr

\*\* LORIA - INRIA-Lorraine / Campus Scientifique - B.P. 239  
54506 Vandœuvre-lès-Nancy / France  
{dutech,charp}@loria.fr

---

RÉSUMÉ. A définir par la commande `\resume{...}`

ABSTRACT. An original Reinforcement Learning (RL) methodology is proposed for the design of multi-agent systems. In the realistic setting of situated agents with local perception, the task of automatically building a coordinated system is of crucial importance. To that end, we design simple reactive agents in a decentralized way as independent learners. But to cope with the difficulties inherent to RL used in that framework, we have developed an incremental learning algorithm where agents face a sequence of progressively more complex tasks. We illustrate this general framework by computer experiments where agents have to coordinate to reach a global goal.

MOTS-CLÉS: A définir par la commande `\motscles{...}`

KEYWORDS: Reinforcement Learning, Multi-Agent Systems, Partially Observable Markov Decision Processes, Shaping, Policy-Gradient

---

---

1. This work has been conducted in part in NICTA's Canberra laboratory.

Abbreviations used in this document :

- RL - Reinforcement Learning
- MAS- Multi-Agent System
- MDP -Markov Decision Process
- POMDP -Partially Observable Markov Decision Process

## 1. Introduction

The problem of automating the design of a Multi-Agent System (MAS) is at the heart of much research. It is an important issue mostly related to a crucial question : how to link the *global* description of a task with agents whose behaviours depend on a necessarily *partial* and *local* view of this task.

This design problem is all the more tricky as “reactive” and “cooperative” MAS are considered, since these systems rely on interactions among—often many—agents in order to produce a complex collective behaviour relying in particular on self-organisation phenomena. The main difficulty comes from the fact that self-organisation is a process that is not well understood and not easy to control. Thus, the only solution for the designer is often to undergo a tedious task of tuning the parameters of the reactive behaviours so as to obtain the desired collective behaviour. On the other hand, the greatest advantage of these systems is the simplicity of the agents, which eases their design.

Reinforcement Learning (RL) ( , ) is a common approach to decision-making under uncertainty. It is also an appealing tool to tackle the automation of MAS design ( , ) in a decentralised manner. This is the approach adopted in the present paper, in which independent learners try to achieve a common goal. Our interest is to see each agent learn locally how to optimise a global performance. There are many advantages to this approach :

- Because we employ Reinforcement Learning, a simple scalar signal evaluating the system behaviour (the reward) is sufficient to learn. It is not necessary to have a teacher knowing the problem’s solution beforehand.
- The decentralised framework often makes the task faced by each agent of a reactive MAS rather simple. In many cases, it is easier for each agent to learn its local reactive behaviour than to try learning the system’s collective behaviour. This is a way to avoid an important pitfall of RL : a combinatorial explosion is all the more probable that the problem is complex.

Moreover, Reinforcement Learning methods are based on the mathematical formalism of *Markov Decision Processes* (MDP) that details the assumptions to be satisfied for an algorithm to converge. This mathematical framework also gives a formalism for the study of a MAS’s collective behaviour.

The use of Reinforcement Learning in a decentralised fashion for Multi-Agent Systems causes some difficulties. Indeed, our approach is not in the precise framework of MDPs (because of the multi-agent partially observable setting), which leads to the loss of the usual guarantees that the algorithm converges to an optimal behaviour. This is due to the fact that each agent is constrained to a local and partial view of the system and, as a consequence, generally does not know the system’s global state<sup>1</sup>.

Common model-free learning algorithms are for example  $Q$ -learning and Sarsa ( ). Under the Markov assumption, they efficiently find a globally optimal deterministic policy (one such policy exists in this case). Yet, because each of our learners is facing a non-Markovian problem (due to a decision made with insufficient information), we have to use direct policy search algorithms instead, looking for an—at least—*locally* optimal *stochastic* policy.

Based on such a classical RL technique to find stochastic policies, our proposition is then to make use of a *decentralised incremental* learning algorithm :

- The learning is *decentralised* because the group’s behaviour evolves through the independent learning processes of each agent.
- By *incremental*, we mean that agents are progressively pitted against harder and harder tasks so as to progressively learn a more complex behaviour.

Another way in which our approach could be called incremental is that initially learning is performed with very few agents, so as to minimise coordination and cross-work actions, and then the resulting basic behaviours are exported to learning tasks with more and more agents. Eventually, behaviours can be further refined by learning in these more demanding environments. Thus, the originality of our approach is twofold : learning is decentralised and incremental.

**Note :** In a broader sense, incremental RL is generally referred to as *shaping* (Section 4.2 comes back to this subject).

In this paper, we present our incremental (shaping) method for learning in a multi-agent system and an experiment on which we tested our methodology. Section 2 discusses related work, Section 3 brings some background knowledge, Section 4 describes the approach we follow, and Section 5 describes the experiments conducted to test the viability of our approach and the results obtained. A discussion of our work follows in Section 6, highlighted by some comparisons with other similar works. Future directions and conclusive remarks end the paper in Section 7.

---

1. Note that most other studies make the opposite assumption of a total perception by each agent.

## 2. Related Work

### 2.1. *On explicit coordination*

Boutilier ( ) has studied the problem of coordination from a theoretical point of view, using Multi-agent Markov Decision Processes (MMDP). In his framework, each agent can compute a globally optimal policy where only coordination problems are to be solved, i.e. when the optimal joint action is not unique. As he points out, such a coordination can be reached using social laws, communication or a learning mechanism. A major drawback of this planning framework is that it requires not only a global perception, but also the ability to recognise other agents, which is rarely the case in a reactive MAS.

#### Modelling/Recognising Other Agents —

Even without Boutilier's strong hypotheses, the ability to recognise other agents can be useful to understand their behaviours (learn their policies or determine their goals). Taking this information into account to choose an action may help improving the coordination and global cooperation. Yet, the other agent's model must be accurate since, as shown by Hu and Wellman ( ), a bad model may be worse than no model. Historically, the modelling of other agents has mainly been used in competitive situations, other agents being viewed as opponents ( , ). Works often consider game theory and the simple prisoners dilemma ( ).

#### Communication —

Communication could also be used to solve the problem of explicit coordination. But attention must be paid to the fact that communication in itself has a cost ( ): it is a new resource to manage. Moreover, communication may have an important impact on the complexity of decision-making ( ). An important aspect of communication is the question of its content : intended actions, utility of actions, belief state, perceptions... ? A possible direction for tackling this question is that of learning (or "agreeing on") the interpretation of messages that have no prior common meaning ( , ).

### 2.2. *Reward Definition*

#### Individual Viewpoints —

It is very important to remember that defining the reward function is a crucial but difficult phase. A first point is that a "truly" autonomous agent should not rely on external help to receive reinforcement signals, but on an internal mapping from observations and actions to a scalar reward.<sup>2</sup> This is similar to the viewpoint adopted by Fernández and Parker ( ), where the team tries to maximise the accumulated reward

---

2. But an agent learning in "controlled conditions" could get its reward from a third party, which is not our choice.

of all teammates despite the lack of communication. On the contrary, much research on multi-agent frameworks relies on complete observability and, as such, a common global reward function may be used by each agent, ensuring that they work toward a common objective.

#### Multi-Agent Credit Assignment Problem —

In all cases, as we mentioned in Section 3.3, the multi-agent credit assignment problem (how to define individual reward functions when considering a global—cooperative—task) remains difficult, as can be observed in Tumer and Wolpert’s work on this topic (Collective INtelligence ( )). The main results obtained only concern situations where agents can be organised in sub-groups working on totally independent problems.

#### Modifying the Reward Function —

The reward function can be modified to efficiently guide the learning agent. This has been applied in Matarić’s work with multi-robots systems ( ), where she takes the option of adapting the reward function and the agents’ actions and perceptions to use a simple reinforcement algorithm. The reward is not a binary process (no reward / big reward at goal) but uses a “progress estimator”, i.e. a smoother reward adapted to give very frequent hints to each individual agent. This method requires an important work from the human designer and is strongly task-dependent. Another example is that of Stone and Veloso’s work on simulated robotic soccer ( ), where they also choose to define a reward function giving an estimate of the team’s progress.

The definition of a reward function—and how to use it—is an important problem that should always be kept in mind in multi-agent Reinforcement Learning. Moreover, theoretical advances are still required in the multi-agent framework, so as to ensure that individual reinforcement signals appropriately lead to an intended global goal.

### 2.3. Learning Algorithm

#### Partial Observability –

In a partially observable setting, well informed decisions can be made using past observations to disambiguate the current situation.

One approach is that of computing a probability distribution over states (what gives a “belief state”) ( ). But this requires knowing the underlying model of the system and can be computationally very expensive. Furthermore, in our setting, the underlying MDP changes when the complexity of the problem increases. This would make it very difficult to reuse any knowledge. Another approach consists in defining a policy on a sequence of past observations ( , ), which is as scalable as the “instant” policy we are using. The main difficulty is that learning consumes much more time and memory when considering past observations.

To our knowledge, most practical experiments in multi-agent settings have only involved policies depending only on the immediate observation ( $s, a$ ). In this case, direct policy search (looking for stochastic policies) has had more success than dynamic programming (producing a deterministic policy) when both types of algorithm have been compared, as done by Salustowicz et al. ( ) or as we experienced ourselves using  $Q$ -learning in earlier experiments. Work by Peshkin et al. ( ) and Baxter et al. ( ) also make use of a direct policy search in multi-agent settings.

A particular case of RL based on current observation is that of TPOT-RL ( ), which seems specific to problems like the robot soccer it has been designed for. Indeed, it considers that the next state is not accessible when the ball is being passed to another player for example. As a consequence, a classical update using  $V(s')$  to compute  $Q(s, a)$ —if the transition is  $(s, a) \rightarrow s'$ —is not possible, hence the idea of using a Monte-Carlo sampling to evaluate the expected future reward.

#### Multi-Agent Aspect –

In a multi-agent setting, better results logically require the agent to reason about other agents' reasoning ( ). But this requires knowledge about this other agent's behaviour, while we have seen in Sec. 2.1 that it is not convenient to distinguish agents and that it would again lead to computationally expensive algorithms.

#### Architecture –

In some challenging applications of Reinforcement Learning, a monolithic architecture seems often insufficient to tackle the complexity of the environment. A good way to go can be to decompose the problem according to different levels of abstraction (from low-level controllers to high-level strategies). This is quite common in robotics, with examples in foraging ( ), robotic soccer ( , ). In this domain, different questions arise as : 'What should the various levels be?' and 'Can they be automatically defined ( )?' We do not make use of a hierarchical architecture, so as to focus on our proposed shaping methodology.

### 2.4. Automated Shaping

Existing work in the field of shaping ( , , ) shows how efficient shaping can be, and gives some theoretical results on how a problem can be modified without altering the solution. Yet, how to modify a problem so that it is easier to solve remains an open question. It is probable that if one knows how to modify the problem, one probably knows a lot about the solution. Two experiments using reward shaping in a multi-agent application have been mentioned in Sec. 2.2 ( , ).

Asada's completely observable model ( ) made it possible to automate the definition of the agent's training. It was possible to evaluate the distance to the goal thanks to an ordering of perceptions. In the case of factored perceptions (a perception being described by a vector of variables), an interesting direction could be to analyse the

frequency of changes in the variables, as done by Hengst ( ) to discover hierarchy in Markov Decision Processes.

### 3. Background

In this section, we first describe the precise problem of Multi-Agent System design we want to address. We then present Reinforcement Learning, focusing on the details which are of interest to our work. Lastly, we explain which particular difficulties are met due to the use of RL in a MAS. This will lead to the original methodology proposed in Section 4.

#### 3.1. Designing Multi-Agent Systems

We are interested in automating the design of cooperative Multi-Agent Systems by having each individual agent learn its own behaviour. As mentioned earlier, we have decided to work with very simple reactive agents for complexity reasons. Besides, it allows us to concentrate on the learning aspect of the design.

Among many possible choices, our agents can be characterised as :

- **reactive** : Agents have “reflex” behaviours, act based on current observation only (memoryless behaviour).
- **situated with local perception** : Even if “partial” observations are a disadvantage when learning, their “local” aspect is of benefit as it limits the risk of combinatorial explosion (only a few elements are perceived).
- **possibly heterogeneous** : Although they may have the same abilities for perception and action, each agent can acquire a different behaviour from the others, as agents learn individually in the adopted learning process.
- **cooperative** : All agents share the same goal and they *will* have to coordinate to reach it.

Agents may have other characteristics (such as communication skills), yet we do not have any preferences on them, as they do not play any crucial role in our approach.

The question raised is then the following : given a global task (i.e. a task that a priori requires a global view of the system in order to be solved), how to design – independently and in decentralised way– the individual behaviours of agents having a local and partial view of their environment ? For the reasons mentioned in the introduction (formalism, top-down approach with no supervision), we propose to tackle this problem through Reinforcement Learning (interested readers can find an introduction to Reinforcement Learning written by Sutton and Barto ( )).

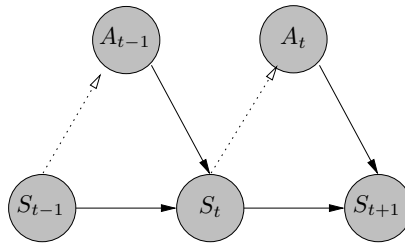


### 3.2. Reinforcement Learning

#### 3.2.1. Markov Decision Processes

We first consider the *ideal* theoretic framework for Reinforcement Learning, that is to say Markov Decision Processes ( , ).

Let  $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, r \rangle$  be a Markov Decision Process (MDP) where  $\mathcal{S}$  is a finite set of **states** and  $\mathcal{A}$  a finite set of **actions**. This process is Markov since state transitions are ruled by the **transition function**  $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$  with  $\mathcal{T}(s, a, s') = Pr(S_{t+1} = s' | A_t = a, S_t = s)$  (Figure 1 presents MDPs as a graphical model).  $r$  is a mapping from  $\mathcal{S} \times \mathcal{A}$  to  $\mathbb{R}$  that defines the **reward** gained by the system after each state transition. An action **policy**  $\pi$  is a mapping from states to distributions over actions ( $\pi : \mathcal{S} \rightarrow \Pi(\mathcal{A})$ , with  $\Pi(\mathcal{A})$  a probability distribution over actions). The problem is then to find a policy optimising a performance measure based on the reward, named **utility** and denoted  $V$ . Typically, the utility can be the sum of rewards to a finite horizon, the weighted sum of this reward on an infinite horizon ( $\sum_{t=0}^{\infty} \gamma^t r_t$  where  $\gamma \in [0, 1)$ ), or the average reward gained during a transition.

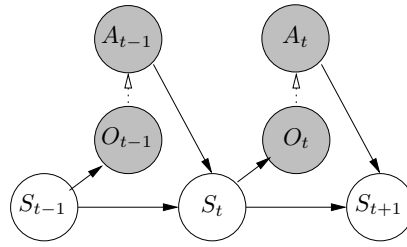


**Figure 1.** Graphical Model of an MDP. Dotted arrows indicate that an action is chosen depending on a state. Everything here is observable (indicated by the grey background), but rewards are not represented (to keep the figure simple).

In this paper, only *model free* RL ( , ) is considered : agents look for optimal policies with no knowledge of the system's model ( $\mathcal{T}$  and  $r$ ). If a model could be computed (when the dynamics of the system are known), multi-agent problems usually lead to huge state spaces (size exponential in the number of agents). The general principle of model-free RL algorithms is to have the learning agent evolve in its environment and gather results of experiments (in state  $s_t$ , action  $a_t$  may lead to state  $s_{t+1}$  with reward  $r_t$ ). Through stochastic approximation methods, the agent can then directly learn (using  $Q$ -learning ( ) or Sarsa ( ) for example) a value function on each of its states and deduce an optimal deterministic policy (whereas general policies may be stochastic), the Markov property being a guarantee that there is no local optimum in this process and that one deterministic optimal policy exists.

### 3.2.2. Partially Observable MDPs

In our framework, agents only deal with partial observations. Therefore, they do not have access to a complete state of the system and are rather facing a Partially Observable Markov Decision Process (POMDP). Such a POMDP is defined by adding to the  $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, r \rangle$  tuple a finite set  $\Omega$  of possible observations, and an observation function  $\mathcal{O}$  linking states to the observations they may cause :  $\mathcal{O}(s, o) = Pr(O_t = o | S_t = s)$  (see Figure 2). Besides, in our model-free problem, an agent has no knowledge of the underlying MDP ( $\mathcal{S}, \mathcal{T}$  and therefore  $\mathcal{O}$ ).



**Figure 2.** Graphical Model of a POMDP (see also Figure 1). In the case of a POMDP, states are hidden (represented by a white background). Note : the choice of action depends on the current observation only.

What an agent experiences is then only linked to observations instead of states. As a result, the process  $(O_t)$  obtained is not necessarily Markov anymore ( $Pr(O_t | O_{t-1}) \neq Pr(O_t | O_{t-1}, O_{t-2}, \dots)$ ), and there may not be a deterministic policy among the optimal solutions. This requires looking for stochastic policies of the form :  $\pi : \Omega \rightarrow \Pi(\mathcal{A})$ . The dynamic programming algorithms we mentioned for MDPs are not suitable in this new framework (since the Markov property is not satisfied), but may be replaced by policy search algorithms as –for example– the online policy-gradient we have used (based on Baxter et al. (, )). Early experiments have also been performed with  $Q$ -learning, showing that looking for a stochastic policy leads to better results, and turning a deterministic policy in a stochastic one using a soft-max is no satisfying solution.

This choice of a policy-gradient is not so usual, as it often appears to be sufficient to use dynamic programming algorithms, which are an easy pick in many cases. Yet policy-gradients are not only more appropriate in theory (optimising in the space of stochastic policies), but appear in recent works as very promising in problems involving function approximation techniques (, ) and in probabilistic planning ( ). In both cases, policy-gradients make it possible to optimise a policy in a large state-action space by using either function approximation methods or a carefully chosen controller.

Learning is all the more difficult under these new conditions, as there may now be local optima into which an agent can permanently fall. Interested readers may also refer to papers by Littman et al. ( ) and Singh et al. ( ) to learn about the use of

MDP algorithms in partially observable settings, and by Jaakkola et al. ( ) for another algorithm that could be used instead of the mentioned online policy-gradient.

### 3.3. *RL and MAS*

As pointed out by Boutilier ( ), the evolution of a Multi-Agent System can be modelled as an MDP if considered from an external point of view : depending on the *global* system state, there is a *joint action* to choose (the set of all agents' individual actions). Indeed, most theoretical works on Reinforcement Learning in MAS are grounded on this formalism, as can be observed from thorough reviews by Stone and Veloso ( ) and Shoham et al. ( ) (the work by Gmytrasiewicz and Doshi ( ) gives a counter-example).

Yet, such an approach appears not to be reasonable for practical applications. The reason for this is twofold :

- The global state space will grow exponentially and will shortly be unmanageable (this is already often the case in simpler mono-agent settings).
- Realistic perceptions are generally limited to a local view of an agent's environment.

Our motivation to deal with these two limitations is the main argument for working in a decentralised and partially observable framework.

Unfortunately, as shown by Bernstein et al. ( ), solving the problem in a non-centralised way when the agents only have a partial perception of the system's state is NEXP-complete, i.e. there is provably no polynomial algorithm to solve the problem. The additional complexity of our problem –compared to the classical MDP setting– is due to three major difficulties being faced :

1) **Partial observability.** As already mentioned, each agent's perception is local, which makes them uninformed of the global state of the problem. As such, the problem at hand belongs to the class of *partially observed* Markov decision models (see Section 3.2.2).

2) **Non-stationary transitions.** Reactive agents with a local view of the environment cannot use joint actions to solve the problem. In fact, other agents are hardly predictable elements of the environment : as agents learn their policies simultaneously (in our work), each agent lives in an environment with non-stationary transitions (other agents are part of the agent's environment).

3) **Multi-Agent Credit Assignment**<sup>3</sup>. In a Multi-Agent problem, an important difficulty is to reward agents appropriately. Intuitively, an agent whose decisions have not contributed to a recent success of the group should not get any reinforcement signal. Yet, if an MDP point of view is adopted -all agents getting identical rewards-

3. Not to be confused with the *temporal* (mono-agent) credit assignment problem : which past decision should be reinforced because of the present reward ?

an agent may “understand” which decisions are really useful as only some of them regularly lead to successes.

In our framework, reinforcement signals are internal to the agent, depending on its immediate observations. Thus, we face the problem of defining individual reward functions that correctly represent a common group objective (a question raised in more details in the COLlective INTelligence approach ( , )).

Note that in practice, e.g. in traffic control, local rewards make learning often easier, even if they may not lead to an optimal controller.

Classical stationary Partially Observed Markov Decision Processes are nearly impossible to solve when there are more than a hundred states ( ). The combination of the first two problems (i.e non-stationarity and partial observations) makes the problem non-solvable without using approximations. The multi-agent credit assignment issue is a rather independent question compared to non-stationarity and partial observations. We simply assume in this paper that the reward functions employed appropriately leads to a cooperative system.

#### 4. Shaping : incremental reinforcement learning

As there are no exact algorithms looking for optimal behaviours for the class of agents we consider, we will use approximation methods. Even if the learning conditions are worse than those of a POMDP (see Section 3.3), we have decided to make use of an algorithm suited to this partially observable mono-agent situation. Each agent will use a gradient descent RL algorithm (or policy-gradient) ( , ) suitable for on-line learning in a model-free POMDP, though it is not designed for use in our multi-agent framework.

After a brief description of this policy-gradient algorithm, we will focus on the main point of this paper : the use of a shaping method to design Multi-Agent Systems. This presentation is done by first introducing the general idea behind shaping and then detailing how this approach is here adapted to a MAS.

##### 4.1. Local learning algorithm used

As stated earlier, *each* agent uses its own policy-gradient algorithm to learn its policy ( $Q$ -learning was tried in early experiments, but was found to be unstable, most likely as it was looking for deterministic policies in a non-Markovian framework). We use an on-line version of the algorithm (presented here in its usual *mono*-agent setting).

As proposed by Baxter et al. ( , ), a policy  $\pi$  depends on a set of parameters  $\Theta$ . This leads to an expected utility  $V(\pi_\Theta) = V(\Theta)$ . The policy-gradient leads to obtaining a locally optimal policy by finding  $\Theta^*$  that makes the gradient equal to zero :  $\nabla V(\Theta^*) = 0$ .

The set of parameters we choose is  $\Theta = \{\theta(o, a), o \in \Omega, a \in \mathcal{A}\}$ , with  $\theta(o, a)$  a real valued parameter. The policy is defined by the probabilities of taking action  $a$  in state  $o$  as :

$$\pi_{\Theta}(o, a) = \frac{e^{\theta(o, a)}}{\sum_{b \in \mathcal{A}} e^{\theta(o, b)}}$$

Although this algorithm is theoretically not suited to MAS, cooperating agents prove to simultaneously evolve toward coordinated behaviours. Some readers may also be interested in the fact that such an algorithm may be used to learn controllers for several cooperating agents in a centralised way : to that end, each agent  $A_j$  has to be described by a subset  $\Theta_j$  of parameters linking its observations to its actions ( , , ).

#### 4.2. Incremental RL : Shaping

To speed up learning and improve the efficiency of resulting behaviours (mainly by avoiding being stuck in low local optima), we propose applying a progressive learning method. This approach, inspired by works in psychology ( , ) (Randløv and Alstrøm ( ) give an historical introduction), has been employed successfully in mono-agent RL, but only in completely observable MDPs ( , ).

The main idea is to ease the learning task by turning the problem the agent is facing into a simpler problem, and then progressively coming back to the original one. In fact, as far as Markov Decision Processes are concerned, a problem may be modified in various ways, as :

- The reward function may be redefined ( , , ). This is even done intuitively in some cases by rewarding the agent when it goes *toward* its goal while a reward when the goal is reached should be sufficient. Note that this may be hazardous (agent going back and forth to accumulate reward), but there are some theoretical guarantees ( ).
- The physics of the system may be altered ( ). This corresponds to modifying the MDP's transition function. A simple example is that of adding training wheels to a bike, and then progressively raising them ( ).

Whereas very appealing, the idea of shaping is not so easily applied : it may be difficult to determine what is a “simpler” problem. As noticed in Laud's PhD thesis ( ), shaping is generally a way to bring prior knowledge to a learning agent. In a way, it often turns Reinforcement Learning into a slightly supervised learning.

In a third shaping approach (although its author does not mention shaping) an algorithm can successfully find “simpler” problems by itself. In this work by Asada ( ), the learning agent is helped by being first confronted by states “close” to the goal to achieve, and then being progressively put in more and more complex situations (far from this goal). The result is to help the spreading of rewards in the value function (or  $Q$ -values). In this particular approach, and if a model is known (transition and reward functions  $\mathcal{T}$  and  $r$ ), the definition of the training used with the agent may be

automated, since states close to rewarded transitions can be identified. This makes it effectively possible to determine what is a “simpler” problem.

This third type of shaping is not completely orthogonal to reward-based shaping. It relies indeed on a progress estimator (since it uses start states close to the goal), and using a progress estimator is an ideal way to shape a reward function. Yet they can lead to different behaviours : reward-based shaping won’t be efficient in a maze, since the distance to the goal is often a bad progress estimator, while Asada’s shaping acts more as a backward search as it starts from the goal. Transition-based shaping is often difficult to implement and to benefit from : in the mountain-car problem ( ), starting from a flat mountain and progressively raising it does not help, as there is a critical point where the policy has to switch from moving directly to the goal to moving back and forth to get momentum.

This comparison between these three shaping approaches led us to prefer Asada’s method. But different contexts would lead to different choices. A second, and stronger, argument is that it naturally extends to our shaping approach for Multi-agent Systems as described hereafter.

Note : the term “shaping” sometimes appears to refer to design methods mixing engineering and automatic design algorithms (as reinforcement learning and evolutionary algorithms) ( ). This is slightly different from the terminology used in our context.

### 4.3. *Shaping for Multi-Agent Systems*

To speed up learning and reduce the problems of complexity and credit assignment<sup>4</sup>, we propose a methodology for shaping in a multi-agent setting. A first step follows the same idea as in Asada’s work, and a second one is based on a growing MAS.

#### *Growing Task Complexity*

Incremental learning is possible along the complexity dimension of the problem. Agents are pitted against harder and harder tasks. First tasks are “near” (in term of number of actions) positive reinforcement positions, then further and further away. While the learning progresses, agents have more and more freedom of action and can explore their environment further ahead.

To be more precise, a *step* consists in all agents making one move simultaneously. Then, we define a *trial* as a sequence of  $n$  steps beginning in a given situation (close to the goal at the beginning). This *trial* must be repeated sufficiently ( $N$  times) to be useful. This succession of *trials* will be called an *experiment* for our agents. The trainer has to define a sequence of progressive *experiments* to help learning. Algorithm 1 gives

---

4. Mono-agent credit assignment problem mentioned in a previous footnote.

a more formal presentation of the process, which will also be usefully illustrated in our experiments (Section 5.2).

---

**Algorithm 1** Complexity shaping
 

---

**Require:** A system  $\Sigma$  consisting of a set  $\mathcal{A}$  of agents and an environment.

A training  $T$ .

```

1: for all experiment  $E$  of the training  $T$  do
2:    $e =$  trial defined in  $E$  by :
3:     *  $s_0$  initial state of the system
4:     *  $n$  number of steps to accomplish from  $s_0$ 
5:    $N =$  number of times  $e$  has to be repeated
6:   for all trial  $i \in [1..N]$  do
7:     Put system in state  $s_0$ 
8:     for all step  $j \in [1..n]$  do
9:       for all agent  $\alpha \in \mathcal{A}$  do
10:        Give  $\alpha$  its perceptions  $o_t^\alpha$ 
11:        Let  $\alpha$  learn from its last experience
12:        Ask  $\alpha$  for its decision  $a_t^\alpha$ 
13:       end for
14:       Simulate system's evolution :  $\Sigma_t \rightarrow \Sigma_{t+1}$ 
15:        $t \leftarrow t + 1$ 
16:     end for
17:   end for
18: end for
19: repeat
20:   [ lines 9 to 15 ]
21: until stopping criterion of the learning algorithm verified
Ensure: Policies of agents in  $\mathcal{A}$ .

```

---

### *Growing MAS*

This phase is more specific to the conception of Multi-Agent Systems. It consists of learning with a reduced number of agents, and then multiplying them. This is simply done by cloning the first agents obtained, and then letting them all learn again. But one should not forget that all agents always learn their own behaviours, clones may therefore evolve in different ways.

Simultaneously, we also add other objects in the environment, which –from an agent's point of view– will increase the complexity of learning in a similar fashion as adding other agents. Indeed, if the goal remains the same, more objects are interacting while the agent only perceives few of them. An important remark at this stage is that the agents' internal architecture must be adapted to a variable number of objects/agents in the environment. This problem has been answered in our experimentation with a rather simple method based on the notion of local perceptions (see the problem description in Section 5.1 for more details). This shows once again how important

locality is in Multi-Agent Systems, even if this restricts each agent’s knowledge about the world.

Note : the question of designing such a *scalable* agent (able to handle a variable number of perceptions and motivations) also led to other developments ( ) that fall out of the scope of the present paper (see discussion in Section 6).

## 5. Experimenting with shaping

An application of the shaping method presented in the previous section is given in the experiments described here. After a short description of the problem, we give the details of the experiments conducted, which separately analyse both aspects of the shaping used (growing complexity and growing MAS), and also study the influence of a MAS of variable size.

### 5.1. Problem description

The task chosen involves agents (either yellow or blue) in a grid world whose goal is to push light green cubes against dark green ones<sup>5</sup>. When two agents coordinate their movements to attain this goal –pushing *together* a pair of cubes– both cubes temporarily disappear to randomly reappear elsewhere on the grid. Simultaneously, agents responsible for this fusion receive a positive reward. The agent’s objective is then to merge pairs of cubes as often as possible, not just foraging.

#### *Agents’ description*

– **actions** : Agents only have four possible actions corresponding to moving North, East, South and West (they always try to move). Agents can push other agents and other cubes, which makes the consequences of their actions stochastic. Indeed, when several agents are influencing a cube’s movement (for example), only one –randomly chosen– influence will have a real effect.

– **perceptions** : As shown in Figure 3, an agent’s perceptions are made up of the following information :

- `dir(oa)` : direction of nearest agent of the opposite colour (N-E-S-W),

- `dir(cl)/dir(cd)` : direction of nearest light cube (and dark cube) (N-NE-E-SE-S-SW-W-NW),

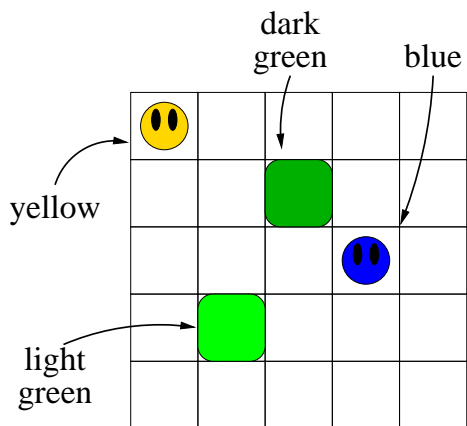
- `near(cl)/near(cd)` : is there a light cube (dark cube) in one of the eight nearest positions (true|false).

Combining these, there exists a maximum of 1024 observations (some combinations of perceptions are not possible). We reduce this number to 256 by using symmetries

5. From an agent’s point of view, light and dark green cubes are interchangeable.



of the problem. This number is small compared to the 15 249 024 states of the  $8 \times 8$  totally observed centralised problem, and also it is *independent of the world's size*.



agent	dir(cl)	dir(cd)	dir(oa)	near(cl)	near(cd)
yellow	S	E	SE	no	no
blue	W	NW	NW	no	yes

*cl : light cube - cd : dark cube - oa : other agent*

**Figure 3.** *perceptions' examples (two agents in a simple world)*

– **reward** : The reward function we have chosen eases the credit-assignment problem. When an agent takes part in the fusion of two cubes, it gets a reward (+5), while the reward is zero the rest of the time. Indeed, an agent knows when it has taken part in such a fusion as this corresponds to the event of seeing a pair of aligned blocks disappearing when the agent moves in their direction. A “global” reward would not be very consistent with agents having only local information.

There is no guarantee that this local reward function leads to a complete cooperation, as would be the case with a global one. Only experiments can easily confirm/infirm the choice we made.

We conclude by some remarks about the simulation itself. To focus on the learning problem, we have implemented simple mechanisms to avoid some unrelated problems. For example, cubes cannot go on border cells of the grid, neither by being pushed, nor when reappearing on the grid (this could otherwise lead to blockings, with cubes that could not be moved away from the border).

## 5.2. The 2-agent and 2-cube case

### 5.2.1. Reduced problem

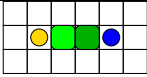
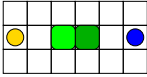
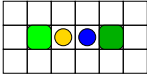
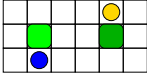
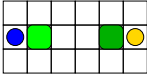

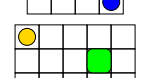
The first step in our incremental learning scheme is to use a small sized world with a minimal number of agents and objects : one agent and one cube of each colour. Then, beginning with positive reinforcement situations, the agents will face harder and harder problems.

### 5.2.2. Progressive task

For our problem, there is only one situation that leads to a positive reward. This situation is thus the starting point in the shaping process Agents face a sequence of tasks to learn before ending in a standard  $8 \times 8$  environment where they keep on learning.

Table 1 shows a sequence of *experiments* we used to help our agents in their *training*. The *starting configuration* of the first *trial*, on a  $6 \times 3$  world, needs only one move to reach the goal, each agent pushing toward the cubes. However, they have up to 6 *steps* (the duration of a *trial*) to reach it (so they can explore different moves), and they can try this 150 times (duration of this *experiment*).

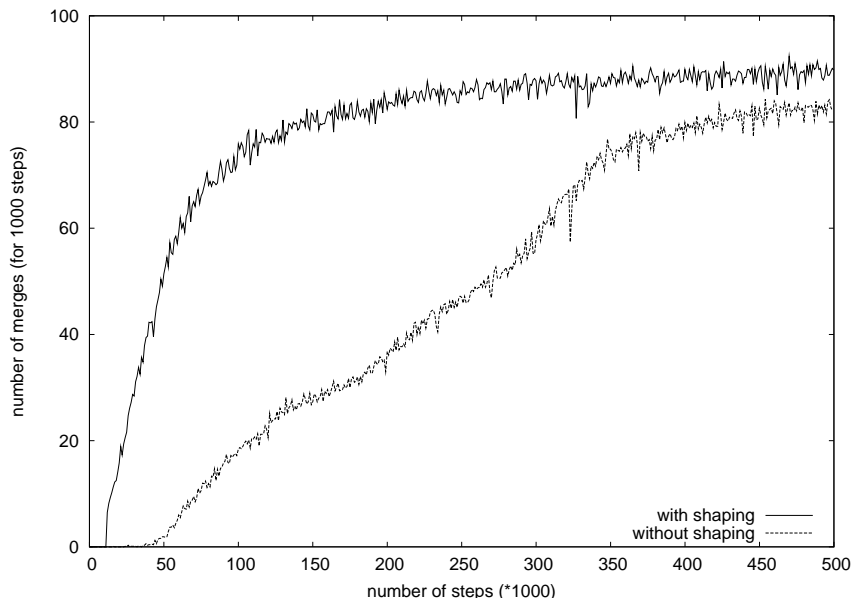
**Tableau 1.** *The sequence of experiments we used for incremental learning.*

Starting configuration	n (moves)	N (trials)
	6	150
	6	100
	10	150
	20	150
	20	150
	100	15
	100	15

In order to estimate the efficiency of our approach, a standard learning (from scratch) is compared to the incremental learning (shaping) proposed. By “from scratch”, we mean the use of the same policy-gradient algorithm, but always starting with a blank policy i.e., with no prior knowledge. In both cases, we count the number of cube merges accomplished during 1000 time steps to evaluate the quality of the agents’ policies, these agents being put in an  $8 \times 8$  environment once the training has ended (when there is one). We will now observe and analyse the results obtained.

### Results

Figure 4 shows the evolutions in the pair of agents’ efficiency with and without using shaping (average of 10 independent simulations). The curve representing the efficiency of learning after a period of assisted training only begins after the 12000 time steps of this training (Table 1).



**Figure 4.** 2 agents, 2 cubes : learning from scratch vs. shaping

In both cases, agents tend toward behaviours allowing about 90 merges each 1000 time steps (on average, two cubes are merged every 11 time steps). On the other hand, once the training time has elapsed, the convergence toward very good performance is achieved notably faster than through “standard” learning. If agents do not learn everything through the script that guides them during early times, at least do they benefit from it to efficiently learn their policies.

### *Designing the Sequence of Experiments*

The sequence of experiments should provide problems of growing complexity. Here, it is rather easy to design appropriate starting situations, since it suffices to move agents away from the blocks and blocks away from each other. Because there is a positive reward only in one state, an increasing distance to this state (in the state space) corresponds generally to an increasing complexity. We further discuss the problem of designing this sequence in Sec. 6.5.

### **5.3. More agents and more cubes**

We now consider the use of more agents and more cubes. As done previously, learning with shaping will be compared with standard learning (without shaping). But we will begin by considering the efficiency of the policies learned in the previous experiments (with 2 agents and 2 cubes, which will be denoted  $2a2c$ ) when they are reused in more populous worlds.

Note : Here, agents that do not start learning from scratch (either to simply reuse policies or to improve them) have their policies initialised with policies learned in  $2a2c$ .

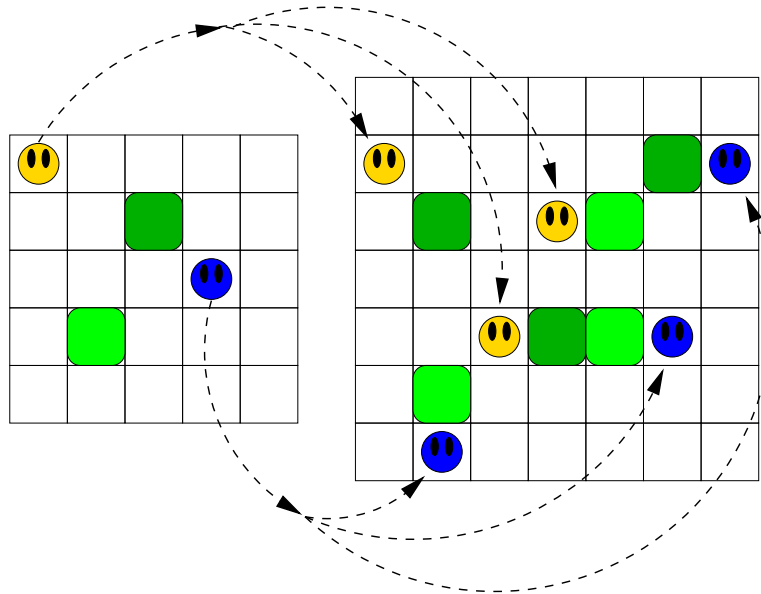
#### *5.3.1. Results : simple reuse of policies*

In this part of our work, the first interest was to check the efficiency of different numbers of agents having to deal with different numbers of cubes (taking the same number of agents (or cubes) of each colour). So, we first use agents whose fixed behaviours have been learned in the  $2a2c$  case (as illustrated on Figure 5), i.e. which are not supposed to be very efficient with more cubes. Nevertheless, this gives them enough good reactions to have some good results.

Several tests were carried out with 2, 4, 6, 8 or 10 cubes and 2, 4, 8, 16 or 20 agents (always in a world of size  $10 \times 10$ ). We used series of 1000 consecutive time steps, these series beginning in random configurations. And as blocking situations could sometimes occur, 100 such series were made in each case in order to compute an average efficiency.

Table 2-a gives the average efficiency in each of the 25 situations (the efficiency is the number of merges made in 1000 steps). These results can be compared with those from Table 2-b that measure the efficiency of agents having a completely random behaviour but placed in the same environments. It then clearly appears that behaviours learned by agents having to handle 2 cubes are still efficient in more complex environments. Through these results, a trend seems to take shape : except for 2 cubes, a growing number of agents improves the results up to the point where agents cramp each other and bring too many coordination problems.

For a given number of agents, and with a growing number of cubes, there seem to be a threshold beyond which additional cubes lead to a deterioration of the group's per-



**Figure 5.** Replicating behaviours from 2 to  $n$  agents. Policies are simply copied from the two original agents into their “clones”.

**Tableau 2.** Agents’ mean efficiencies (number of merges in 1000 steps)

a. Reuse of 2a2c agents						b. Random agents					
cubes	agents					cubes	agents				
↓	2	4	8	16	20	↓	2	4	8	16	20
2	40.4	30.0	20.0	12.7	11.0	2	0	0	0.1	0.3	0.4
4	7.6	17.1	17.5	13.9	12.9	4	0	0.1	0.2	1.2	1.8
6	3.4	11.2	14.7	15.7	16.5	6	0	0	0.5	1.9	3.6
8	1.9	8.6	13.5	15.9	18.0	8	0.1	0.2	1.0	4.1	6.0
10	1.6	6.7	11.0	17.7	20.6	10	0.1	0.3	1.1	6.1	7.3

formance. A more qualitative analysis of observed behaviours has shown that agents remain stuck in sequences of oscillatory movements. Because of their partial perceptions and of their lack of communications, agents seem to have difficulties working together on a same subset of cubes and constantly hesitate between two options that alternatively appear to be of interest.

5.3.2. Results : Incremental learning along the number of agents

Rather than just reusing behaviours learned for a 2 agents and 2 cubes situation in more cluttered environments, the second phase of our methodology –as defined in

Section 4.3– consists in using such behaviours as a starting point to adapt agents to new situations by continuing the reinforcement learning.

To assess this process, we compare the learning curve when the agents are initially novices (starting from scratch) to that when the agents are endowed with *2a2c* policies. Figures 6 a, b, c and d show the results in four distinct situations.

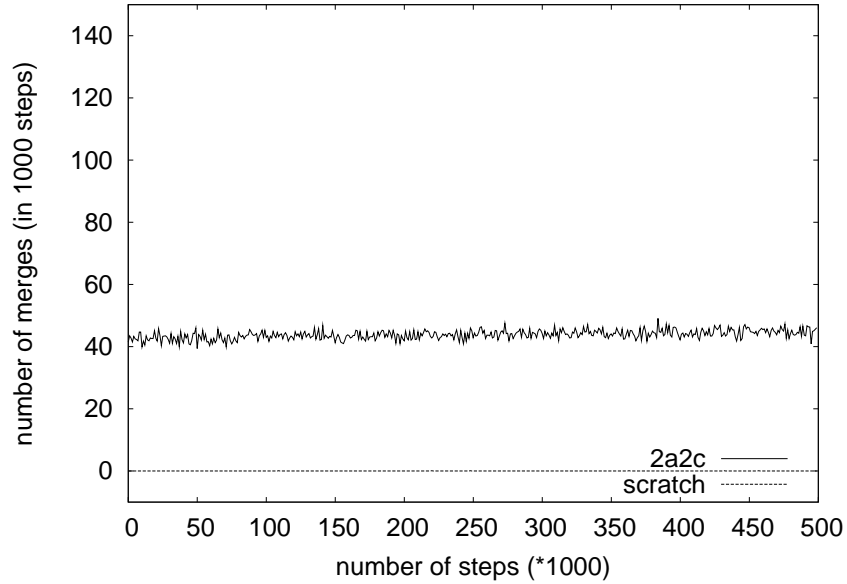
The learning for novice agents is difficult, especially in situations a and b because the size of their environment (larger than the environment used in Section 5.2) leads to scarcer non-zero rewards. In fact, in cases a and b, novice agents do not learn anything and their behaviours are the same as randomly moving agents. Indeed, in each case, novice agents evaluated at time  $t = 0$  are simply random agents having spent 1 000 time steps in the environment. With more agents, even novice agents can learn, as the frequency of rewarded events is higher. This is an interesting phenomenon since, otherwise, more agents makes learning harder. It can be related to difficulties encountered when learning to solve planning problems : a reward being provided only when a goal is reached, a bigger state space leads to infrequent rewards and therefore a harder learning task.

Coming back to the initial problem in this study, these experiments clearly confirm that agents having some experience from a problem of the same kind have a much faster progression than beginners. In case a, the efficiency level is even optimal from the beginning (which is not surprising as the conditions are nearly the same as in the initial learning problem). In addition to an improved learning rate, the maximum efficiency reached by the agents through shaping is by far better than performances obtained when learning from scratch. The reason for this is that agents in a noisy environment (noise here due to a large number of objects/agents) find it difficult to learn that they should sometime go on the other side of a pair of cubes, whereas this knowledge is present in *2a2c* behaviours and can then be reused.

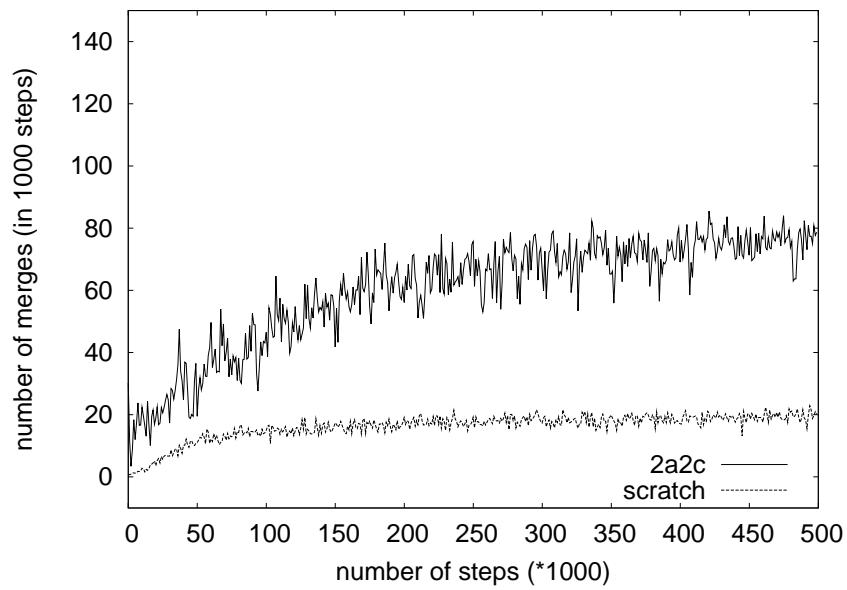
## 6. Discussion

### 6.1. *On explicit coordination*

Our experiments show that this work could benefit from explicitly addressing coordination problems. In particular, it is often the case that two agents, when placed in a world with more than two cubes, are not able to coordinate so as to push the same pair of cubes. Moreover, when too many agents are present, they easily work at cross-purposes. Making appropriate decisions would require taking into account many more agents and objects than each agent can through their limited perceptions. Instead, agents act often in an inconsistent manner (nearly randomly) because they are not “paying attention” to most aspects of their current situation but act greedily.

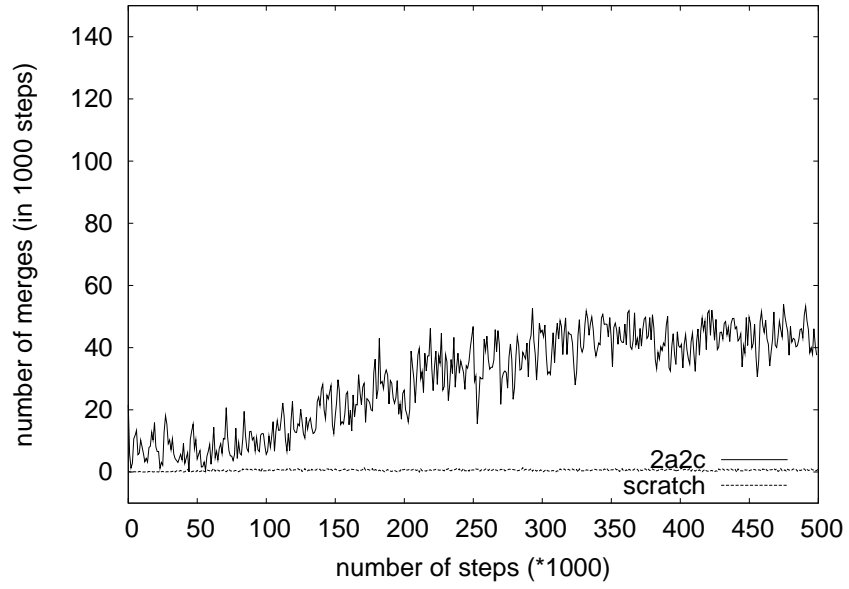


a- 2a2c

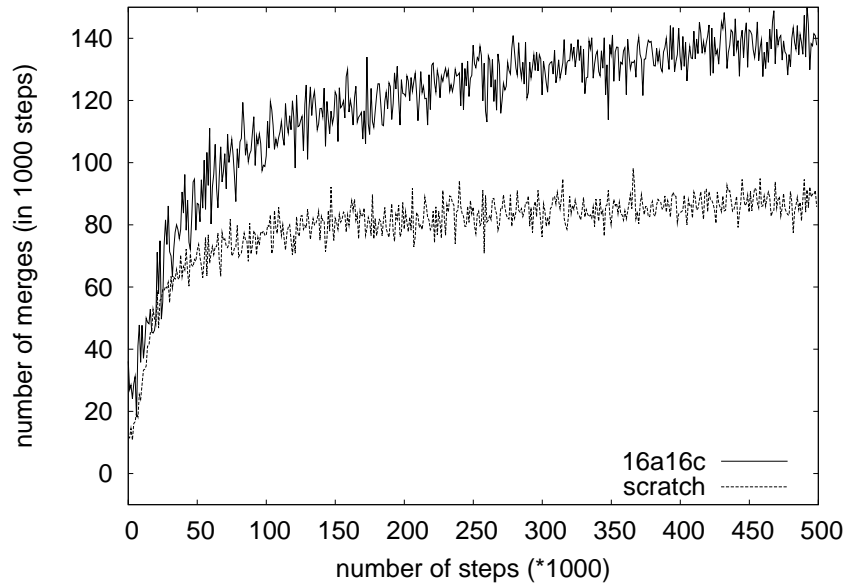


c- 8a8c

**Figure 6.** Comparison between standard learning (from scratch) and incremental learning (reusing behaviours learned in a 2a2c problem in a small grid) in various environments. During the learning process, the efficiency is measured by the number of merges done in 1 000 time steps. Each curve is the average of 100 simulations.



b- 4a4c



d- 16a16c

Figure 6. (con't)



### Other Agents' Modelling and Communication —

Considering both other agents' modelling and communication, it is important to notice how computationally expensive these approaches can be, as they often lead to huge growths of state-action spaces : in one case agents try to guess what others will be doing, and in the other case agents have to decide when and what to communicate (new actions) and have to incorporate received communications in augmented perceptions. Moreover, the work presented in this paper focused on reactive agents, whereas these approaches are quite "high-level" (imply more cognitive agents).

### 6.2. *Reward Definition*

#### Reward Shaping —

Modifying the reward function to help learning—as described in Section 2.2 and as encountered in Matarić's work ( )—is known as *reward shaping*. As a shaping process, it shares some similarities with the "progressive training" part of our approach, but Matarić's experiments can hardly be compared with ours as this real robotic problem involves rather high-level macro-actions : each macro-action is a complete behaviour, which would correspond to a sequence of actions in our case. Moreover, interactions between robots are not strong in the foraging task. There is no coordination problem to solve.

To come back to the other example of Stone and Veloso's work on simulated robotic soccer ( ), it could be interesting to try our "cloning" approach on robotic soccer (starting with a  $2 \times 2$  game for example), but a major difference that could make things more difficult is that the soccer-playing agents will probably end up having different roles, each being assigned a specific position.

### 6.3. *Scalable Architecture and Multiple Motivations*

An important aspect of our method is that it requires agents whose internal architecture is adapted to a variable number of objects or agents in their environment.

In our experiments, this constraint is satisfied by designing a perception process always selecting the same number of visible objects (the closest ones). This prevents from taking into account more than two objects and one agent simultaneously, and makes it impossible to have a different behaviour depending on which agent is considered. Moreover, as the visible agent may be different from one time step to another, it does not seem appropriate to make decisions based on a sequence of observations.

We have tried to address this issue by working on decision making when multiple motivations are faced. Because the Multi-Agent Systems considered are reactive and with local perceptions, each agent is often facing multiple parallel motivations, and has thus to make a choice in a "multi-criteria" problem. In our block-merging example, limiting the perceptions ensures that only one motivation is faced. But work on "ac-

tion selection” ( ) could be a way to address this particular problem. We conducted some work in a mono-agent framework, obtaining an agent able to find which basic behaviours it should use ( , ). Applying our method in a multi-agent framework still needs to be done.

#### 6.4. *Learning Algorithm*

For the reasons mentioned in Section 2.3, we have decided to use a reinforcement learning algorithm mapping the immediate observation to a probability distribution over actions. Some experiments with more classical algorithms ( $Q$ -learning) producing a deterministic policy have shown that this is not a viable solution in our toy problem : the ambiguity on current situation leads agents to looping behaviours (agents pushing their blocks and turning around them). Let us recall that both the partial observability and the multi-agent setting are reasons for deterministic policies to fail.

#### 6.5. *Automated Shaping*

Coming back to the shaping process proposed in this paper, it appears to be mainly constrained by the need to decompose the problem at hand into less and less complex tasks (Section 4.3). The problem is partially made easier in our example as the starting points of this decomposition are rare positive reward situations (the reward being null most of the time). Yet, it remains difficult because of the partial observations. Moreover, our shaping method relies on cloning agents, which is obviously limited to settings with relatively homogeneous Multi-Agent Systems (with a few types of agents). This assumption seems reasonable in many domains.

Also, considering multi-agent systems, our approach involving cloning agents raises issues regarding task/role allocation : is it easy for agent to specialise for a particular role with this type of shaping ? If the cloning process is creating copies of the same policy, further learning can lead to a differentiation. But task/role allocation does not necessarily imply having agents with heterogeneous policies. If agents with identical policies have different perceptions or internal states, they can choose to take on different roles. But this view is quite different from most research on task allocation ( ), where tasks are explicitly defined, appropriate behaviours are known for each task, and the problem is for the agents to decide who will perform which task.

### 7. Conclusion

In this paper, we have addressed the problem of automatically designing Multi-Agent Systems made of reactive and cooperating agents, each acting as an independent learner. We propose the use of Reinforcement Learning algorithms so that each agent individually adapts its local behaviour in order to achieve a global task. Up to now,

this learning problem has no theoretical solution because of its decentralised aspect and because of the partial perceptions of the learning agents.

To get around this difficulty, we have emphasised the use of an incremental learning (shaping) scheme where more and more agents are faced with harder and harder problems. This method is quite generic as it can be adapted to any task without adapting the agents to the particular problem at hand. Its applicability is mainly limited by the agents' architecture, i.e., how they handle perceptions. It is indeed required to be able to re-use a learned policy in a new setting with more objects. This is feasible with reactive agents using simple sensors (as in our experiments). It could be extended to more complex action-selection architectures based on a combination of simple behaviours ( ).

We have tested our approach on a simulated environment where agents have to coordinate in order to reach a shared goal : the fusion of different coloured cubes. The experiments support our framework as they show the efficiency of incremental learning. Incremental learning leads to better learning rates than unassisted learning from scratch. Furthermore, it is more efficient to learn a more complex task after an initial stage of incremental learning than learning directly this more complex task from scratch : this shaping helps overcoming local optima.

Still, there is room for improvement. We have discussed several ways to overcome these limitations, like using communication for addressing explicit coordination or modelling other agents to better adapt to their behaviours. Defining individual reward functions representing a common goal also remains a crucial issue. Yet, the main improvement regarding our shaping approach would be to automate the definition of training phases in our progressive learning (maybe by analysing perceptions as done in some recent works ( )).

Furthermore, this work was a good opportunity to use a direct (stochastic) policy search algorithm in cooperative agents with local perceptions. It also made it possible to observe some phenomena in such a multi-agent learning setting : the main one being that a large number of agents induces more interactions and thus a faster learning, but not necessarily with the best behaviour since the incremental process is able to produce better behaviours.

## 8. Bibliographie

- Asada, M., S. Noda, S. Tawaratsumida, and K. Hosodaal : 1996, 'Purposive Behavior Acquisition for a Real Robot by Vision-based Reinforcement Learning'. *Machine Learning* **23**(2-3), 279-303.
- Bartlett, P. and J. Baxter : 1999, 'Hebbian Synaptic Modifications in Spiking Neurons that Learn'. Technical report, Australian National University.
- Baxter, J. and P. Bartlett : 2001, 'Infinite-Horizon Policy-Gradient Estimation'. *Journal of Artificial Intelligence Research* **15**, 319-350.

- Baxter, J., P. Bartlett, and L. Weaver : 2001, ‘Experiments with Infinite-Horizon, Policy-Gradient Estimation’. *Journal of Artificial Intelligence Research* **15**, 351–381.
- Bernstein, D., R. Givan, N. Immerman, and S. Zilberstein : 2002, ‘The Complexity of Decentralized Control of Markov Decision Processes’. *Mathematics of Operations Research* **27**(4), 819–840.
- Bertsekas, D. and J. Tsitsiklis : 1996, *Neurodynamic Programming*. Athena Scientific.
- Boutilier, C. : 1996, ‘Planning, Learning and Coordination in Multiagent Decision Processes’. In : Y. Shoham (ed.) : *Proceedings of the 6th Conference on Theoretical Aspects of Rationality and Knowledge (TARK '96)*. pp. 195–210.
- Buffet, O. : 2003, ‘Une double approche modulaire de l’apprentissage par renforcement pour des agents intelligents adaptatifs’. Ph.D. thesis, Université Henri Poincaré, Nancy 1. Laboratoire Lorrain de recherche en informatique et ses applications (LORIA).
- Buffet, O. and D. Aberdeen : 2006, ‘The Factored Policy Gradient planner (IPC-06 Version)’. In : A. Gerevini, B. Bonet, and B. Givan (eds.) : *Proceedings of the Fifth International Planning Competition (IPC-5)*. pp. 69–71. [Winner, probabilistic track of the 5th International Planning Competition].
- Buffet, O., A. Dutech, and F. Charpillet : 2004, ‘Self-Growth of Basic Behaviors in an Action Selection Based Agent’. In : S. Schaal, A. Ijspeert, A. Billard, S. Vijayakumar, J. Hallam, and J.-A. Meyer (eds.) : *From Animals to Animats 8 : Proceedings of the Eighth International Conference on Simulation of Adaptive Behavior (SAB'04)*. pp. 223–232.
- Buffet, O., A. Dutech, and F. Charpillet : 2005, ‘Développement autonome des comportements de base d’un agent’. *Revue d’Intelligence Artificielle* **19**(4–5), 603–632.
- Carmel, D. and S. Markovitch : 1996, *Adaption And Learning In Multi-Agent Systems*, Vol. 1042 of *Lecture Notes in Artificial Intelligence*, Chapt. Opponent Modeling in Multi-agent Systems, pp. 40–52. Springer-Verlag.
- Cassandra, A. R. : 1998, ‘Exact and Approximate Algorithms for Partially Observable Markov Decision Processes’. Ph.D. thesis, Brown University, Department of Computer Science, Providence, RI.
- Drigo, M. and G. Di Caro : 1999, ‘Ant Colony Optimization : A New Meta-Heuristic’. In : P. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, and A. Zalzala (eds.) : *Proceedings of the Congress on Evolutionary Computation (CEC-99)*. pp. 1470–1477.
- Dutech, A. : 2000, ‘Solving POMDP using selected past-events’. In : W. Horn (ed.) : *Proceedings of the Fourteenth European Conference on Artificial Intelligence (ECAI'00)*. pp. 281–285.
- Fernández, F. and L. Parker : 2001, ‘Learning in Large Cooperative Multi-Robot Domains’. *International Journal of Robotics and Automation* **16**(4), 217–226.
- Gerkey, B. and M. Mataric : 2004, ‘A formal analysis and taxonomy of task allocation in multi-robot systems’. *International Journal of Robotics Research* **23**(9), 939–954.
- Gmytrasiewicz, P. and P. Doshi : 2004, ‘Interactive POMDPs : Properties and Preliminary Results’. In : *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'04)*.
- Goldman, C., M. Allen, and S. Zilberstein : 2004, ‘Decentralized Language Learning Through Acting’. In : *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'04)*.

- Hengst, B. : 2002, 'Discovering Hierarchy in Reinforcement Learning with HEXQ'. In : C. Sammut and A. G. Hoffmann (eds.) : *Proceedings of the Nineteenth International Conference on Machine Learning (ICML'02)*. pp. 243–250.
- Hu, J. and M. Wellman : 1998, 'Online learning about other agents in a dynamic multiagent system'. In : K. P. Sycara and M. Wooldridge (eds.) : *Proceedings of the 2nd International Conference on Autonomous Agents (Agents'98)*. pp. 239–246.
- Jaakkola, T., M. Jordan, and S. Singh : 1994, 'On the Convergence of Stochastic Iterative Dynamic Programming Algorithms'. *Neural Computation* **6**(6), 1186–1201.
- Jong, E. D. : 2000, 'Attractors in the Development of Communication'. In : J.-A. Meyer, A. Berthoz, D. Floreano, H. L. Roitblat, and S. W. Wilson (eds.) : *From Animals to Animats 6 : Proceedings of the 6th International Conference on Simulation of Adaptive Behavior (SAB-00)*.
- Kaelbling, L., M. Littman, and A. Moore : 1996, 'Reinforcement Learning : A Survey'. *Journal of Artificial Intelligence Research* **4**, 237–285.
- Laud, A. : 2004, 'Theory and Application of Reward Shaping in Reinforcement Learning'. Ph.D. thesis, University of Illinois at Urbana-Champaign.
- Littman, M., A. Cassandra, and L. Kaelbling : 1995, 'Learning policies for partially observable environments : scaling up'. In : A. Prieditis and S. J. Russell (eds.) : *Proceedings of the Twelfth International Conference on Machine Learning (ICML'95)*. pp. 362–370.
- Matarić, M. : 1997, 'Reinforcement Learning in the Multi-Robot Domain'. *Autonomous Robots* **4**(1), 73–83.
- McCallum, R. A. : 1995, 'Reinforcement Learning with Selective Perception and Hidden State'. Ph.D. thesis, University of Rochester.
- Ng, A., D. Harada, and S. Russell : 1999, 'Policy invariance under reward transformations : Theory and application to reward shaping'. In : I. Bratko and S. Dzeroski (eds.) : *Proceedings of the Sixteenth International Conference on Machine Learning (ICML'99)*. pp. 278–287.
- Peshkin, L., K. Kim, N. Meuleau, and L. Kaelbling : 2000, 'Learning to Cooperate via Policy Search'. In : C. Boutilier and M. Goldszmidt (eds.) : *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI'00)*. pp. 489–496.
- Peters, J., S. Vijayakumar, and S. Schaal : 2005, 'Natural Actor-Critic'. In : J. Gama, R. Camacho, P. Brazdil, A. Jorge, and L. Torgo (eds.) : *Proceedings of the Sixteenth European Conference on Machine Learning (ECML'05)*, Vol. 3720 of *Lecture Notes in Computer Science*.
- Puterman, M. L. : 1994, *Markov Decision Processes—Discrete Stochastic Dynamic Programming*. John Wiley and Sons, Inc., New York, USA.
- Pynadath, D. and M. Tambe : 2002, 'The Communicative Multiagent Team Decision Problem : Analyzing Teamwork Theories and Models'. *Journal of Artificial Intelligence Research* **16**, 389–423.
- Randløv, J. : 2000, 'Shaping in Reinforcement Learning by Changing the Physics of the Problem'. In : P. Langley (ed.) : *Proceedings of the Seventeenth International Conference on Machine Learning, (ICML'00)*. pp. 767–774.
- Randløv, J. and P. Alstrøm : 1998, 'Learning to Drive a Bicycle using Reinforcement Learning and Shaping'. In : J. W. Shavlik (ed.) : *Proceedings of the Fifteenth International Conference on Machine Learning, (ICML'98)*. pp. 463–471.

- Rogowski, C. : 2004, 'Model-based Opponent Modelling in Domains Beyond the Prisoner's Dilemma'. In : *Proceedings of Modeling Other Agents from Observations (MOO 2004), AAMAS'04 workshop*.
- Salustowicz, R., M. Wiering, and J. Schmidhuber : 1998, 'Learning Team Strategies : Soccer Case Studies'. *Machine Learning* **33**, 263–282.
- Shoham, Y., R. Powers, and T. Grenager : 2003, 'Multi-agent reinforcement learning : a critical survey'. Technical report, Stanford.
- Singh, S., T. Jaakkola, and M. Jordan : 1994, 'Learning without state estimation in Partially Observable Markovian Decision Processes'. In : W. W. Cohen and H. Hirsh (eds.) : *Proceedings of the Eleventh International Conference on Machine Learning (ICML'94)*.
- Skinner, B. : 1953, *Science and Human Behavior*. Collier-Macmillan, New-York.
- Staddon, J. : 1983, *Adaptative Behavior and Learning*. Cambridge University Press.
- Stone, P. and M. Veloso : 2000a, 'Layered Learning'. In : R. L. de Mántaras and E. Plaza (eds.) : *Proceedings of the Eleventh European Conference on Machine Learning (ECML'00)*, Vol. 1810 of *Lecture Notes in Computer Science*.
- Stone, P. and M. Veloso : 2000b, 'Multiagent systems : a survey from a machine learning perspective'. *Autonomous Robotics* **8**(3).
- Stone, P. and M. Veloso : 2000c, 'Team-Partitioned, Opaque-Transition Reinforcement Learning'. In : *Proceedings of the Third International Conference on Autonomous Agents (Agents'00)*.
- Sutton, R. and G. Barto : 1998, *Reinforcement Learning : an introduction*. Bradford Book, MIT Press, Cambridge, MA.
- Sutton, R., D. McAllester, S. Singh, and Y. Mansour : 1999, 'Policy Gradient Methods for Reinforcement Learning with Function Approximation'. In : S. A. Solla, T. K. Leen, and K.-R. Müller (eds.) : *Advances in Neural Information Processing Systems 11 (NIPS'99)*, Vol. 12. pp. 1057–1063.
- Tumer, K. and D. Wolpert : 2000, 'Collective Intelligence and Braess' Paradox'. In : *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI'00)*. pp. 104–109.
- Tyrrell, T. : 1993, 'Computational Mechanisms for Action Selection'. Ph.D. thesis, University of Edinburgh.
- Vidal, J. and E. Durfee : 1997, 'Agent learning about agents : a framework and analysis'. In : S. Sen (ed.) : *Collected papers from the AAAI-97 workshop on multiagent learning*. pp. 71–76.
- Watkins, C. : 1989, 'Learning from delayed rewards'. Ph.D. thesis, King's College of Cambridge, UK.
- Wolpert, D. and K. Tumer : 1999, 'An introduction to collective intelligence'. Technical Report NASA-ARC-IC-99-63, NASA AMES Research Center.
- Wolpert, D., K. Wheeler, and K. Tumer : 1999, 'General principles of learning-based multi-agent systems'. In : *Proceedings of the 3rd International Conference on Autonomous Agents (Agents'99)*. pp. 77–83.
- Xuan, P., V. Lesser, and S. Zilberstein : 2000, 'Communication in Multi-Agent Markov Decision Processes'. In : S. Parsons and P. Gmytrasiewicz (eds.) : *Proceedings of ICMAS Workshop on Game Theoretic and Decision Theoretic Agents*.