Multi-Agent Systems by Incremental Gradient Reinforcement Learning

Alain Dutech **Olivier Buffet** **François Charpillet**
{buffet,dutech,charp}@loria.fr
LORIA, BP 239
54506 Vandoeuvre-lès-Nancy
France

Abstract

A new reinforcement learning (RL) methodology is proposed to design 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. We use simple reactive agents which learn their own behavior in a decentralized way. To cope with the difficulties inherent to RL used in that framework, we have developed an incremental learning algorithm where agents face more and more complex tasks. We illustrate this general framework on a computer experiment where agents *have* to coordinate to reach a global goal.

1 Introduction

Multi-Agent Systems (MAS) - systems where autonomous entities called agents interact with each other - are a growing interest in the artificial intelligence community, both in terms of research and applications (see [Ferber, 1999]). Whereas MAS are usually build "by hand" using simulations to tune up the system to reach a desired global behavior, this paper deals with a key problem : the design of MAS in an automated fashion *through learning*.

Learning in MAS can indeed take many forms [Stone and Veloso, 2000]. We opted for Reinforcement Learning (RL) methods [Sutton and Barto, 1998] as they do not require a teacher knowing before hand a solution of the problem. An evaluation of the current agents' actions, using a scalar value for example, is enough to learn.

Furthermore, we consider Multi-Agent Systems composed of simple reactive situated agents with local perceptions. Thus, the system is easier to build and can nevertheless solve complex problems as, in this **decentralized** framework, each agent faces simpler tasks.

As RL suffers from combinatorial explosion, decentralization of the learning process should bring the same benefits, i.e. each agent learns its own behavior by itself which simplifies the task to be learned. Besides, it is consistent with the localized aspect of realistic situated agents. But, in this context, two major difficulties burden RL :

- **hidden global state.** Usual situated agents can only rely on an imperfect, local and partial perception of their world. Then, the global state of the system stays unknown, which prevents classical RL algorithms from finding an optimal policy, as shown by [Singh *et al.*, 1994].
- **credit assignment problem.** When a positive reward is given by the environment, it is not always evident to credit positively the "good" actions that led to this reward. With many agents, this problem is even more crucial as we must also decide which agents to reward.

Our answer to these problems is a decentralized **incremental** learning algorithm based on a classical RL technique to find **stochastic** policies. Actually, agents with stochastic behaviors are more adapted to the problem of partial perception. By *incremental*, we mean that agents are progressively pitted against harder and harder tasks so as to progressively learn a complex behavior. Another aspect of the incremental learning is to begin learning with very few agents, so as to minimize coordination and cross-work actions, and then export these basic behaviors to tasks with more and more agents. Eventually, behaviors can be further refined by learning in these more demanding environments. Thus, the originality of our approach is twofold : learning is decentralized and incremental.

In this paper, we present our method for learning in a multi-agent system and an experiment on which we tested our ideas. Section 2 gives the details of our framework, then, Sections 3, 4 and 5 describe the experiments conducted to test the viability of our approach. A discussion about our work follows in Section 6, highlighted by some comparisons from other similar works. Future directions and conclusive remarks end the paper in Section 7.

2 Our Framework

In this part, we first present the kind of agents we consider. Then we show the problems one faces when using reinforcement learning in a multi-agent setting. Lastly, we explain how incremental learning brings a partial solution to these problems.

2.1 The agents

We are interested in designing Multi-Agent Systems (MAS) by having each individual agent learn its behavior. As written 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 characterized as :

- **situated with local perception** : Local perceptions are more thoroughly discussed in Section 2.4.
- **possibly heterogeneous** : through the learning process where agents learn individually, each agent can acquire a different behavior from the others.
- **cooperative** : all agents share the same goal and they *will* have to coordinate to reach it.

We say nothing about other characteristics (for example communications) as they are not important for our framework and could be easily incorporated into the agents.

2.2 Limitations of classical RL

Reinforcement Learning (RL) methods are very appealing ways of learning optimal memoryless behaviors for agents as they only require a scalar feedback from the system to the agents for them to learn. Besides, these techniques can be used when there is uncertainty in the world's evolution.

But the convergence of RL algorithms (like *Q-Learning* or *TD(λ)*) has only been proven for Markov Decision Processes (MDP). A MDP is defined as a $< \mathcal{S}, \mathcal{A}, T, r >$ tuple, $\mathcal{S}$ being a finite set of states and $\mathcal{A}$ a finite set of actions. When the system is in given state s, an action a being chosen, the probability for the system to end in state s' is given by $T(s, a, s')$. After each transition, the environment generates a reward given by $r(s, a)$. The problem is then to find the optimal mapping $\pi(s, a)$ between states and actions so as to maximize the reward received over time, usually expressed as a utility function $V(s) = \sum_{t=0}^{\infty} \gamma^t E(r_t | s_0 = s)$. Such a mapping is called a policy.

As pointed out by [Boutilier, 1996] the evolution of the kind of MAS we are interested in is a MDP. As such, it could be solved using classical reinforcement learning algorithms where the state of the system is the composition of the states of all agents and an action is a joint action composed of all individual actions of the agents. Thus, the number of states and actions in a centralized view of the problem should quickly prove to be too big for RL to be applied, as illustrated in Section 3.1. Besides, solving our problem this way would mean to solve it in a centralized way whereas we aim at a non-centralized solution as each agent should learn by itself.

Unfortunately, as shown by [Bernstein *et al.*, 2000], solving the problem in a non-centralized 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. We face two major difficulties :

1. **Non-stationary transitions.** Reactive agents with a local view of the environment can not use joint actions to solve the problem. In fact, other agents are unpredictable elements of the environment. As a consequence, the transitions from one state of the system to another, as seen by an agent, are non-stationary : for example the probability for an agent to move ahead depends greatly on the actions of other adjacent agents.
2. **Partial observability.** As the agent's perception is local they can not know the global state of the problem. As such, the problem at hand belongs to the class of *partially observed* Markov decision models (see [Littman *et al.*, 1995]).

Classical stationary Partially Observed Markov Decision Processes are nearly impossible to solve when there are more than a hundred states [Dutech, 2000]. The combination of the two problems (i.e non-stationarity and partial observation) makes the problem non-solvable without using approximations.

2.3 Incremental gradient reinforcement learning

We propose to find approximate solutions by using incremental gradient RL. The main idea of this methodology is to progressively scale up with the complexity of the problem. Agents run their own local version of RL, here a gradient descent described in [Baxter and Bartlett, 1999].

Stochastic behaviors

In the context of partially observed MDP, stochastic behaviors perform better than deterministic policies (see [Singh *et al.*, 1994]). As previous experiments (see [Buffet *et al.*, 2001]) with Q-Learning were not conclusive to this regard, we chose to work with a gradient descent reinforcement learning algorithm especially suited for stochastic policies. This algorithm, designed by Baxter [Baxter and Bartlett, 1999] is detailed in Section 3.2.

Incremental learning

To speed up learning and reduce the problems of complexity and credit assignment, we propose a methodology for incremental learning. The most obvious way is to use :

- **growing number of agents** : learning starts with a small number of agents, each learning its own strategy. There must be enough agents to solve the problem. Then, more agents are added, with initial policies taken from the original agents and refined through learning if needed.

Incremental learning is also 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. So, we use :

- **progressive tasks** : learning begins with a very simple version of the task to be executed, or with the agent being heavily guided to solve the task. Then, as learning progresses, the task is made harder usually by giving more freedom of action to the agents

2.4 On local perception

We use situated agents with local perceptions to reduce the complexity of the problem as, very often, centralized problems are often too huge to be solved (see Section 3.1). This means that an agent does not know the global state of the world, which makes RL difficult to use.

However, local perceptions are a prerequisite for incremental learning, as behaviors of the agents can be based on the "nearby" world. Thus, they scale immediately with the number of agents, the dimension of the world or the complexity of the task.

3 Experimenting with incremental learning

An application of this general notion of incremental learning is given in the experiments described here. After a short description of the problem, we give the details of the incremental tasks we used : harder tasks first and then more agents.

3.1 Problem description

The task chosen involves agents (either yellow or blue) in a grid world whose goal is to push yellow cubes against blue ones[1]. When two agents coordinate their movements to attain this goal -pushing *together* a pair of cubes- both cubes temporarily disappear. Simultaneously, agents responsible for this fusion receive a positive reward. The goal is to merge as many cubes as possible in a given time.

Considering our agents' abilities, they simply 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 so the consequences of their actions are stochastic, depending on which constraints will be considered first.

Agents' perceptions, as shown on figure 3.1, are :

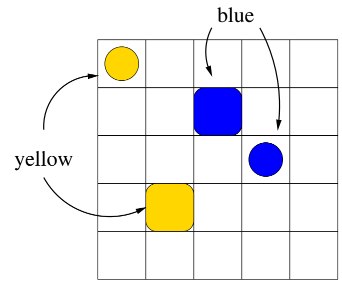
- `dir(oa)` : direction of nearest agent from opposite color `(N-E-S-W)`,
- `dir(cy)` : direction of nearest yellow cube `(N-NE-E-SE-S-SW-W-NW)`,
- `dir(cb)` : direction of nearest blue cube `(N-NE-E-SE-S-SW-W-NW)`,
- `near(cy)` : is there a yellow cube in one of the eight nearest positions `(true|false)`,
- `near(cb)` : is there a blue 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 of the problem. This number is very small compared to the 15.249.024 states of the 8×8 totally observed centralized problem, and also *independent of the world's size*.

The reward function we have chosen eases the credit-assignment problem. Only the two agents identified as having originated the fusion of two cubes get a reward $(+5)$, whereas in other cases the reward is zero for all agents. A "global" reward would not be very coherent with agents having only local informations.

We conclude by some remarks about the simulation itself. To focus on the learning problem, we have implemented simple mechanisms to avoid some non-significant problems. For example, cubes cannot go on border cells of the grid. Similarly, when cubes reappear on the grid, they can only be put on inner cells.

[1]Colors of agents and cubes are not related in this problem.



agent	dir(cy)	dir(cb)	dir(oa)	near(cy)	near(cb)
yellow	S	E	SE	no	no
blue	W	NW	NW	no	yes

cy : yellow cube - cb : blue cube - oa : other agent

Figure 1: perceptions' examples (two agents in a simple world)

3.2 Learning algorithm used

As said earlier, each agent uses its own gradient descent algorithm to learn its policy . We precisely use an on-line version of the algorithm.

As proposed by Baxter [Baxter and Bartlett, 1999], a policy π depends on a set of parameters Θ. This leads to an expected utility $V(\pi_\Theta) = V(\Theta)$. The gradient descent permits [2] to find a locally optimal policy by finding Θ^* that maximizes $V(\Theta)$.

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

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

4 The 2-agent and 2-cube case

4.1 Reduced problem

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

4.2 Progressive task

For our problem, there is only one situation which leads to a positive reward. This situation is thus the starting point of incremental learning. Agents face a sequence of tasks to learn before ending in a standard 8×8 environment where they keep learning.

To be more precise, we define a *try* as a sequence of n steps[3] beginning in a given situation. This *try* must be repeated sufficiently (N times) to be useful. This succession of *tries* will be called an *experiment* for our agents. The trainer has to define a sequence of progressive *experiments* to help learning. Following [Asada *et al.*, 1996], we designed a training sequence where starting situations are more and

[2]It is true if V only depends on this agent's policy, which is not the case in a MAS.

[3]In a *step*, all agents make one move simultaneously.

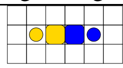
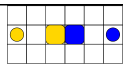
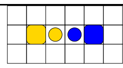
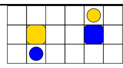
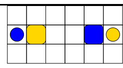
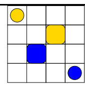
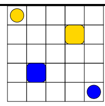
Starting configuration	n (moves)	N (tries)
	6	750
	6	500
	10	750
	20	750
	20	750
	100	75
	100	75

Table 1: The sequence of experiments we used for incremental learning.

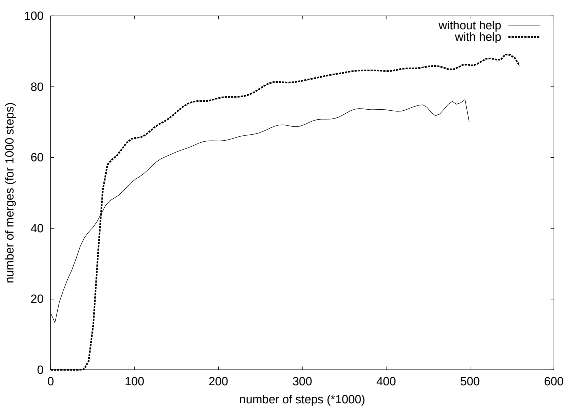


Figure 2: 2 agents, 2 cubes : incremental vs. raw learning

more "distant" from the goal, as Asada showed the efficiency of such a method.

Table 1 shows a sequence of experiments we used to help our agents in their training. The first *starting configuration*, on a 6×3 world, needs only one move to reach the goal, each agent pushing toward the cubes. However, they have up to 6 steps to reach it (so they can explore different moves), and they can try this 750 times.

Figure 2 shows the evolutions in the pair of agents' efficiency whether using or not progressive learning. The curve showing the efficiency of learning after a period of assisted training only begins after the 60000 steps of this training. Once this delay elapsed, notably better performances are reached with the assistance provided by our method.

cubes	agents				
↓	2	4	8	16	20
2	**40.4**	30.0	20.0	12.7	11.0
4	7.6	17.1	**17.5**	13.9	12.9
6	3.4	11.2	14.7	15.7	**16.5**
8	1.9	8.6	13.5	15.9	**18.0**
10	1.6	6.7	11.0	17.7	**20.6**

Table 2: Average efficiencies
Number of merges for 1000 steps

5 More agents and more cubes

5.1 The next step in incremental learning

Until now, only two agents were used. More could be necessary to reach our goal in a world with more cubes.

In this section, we first have a look at the efficiency of a policy learned with $2a2c$ and used with more agents and more cubes, and then we let agents' policies evolve in this more populated worlds.

Note : Here, we use agents that have either already learned a policy through the *2a2c* case with help or have never learned anything at all.

5.2 Influence of the number of agents and cubes

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 color). In this experiment, we use agents whose fixed behaviors have been learned in the $2a2c$ case, which is not supposed to be very efficient with more cubes. Nevertheless it 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 10×10). We used series of 1000 consecutive steps, these series beginning in random configurations. But as blocking situations could sometime occur, 100 such series were made in each case.

Table 2 gives the average efficiency in each of the twenty-five cases (the efficiency is the number of merges made in 1000 steps). It quickly shows that there seems to be an optimal number of agents for a given number of cubes as denoted by bold numbers in Table 2.

A growing number of agents improves the results, until they cramp each other and bring too many coordination problems. With a growing number of cubes, the results are improved only up to an optimal number beyond which agents hesitate on which cubes to work with and fall easily into oscillatory sequences of moves.

After another set of tries, in the same conditions but with policy-less agents, it appears that agents with a random behavior have always bad results, whatever their number.

5.3 Incremental learning along number of agents

With agents whose fixed behaviors have been learned in the 2 agents and 2 cubes case (*2a2c-policies*), we have seen that a growing number of cubes induces problems that can be solved

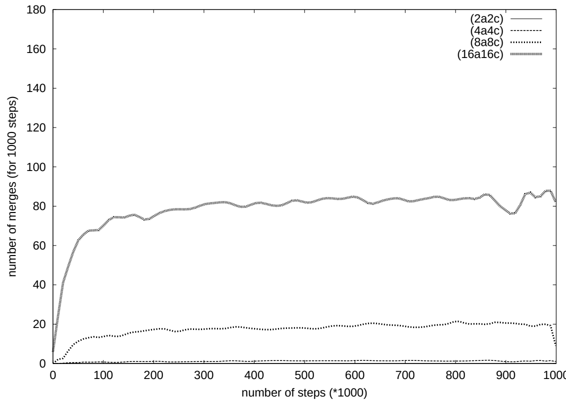


Figure 3: evolution while learning from scratch
(The $(4a4c)$ and $(2a2c)$ curves are practically confounded with abscissa axis.)

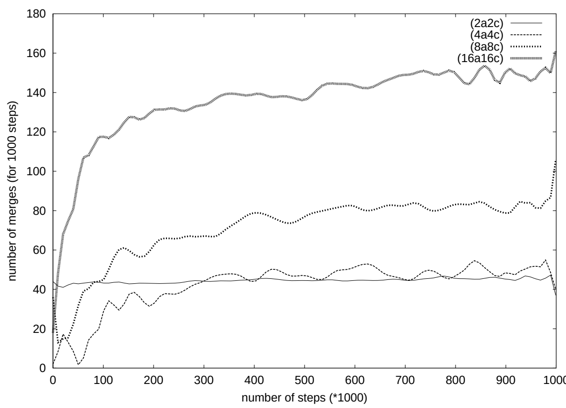


Figure 4: evolution while learning from *2a2c behaviors*

with a sufficient number of agents. The incremental learning solution is to improve the policies by keeping on learning with more agents.

Figures 3 and 4 show the efficiencies evolving with time in different learning cases : different number of objects, and agents learning from scratch or not. As shown on figure 3, agents learning from scratch have a slow evolution if there are only a few objects (agents and cubes) in the grid-world. But their learning speed rises with the rate $n_{objects}/size_{grid-world}$. When a group of agents seems to have reached a maximum efficiency, their results are better than the results obtained in Section 5.2 with the same situation but fixed policies. This confirms the interest of adaptation through learning.

One can notice that the optimal efficiency reached by agents using incremental learning is by far better than the performances reached using learning from scratch. As forecast, agents using *2a2c behaviors* come with some useful knowledge that allows them to find a better local optimum.

6 Discussion and similar works

6.1 On explicit coordination

Our experiments show that our work can benefit from addressing explicitly 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-purpose.

In [Boutilier, 1996], Boutilier has studied this problem from a theoretical point of view, using Multi-agent Markov Decision Processes. 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. Such coordination can be reached using social laws, communication or a learning mechanism. However, this centralized planning framework can not be used for decentralized RL. Other works, like [Hu and Wellman, 1998], more oriented towards reinforcement learning, settle on game-theory and Nash equilibria to solve coordination problems. Once again, these theoretical works are difficult to implement, especially with more than two agents.

Another field of research deals with explicitly taking into account other agents when learning, or planning, in a MAS (see [Carmel and Markovitch, 1996] or [Vidal and Durfee, 1997]). Usually, an agent builds a model of the other agents, so as to estimate their next actions. But, in the general case, it seems that not modeling the others is better than having a bad model and modelization leads to more complex agents.

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 [Xuan *et al.*, 2000]. Is it worthwhile to pay this cost whereas, simply putting more agents seems enough to increase the performances, even if the coordination problem of two particular agents is not really and explicitly solved ? This is one of the questions we would like to study in the near future. On the same subject, we would like to investigate the use of short term memory to provide our agent with attention mechanisms and help them to coordinate.

6.2 Reward

Matarić, in her work with multi-robots systems (see [Matarić, 1997]), took the option of adapting the reward function and the agents' actions and perceptions to use a simple reinforcement algorithm. The reward is not the kind of boolean process we have (no reward / big reward at goal) but rather a smoother reward adapted to give very frequent hints to each individual agent. By using behaviors, that can be seen as macro-actions, the size of state and action space is greatly reduced and learning is easier. On the other hand, this kind of specialization is very task-dependent whereas our agents could be made to learn another task without changing very much the agents or the learning process.

The COIN framework [Wolpert *et al.*, 1999] (COllective INtelligence) is similar to our framework : designing MAS through learning. The main ideas behind their work is to adapt reward functions and clusters of agents to build a *subworld-factored system*. Dealing with our problem , it would mean creating only one sub-group sharing the same reward, and other experiments we conducted show that learning becomes difficult and slow. Whereas this framework is useful for finding "independent" sub-groups of agents, our ideas seem better adapted for in-group learning.

6.3 Automatic incremental learning

As explained in 2.3, we need to decompose the problem at hand in more and more complex tasks. The starting points of this decomposition are positive reward situations. The decomposition process could be automatically performed online by the learning agents. In addition to classical methods, the agents could randomly explore their environment, mark "interesting situations" and then work around them. Knowledge thus learned could then be transfered to other agents.

7 Conclusion

In this paper we have addressed the problem of automatically designing large Multi-Agent Systems. Our framework is based on each individual agent using a Reinforcement Learning algorithm to adapt its behavior to the desired global task. This learning problem is generally unsolvable because of its decentralized aspect and more classical limitations like partial observability of state, credit assignment and combinatorial explosion. To that effect, we have emphasized the use of an incremental learning scheme where more and more agents are faced with harder and harder problems. This method is very generic as it can be adapted to any task without adapting the agents to the particular problem at hand.

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 colored cubes. The experiments give credit to our framework as they show the efficiency of incremental learning. Incremental learning leads to better learning rates than raw unassisted learning. 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. As a whole, our framework led us to conclude that "the more the better".

Still, there is room for improvement. We have discussed several ways to overcome these limitations, like using communication for addressing explicit coordination, short-term memory and intentions to deal with oscillatory behaviors and policy-search reinforcement learning as a possibly more adequate learning algorithm. We have also considered modeling the behavior of the other agents as a mean to predict their actions and thus improve the behavior of agents.

8 Acknowledgments

We are particularly thankful to Bruno Scherrer, Iadine Chadès and Vincent Chevrier for numerous discussions and their invaluable help in writing this paper.

References

[Asada *et al.*, 1996] M. Asada, S. Noda, S. Tawaratsumida, and K. Hosoda. Purposive behavior acquisition for a real robot by vision-based reinforcement learning. *Machine Learning*, 23:279–303, 1996.

[Baxter and Bartlett, 1999] J. Baxter and P. Bartlett. Direct gradient-based reinforcement learning: I. gradient estimation algorithms. Technical report, The Australian National University, Canberra, Australia, 1999.

[Bernstein *et al.*, 2000] D.S. Bernstein, S. Zilberstein, and N. Immerman. The complexity of decentralized control of markov decision processes. In *Proc. of the 16th Conf. on Uncertainty in Artificial Intelligence*, 2000.

[Boutilier, 1996] C. Boutilier. Planning, learning and coordination in multiagent decision processes. In *Proc. TARK'96, De Zeeuwse Stromen, The Netherlands*, 1996.

[Buffet *et al.*, 2001] O. Buffet, A. Dutech, and F. Charpillet. Incremental reinforcement learning for designing multiagent systems. In *Proc. of Agents'01*, 2001.

[Carmel and Markovitch, 1996] D. Carmel and S. Markovitch. Opponent modeling for learning in multi-agent systems. In *Proceedings of IJCAI'95 Workshop*. Springer, 1996.

[Dutech, 2000] A. Dutech. Solving pomdp using selected past-events. In *Proc. of the 14th European Conf. on Artificial Intelligence, ECAI2000*, 2000.

[Ferber, 1999] J. Ferber. *Multi-Agent Systems. An introduction to Distributed Artificial Intelligence.* John Wiley & Sons Inc., New York, 1999.

[Hu and Wellman, 1998] J. Hu and M. Wellman. Multiagent reinforcement learning: theoretical framework and an algorithm. In *Proc. of ICML-98*, 1998.

[Littman *et al.*, 1995] M. Littman, A. Cassandra, and L. Kaelbling. Learning policies for partially observable environments: scaling up. In *Proc. of ICML-95*, 1995.

[Matarić, 1997] M. Matarić. Reinforcement learning in the multi-robot domain. *Autonomous Robots*, 4(1):73–83, 1997.

[Singh *et al.*, 1994] S. Singh, T. Jaakkola, and M. Jordan. Learning without state estimation in partially observable markovian decision processes. In *Proc. of ICML-94*, 1994.

[Stone and Veloso, 2000] P. Stone and M. Veloso. Multiagent systems: a survey from a machine learning perspective. *Autonomous Robotics*, 8(3), 2000.

[Sutton and Barto, 1998] R. Sutton and G. Barto. *Reinforcement Learning*. Bradford Book, MIT Press, 1998.

[Vidal and Durfee, 1997] J. Vidal and E. Durfee. Agent learning about agents: a framework and analysis. In S. Sen, editor, *Collected papers from the AAAI-97 workshop on multiagent learning*. AAAI Press, 1997.

[Wolpert *et al.*, 1999] D. Wolpert, K. Wheeler, and K. Tumer. General principles of learning-based multiagent systems. In *Proc. of Agents'99, Seattle*, pages 77–83, 1999.

[Xuan *et al.*, 2000] P. Xuan, V. Lesser, and S. Zilberstein. Communication in multi-agent markov decision processes. In *Proc. of ICMAS Workshop on Game Theoretic and Decision Theoretic Agents*, 2000.