Solving Multiagent Markov Decision Processes: A Forest Management Example

[1]**Iadine Chadès and Bertrand Bouteiller**

[1]Institut National de Recherche Agronomique
Unité de Biométrie et Intelligence Artificielle iadine.chades@toulouse.inra.fr

Keywords: *Multiagent systems, Stochastic Dynamic Programming, Multiagent Reinforcement Learning*

EXTENDED ABSTRACT

In the Artificial Intelligence community, Markov Decision Processes (MDPs) and Reinforcement Learning (RL) are used to solve sequential decision making problems under uncertainty. However, when problems involve several agents, solving Multi-agent MDPs becomes untractable due to the high complexity level: exponential in the number of agents at least. Nevertheless, these problems may often be represented in a compact way and be decomposed into subproblems weakly coupled. In this paper, we propose to go further in the understanding of previously proposed multi-agent reinforcement learning algorithms. We have evaluated their fitness on a forest management problem.

The studied forest is formed by a finite number of stands. Each stand is composed of the same aged trees. The benefit provided by the exploitation and the cut of a stand depends on the quality of the trees, that is to say the age of the trees. Once a stand is cleared, new trees grow until the next harvest and so on. The clear decision has a fixed cost of achievement and also depends on how far is a stand from the previous cleared stands. The probability of losing several stands is represented by the risk of fire occurrences. The forest management yields important incomes that can be optimized: the aim is to find an optimal strategy (also called policy) that maximizes the benefits while taking into account uncertainty of fire occurrence. To solve this problem one must decide the best sequential harvest over a finite period of time.

This problem becomes very interesting when the management involves a large amount of stands that are spatially linked. Classically, two approaches are usually investigated to solve Markov decision problems, depending on the quantity and quality of knowledge available:

- Planning methods. These are used when all the components of the studied system are known and modeled.
- Learning methods. These are used when the dynamic of system is too complex to be modeled or when the dynamic of the system is partially unknown. However, a simulator can be used to learn optimal solutions using simulation techniques and reward functions.

In practice, even for small size problems, planning algorithms can fail to compute optimal policies because of a lack of memory space, whereas Reinforcement Learning (RL) techniques succeed. Nevertheless, despite the fact that convergence of RL algorithms toward optimal policies have been proved, when applied, RL algorithms are more able to give near optimal policies than optimal policies due to important time convergence. Solving very large optimal decision making problems under uncertainty remains an important challenge in Artificial Intelligence and more specifically using MDPs. In order to find near optimal solutions, Multiagent systems provide an interesting alternative to deal with the global complexity of large problems. In designing agents that perceive, decide, and interact locally, one can significantly reduce the global complexity of solving a global problem. We have previously studied this alternative using planning methods (Chadès, Scherrer & Charpillet 2002, Chadès 2003, Chadès 2004), but we are now interested in studying the Multiagent Reinforcement Learning alternative to solve large Markov decision problems.

As a preliminary work, we propose in this paper to focus on the heuristics proposed by Schneider, Wong, Moore & Riedmiller (1999). The multi-stand forest management problem chosen offers a way to evaluate these Multiagent Reinforcement Learning algorithms. For this particular problem, the results show that Multiagent Reinforcement Learning algorithms with spatial local rewards find better policies than Multiagent Reinforcement Learning algorithms with global rewards.

1. INTRODUCTION

In this paper, we propose to go further in the understanding of previously proposed multiagent reinforcement learning algorithms. We have evaluated their fitness on a forest management problem. This problem becomes very interesting when the management involves a large amount of stands that are spatially linked. Unfortunately, solving very large decision problems remains an important challenge in Artificial Intelligence and more specifically using MDPs due to the high complexity level. To deal with this complexity heuristic approaches have been developed:

- State aggregation methods. States are grouped in subsets sharing the same features, thus reducing the size of the MDP.
- Decomposition methods. The original problem is split into smaller subproblems that are solved independently. The elementary solutions are then combined in order to provide an approximately optimal solution to the global problem.
- Multiagent methods. The original problem is decomposed in subproblems that are solved using interaction between agents. Multiagent reinforcement learning combine RL and multi-agent methods.

The body of work in AI on multiagent RL is still small (see Shoham, Powers & Grenager (2003) for a critical survey), this is particularly the case when agents use local perceptions and local actions to achieve together a global task (Schneider et al. 1999, Dutech, Buffet & Charpillet 2001). In the field of Markov decision models, the Decentralized MDP (Dec-MDP) formalizes the local observations and the local decision making of a cooperative multiagent systems. However, it has been shown (in Bernstein, Zilberstein & Immerman 2000) that the decentralized MDP class problem is NEXP-complete for a number of agents greater or equal 2. Finding an exact solution for a Dec-MDP problem is therefore untractable. Heuristics for this theoretical problem have been proposed recently in the literature (Chadès et al. 2002, Nair, Tambe, Yokoo, Pynadath & Marsella 2003, Shen, Lesser & Carver 2003).

This paper follows the work done by Garcia & Sabbadin (2001) on the multi-stand forest management problem. We propose to go further into the use of Multi-agent Reinforcement Learning algorithms. To this end, we change the existing model to reduce the computer memory requirements and to take into account the local properties of the agents. Moreover, we add a spatial dependence between the agents to enhance coordination phases. The algorithms studied were proposed in Schneider et al. (1999) with a small number of agents (10 agents). Our results show that only some of these algorithms gave good solutions.

The paper is organized as follows. In the first section, we introduce the MDP model and the Q-Learning algorithm. Then, we present our application and show that exact method can't compute optimal policies with more than 10 stands and 6 classes of age. In section 4, we introduce multiagent reinforcement learning heuristic solutions, and our model based on the use of local spaces, and spatial reward function. In section 5, we report the results of our simulations. The assessments allow us to discuss future work in the last section.

2. MARKOV DECISION PROCESSES

Markov Decision Processes (MDP) are basic and generic models that enable us to formulate the problems we are interested in. An MDP is defined as a tuple $\langle S, D, T, R\rangle$ where:

- S is a finite set of states.
- D is a finite set of actions.
- $T : S \times D \times S \longmapsto [0, 1]$ is a transition function. $T(s, d, s') = P(s'|s, d)$ denotes the probability of moving from state s to state s' when action d is performed.
- $R : S \longmapsto \mathbb{R}$ is a reward function.

In an MDP, given a policy $\pi : S \longmapsto D$ and a starting state s_0 whose utility is $V^\pi(s_0)$, the expected long-term reward is characterized by the following linear system :

$$\forall s \in S, V^\pi(s) = R(s) + \gamma . \sum_{s' \in S} T(s, \pi(s), s')V^\pi(s') \quad (1)$$

V^π is the so-called value function. The problem is to find some policy that maximizes this expected long-term criterion. It is proved that there exists one optimal value function (Puterman 1994). Besides, deterministic optimal policies can easily be found using an appropriate algorithm, such as *Value Iteration* or *Policy Iteration* (Puterman 1994). The complexity of Value Iteration is in $O(|D||S|^2)$ precluding its use when state or action spaces are large. Moreover, it cannot be applied when the transition probabilities or rewards are unknown. The reinforcement learning algorithms are designed to overcome these difficulties by applying two

principles: unknown quantities are estimated by means of simulation; large state or action spaces are handled through function approximation.

The Q-learning algorithm (Watkins & Dayan 1992) is a standard reinforcement learning algorithm which can be used when probabilities of transitions are unknown and, with the joint use of a function approximator, when the state or action spaces are large. It consists of iteratively computing the optimal value function: for each state s at each instant t, the optimal value $Q^*(s,d)$ of each decision d is estimated on the basis of simulated transitions. When all these values have been correctly estimated, the optimal policy can be derived through:

$$\forall s \in S \pi^*(s) = \arg\max_{d \in D} Q^*(s,d)$$

To estimate these state/decision values, the algorithm performs Bellman's updates in the style of equation 1 but on the basis of a sample of simulated transitions instead of the actual probabilities and rewards (see Algorithm 1).

Algorithm 1 Q-Learning

```
Initialize(Q_0);
for n ← 0 to N_traj do
    s ← InitializeState;
    for t ← 1 to τ do do
        d ← SelectAction;
        (s', r) ← SimulateTransition(s, d);
        {Update of the simulated state/action pair value;}
        d ← r + γ max_b Q(s', b, t+1) − Q(s, d, t);
        Q(s, d, t) ← Q(s, d, t) + α(s, d, t)d;
        { α(s, d, t) ∈ [0, 1] is the learning rate;}
    end for
end for
```

If every action in each state at each instant is tried infinitely often and if the learning rate properly decreases, then $Q \rightarrow Q^*$ with probability 1. In practice, one needs some efficient control of the search to focus on the most relevant state/action pairs. This is the role of the SelectAction function which has to handle the classical trade-off between exploration and exploitation (Sutton & Barto 1998).

3. GLOBAL MODEL AND AN EXACT DYNAMIC PROGRAMMING SOLUTION METHOD

The problem that will illustrate the various approaches to solving "large" MDPs is a multi-stand forest management problem, in which we want to maximize the long-term revenue from timber sales. We model the problem in a MDP framework, following similar work from Garcia & Sabbadin (2001).

3.1. States of the system

The forest is formed by N homogeneous stands (same tree species), that can be of different size and shape. Each stand n is simply defined by an age group a_t^n at current time t, with $a_t^n \in \mathcal{A} = \{1, \ldots, A\}$. A is the last class[1]. Thus, the state of stand n is defined by a_t^n. The global state of the system at time period t is defined by a vector $s_t = (a_t^1, \ldots, a_t^N) \in S = \mathcal{A}^N$

3.2. Decisions

At time t, the system is in state s_t, we decide which stands will be clearcut within the next time period. The set of global decisions of the system is defined by the set of vectors $d_t = (d_t^1, \ldots, d_t^N)$. Where d_t^n is the decision in stand n. For each stand n, d_t^n can take two values "clearcut" or "do nothing". The cutting decision may not have the desired effect if a fire occurs within the $[t, t+1]$ period. Another decision to take is the fire protection expenditure level, $e_t \in \epsilon = \{1, \ldots, E\}$ applied globally for the forest for the current period. e_t may consist of funds allocated to the fire tower network or road maintenance. It shall be noticed that e_t is the only factor that links the dynamics of the different stands, and thus prevent us from considering them as independent.

3.3. State Transition Function

The state transition function is stochastic, due to the stochastic nature of the fire event, modeled by the probability table $P_{fire}(n, a_t^n, e_t)$ indexed by stand number, age, and fire protection level. Each stand evolves under the conjugate effect of two processes:

- Harvest. If $d_t^n = 0$ (clearcut), $a_{t+1}^n = 1$.
- Growth. If $d_t^n = 1$ the age of trees at the next time period depends on the fire event: if there is a fire (with probability $P_{fire}(n, a_t^n, e_t)$) $a_{t+1}^n = 1$ and if not, $a_{t+1}^n = \min(a_t^n + 1, A)$.

We then define a transition probability $P_{d_t^n, e_t}(a_t^n, a_{t+1}^n)$ for every stand. At the global level, due to the independence of the various stands, e_t being fixed:

$$P(s_{t+1}|s_t, e_t, d_t) = \prod_{n=1}^{N} P_{d_t^n, e_t}(a_t^n, a_{t+1}^n)$$

3.4. Rewards

The incomes are the result of the clearcut decisions and depend on the features of each stand (quality, quantity, and age group of the stand). To these incomes, we must remove the cost of the clearcut

[1] We consider that all trees older than A keep the same properties, and thus need not be distinguished

A \ N	4	6	10	15
3	3.77	5.23	8.92	28.63
4	3.89	5.47	10.05	/
6	3.93	5.77	/	/

Table 1. CPU time (in s.) needed to solve the global forest management: exact method.

achievements ($k'(d_t)$), and the cost of fire prevention ($k(e_t)$).

$$r_t = r(s_t, e_t, d_t, s_{t+1})$$

$$= -k(e_t) - k'(d_t) + \sum_{n=1}^{N} r_n(d_t^n, a_t^n, a_{t+1}^n, n)$$

with $r_n(d_t^n, a_t^n, a_{t+1}^n, n)$ price of the whole stand timber stock if $d_t^n = 1$, and of the salvaged timber if there is a fire. $r_n(d_t^n, a_t^n, a_{t+1}^n, n) = 0$ if we choose not to cut and there is no fire.

3.5. Exact Solution

This MDP can now be solved using the exact dynamic programming method. Note that the global MDP model of the forest has A^N states and $2^N \times E$ decisions. The size of the problem grows exponentially with N and A. Table 1 shows the time needed to solve the forest management problem using a global exact method. For $A = 4$ and $N = 15$ and $E = 2$ the memory size is too huge to compute an exact solution.

4. HEURISTIC SOLUTION METHODS

4.1. Previous Work on Multiagent RL

In a previous work (Garcia & Sabbadin 2001), the authors began to study Multiagent RL, used as a means of decomposing the initial problem. This method is based on a different representation of the problem. The idea underlying this is that each stand is managed independently by an agent, and the fire protection level is managed by a $N + 1^{th}$ agent. The motivation is to limit the size of the computer memory needed for storing the $N + 1$ Q-functions. This implies limiting the information available to each agent. Garcia & Sabbadin (2001) chose to use the following relevant factors for each stand agent: age a_t^n of trees in stand n; average age $\bar{a}_t$ of trees in forest; number of stands cut in the preceding period Cut_{t-1}; current fire prevention level e_{t-1}. Concerning the prevention agent, an aggregated representation of the ages of trees would be an important factor, as well as the current prevention level: age repartition $n_t^1, \ldots, n_t^A$; current prevention level e_{t-1}. The global size needed to store the Q-functions is in $O(E \times (A^2 \times N^2 + N^{A-1}))$. The authors used a Q-Learning algorithm using the aggregate states described above to solve this problem. This algorithm is very close to the original Q-learning algorithm. Each stand agent chooses an action, updating the global state, then the fire protection agent chooses a new protection level. Then all Q-functions are updated.

For $N = 13$ and $A = 6$, or $N = 100$ and $A = 3$ with $E = 3$, the space memory needed to compute the Q functions becomes too huge to process. Indeed, the Q-function of the prevention agent is $O(N^A \times E)$. However, the aggregation method and the Q-learning algorithm allow us to deal with more than one hundred stands. Garcia & Sabbadin (2001) choose a quality criterion to compare the quality of computed policies: on a small problem the Q-learning algorithm scored 0.96 (1 is optimal).

4.2. New spatial representation

This previous study has shown that Multi-agent Q-learning can be an interesting approach for solving forest management problems. We propose to go further into this study. First, we would like to better use the paradigm of MAS with more interactions between agents, and our second purpose is to deal with larger problems. As a means of reducing the memory space, the previous model erased the spatial dimension of the initial statement of the problem, and forbade dealing with spatial local interactions. We chose to improve this model by integrating a new interaction between the stands: if a fire occurs in a stand, the neighborhood of this stand is also affected by the fire for a certain distance. With a spatial model we can also improve the simulation of the harvest activities: we can assume that it costs less to clearcut stands that are closer than distant stands or isolated stands. To design this spatial representation, we compute for the system a new *neighborhood matrix*: for each stand n we store the 8 neighbors.

We propose studying the behavior of Q-learning algorithms using the following multi-agent model. N agents represent the N stands of the forest. Our spatial distributed formulation of the problem makes the following changes:

- State space. S represents the set of states of the global system (the ages of all the stands). $|S_n|$ represents the local state space of agent n. An agent observes its own age and the mean age of its 4 direct neighbors. $|S_n| = |A^2|$.
- Action space. D_n represents the local action space of agent n. An agent has the ability to choose a decision d_n: "clearcut" or "do nothing". $|D_n| = 2$.

- State transition function. The transition function remains stochastic and is slightly different from the previous model because of the spatial consequences of fire occurrence.

- Rewards. $R(s,d)$ represents the global reward of the system, and $R_n(s_n,d_n)$ the local one, with $R(s,d)=\sum_{n=1}^{N} R_n(s_n,d_n)$.

In the model describe above each stand had a probability to be burned, depending on its age, but the effects of the fire were limited to only one stand for each occurrence. Using the neighborhood matrix, we can now model the propagation of the fire to the neighborhood of the stand involved. Thus, we define the *dynamic of fire occurrence* for each time period as following: A number of fire occurrences is randomly chosen; For each fire occurrence a random process determines which stand is first involved and then the intensity and the propagation to the neighborhood are computed; Finally all the stands involved burn and a penalty is given to each, following the degree of propagation.

The last improvement concerns the *reward function*. Now, the cost of a clearcut decision for an agent depends on the number of stands cut around at the same time period. It's harder to access a distant stand and a path must be designed. To reflect this spatial dependence at each time period, we introduce a new feature: the size of each group of stands cut. We also bound this parameter to a maximum size of stands to avoid artefact effects like clearcuting all the forest. The neighborhood matrix allows us to deal with the dynamic costs of the clearcut decision, and with the management of groups of stands cut. The local reward that each agent receives in state s_n for decision d_n is now defined by: $R_n(s_n,d_n)=r_n(s_t^n,d_t^n)-k_t^n(d_n)$ with $r_n(s_t^n,d_t^n)$ the price of the whole stand timber stock if $d_t^n=1$ (this is also the price of the salvaged timber if there is a fire). $r_n(s_t^n,d_t^n)=0$ if an agent chooses not to cut and there is no fire. $k_t^n(d_n)=c_t^n(Path)/size_t^n(group_{cut})$ is the new cost of the clearcut. Note that this new cost is spatially linked to the decisions of the other agents. Contrary to the original model, we choose not to take into account the cost of fire protection.

4.3. Multiagent Reinforcement Learning Algorithms Chosen

In Schneider et al. (1999), the authors studied a different way to use a distributed value function. As we now focus on the use of the local state, there are no results on the convergence properties of the different distributed Q-learning algorithms. To go further in understanding of the use of local states and local decisions for each agent, we only focus on heuristics using local Q-functions or distributed local Q-functions. We propose to evaluate the behavior of 4 Q-functions.

Algorithm ($\mathcal{A}$): Global reward. For each agent, the reward is global: $R(s,d)$. The Q-function for each agent n is therefore defined by:

$$Q_n(s_n,d_n)=(1-\alpha)Q_n(s_n,d_n)+\alpha(R(s,d)$$
$$+\gamma\max_{d'_n\in D_n}(Q_n(s'_n,d'_n)))$$

Here we assume that an agent is an independent learner. In other words, they perform their actions, obtain a global reward and update their Q-values. This algorithm is the same as used in the previous section in Garcia & Sabbadin (2001), the difference concerns the model chosen.

Algorithm ($\mathcal{B}$): Global reward and distributed Q-functions. For each agent, the reward is global $R(s,d)$ but this time agents take into account the value of their neighbors' Q-functions. The Q-function for each agent n is:

$$Q_n(s_n,d_n)=(1-\alpha)Q_n(s_n,d_n)+\alpha(R(s,d)$$
$$+\gamma\sum_j f(n,j)\max_{d'j\in D_j}(Q_j(s'_j,d'_j)))$$

and

$$f(n,j)\leftarrow\begin{cases}\frac{1}{(|neighb|+1)} & \text{if } j=n\\ \text{or if } j\in neighb(n) & \\ 0 & \text{else}\end{cases}$$

This Q-function was not proposed in Schneider et al. (1999).

Algorithm ($\mathcal{C}$): Local spatial rewards. For each agent, the reward is local: $R_n(s_n,d_n)$. The Q-function for each agent n is:

$$Q_n(s_n,d_n)=(1-\alpha)Q_n(s_n,d_n)+\alpha(R_n(s_n,d_n)$$
$$+\gamma\max_{d'_n\in D_n}(Q_n(s'_n,d'_n)))$$

Here we assume that an agent is an independent learner, it performs its action, obtains a local reward and updates its Q-value. Note that the local reward function is spatially linked to the actions of neighbors.

Algorithm ($\mathcal{D}$): Local spatial rewards and distributed Q-functions. For each agent, the reward is local: $R_n(s_n,d_n)$ and the agents take into account the value of their neighbors' Q-functions. The Q-function for each agent n is:

$$Q_n(s_n,d_n)=(1-\alpha)Q_n(s_n,d_n)+\alpha(R_n(s_n,d_n)$$

$$+\gamma \sum_j f(n,j) \max_{d'_j \in D_j} (Q_j(s'_j, d'_j)))$$

Note that the local reward function is spatially linked to the actions of the neighbors.

The algorithms $\mathcal{C}$ and $\mathcal{D}$ are similar to those proposed in Schneider et al. (1999), except in the definition of the spatially local reward function. Indeed, we use the neighborhood matrix to define $R(s_n, d_n)$ whereas Schneider et al. (1999) used a $f(n,j)$ function. Note that the communications between agents are done by the use of these Q-functions. Because the learning is done online as the system actually passes through various states, there is no need for neighbors to specify to each other what state or action they've been in or taken. It is only necessary for them to transmit their current estimated value of the state they land in at each iteration.

5. SIMULATIONS AND ASSESSMENTS

5.1. First results

We first study the general behavior of each algorithm during the learning phase with $N = 25$ stands, $N = 100$ stands and $A = 6$ (Figure 1). As it is not possible to compute an optimal policy for this problem, so we decided to run a deterministic and uniform policy as a means of comparison (Figure 2 and Table 2). This policy is defined by the cut decision when stands are 6, i.e. when outcomes are higher.

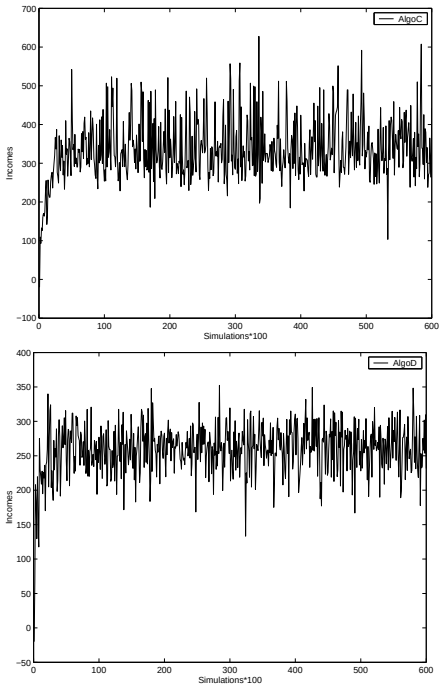

Figure 1. Evolutions of algorithms $\mathcal{C}$ $\mathcal{D}$ during the learning process.

Algorithms	Mean	Standard deviation	Rank
$\mathcal{A}$	211.75 - 163	19.75 - 51.4	4 - 5
$\mathcal{B}$	101.08 - 197	17.94 - 45.76	5 - 4
$\mathcal{C}$	329.73 - 1303.5	68.46 - 87.38	1
$\mathcal{D}$	266.76 - 1148.5	32.81 - 78.62	3
$= 6$	325.24 - 1288.6	65.47 - 72.52	2

Table 2. Cumulated rewards obtained during the simulation with 25 and 100 stands.

Obviously, algorithm $\mathcal{B}$ computes very poor policies. The individual learning strategy used in $\mathcal{A}$ didn't succeed to take into account the spatial dependence between agents decisions whereas $\mathcal{C}$ succeeds to converge toward a better policy than the deterministic policy. These results show that algorithm $\mathcal{C}$ performs well in a spatially linked environment. Besides spatial local rewards, $\mathcal{D}$ gives worse policies than $= 6$. This means that either the Q-functions of the neighbors are not relevant to computing an agent Q-function for this application, or $\mathcal{D}$ needs more iterations to converge to a better policy. To answer this question, we have run algorithm $\mathcal{D}$ for 200000 iterations. Results show that the mean of rewards achieved is slightly better (1164) but still less than the rewards accumulated by the deterministic policy.

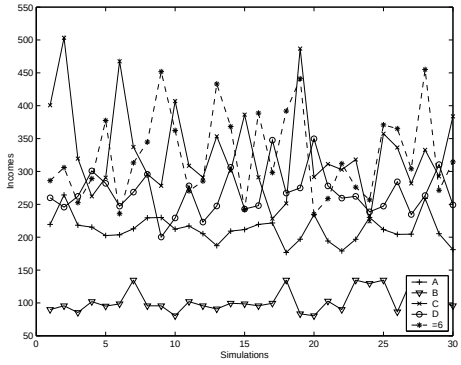

Figure 2. Profiles of the 4 Q-learnings and the best deterministic policy.

5.2. Increasing the local actions space

To improve the exploration of Q-learning, we tried to change the model of the actions space. Previously, the action space was defined by the cut or the do nothing decisions for each agent at each time step. The set of actions is now the time horizon before the cut. So the decision is now after how long an agent will cut a stand. 0 means cut at current time t. Obviously, this new local actions space is limited by the number of age groups.

With the same number of iterations, new profiles show that during the learning phase, the convergence of the algorithms is faster. However, the same results are

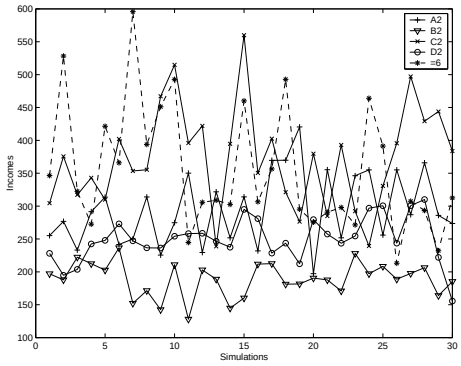

Figure 3. Profiles of the 4 Q-learnings and the best uniform policy.

Algorithms	Mean	Standard deviation	Rank
$\mathcal{A}2$	295.44 -	54.70 -	3
$\mathcal{B}2$	189.05 -	25.59 -	5
$\mathcal{C}2$	372.45 -	75.62 -	1
$\mathcal{D}2$	249.63 -	33.42 -	4
$= 6$	353.75 -	93.55 -	2

Table 3. Simulation results with 25 stands.

achieved (Figure 3, Table 3, except with $\mathcal{A}2$ which converges to a significantly better policy than in the previous model. As our algorithms only deal with local states, increasing the size of the local action space is still a good trade off between exploration and time of convergence. Obviously, this is not suitable for a centralized solving process. Only $\mathcal{C}2$ performs a better policy than the uniform policy $= 6$. We can conclude that a coordination process between agents is effective. The only way to beat the uniform policy at the maximum age is to decide to cut stands by group of neighbors, and thus agents avoid the cost of a single harvesting operation.

6. DISCUSSION

In this paper, we have evaluated multiagent Q-learning algorithms using local states and local actions on a forest management problem. The assessments of our experiments show that among the 4 Q-functions proposed, the local spatial rewards algorithm gave the best policies. Therefore, for this application and the chosen model, it is not interesting to take into account the neighbors' Q-functions when spatial rewards are used. We still have to improve our assessments and comparisons with other heuristic approaches but we believe that the local properties and interactions of MAS approaches seem difficult to beat in terms of memory complexity only in $O(N|A_n|^{|S_n|})$. However, this preliminary work opens several questions in the field of Multiagent RL that have still to be answered. First, the convergence properties of algorithms are still a drawback of these heuristic approaches. Then, under specific conditions, which quality of policy can be expected? The major difficulty in proving convergence properties is that the MAS paradigm deals with partial observations in the field of Markov decision modeling.

7. REFERENCES

Bernstein, D. S., Zilberstein, S. & Immerman, N. (2000), The complexity of decentralized control of markov decision processes, *in* 'Proc. of UAI'.

Chadès, I. (2003), Planification distribuée dans les SMA l'aide de processus décisionnels de Markov, PhD thesis, Universite Nancy 1, LORIA.

Chadès, I. (2004), Multiple equilibria solution for the multi-agent mdp coordination problem, *in* 'Workshop of Multi-Agent Markov Decision Processes: Theories and Models, ECAI04'.

Chadès, I., Scherrer, B. & Charpillet, F. (2002), A heuristic approach for solving decentralized-pomdp: Assessment on the pursuit problem, *in* 'the 2002 ACM Symposium on Applied Computing'.

Dutech, A., Buffet, O. & Charpillet, F. (2001), Multi-agent systems by incremental gradient reinforcement learning, *in* 'Proceedings of IJCAI'01'.

Garcia, F. & Sabbadin, R. (2001), Solving large weakly coupled markov decision processes: Application to forest management, *in* 'MODSIM 2001'.

Nair, R., Tambe, M., Yokoo, M., Pynadath, D. & Marsella, S. (2003), Taming decentralized pomdps: Towards efficient policy computation for multiagent settings, *in* 'IJCAI'03'.

Puterman, M. L. (1994), *Markov Decision Processes–Discrete Stochastic Dynamic Programming*, John Wiley and Sons, Inc., New York, USA.

Schneider, J., Wong, W.-K., Moore, A. & Riedmiller, M. (1999), Distributed value functions, *in* 'Proc. ICML 99', Morgan Kaufmann, San Francisco, CA, pp. 371–378.

Shen, J., Lesser, V. & Carver, N. (2003), Minimizing communication cost in a distributed bayesian network using a decentralized mdp, *in* 'the second international joint conference on Autonomous agents and multiagent systems', pp. 678–685.

Shoham, Y., Powers, R. & Grenager, T. (2003), Multi-agent reinforcement learning: a critical survey, Technical report, Stanford University.

Sutton, R. & Barto, G. (1998), *Reinforcement Learning: an introduction*, Bradford Book, MIT Press, Cambridge, MA.

Watkins, C. & Dayan, P. (1992), 'Q-learning', *Machine Learning* **8**, 279–292.