HAL
open science

Reinforcement learning for spatial processes

Nicklas Paul Forsell, Frederick F. Garcia, Régis Sabbadin

To cite this version:

Nicklas Paul Forsell, Frederick F. Garcia, Régis Sabbadin. Reinforcement learning for spatial processes. 18. World IMACS Congress and MODSIM09. International Congress on Modelling and Simulation, Jul 2009, Cairns, Australia. 7 p. hal-02752972

HAL Id: hal-02752972

https://hal.inrae.fr/hal-02752972

Submitted on 3 Jun 2020

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.

18th World IMACS / MODSIM Congress, Cairns, Australia 13-17 July 2009
http://mssanz.org.au/modsim09

Reinforcement learning for spatial processes

Forsell, N. [1,2], **F. Garcia** [1] **and R. Sabbadin** [1]

[1] *Department of Forest Resource Management, SLU, Umeå, SWEDEN*
[1] *Unité de Biométrie et Intelligence Artificielle, INRA de Toulouse, FRANCE*
Email: nicklas.forsell@srh.slu.se

Abstract: Markov decision processes (MPDs) have become a popular model for real-world problems of planning under uncertainty. A vide range of applications has been published within the fields of natural resources management, forestry, agricultural economics, and robotics. An MDP can be used to represent and optimize the management of an environment. A state variable is used to represent the current state of the environment and the interaction with the environment is expressed with an action variable, allowing for stochastic transition from one state to another state. A transition function is used to express the transition between the different states of the environment, and a reward function expresses the rewards received by applying an action when the environment is in a specific state. MDPs can thereby be used to optimize management strategies for problems with uncertain effects of actions, multiple and conflicting objectives, stochastic events, and stochastic environments.

A common problem when optimizing management strategies for natural resources is that large areas have to be considered, rendering the state and action space of the MDP large. For example, an agricultural area consisting of thousands of crop fields, or forests consisting of thousands of stands. Also, spatial aspects may have to be considered. Some examples are management of epidemics, invasive or endangered species, management to prevent wind damage and wild fire. However, the structure of the problem can be used to model the problem on a compact form. For example is the probability of a tree being damaged by wind only on the state of the neighbouring trees. This type of local dependency and internal structure can be used to express the problem on a compact factored form. For example, instead of modelling the state of forest with a single state variable, a set of state variables is utilized, each expressing the state of a part of forest. A number of different representational models have been proposed for this type of problems. Factored MDPs and collaborative multiagent MDPs are two frameworks that have recently received a loot of attention.

The graph-based Markov decision process (GMDP) framework has recently been developed specifically for modelling a range of large-scale spatiotemporal problems. As the name suggests, the GMDP uses a graph structure to represent the transition and reward functions on a compact form. Two algorithms for computing approximate solutions for GMDPs have been proposed. The algorithms are capable of solving large-scale GMDPs as their running time increases only linearly and polynomially with the size of the problem for a fixed induced width of the graph. However, the algorithms are model-dependent and require the transition and reward functions to be known. Simulation based models of the transition and reward functions can therefore not be used. Since there are a large number of simulation based models of real-world management problems, for example (Blennow & Sallnäs, 2004; Finney, 1994), the development of model-free solution algorithms is an important extension of the GMDP framework and will increase the applicability of GMDPs.

In this paper, we describe a number of model-free reinforcement learning (RL) algorithms for GMDPs derived from existing model-free RL algorithms for collaborative multiagent MDPs. The collaborative multiagent MDP framework is similar to the GMDP framework in a number of aspects, and in this paper we use these similarities to propose RL algorithms for GMDPs. The performance of the algorithms is compared to model-dependent algorithms for a medium- and large-scale forest management problem. Our experimental results show that the proposed RL algorithms are able to efficiently compute policies demonstrating near-optimal performance. The policies computed by the RL algorithms are of similar quality to policies computed by the model-dependent algorithms.

Keywords: *Reinforcement Learning (RL), planning under risk and uncertainty, spatial processes, Markov decision processes (MDPs)*

1. INTRODUCTION

Markov decision processes (MDPs) are commonly used for representing and solving sequential decision-making problems under uncertainty. Real-world problems are commonly described on a factored form, making it possible to utilize the structure of the problem to compactly represent the transition and reward functions either utilizing a *factored* MDP model (Guestrin *et al.*, 2003; Boutilier *et al.*, 2000), or a *collaborative multiagent* MDP model (Guestrin *et al.*, 2003). However, a common feature of factored MDPs and collaborative multiagent MDPs models, is that the underlying MDP is large, rendering the computation of high quality policies impractical. Graph-based Markov decision processes (GMDPs) (Forsell & Sabbadin, 2006; Peyrard & Sabbadin, 2006) is a recently proposed framework specifically developed for modeling a range of large scale spatiotemporal problems. The framework was specifically developed for computing high quality policies for large-scale problems. Two algorithms for computing near-optimal policies for large scale GMDPs has been proposed (Forsell & Sabbadin, 2006; Peyrard & Sabbadin, 2006). However, the algorithms are model-dependent and require complete knowledge concerning the transition and reward functions. Simulated based models of the transition and rewards can therefore not be used.

In this paper we describe a number of model-free RL algorithms for GMDPs derived from existing model-free RL algorithms for collaborative multiagent MDPs. The collaborative multiagent MDP framework is similar to the GMDP framework in a number of aspects, and in this paper we use these similarities to propose RL algorithms for GMDPs. The remainder of this paper is structured as follows. In Section 2 we review the notion of MDPs, GMDPs and collaborative multiagent MDPs. In Section 3, we review several RL algorithms and show how they can be adapted to the GMDP framework. In Section 4, we give experimental results on a medium- and large-scale forest management problem. Finally, we present some conclusions in Section 5.

2. MARKOV DECISION PROCESSES

In the classical formulation (Puterman, 1994) a stationary MDP is defined by a four-tuple $< X, A, p, r >$ where: X is a finite set of possible states of the system; A is a finite set of applicable actions; p is a Markovian transition function $p : X \times X \times A \rightarrow [0,1]$ such that $p(x'|x,a)$ represents the probability of moving from state x to state x' by applying action a; and r is an "immediate" reward function $r : X \times A \rightarrow \Re$ such that $r(x,a)$ represents the reward obtained in state x after taking action a. At each decision step, an action a is selected to be applied, after which the system state changes from x to x'. The discrete set of decision steps is assumed to be infinite. A (stationary) *policy* defines for each possible state of the system, which action to apply. A policy is thus a function $\pi : X \rightarrow A$, that assigns an action to every state. The infinite horizon, discounted, cumulative reward associated with applying a policy π to an MDP, with initial state x, is defined through a *value function* $v_\pi : X \rightarrow \Re$

$$v_\pi(x) = E\left[\sum_{t=0}^{\infty} \gamma^t r\left(x^t, \pi(x^t)\right) \middle| x^0 = x\right] \tag{1}$$

The expectation is taken over all possible trajectories $\tau = < x^0, a^0, x^1, a^1, \ldots, x^t, a^t, \ldots >$ where, from the initial state x, the policy π is applied. The discount factor, $0 \leq \gamma < 1$, ensures that the infinite sum in (1) converges. The problem of finding an optimal policy π^*, solving the MDP, with respect to the discounted criterion (1) can be written as: Find π^* so that: $v_{\pi^*}(x) \geq v_\pi(x)$, $\forall x \in X$, $\forall \pi \in A^X$. It has been shown that there always exists an optimal policy for an MDP. A number of methods for computing the optimal policy have been suggested (e.g. Bertsekas & Tsitsiklis, 1996; Puterman, 1994). However, these algorithms reach their limits when the MDP is large.

2.1. Graph-based Markov decision processes

GMDPs are a class of MDPs, in which the state and action spaces are multidimensional and there are local dependencies between the state and action variables. A GMDP is defined by a five-tuple $< X, A, p, r, G >$ where: X is a Cartesian product of finite sets $X = X_1 \times \ldots \times X_n$; A is a Cartesian product of finite sets $A = A_1 \times \ldots \times A_n$; p is a transition function; r is an reward function; and G is a directed graph $G = (V, E)$ expressing dependencies among the state variables X_i, i = 1,…,n. We represent the joint state of the system as $x = (x_1,\ldots,x_n)$, and the joint action as $a = (a_1,\ldots,a_n)$. Vectors of two or more variables values are highlighted in bold. Upper case letters represent random variables (X, X_i), and lower case letters are used to represent the value of the random variables ($\mathbf{x}$, x_i). The local dependencies expressed by the graph $G = (V, E)$ are modelled through a neighbourhood function: $N : V \rightarrow 2^V$, which is defined on the graph by: $N(i) = \{j \in V \mid (j,i) \in E\}$,

$\forall i \in V$. The transition and reward functions of the GMDP model are factorized according to the local dependencies in the following manner: $p(\boldsymbol{x}'|\boldsymbol{x},\boldsymbol{a}) = \prod_{i=1}^{n} p_i(x_i' \mid \boldsymbol{x}_{N(i)}, a_i)$, and $r(\boldsymbol{x},\boldsymbol{a}) = \sum_{i=1}^{n} r_i(\boldsymbol{x}_{N(i)}, a_i)$, where $\boldsymbol{x}_I = (x_{j1},\ldots,x_{jk})$, $j_l \in I$, $l = 1,\ldots,k$. No dependencies between the action variables A_i are expressed by the graph as each local transition and reward function is only dependent on a single action variable.

Two algorithms for solving large-scale GMDPs have been suggested. One is based on *Mean-Field Approximation* and *Approximate Policy Iteration* (MF-API) (Peyrard & Sabbadin, 2006), and the other on *Approximate Linear Programming* (ALP) (Forsell & Sabbadin, 2006). Both algorithms compute an approximate solution to the problem, and their running time increases polynomially and linearly, respectively, with increases in the size of the problem for a fixed induced width of the graph. However, the two algorithms are model-dependent and require knowledge concerning the transition and reward functions.

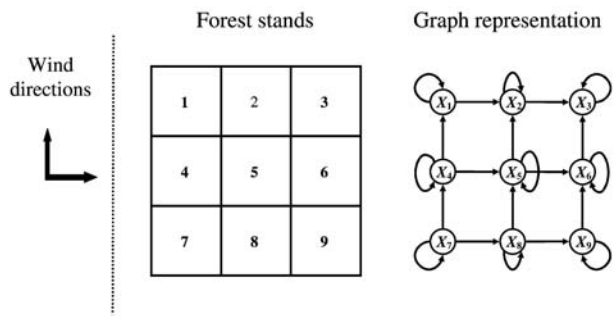

Figure 1. Forest stands and graph representing neighbouring relations between the stands when wind directions are restricted to north and east.

GMDP Example: Forest management under risk of wind damage

In the forest wind damage problem, a manager needs to specify how a forest at risk of being damaged by wind should be managed (Forsell *et al.*, 2009; Meilby *et al.*, 2001). The forest is divided into geographical parts, called *stands*. Each stand is represented with a state variable X_i, representing the age of the stand: $X_i \in$ {0-25 years, 25-50 years, 50-75 years, 75-100 years}. The state variables can thus be used to compute the value of the timber (which is related to the age of the trees) in the stands. The stands are individually managed and the management options are represented by action variables $A_i \in$ {clear-cut, do nothing}. The dynamics of the process have a deterministic and a stochastic component. Natural aging and clear-cuttings of the stands (replacing all trees in a stand with new, "zero-aged", ones) are deterministic, while damage to the stands caused by wind, which forces the forest owner to perform (possibly unplanned) clear-cuttings, occur stochastically. The transition probability of a stand depends on the action locally applied, and the "shelter effect" provided by its neighbouring stands. If a stand is sufficiently old, it can block the wind and decrease the risk of damage to neighbouring stands. The probability of a stand being damaged by wind increases with the aging of the stand, and decreases with the aging of the neighbouring stands. Here, the wind was assumed to only come from two directions, each with equal probability. Figure 1 shows the probabilistic dependencies between the local state variables. A reward is received when a stand is clear-cut and the reward increases with the aging of the stand. A forced clear-cutting due to damage will only provide a small proportion of the benefit obtained by clear-cutting the undamaged stand.

2.2. Collaborative multiagent MDPs

Collaborative multiagent MDP (Guestrin *et al.*, 2003) form another class of MDPs. A collaborative multiagent MDP is defined by a four-tuple $< X, A, p, r >$ where: X is a Cartesian product of finite sets $X = X_1 \times\ldots\times X_m$; A is a Cartesian product of finite sets $A = A_1 \times\ldots\times A_n$; p is a transition function; and r is an reward function. The numbers of local state and action variables do not have to be the same. The transition and reward functions are factored as: $p(\boldsymbol{x}'|\boldsymbol{x},\boldsymbol{a}) = \prod_{i=1}^{n} p_i(x_i' \mid \boldsymbol{x},\boldsymbol{a})$, and $r(\boldsymbol{x},\boldsymbol{a}) = \sum_{i=1}^{n} r_i(x,a)$, where the local reward and transition functions only depend on small subsets of local state and action variables. The system is called multiagent as the number of agents interacting within the system is n, and collaborative as the agents are trying to work together to optimize a shared performance measure. A GMDP can be seen as a special case of a collaborative multiagent MDP, in which the number of local state and action variables is the same (n=m).

Each local transition and reward function only depends on a single local action variable, and a graph structure (with n nodes) describes the dependencies between the local state variables. These similarities will now be used to adapt a number of existing model-free collaborative multiagent MDP RL techniques to the GMDP framework.

3. REINFORCEMENT LEARNING FOR GMDPS

Reinforcement learning (RL) (Sutton & Barto, 1998; Bertsekas & Tsitsiklis, 1996) is an approach that can be used to solve MDPs with unknown transition and reward functions. It is a widely used simulation-based approach that computes a solution to an MDP by iteratively improving an estimation of the optimal solution. Recall that solving an MDP is the problem of finding the optimal policy π^*, where the optimal policy can be characterized as the one that maximizes the expected discounted future reward (1) for each state x. The expected discounted future reward can also be represented for each state x and action a, with the use of a Q-function. A Q-function, also known as action-value function, represents the expected discounted future reward for a state x when action a is selected to be performed:

$$Q_\pi(x,a) = E\left[\sum_{t=0}^{\infty} \gamma^t r\left(x^t, \pi(x^t)\right) \middle| x^0 = x, \pi(x^0) = a)\right] \tag{2}$$

The Q-functions can be used to characterize the optimal policy π^* as the one that maximizes (2) for each state-action pair (x,a). The problem of solving an MDP is thus the same as the problem of finding the optimal Q-function Q^*. The approach is appealing as it has been shown that the optimal Q-function Q^* satisfies the Bellman equation (Sutton & Barto, 1998; Puterman, 1994):

$$Q^*(x,a) = r(x,a) + \gamma \sum_{x' \in X} p(x' | x,a) \max_{a' \in A} Q^*(x',a') \tag{3}$$

Reinforcement learning uses this observation to solve an MDP by computing an approximation of the optimal Q-function Q^*. The method iteratively improves an estimation of the optimal Q-function by simulations. A trivial approximation of the Q-function is initiated, after which N trajectories τ are simulated. The trajectories start at a randomly selected state x^0, and for each time step t, an action a^t is selected to be performed. A commonly used strategy for exploring the actions is the ε-greedy strategy. In this approach the greedy action $a = \text{argmax}_{a \in A}\, Q(x,a)$ is selected with a high probability, while a random action is selected with a small probability ε. After the action has been selected, the future state x^{t+1} and reward $r(x^t,a^t)$ are simulated, based on the state-action pair (x^t,a^t). After each simulation, the Q-function is updated with a combination of the current and simulated values:

$$Q(x,a) \leftarrow (1-\alpha)Q(x,a) + \alpha\left(r(x,a) + \gamma \max_{a' \in A} Q(x',a')\right) \tag{4}$$

where $\alpha \in (0,1)$ is the learning rate. The RL approach is known to converge to the optimal Q-function Q^* when every state-action pair is sampled infinitely many times.

For GMDPs it is however necessary to decompose the Q-function. The optimal Q-function Q^* can of course be found by iteratively applying (3) to Q-functions defined over the joint state and action variables $(\mathbf{x},\mathbf{a})$. However, such an approach quickly becomes intractable for large problems, as a matrix of size $|X| \times |A|$ is required to represent the Q-function. Instead, the global Q-function can be expressed as a sum of local Q-functions, where each local Q-function is defined over a subset of the state and action variables, and is thus of a manageable size. However, this is an approximation as generally the standard convergence proof for Q-learning is no longer valid. How then, should the local Q-functions be defined, to restrict the loss of quality in the computed policy while keeping the size of the local Q-functions at a minimum? Furthermore, how should these local Q-functions be updated? Several decompositions of the global Q-function together with update rules have been suggested for collaborative multiagent MDP (e.g. Kok & Vlassis, 2006; Guestrin *et al.*, 2002; Schneider *et al.*, 1999; Claus & Boutilier, 1998). We now show how these algorithms can be adapted to the GMDP framework and how they can be used to compute an approximate solution to a GMDP.

3.1. Decomposition of the global Q-function

Q-function decomposition methods for the collaborative multiagent MDP framework can be divided into two general approaches. In the first, the global Q-function is decomposed over the agents in the system (5) - (7) (Guestrin *et al.*, 2002; Schneider *et al.*, 1999; Claus & Boutilier, 1998). In the GMDP framework, this

implies that the global Q-function is approximated by a sum of n local Q_i-functions. Thus, it can be seen as decomposition of the global Q-function over the nodes of the graph $G = (V, E)$. The following three versions of this decomposition approach can be derived from the collaborative multiagent MDP literature. In the second approach, is the global Q-function decomposed over pairs of dependent action variables (8) (Kok & Vlassis, 2006). In the GMDP framework, this implies that the global Q-function can be approximated by a sum of $|E|$ local Q_{ij}-functions. Thus, it can be seen as decomposition of the global Q-function over the edges of graph $G = (V, E)$.

$$Q(\boldsymbol{x},\boldsymbol{a}) = \sum_{i=1}^{n} Q_i(\boldsymbol{x}, a_i) \tag{5}$$

$$Q(\boldsymbol{x},\boldsymbol{a}) = \sum_{i=1}^{n} Q_i(\boldsymbol{x}_{N(i)}, a_i) \tag{6}$$

$$Q(\boldsymbol{x},\boldsymbol{a}) = \sum_{i=1}^{n} Q_i(\boldsymbol{x}_{N(i)}, \boldsymbol{a}_{N(i)}) \tag{7}$$

$$Q(\boldsymbol{x},\boldsymbol{a}) = \sum_{(i,j)\in E}^{n} Q_{ij}(\boldsymbol{x}_{N(j)}, a_i, a_j) \tag{8}$$

3.2. Update rule

A number of update rules can be used when the global Q-function has been decomposed. Three update rules have been proposed for the decompositions (5), (6) and (7) (Guestrin *et al.*, 2002; Schneider *et al.*, 1999; Claus & Boutilier, 1998). We present them in the case where decomposition (6) is used:

$$Q_i(\boldsymbol{x}_{N(i)}, a_i) \leftarrow (1-\alpha)Q_i(\boldsymbol{x}_{N(i)}, a_i) + \alpha\left(r_i(\boldsymbol{x}_{N(i)}, a_i) + \gamma \max_{a_i \in A_i} Q_i(\boldsymbol{x}'_{N(i)}, a'_i) \right) \tag{9}$$

$$Q_i(\boldsymbol{x}_{N(i)}, a_i) \leftarrow (1-\alpha)Q_i(\boldsymbol{x}_{N(i)}, a_i) + \alpha\left(r_i(\boldsymbol{x}_{N(i)}, a_i) + \gamma\left(\sum_{j\in N(i)} f(i,j) \max_{a'_j \in A_j} Q_j(\boldsymbol{x}'_{N(j)}, a'_j) \right)\right) \tag{10}$$

$$Q_i(\boldsymbol{x}_{N(i)}, a_i) \leftarrow Q_i(\boldsymbol{x}_{N(i)}, a_i) + \alpha\left(r(\boldsymbol{x},\boldsymbol{a}) + \gamma\left(\max_{a'\in A} Q(\boldsymbol{x}',\boldsymbol{a}') - Q(\boldsymbol{x},\boldsymbol{a})\right)\right) \tag{11}$$

These update rules can be adapted to the other decompositions, by changing the scopes of the local Q-functions. The weight function f(i,j) defines how much Q_j contributes to updates of Q_i and is defined by the user. If the global Q-function is decomposed according to the edge-based Q-functions (8), the local Q-functions can be updated according to the following update rule:

$$Q_{ij}(\boldsymbol{x}_{N(j)}, a_i, a_j) \leftarrow (1-\alpha)Q_{ij}(\boldsymbol{x}_{N(j)}, a_i, a_j) + \alpha\left(\frac{r_j(\boldsymbol{x}_{N(j)}, a_j)}{|N(j)|} + \gamma\left(\max_{a'_i \in A_i, a'_j \in A_j} Q_{ij}(\boldsymbol{x}'_{N(j)}, a'_i, a'_j) \right)\right) \tag{12}$$

3.3. Algorithms

The decompositions and the update rules can be combined in a number of ways. Some specific combinations have been described in algorithms that are well studied:

- *Independent learners* (IL) (Claus & Boutilier, 1998): The global Q-function is decomposed according to (5), which are updated according to an adapted version of update rule (9).
- *Distributed value functions* (DVF) (Schneider *et al.*, 1999): In this approach the global Q-function is decomposed according to (6), and the components are updated according to (10). In the GMDP case, the weight function can be defined according to the structure of the graph $G = (V, E)$. A commonly used approach is to apply equal proportionally over all neighbours and express the weight function as: f(i,j) = 1|i ∪ N(j)|, if j ∈ N(i), and zero otherwise.
- *Agent-based decomposition for sparse cooperative Q-learning* (AbSparceQ) (Kok & Vlassis, 2006): This approach combines decomposition (6) with update rule (9).
- *Edge-based decomposition for sparse cooperative Q-learning* (EbSparceQ) (Kok & Vlassis, 2006): In this approach the global Q-function is decomposed according to (8) and updated according to (12). A drawback of this approach is that the update of Q_i requires the knowledge concerning the actions (a_i*,a_j*) that maximizes Q_{ij}: $\{a_i^*, a_j^*\} = \text{argmax}_{a_i' \in A_i, a_j' \in A_j} Q_{ij}(\boldsymbol{x}'_{N(i) \cup N(j)}, a_i', a_j')$. Two algorithms have been proposed for computing this term, a variable elimination algorithm (VE) (Guestrin *et al.*, 2003), and a max-plus algorithm (MP) (Kok & Vlassis, 2006). The VE algorithm is exact and the MP algorithm is an approximate alternative to the VE algorithm that is faster in computational time for large-scale problems. We will only be using the VE algorithm as it is exact and always computes the optimal global action. It

should be noted that the VE algorithm is also required in the ε-greedy strategy, where the VE algorithm has to be used to compute the greedy action.

- *Coordinated reinforcement learning* (CoordRL) (Guestrin *et al.*, 2002): The global *Q*-function is decomposed according to (6), and updated according to (11). As in the EbSparceQ approach, the VE algorithm is required to compute the joint action $\boldsymbol{a}^* = \text{argmax}_{\boldsymbol{a}' \in A}\, Q(\boldsymbol{x}', \boldsymbol{a}')$, and the greedy action for the ε-greedy strategy selection.

4. EXPERIMENTS

To evaluate the performance of the different decomposition and update rules, we used the forest management problem described in Section 2.1. This particular problem was selected as the problem size in real-world applications is often very large. Problems consisting of thousands of stands are commonly considered. The problem therefore provides good illustration of the abilities of a GMDP to handle extremely large state and action spaces. However, as some of the described algorithms are unable to tackle large-scale problems, a medium-scale problem was first used to evaluate the performance of the algorithms. The performance of the scalable algorithms was subsequently evaluated on a large-scale forest management problem. The square topology presented in Figure 2 was used for both problems.

To assess the quality of the algorithms in Section 3.3, we compared them to the previously proposed model-dependent ALP and MF-API algorithms, a heuristic greedy policy, and a utopian upper bound on the value of the optimal policy. Two additional algorithms were also evaluated. The first algorithm can be seen as an extended version of the AbSparceQ approach as it combines decomposition (7) with update rule (9). The second algorithm can be seen as an extended version of the CoordRL approach as it combines decomposition (7) with update rule (11). The utopian upper bound on the value of the optimal policy was computed by assuming that maximal sheltering effect is always given to the stands by their neighbours. This is a utopian bound since the shelter provided by a stand varies with the age of the stand. In our implementation a stand of age 75-100 years provides 40% more shelter than a stand of age 50-75 years. The algorithms were tested on the problems with the following parameters: α = 0.2, ε = 0.2 and γ = 0.9. All the *Q*-learning algorithms were initialized with *Q*-values of zero, after which the learning iterations were performed. There were 50,000 iterations for the medium-scale problem, 40,000 iterations for the large-scale problem, and every 2,000 iterations the policies were computed and simulated with standard Monte Carlo simulations. The estimated value of managing the forest according to the policy was computed as the average over 50 randomly selected start points, where the value for a specific start point was averaged over 100 44-step long runs. The algorithms were implemented in Scilab (2004) and tested using an Intel Pentium 3.6 GHz machine with 1 GB internal memory.

For the medium-scale problem, a forest consisting of nine stands was considered. Figure 2 (top) and

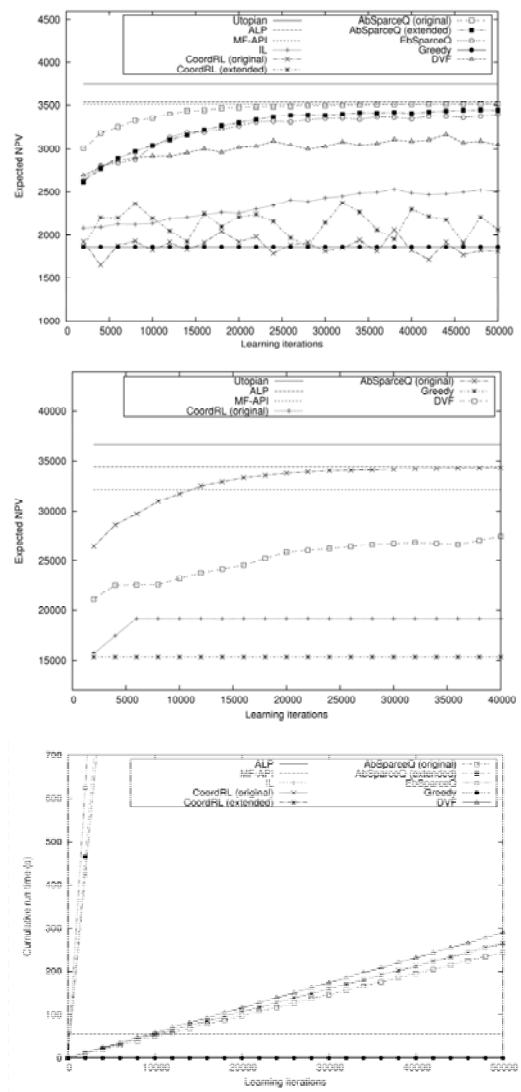

Figure 2. Performance of the RL algorithms on the forest management problem: (top) estimated value of the computed policies for a forest consisting of 9 stands, (middle) estimated value of the computed policies for a forest consisting of 100 stands, (bottom) running time of the algorithms for the forest consisting of 9 stands.

(bottom) show the simulated value and running time of the algorithms. The results show that three of the RL approaches are able to compute a policy that is similar to the policies computed by the ALP and MF-API algorithms in terms of estimated value. These are the EbSparceQ, AbSparceQ and the extended version of the AbSparceQ approach. EbSparceQ and the extended version of the AbSparceQ approach gave similar results in terms of the value of the computed policy and the computational time. They are however outperformed by the AbSparceQ approach as its policy has a slightly higher value, and it is much faster. This is because it does not consider coordinated action selection and therefore does not require the VE algorithm. For the large-scale forest management problem a forest consisting of 100 stands was considered. This resulted in $4^{100} \approx 1.607*10^{60}$ joint states and $2^{100} \approx 1.268*10^{30}$ joint actions. The RL algorithms evaluated on this problem were the DVF, and CoordRL algorithms. The IL algorithm was not evaluated on this problem as it requires the storing of matrices that are too large for the available computational resources. EbSparceQ, the extended versions of AbSparceQ, and CoordRL were not evaluated on the large-scale forest management problem as they require the VE algorithm whose computational time is too long for this large-scale problem. Figure 2 (middle) shows the estimated values of the policies computed by the algorithms. The results show that the AbSparceQ algorithm is capable of computing a policy that is similar to the best policy computed by the ALP and MF-API algorithms in terms of estimated value.

5. DISCUSSION

We have presented a set of model-free reinforcement learning algorithms for solving a GMDP. In contrast to the previously proposed algorithms for solving a GMDP, the presented RL algorithms are model-free and do not require knowledge of the transition and reward functions. This is an important extension of the GMDP framework as there are large numbers of simulation-based models that can be used to simulate the development of forest and agricultural areas over time, especially for the kinds of natural resource management problems considered in this report (Gardiner *et al.*, 2005; Blennow & Sallnäs, 2004).

REFERENCES

Bertsekas, D.P. & Tsitsiklis, J.N. (1996). *Neuro-dynamic programming*. Athena Scientific.

Blennow, K. & Sallnäs, O. (2004). WINDA-a system of models for assessing the probability of wind damage to forest stands within a landscape. *Ecological Modelling* 175, 87-99.

Boutilier, C., Dearden, R. & Goldszmidt, M. (2000). Stochastic Dynamic Programming with Factored Representations. *Artificial Intelligence* 121(1), 49-107.

Claus, C. & Boutilier, C. (1998). The Dynamics of Reinforcement Learning in Cooperative Multiagent Systems. In: *Proceedings of AAAI/IAAI* pp. 746-752.

Finney, M.A. (1994). Modeling the spread and behavior of prescribed natural fires. In: *Proceedings of 12th Conference on Fire and Forest Meteorology* pp. 138-143.

Forsell, N. & Sabbadin, R. (2006). Approximate Linear-Programming Algorithms for Graph-Based Markov Decision Processes. In: *Proceedings of ECAI*, Riva del Garda, Italy pp. 590-599.

Forsell, N., Wikström, P., Garcia, F., Sabbadin, R., Blennow, K. & Eriksson, L.O. (2009). Management of the risk of wind damage in forestry: a graph-based Markov decision process approach. *Annals of operations research*

Gardiner, B.A., Marshall, B., Achim, A., Belcher, R.E. & Wood, C.J. (2005). The stability of different silvicultural systems: a wind-tunnel investigation. *Forestry* 78(5), 471-484.

Guestrin, C., Koller, D., Parr, R. & Venkataraman, S. (2003). Efficient Solution Algorithms for Factored MDPs. *Journal of Artificial Intelligence Research* 19, 399-468.

Guestrin, C., Lagoudakis, M.G. & Parr, R. (2002). Coordinated Reinforcement Learning. In: *Proceedings of ICML* pp. 227-234.

Kok, J.R. & Vlassis, N.A. (2006). Collaborative Multiagent Reinforcement Learning by Payoff Propagation. *Journal of Machine Learning Research*(7), 1789-1828.

Meilby, H., Strange, N. & Thorsen, B.J. (2001). Optimal spatial harvest planning under risk of windthrow. *Forest Ecology and Management* 149, 15-31.

Peyrard, N. & Sabbadin, R. (2006). Mean Field Approximation of the Policy Iteration Algorithm for Graph-Based Markov Decision Processes. In: *Proceedings of ECAI* pp. 595-599.

Puterman, M.L. (1994). *Markov Decision Processes*. New York: John Wiley and Sons.

Schneider, J.G., Wong, W.K., Moore, A.W. & Riedmiller, M.A. (1999). Distributed Value Functions. In: *Proceedings of ICML* pp. 371-378.

Scilab--a free scientific software package, h.w.s.o., INRIA (2004).

Sutton, R.S. & Barto, A.G. (1998). *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press