International Journal of Approximate Reasoning 53 (2012) 66–86

Contents lists available at SciVerse ScienceDirect

International Journal of Approximate Reasoning

journal homepage: www.elsevier.com/locate/ijar

A framework and a mean-field algorithm for the local control of spatial processes

Régis Sabbadin [a,*], Nathalie Peyrard [a], Nicklas Forsell [a,b]

[a] *Unité de Biométrie et Intelligence Artificielle UR 875, Département de Mathématiques et Informatique Appliquées, INRA, Centre de Toulouse, F-31326 Castanet-Tolosan, France*

[b] *Centre for Applied Mathematics, MINES ParisTech, France*

ARTICLE INFO

Article history:
Received 26 March 2010
Received in revised form 23 September 2011
Accepted 23 September 2011
Available online 1 October 2011

Keywords:
Decision-theoretic planning
Factored Markov decision processes
Approximate policy iteration
Mean-field principle
Approximate linear programming

ABSTRACT

The Markov Decision Process (MDP) framework is a tool for the efficient modelling and solving of sequential decision-making problems under uncertainty. However, it reaches its limits when state and action spaces are large, as can happen for spatially explicit decision problems. Factored MDPs and dedicated solution algorithms have been introduced to deal with large factored state spaces. But the case of large action spaces remains an issue. In this article, we define graph-based Markov Decision Processes (GMDPs), a particular Factored MDP framework which exploits the factorization of the state space *and* the action space of a decision problem. Both spaces are assumed to have the same dimension. Transition probabilities and rewards are factored according to a single graph structure, where nodes represent pairs of state/decision variables of the problem. The complexity of this representation grows only linearly with the size of the graph, whereas the complexity of exact resolution grows exponentially. We propose an approximate solution algorithm exploiting the structure of a GMDP and whose complexity only grows quadratically with the size of the graph and exponentially with the maximum number of neighbours of any node. This algorithm, referred to as MF-API, belongs to the family of Approximate Policy Iteration (API) algorithms. It relies on a *mean-field approximation* of the value function of a policy and on a search limited to the suboptimal set of *local policies*. We compare it, in terms of performance, with two state-of-the-art algorithms for Factored MDPs: SPUDD and Approximate Linear Programming (ALP). Our experiments show that SPUDD is not generally applicable to solving GMDPs, due to the size of the action space we want to tackle. On the other hand, ALP can be adapted to solve GMDPs. We show that ALP is faster than MF-API and provides solutions of similar quality for most problems. However, for some problems MF-API provides significantly better policies, and in all cases provides a better approximation of the value function of approximate policies. These promising results show that the GMDP model offers a convenient framework for modelling and solving a large range of spatial and structured planning problems, that can arise in many different domains where processes are managed over networks: natural resources, agriculture, computer networks, etc.

* Corresponding author. Tel.: +33 (0)5 6128 5476; fax: +33 (0)5 6128 5335.
E-mail addresses: sabbadin@toulouse.inra.fr (R. Sabbadin), peyrard@toulouse.inra.fr (N. Peyrard), nicklas.forsell@mines-paristech.fr (N. Forsell).

0888-613X/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.ijar.2011.09.007

1. Introduction

Within Decision Theory, the *Markov Decision Process* (MDP) framework [1] is used to model and solve sequential decision-making problems under uncertainty. However, MDPs reach their limits when state and action spaces are *factored*, which is the case in domains where spatial aspects are introduced, as in the management of epidemics or invasive/endangered species. The same limits are obviously present in the fields of operations research or computer science (computer networks management, information queries on the World Wide Web, etc.). A common approach used to circumvent the computational burden linked to factored state spaces is the *Factored MDP* (FMDP) framework [2–5].

In this framework, the structure of the state space is exploited to model the decision problem compactly. A state of the FMDP is described by the value of several state variables. The global transition and reward functions of a FMDP are defined from *local* functions depending on a small subset of the state variables: respectively a product of local transition functions and a sum of local reward functions. Usually, only the state space structure is exploited, not the action space structure. Standard solution algorithms for MDPs are of limited use for solving FMDPs, since they manipulate a *global value function* over the whole state space, requiring full enumeration of this exponentially large space. Unfortunately, there is no guarantee that the structure of a factored MDP is reflected in the structure of its global value function, meaning that exact solution of FMDPs is generally unattainable.

However, solution methods such as *Stochastic Programming Using Decision Diagrams* (SPUDD) [3] try to retain as much as possible of the structure of the problem to compute exact (or approximate [4]) optimal policies for FMDPs. On the other hand, other approaches, such as *Approximate Linear Programming* (ALP) [6] use parametrised linear approximations of the value function for approximately solving large MDPs (factored or not). The linear approximation of the value function is a linear combination of predefined *basis functions*, defined over small subsets of the state variables. However, one limitation to the size of problems that can be solved remains the size of the action space of the Factored MDP.

In this article, we adopt an alternative and original approach in order to develop an efficient approximation algorithm for a subclass of FMDPs where state space and action space are factored: *graph-based MDPs* (GMDPs). In a GMDP, the state and action spaces are multidimensional and of identical dimension, n. The global transitions and rewards are assumed to be decomposable into local transition and reward functions, each depending only on a single action variable and on few state variables. This decomposition generates a neighbourhood relationship between variables, represented by a graph where nodes are associated with pairs of state/decision variables and edges between two nodes mean that one of the state variables is within the scope of the reward or the transition function of the other. The GMDP framework can be used to model a large range of spatially explicit or structured planning problems where local decisions influence state variable dynamics.

The approximate solution algorithm we propose belongs to the family of Approximate Policy Iteration (API [7]) algorithms. It relies on an approximation of the value function by a sum of functions of limited scope and on a search limited to the suboptimal set of *local policies*, as would be the case for an ALP algorithm. However, here the approximate value function is derived from a *mean-field* approximation of the Markov process induced by a fixed policy. The mean-field principle arises from statistical mechanics [8] and from the study of systems of particles in interaction. It has since been successfully used in other fields such as image analysis, epidemiology and ecology, for statistical model estimation, graphical model inference, and analysis of large Ordinary Differential Equation systems. When applied to GMDPs, the mean-field approximation amounts to an approximation of the transition function of the GMDP Markov chain for a fixed policy by n independent unidimensional Markov chain transition functions, chosen to minimise a given distance function to the exact transition function. The minimum is reached by independent systems obtained by fixing the neighbourhood influence to its mean value. Note that spatial interaction is not entirely lost with the mean-field approximation since the neighbourhood influence is taken into account. Furthermore, the approximation results in time-dependent transition functions, allowing to reflect the propagation of influence between neighbours. Each iteration of the corresponding algorithm, named MF-API, has a time complexity which is quadratic in n and exponential in the maximum number of neighbours of any node (ν). Representation of a local policy is also of limited space complexity (exponential in ν and linear in n).

The paper is organised as follows. In Section 2, we describe the Graph-based Markov Decision Process (GMDP) framework, and we illustrate the model on two simplified problems, derived from crop disease management and forestry. In Section 3 we give a full description of the MF-API algorithm which constitutes the main contribution of the paper. Then, in Section 4 we discuss existing models and algorithms which can be adapted to solve GMDPs. In particular, after having shown the inability of the exact SPUDD algorithm to solve GMDPs, we describe an adaptation of the ALP method to GMDPs, which was proposed by [9]. In Section 5 we provide an experimental comparison of the behaviour of the MF-API and ALP algorithms on the two illustrative applications, for different problem sizes and graph structures. We discuss more thoroughly the consequences of the main distinction between FMDPs and GMPDs, namely the assumption of a multidimensional action space, in Section 6.

2. Graph-based Markov Decision Processes

In this section we introduce the key elements of the GMDP framework: the notion of locality in transitions, rewards, and policy. We show how two classical management problems, in agronomy and forestry, can be modelled as GMDPs.

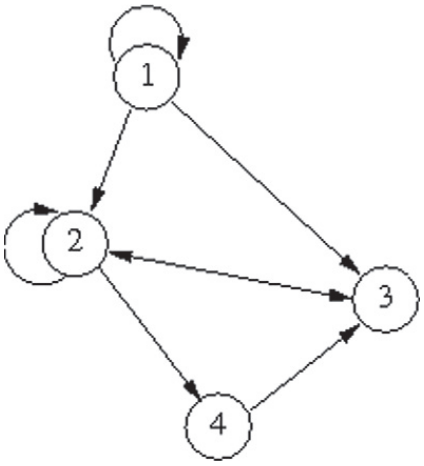

Fig. 1. Graph representation of a GMDP of dimension $n = 4$. $N(1) = \{1\}$, $N(2) = \{1, 2, 3\}$, $N(3) = \{1, 2, 4\}$ and $N(4) = \{2\}$.

2.1. The GMDP model

In its classical formulation [1], an infinite-horizon stationary MDP is described by a four-tuple $\langle \mathcal{X}, \mathcal{A}, p, r \rangle$ where: $\mathcal{X}$ represents the finite set of states that can be reached by the system, $\mathcal{A}$ represents the finite set of actions that can be applied at each time step, p is a state transition function, and r is an "immediate" reward function. The application of an action may change the state of the system and the state $x^{t+1} \in \mathcal{X}$ at time step $t + 1$ depends (stochastically) only on state $x^t \in \mathcal{X}$ at time t, and on the action $a^t \in \mathcal{A}$ applied at t (Markov property). The *stationary* state transition function p can thereby be expressed as $p(x'|x, a)$, the probability that the system will change from state $x^t = x$ to state $x^{t+1} = x'$, given that action $a^t = a$ has been applied. The *reward* obtained when action a is applied in state x is $r(x, a)$.

Let us now consider the situation where the state of the system is described by a vector $x = (x_1, \ldots, x_n)$ of state variable values that are not stochastically independent, and where an action is also described by a vector $a = (a_1, \ldots, a_n)$ of action variable values, of the same size n. In a GMDP the state space is a Cartesian product $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$, and the action space is $\mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_n$. Note that the multidimensional nature of the action space of a GMDP makes it different from and more difficult to solve exactly than a FMDP, in which the state space only is multidimensional. We will come back to this point in more details in Section 4.

In a GMDP, the global transition $p(x|x', a)$ is factored into a product of local transition probabilities $p_i(x'_i|\{x_j, j \in N(i)\}, a_i)$, where $N(i) \in \{1, \ldots, n\}$ is the set of indices of the state variables which actually influence the dynamics of the ith state variable.[1] The set $\{x_j, j \in N(i)\}$ will be denoted $x_{N(i)}$. Then we have the following factorisation assumption:

$$p(x'|x, a) = \prod_{i=1}^{n} p_i(x'_i|x_{N(i)}, a_i), \quad \forall x \in \mathcal{X},\ \forall x' \in \mathcal{X},\ \forall a \in \mathcal{A}.$$

Similarly, in a GMDP the global reward $r(x, a)$ is assumed to be decomposable:

$$r(x, a) = \sum_{i=1}^{n} r_i(x_{N(i)}, a_i), \quad \forall x \in \mathcal{X},\ \forall a \in \mathcal{A}.$$

$r_i(x_{N(i)}, a_i)$ is the local reward obtained when action a_i is performed and the local state of the system is $x_{N(i)}$. For some problems, it can occur that the state variables influencing the dynamics of x_i and the ones involved in r_i are not identical. In these cases, $N(i)$ will be defined as the union of these two sets, without loss of generality.

The sets of indices $1, \ldots, n$ and the relationship $N(i)$ can be represented by an oriented graph $G = (V, E)$, where $V = \{1, \ldots, n\}$ is a set of nodes[2] and $E \subseteq V^2$ is a set of oriented edges. An edge (j, i) from j to i indicates that $j \in N(i)$, i.e. the state variable x_j influences the dynamics of x_i and r_i depends on x_j. G may contain loops, and in particular self-loops (i, i) (see Fig. 1) because the transition of a state variable usually depends on its state at the previous time step. Based on this graphical representation, $N(i)$ will sometimes be referred to as the in-neighbourhood of i. In a GMDP, two variables are attached to a node i of G: the state variable x_i and the action variable a_i. Since a_i only affects the dynamics of x_i, it is unnecessary to represent a_i explicitly in the graph.

Let $\sigma = \max_i |\mathcal{X}_i|$, $\alpha = \max_i |\mathcal{A}_i|$ and $\nu = \max_i |N(i)|$. The spatial complexity of the representation of a GMDP is in $O(n\alpha\sigma^{\nu+1})$. The complexity is linear in the number of nodes and exponential in the maximum in-neighbourhood size, ν. Note that this representation of a structured decision problem is similar to that of [11].

[1] The set $N(i)$ is equivalent to the set of indices of the parents of a state variable in a Dynamic Decision Network [10].

[2] Here, notations for nodes and indices of nodes coincide.

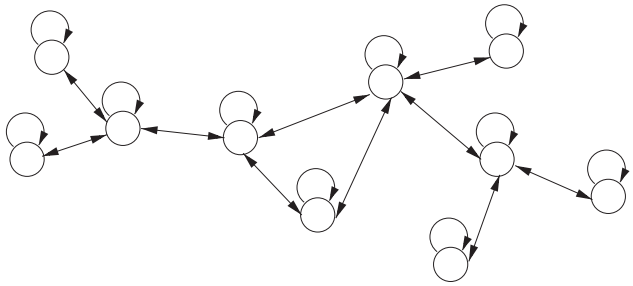

Fig. 2. Graph representing the topology of an agricultural area. There is one node per crop field and edges express the fact that two crop fields share a common border.

Table 1
Local transition probabilities $p(x_i'|x_{N(i)}, a_i)$, for the disease management problem.

	$x_i = 1$	$x_i = 2$	$x_i = 3$	$x_i = 4$
$a_i = 1$				
$x_i' = 1$	$1 - P$	0	0	0
$x_i' = 2$	P	$1 - P$	0	0
$x_i' = 3$	0	P	$1 - P$	0
$x_i' = 4$	0	0	P	1
$a_i = 2$				
$x_i' = 1$	1	q	q/2	q/3
$x_i' = 2$	0	$1 - q$	q/2	q/3
$x_i' = 3$	0	0	$1 - q$	q/3
$x_i' = 4$	0	0	0	1-q

Table 2
Local rewards $r(x_i, a_i)$, for the disease management problem in a crop field.

	$a_i = 1$	$a_i = 2$
$x_i = 1$	r	0
$x_i = 2$	$r/2$	0
$x_i = 3$	$r/3$	0
$x_i = 4$	$r/4$	0

2.2. Example GMDPs

The GMDP framework is particularly well adapted for representing planning problems with spatial features. In these situations, decisions are usually broken down into several spatial decision units and the dynamics of the system are the result of local interaction between neighbouring units. We now present two such examples derived from real-world agricultural problems.

2.2.1. Disease management in crop fields

We illustrate the GMDP framework on a simplified example of disease control in crop fields, a didactic version of a more realistic application of GMDPs to phoma stem canker management on oil seed rape crops [12]. Let us consider a set of crop fields in an agricultural area. Each field is susceptible to contamination by a pathogen. Contamination can either be long range, or over a short distance between neighbouring fields. When a field is contaminated, the yield decreases. Decisions about crop field management are taken each year and two actions ($\mathcal{A}_i = \{1, 2\}$) can be applied to each field: a normal cultural mode ($a_i = 1$) is used or the field is left fallow and treated ($a_i = 2$). The problem is to optimise the choice of a long-term policy in terms of expected yield. The state and action spaces associated with this problem are multidimensional since the problem involves managing several fields in interaction.

The topology of the area is represented by a graph $G = (V, E)$ (Fig. 2). There is one node per crop field. An edge is drawn between two nodes if the fields share a common border. The neighbourhood relationship is symmetric since we assume that infection can spread in any direction. Furthermore, in this example, node i is included in $N(i)$.

Each crop field can be in one of four states: $x_i = 1$ if it is uninfected and $x_i = 2$ to $x_i = 4$ for increasing degrees of infection. Transition probabilities, $p_i(x_i' \mid x_{N(i)}, a_i)$, are parametrised by the following variables: the probability ε of long-distance contamination, the probability p that a field be contaminated by an infected neighbouring field j (with $x_j \geq 2$), and the number n_i of neighbouring fields of node i that are infected. The probability that a field moves from state x_i to state $x_i + 1$ when a normal cultural mode is chosen is equal to $P = P(\varepsilon, p, n_i) = \varepsilon + (1 - \varepsilon)(1 - (1 - p)^{n_i})$. Finally, q denotes the probability that the level of infection of a field decreases when it is left fallow. The corresponding transition probability model is summarised in Table 1. Harvest is affected by the state of the crop field and the chosen action. The maximal yield (r) can be achieved for an uninfected field with normal treatment. Otherwise, the yield decreases with the level of the attack (see Table 2). A field left fallow produces no reward.

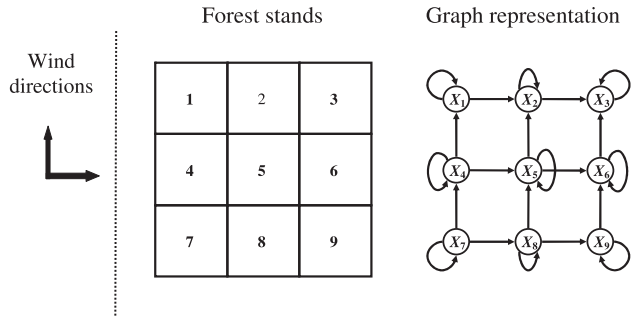

Fig. 3. A GMDP representing the forest management problem under the assumption of limited wind directions. The nodes represent the stands and the oriented edges represent potential wind protection.

Table 3
Local transition function for stand i in the forest management problem.

State	Action	Damage	Next state
x_i	c_1	Not damaged	$x_i' = min(m, x_i + 1)$
x_i	c_2	Not damaged	$x_i' = 1$
x_i	c_1, c_2	Damaged	$x_i' = 1$

2.2.2. Forest management under risk of wind damage

This example is also a didactic version of a more realistic application of GMDPs to forest management [13]. The forest is divided into *stands*, and in this model the state of each stand is represented by the value of a state variable x_i. The state variable, modelled as the age of the trees, describes both the timber value of the stand and the protection from wind that it provides to neighbouring stands. The revenue from a stand, when harvested, also depends on the state. More specifically, the state space of each state variable is the set $\mathcal{X}_i = \{1, 2, \ldots, m\}$, $m \in \mathbb{N}$. If the length of the time period is Y years and the state of the stand i is x_i, then the trees in the stand are between $Y(x_i - 1)$ and $Yx_i - 1$ years old. For simplicity, it is assumed that once a stand is old enough (more than $Y(m - 1)$ years), the state of the stand no longer changes. An action variable a_i represents the action applied locally to a stand. We consider only two management activities: “do not clear-cut" and “clear-cut", denoted by c_1 and c_2: $\mathcal{A}_i = \{c_1, c_2\}$.

For a specific period of time, the probability of a stand being damaged by wind is dependent on the state of the stand and the protection provided by other stands. Indeed, a stand can block the wind and thereby reduce the risk of other stands being damaged. The geographic layout of the forest specifies which stands can provide protection to which. We assume that the wind can only come from limited directions and we use a directed graph $G = (V, E)$ to specify the pattern of potential wind protection (see Fig. 3). There is one node per stand in V and there exists a directed edge $(j, i) \in E$ if and only if stand j can give shelter to stand i.

We assume that damage to the stands due to wind always occurs after a decision has been made concerning which management activities will be performed, and before the decision is implemented. It is also assumed that the state of a damaged stand changes to $x_i' = 1$, whatever the management activity chosen. A time period comprises three steps:

1. Management activities are selected according to the management policy and the current state of the stands.
2. Some stands may be damaged due to wind.
3. Management activities selected in step 1 are implemented. Damaged stands are salvage-harvested.

The wind damage event D_i is stochastic. Its probability depends on the state of neighbours and is defined by the conditional probability $p_i(D_i|x_{N(i)})$. Table 3 shows the possible state transitions given the action chosen and the wind damage event. In order to eliminate the wind-damage-event random variable in the expression of the local reward, we define

$$\hat{r}_i(x_{N(i)}, a_i) = p_i(D_i = 1|x_{N(i)})r_i(x_{N(i)}, a_i, D_i = 1) + p(D_i = 0|x_{N(i)})r_i(x_{N(i)}, a_i, D_i = 0),$$
$$\forall x_{N(i)} \in \mathcal{X}_{N(i)}, \quad \forall a_i \in \mathcal{A}_i,\ i = 1, \ldots, n.$$

This corresponds to the approximation that the wind damage events are independent for all stands, given the states of the neighbouring stands.

2.3. Global and local GMDPs solution policies

In a MDP, a function $\delta : \mathcal{X} \to \mathcal{A}$ assigning an action to every state is called a *stationary policy*. Once a policy δ is fixed, it defines a stationary *Markov process* over $\mathcal{X}$, with transitions $p_\delta(x'|x) = p(x'|x, \delta(x))$. The infinite horizon discounted value

$v_\delta(x)$ of a policy δ, applied to a MDP with initial state x, is defined as:

$$v_\delta(x) = E\left[\sum_{t=0}^{+\infty} \gamma^t r(x^t, \delta(x^t))|x^0 = x\right], \quad \forall x \in \mathcal{X}.$$

The expectation is taken over all possible trajectories $\langle x^0, \delta(x^0), x^1, \ldots, x^t, \delta(x^t), \ldots\rangle$ starting from the initial state x^0 and applying policy δ. The discount factor, $0 \leq \gamma < 1$, ensures that the above infinite sum converges. The problem of finding the optimal policy for a stationary MDP can be written as:

$$\text{Find } \delta^*, \mathcal{X} \to \mathcal{A}, \quad \text{s.t. } v_{\delta*}(x) \geq v_\delta(x), \quad \forall x \in \mathcal{X}, \ \forall \delta \in \mathcal{A}^{\mathcal{X}}.$$

It has been shown that there is always an optimal policy, and that it can be computed in time polynomial in the size of $\mathcal{X}$ and $\mathcal{A}$, using *Stochastic Dynamic Programming* algorithms such as *Policy Iteration*, *Value Iteration* or *Linear Programming* algorithms [1].

However, for structured problems, such as GMDPs, the sizes of the state and action spaces of the underlying MDP are exponential in the number of variables of the GMDP. Thus, the time (and space) complexity of dynamic programming or linear programming algorithms applied to GMDPs is exponential in n. The size of GMDP problems that can be solved efficiently by these methods (as for any factored MDP) is therefore considerably limited.

In order to reduce this complexity, we adopt an approach in which the search space is limited to a subset of policies that make best use of the graph structure. A policy δ in a GMDP can be defined by a set of functions $(\delta_1, \ldots, \delta_n)$, where $\delta_i : \mathcal{X} \to \mathcal{A}_i$. Such policies will be referred to as global policies and may require space in $O(n\sigma^n)$ to be represented. We will consider the particular class of *local policies*: a policy $\delta : \mathcal{X} \to \mathcal{A}$ is said to be *local* if and only if $\delta = (\delta_1, \ldots, \delta_n)$ where $\delta_i : \mathcal{X}_{N(i)} \to \mathcal{A}_i$. The main advantage of local policies is that they can be concisely expressed when ν is small: they only require space in $O(n\sigma^\nu)$ to be represented. However, as local polices only form a subset of the set of global policies, there is no guarantee that an optimal policy of a GMDP exists which is local, even though this result was wrongly claimed in [11]. In Appendix A we present an example of GMDP in which no local policy is optimal. However, in general, GMDPs globally optimal policies cannot be represented efficiently, even for small problems (10–20 nodes). Thus, local policies offer a trade-off between complexity and optimality. We will see later that they can perform well in practice.

3. Mean-field approximate policy iteration algorithm for GMDPs

The *Policy Iteration* (PI) algorithm [1] computes an optimal policy for a MDP. This iterative algorithm alternates two steps, an *evaluation* phase of the current policy and an *improvement* phase of this policy. When the current policy cannot be improved further, its optimality is guaranteed. Generally, this algorithm converges after a small number of iterations, but each iteration is costly since policy evaluation requires resolution of a system of linear equations (complexity $O(|\mathcal{X}|^3)$) and the complexity of the policy improvement phase is $O(|\mathcal{A}|.|\mathcal{X}|^2)$.

When state and action spaces are factored, only approximate evaluation and improvement of the current policy are possible. This leads to the family of *Approximate Policy Iteration* (API) algorithms [7], providing a trade-off between optimality of the result and computational efficiency. The complexity of API algorithms remains in general exponential in the number of action variables, therefore they can hardly be applied to GMDP problems with more than a few tens of nodes. To cope with this limitation, we now present an original algorithm that can be used to compute an approximate solution of any GMDP. The originality of the approach, which was initially proposed in [14], is the approximation of the Markov process induced by a fixed policy by one with a simpler dependence structure, instead of a direct approximation of the value function as would be classically done with API. Namely, we consider the *mean-field* approximation of the process. When applied to the GMDP, the mean-field approximation amounts to an approximation of the Markov chain with n state variables by its best representative among the family of n independent unidimensional, *non-stationnary* Markov chains. We will see that this best representent is derived from the exact spatio-temporal process by fixing the influence of neighbouring state variables to its mean value, thus preserving some part of the spatial information. Furthermore, the approximate transition function is time dependent, in order to reflect influence propagation between neighbours. Applying this principle enables us to derive an approximate solution algorithm for GMDPs: the *MF-API algorithm*.

3.1. Expressing the occupation measure of a local policy value function

For an infinite-horizon problem the value function v_δ of policy δ is defined as:

$$v_\delta(x) = E\left[\sum_{t=0}^{+\infty} \gamma^t r(x^t, \delta(x^t))|x^0 = x\right], \quad \forall x \in \mathcal{X}.$$

The value of a policy can be computed by an iterative *successive approximations* algorithm [1]. It can also be computed by methods based on the *occupation measure* [15]. If $\delta : \mathcal{X} \to \mathcal{A}$ is a policy and $x \in \mathcal{X}$ an initial state, the occupation measure $P_{x,\delta} : \mathcal{X} \to [0, 1]$ is a probability distribution defined by:

$$P_{x,\delta}(y) = (1-\gamma)\sum_{t=0}^{+\infty}\gamma^t P_\delta(X^t = y|X^0 = x), \quad \forall y \in \mathcal{X}.$$

The conditional probability $P_\delta(X^t = y|X^0 = x)$ is the probability that $X^t = y$, given that $X^0 = x$ and policy δ is applied. It can be evaluated from the transition probabilities as follows

$$P_\delta(X^t = x^t|X^0 = x^0) = \sum_{x^1,\ldots,x^{t-1}}\prod_{t'=0}^{t-1} p(x^{t'+1}|x^{t'}, \delta(x^{t'})).$$

Intuitively (ignoring the discount factor γ), $P_{x,\delta}$ measures the proportion of time spent in state y by the process when starting in state x and applying δ. The value function v_δ of any policy can be computed from the occupation measure $P_{x,\delta}$ as follows:

$$v_\delta(x) = \frac{1}{1-\gamma}\sum_{y\in\mathcal{X}} P_{x,\delta}(y) r(y, \delta(y)), \quad \forall \delta : \mathcal{X} \to \mathcal{A}, \forall x \in \mathcal{X}.$$

This expression of the value function avoids having to solve a fixed-point equation, but the difficulty is to compute the occupation measure $P_{x,\delta}(y)$. The exact computation of the conditional distribution $P_\delta(X^t = y|X^0 = x)$ is too difficult. Let us consider the specific case of a local policy $\delta = \{\delta_1, \ldots, \delta_n\}$. The value of δ can be decomposed exactly as follows:

$$v_\delta(x) = \sum_{i=1}^{n} v_\delta^i(x), \quad \forall x \in \mathcal{X},$$

where

$$v_\delta^i(x) = \sum_{t=0}^{+\infty}\gamma^t \sum_{y\in\mathcal{X}} P_\delta(X^t = y|X^0 = x) r_i(y_{N(i)}, \delta_i(y_{N(i)})), \quad \forall x \in \mathcal{X},\ i = 1, \ldots, n. \tag{1}$$

This decomposition takes n times as much space to represent than the exact value function, and as such is not appropriate for the efficient computation of optimal policies. However, we propose to look for an approximation Q_δ^t of the conditional probability $P_\delta(X^t = y|X^0 = x)$ which has a simple structure and will allow an approximation of $v_\delta^i(x)$ by a function $\hat{v}_\delta^i$ depending only on $x_{N(i)}$. We will establish that if Q_δ^t is the mean-field approximation, we get $v_\delta^i(x) \approx \hat{v}_\delta^i(x_{N(i)})$, where:

$$\hat{v}_\delta^i(x_{N(i)}) = \sum_{t=0}^{+\infty}\gamma^t \sum_{y_{N(i)}}\left(r_i(y_{N(i)}, \delta_i(y_{N(i)})) \prod_{j\in N(i)} Q_\delta^{t,j}(y_j|x_j)\right).$$

3.2. Mean-field approximation of a structured Markov chain

The main idea of the mean-field principle is to approximate a system of interacting variables by a system of independent ones, by approximating the influence of the neighbourhood by its mean value. The mean-field approximation of a multidimensional distribution $P(u_1, \ldots, u_m)$ is defined as the joint distribution Q of m *independent* variables $u_1, \ldots, u_m$, which is as close to P as possible, according to the Kullback–Leibler divergence, $KL(Q|P) = E_Q[\log(Q/P)]$. $KL(Q|P)$ is a positive quantity [3] and is equal to zero if and only if $P = Q$. The minimum of $KL(Q|P)$ is sought among joint distributions Q satisfying $Q(u_1, \ldots, u_m) = \prod_{i=1}^m Q_i(u_i)$.

Within the framework of GMDPs, the difficulty in computing the conditional distribution $P_\delta(X^t = y|X^0 = x)$ comes from the spatio-temporal dependence between nodes. Indeed, for a given policy δ, each variable X_i^t depends on the state of all neighbour variables at the previous time step, $X_{N(i)}^{t-1}$. If t is large enough, X_i^t may depend on all variables X_i^0. We propose to limit this influence by applying the mean-field principle to the joint probability distribution of X^{t-1} and X^t, fully determined by the distribution of X^{t-1}, $P_\delta(X^{t-1} = x)$, and the transition probability of X^t given X^{t-1}, $p_i(x_i^t \mid x_{N(i)}^{t-1}, \delta_i(x_{N(i)}^{t-1}))$.

The approximation procedure is iterative. In the following, P will denote the exact distribution and Q a distribution of n independent random variables. Let us first consider the joint probability at time $t = 1$. Since usually no *a priori* information is available, we will assume that at $t = 0$ the state variables X_i^0 are independent (for the sake of simplicity but also to ensure the repeatability of the approximation method from one time step to the next). Exact and approximate distributions have the following form:

$$P^0(x^0) = \prod_{i=1}^{n} P^{0,i}(x_i^0), \quad p_\delta(x^1|x^0) = \prod_{i=1}^{n} p_i\left(x_i^1|x_{N(i)}^0, \delta_i\left(x_{N(i)}^0\right)\right),$$

[3] $KL(P|Q) = E_P[\log(P/Q)]$ is also well defined but is not considered for building the mean-field approximation since its evaluation is intractable.

$$Q_\delta^0(x^0) = P^0(x^0), \quad q_\delta^1(x^1|x^0) = \prod_{i=1}^{n} q_\delta^{1,i}(x_i^1|x_i^0). \tag{2}$$

In the approximate model, X_i^1 only depends on X_i^0. Let us denote by $\mathcal{Q}$ the set of all joint distributions of X^{t-1} and X^t that can be decomposed according to (2). Among all the elements in $\mathcal{Q}$, we need to look for the closest one to the exact joint distribution, according to the Kullback–Leibler divergence. Since $Q_\delta^0 = P^0$, the only approximation in the computation of the joint distribution arises from the computation of q_δ^1.

Proposition 1 (Mean-field approximation of the conditional distribution)**.** *The mean-field approximation q_δ^1 defined as the solution of*

$$q_\delta^1 = \arg\min_{q_\delta^1 \in \mathcal{Q}} KL\Big(q_\delta^1(X^1|X^0)P^0(X^0) \mid p_\delta(X^1|X^0)P^0(X^0)\Big), \text{ is}$$

$$q_\delta^{1,i}(x_i^1|x_i^0) \propto \exp\left(E_{P^0}\Big[\log p_i\Big(x_i^1|x_i^0, X^0_{N(i)\setminus i}, \delta_i\Big(x_i^0, X^0_{N(i)\setminus i}\Big)\Big)\Big]\right) \quad \forall x_i^0 \in \mathcal{X}_i, x_i^1 \in \mathcal{X}_i, i = 1, \ldots, n. \tag{3}$$

The expectation is taken over the distribution of $X^0_{N(i)\setminus i}$. The proof is in Appendix B.

Expression (3) is not normalized. The classical procedure is instead to consider the simplified and normalized version obtained by switching expectation and logarithm operators:

$$q_\delta^{1,i}(x_i^1|x_i^0) = E_{P^0}\Big[p_i\Big(x_i^1|x_i^0, X^0_{N(i)\setminus i}, \delta_i\Big(x_i^0, X^0_{N(i)\setminus i}\Big)\Big)\Big].$$

We will use this mean-field approximation of the transition probabilities in the following. This approximation is easy to interpret: the transition probability is replaced with the expectation of the transition probabilities for all possible configurations of the neighbourhood. Given the approximate transition probabilities between time 0 and time 1, and given the initial joint distribution P^0, we can derive the mean-field approximation of the probability distribution of X^1 for a given policy δ:

$$Q_\delta^1(x^1) = \sum_{x^0} q_\delta^1\big(x^1|x^0\big)P^0(x^0) = \prod_{i=1}^{n}\left(\sum_{x_i^0} q_\delta^{1,i}|\big(x_i^1|x_i^0\big)P^{0,i}\big(x_i^0\big)\right).$$

The distribution at time 1, under the mean-field approximation, is also a distribution of n independent variables. Thus, we can iterate the procedure (minimisation of the Kullback–Leibler divergence) and obtain for each time step $t \geq 2$ a mean-field approximation of the transition probabilities:

$$q_\delta^{t,i}\big(x_i^t|x_i^{t-1}\big) = E_{Q_\delta^{t-1}}\Big[p_i\Big(x_i^t|x_i^{t-1},\ X^{t-1}_{N(i)\setminus i},\ \delta_i\Big(x_i^{t-1}, X^{t-1}_{N(i)\setminus i}\Big)\Big)\Big]. \tag{4}$$

One should note that the approximate transition probabilities $q_\delta^t(x^t|x^{t-1})$ depend on t while the exact ones do not. Finally, for all $t \geq 2$, a mean-field approximation of the conditional probability of X^t, given $X^0 = x^0$ and δ, can be obtained as:

$$\begin{aligned} Q_\delta^t\big(x^t|x^0\big) &= \sum_{x^{t-1}} q_\delta^t\big(x^t|x^{t-1}\big)Q_\delta^{t-1}\big(x^{t-1}|x^0\big) \\ &= \prod_{i=1}^{n}\left(\sum_{x_i^{t-1}} q_\delta^{t,i}(x_i^t|x_i^{t-1})Q_\delta^{t-1,i}\big(x_i^{t-1}|x_i^0\big)\right). \end{aligned}$$

The result is an expression of the form

$$Q_\delta^t\big(x^t|x^0\big) = \prod_{i=1}^{n} Q_\delta^{t,i}\big(x_i^t|x_i^0\big), \quad \forall t.$$

with, by definition,

$$Q_\delta^{t,i}\big(x_i^t|x_i^0\big) = \sum_{x_i^{t-1}} q_\delta^{t,i}\big(x_i^t|x_i^{t-1}\big)Q_\delta^{t-1,i}\big(x_i^{t-1}|x_i^0\big).$$

Exact and approximate expressions of the different probability distributions involved here are summarised in Table 4. The recursive definition is initialised with $Q_\delta^1(x^1|x^0) = q_\delta^1(x^1|x^0)$. This procedure provides an approximation of the conditional probability $P_\delta(X^t = y|X^0 = x)$, which simplifies the evaluation of (1).

Table 4
Mean-field approximations of the conditional, transition and joint probabilities at time t of a GMDP.

	GMDP	Approximation
Conditional probability	$P_\delta\left(X^t = x^t \mid X^0 = x^0\right)$	$\prod_{i=1}^n Q_\delta^{t,i}\left(x_i^t \mid x_i^0\right)$
Transition probabilities	$p_i\left(x_i^t \mid x_{N(i)}^{t-1}, \delta\left(x_{N(i)}^{t-1}\right)\right)$	$q_\delta^{t,i}\left(x_i^t \mid x_i^{t-1}\right)$
Joint probability	$P_\delta\left(X^t = x^t\right)$	$\prod_{i=1}^n Q_\delta^{t,i}\left(x_i^t\right)$

Data : $\{G = (V, E), \{p_i(x_i^t \mid x_{N(i)}^{t-1}, a_i)\}, \{r_i(x_{N(i)}, a_i)\}, \{\delta_i(x_{N(i)})\}, P^0, \gamma, T_{max}\}$

Result : $\{\hat{v}_\delta^i(x_{N(i)})\}$

% Initialization ;
compute $p_\delta = \{p_i(x_i^t \mid x_{N(i)}^{t-1}, \delta(x_{N(i)}^{t-1}))\}$;
$Q_\delta^0 = P^0$;
for *all nodes i, all states* $x_{N(i)}^0$ **do**
  $\hat{v}_\delta^i(x_{N(i)}^0) = 0$;
end
$t = 1$;
% Main loop (over time) ;
while $t \leq T_{max}$ **do**
  % Computation of MF transition, conditional and joint probabilities ;
  for *all nodes i, all states* x_i^t, x_i^{t-1}, x_i^0 **do**
    $q_\delta^{t,i}(x_i^t \mid x_i^{t-1}) = \text{pseudominKL}(Q_\delta^{t-1}, p_\delta)$;
    if *t=1* **then** $Q_\delta^{1,i}(x_i^1 \mid x_i^0) = q_\delta^{1,i}(x_i^1 \mid x_i^0)$ **else**
    $Q_\delta^{t,i}(x_i^t \mid x_i^0) = \sum_{x_i^{t-1}} q_\delta^{t,i}(x_i^t \mid x_i^{t-1}) Q_\delta^{t-1,i}(x_i^{t-1} \mid x_i^0)$;
    $Q_\delta^{t,i}(x_i^t) = \sum_{x_i^0} Q_\delta^{t,i}(x_i^t \mid x_i^0) Q_\delta^{0,i}(x_i^0)$;
  end
  % Update of MF value function ;
  for *all nodes i, all states* $x_{N(i)}^0$ **do**
    $\hat{v}_\delta^i(x_{N(i)}^0) = \hat{v}_\delta^i(x_{N(i)}^0) + \gamma^t \sum_{x_{N(i)}^t} \left(r_i(x_{N(i)}^t, \delta_i(x_{N(i)}^t)) \prod_{j \in N(i)} Q_\delta^{t,j}(x_j^t | x_j^0)\right)$;
  end
  $t = t + 1$;
end

Algorithm 1: Pseudo code for mean-field policy evaluation.

3.3. Mean-field approximate policy evaluation

The mean-field approximation can now be exploited to perform the evaluation step of the MF-API algorithm. An approximation of each term $v_\delta^i(x)$ of the value function is derived in a straightforward way:

$$v_\delta^i(x) \approx \hat{v}_\delta^i(x_{N(i)}) = \sum_{t=0}^{+\infty} \gamma^t \sum_{y_{N(i)}} \left(r_i(y_{N(i)}, \delta_i(y_{N(i)})) \prod_{j \in N(i)} Q_\delta^{t,j}(y_j | x_j) \right).$$

The important point is that each term $\hat{v}_\delta^i$ only depends on $x_{N(i)}$.

Proposition 2 (Complexity of approximate policy evaluation). *The complexity of the evaluation of* $\hat{v}_\delta^i$ *is* $O\left(\log(1/\epsilon) n \sigma^{3\nu+1} \nu^2\right)$, *where* ϵ *is the required precision for the computation of* $\hat{v}_\delta^i$.

See Appendix C for a proof. Algorithm 1 provides the pseudo code of the mean-field policy evaluation step. Note that the infinite sum $\sum_{t=0}^{\infty}$ is replaced with a finite sume $\sum_{t=0}^{T_{max}}$, where T_{max} is an input parameter. By increasing T_{max}, the finite sum can be made arbitrarily close to the infinite sum (the infinite sum $\sum_{t=T_{max}+1}^{\infty}$ can be easily upper-bounded). In this algorithm the function pseudominKL evaluates (4).

3.4. Mean-field approximate Policy Improvement

The *improvement* phase of the policy iteration algorithm improves the current policy δ into a policy δ' by applying for all $x \in \mathcal{X}$:

$$\delta'(x) = \arg\max_{a \in \mathcal{A}} \left\{ r(x,a) + \gamma \sum_{x' \in \mathcal{X}} p(x'|x,a) v_\delta(x') \right\}.$$

It is guaranteed that $v_{\delta'}(x) \geq v_\delta(x)$, $\forall x \in \mathcal{X}$ after the improvement phase and that δ is optimal once $v_{\delta'}(x) = v_\delta(x)$, $\forall x \in \mathcal{X}$ [1].

We propose an approximate policy improvement step in order to reduce its complexity. To restrict the search space, we limit the search for approximately optimal policies to local ones. This allows a reduction in space and time complexity. Using the mean-field approximation, $v_\delta(y) = \sum_{i \in V} \hat{v}^i_\delta(y_{N(i)})$ and the local properties of rewards and transition probabilities, the exact maximisation can be replaced with the maximisation of the following expression:

$$\delta'(x) = \arg\max_{a \in \mathcal{A}} \left\{ \sum_i r_i(x_{N(i)}, a_i) + \gamma \sum_{y_{N(i)}} \Big(\prod_{j \in N(i)} p_j(y_j|x_{N(j)}, a_j) \hat{v}^i_\delta(y_{N(i)}) \Big) \right\}.$$

Let us consider a particular node i. The terms which depend on a_i are

$$r_i(x_{N(i)}, a_i) + \gamma \sum_{k \in N^{-1}(i)} \sum_{y_{N(k)}} \Big(\prod_{j \in N(k)} p_j(y_j|x_{N(j)}, a_j) \hat{v}^k_\delta(y_{N(k)}) \Big).$$

Thus, even if δ is local, the resulting policy δ' after improvement is no longer local. δ'_i is a function of all the variables associated with the nodes in $N^2(N^{-1}(i)) \cup N(i)$ [4] because of the terms

$$p_k\Big(y_{N(k)}|x_{N(N(k))}, a_{N(k)}\Big) = \prod_{j \in N(k)} p_j(y_j|x_{N(j)}, a_j), \quad \forall k \in N^{-1}(i)$$

in the update formula. That is,

$$\delta'(x) = \arg\max_{a \in \mathcal{A}} \left\{ \sum_i \phi_i\Big(x_{N^2(N^{-1}(i)) \cup N(i)}, a_{N(N^{-1}(i)) \cup \{i\}}\Big) \right\}.$$

This maximisation is hard to perform exactly and, more importantly, the resulting policy is no longer local. In order to compute a local policy, we approximate all terms $p_k(y_{N(k)}|x_{N(N(k))}, a_{N(k)})$ by terms $\hat{p}_k(y_{N(k)}|x_{N(i)}, a_i)$. First, we adopt a parallel scheme, in order to compute each component δ'_i of the updated policy. Then, the idea behind the approximate improvement phase is again to use an approximation of each p_k, which averages over undesirable variables in $N^2(N^{-1}(i))$ (those that are not in $N(i)$). For a given $k \in N^{-1}(i)$ and $j \in N(k)$, if $j = i$, $p_j\Big(y_j|x_{N(j)}, a_j\Big)$ is taken exactly. If $j \in N(i) \cap N(k)$, $j \neq i$, $p_j\Big(y_j|x_{N(j)}, a_j\Big)$ is approximated by

$$\hat{p}_j(y_j|x_j, \delta_j) = \frac{\sum_{x_{N(j)\setminus j}} p_j(y_j|x_{N(j)}, \delta_j(x_{N(j)}))}{card(\mathcal{X}_{N(j)\setminus j})},$$

where the function *card* counts the number of elements of a set. If $j \in N(k) \cap \overline{N(i)}$, $j \neq i$, the same transition probability is approximated by

$$\hat{p}_j(y_j|\delta_j) = \frac{\sum_{x_{N(j)}} p_j(y_j|x_{N(j)}, \delta_j(x_{N(j)}))}{card(\mathcal{X}_j)}.$$

Finally, $\forall k \in N^{-1}(i)$, $p_k(y_{N(k)}|x_{N(N(k))}, a_{N(k)})$ is replaced with

$$\hat{p}_k(y_{N(k)}|x_{N(i)}, a_i) = p_i(y_i|x_{N(i)}, a_i) \left(\prod_{j \in N(i) \cap N(k), j \neq i} \hat{p}_j(y_j|x_j, \delta_j) \right) \left(\prod_{j \in N(k) \cap \overline{N(i)}, j \neq i} \hat{p}_j(y_j|\delta_j) \right). \tag{5}$$

Proposition 3 (Complexity of Approximate Policy Improvement)**.** *The Approximate Policy Update step is of complexity* $O(n^2 \alpha \nu \sigma^{2\nu})$.

[4] Where $N^{-1}(i) = \{j, i \in N(j)\}$ and $N^2(i) = N(N(i))$.

Data : $\{G = (V, E), \{p_i(x_i^t \mid x_{N(i)}^{t-1}, a_i)\}, \{r_i(x_{N(i)}, a_i)\}, \{\delta_i^{old}(x_{N(i)})\}, \gamma, S_{max}\}$

Result : $\{\delta_i^{new}(x_{N(i)})\}$

% Iterative and parallel updates of all δ_i ;

for *step* $= 1$ *to step* $= S_{max}$ **do**

  % Update of δ_i ;

  for *all nodes i, all states* $x_{N(i)}$ **do**

    % Approximation of $p_k(y_{N(k)}|x_{N(N(k))}, a_{N(k)})$;

    for *all* $k \in N^{-1}(i)$ **do**

      for *all* $j \in N(k)$ **do**

        if $j \in N(k) \cap N(i), j \neq i$ **then**

          Compute $\hat{p}_j(y_j|x_j, \delta_j)$;

        end

        if $j \in N(k) \cap \overline{N(i)}, j \neq i$ **then**

          Compute $\hat{p}_j(y_j|\delta_j)$;

        end

      end

      $\hat{p}_k(y_{N(k)}|x_{N(i)}, a_i) = \mathcal{F}(p_i, \{\hat{p}_j\}, a_i)$;

    end

    $\delta_i^{temp}(x_{N(i)}) = \arg\max_{a_i} \mathcal{H}(r_i, \hat{v}_\delta, \{\hat{p}_k\}, x_{N(i)}, a_i)$;

  end

  if $(\delta^{temp} = \delta^{old})$ **then break else** $\delta^{old} = \delta^{temp}$;

end

$\delta^{new} = \delta^{temp}$;

Algorithm 2: Pseudo code for mean-field policy improvement.

See Appendix C for a proof. Algorithm 2 provides the pseudo code of the mean-field policy improvement step. Since API algorithms are not guaranteed to converge in general (they may cycle over policies), the number of updates is upper bounded by S_{max}. In this way, termination of the algorithm is ensured. In the algorithm, function $\mathcal{F}$ evaluates (5) and function $\mathcal{H}$ is defined as

$$\mathcal{H}(r_i, \hat{v}_\delta, \{\hat{p}_k\}, x_{N(i)}, a_i) = r_i(x_{N(i)}, a_i) + \gamma \sum_{k \in N^{-1}(i)} \sum_{y_{N(k)}} \left(\hat{p}_k(y_{N(k)}|x_{N(i)}, a_i)\hat{v}_\delta^k(y_{N(k)})\right).$$

3.5. Mean-field approximate Policy Iteration

The mean-field approximate policy iteration algorithm can now be defined by the alternation of the approximate evaluation and update phases. Note that, as for most API algorithms, there is no guarantee that the improvement phase, limited here to local policies, actually improves the current policy everywhere. A local policy dominating all local policies may not even exist. The stopping criterion in the MF-API algorithm is the equality of two successive policies, with a maximum number of iterations allowed, to avoid possibly infinite cycles over local policies.

Overall, the MF-API algorithm comes with no theoretical guarantee on the performance of the computed policies, relative to the optimal, global or local, policy of the GMDP. This is the classical drawback of mean-field methods, which are nevertheless largely and successfully used, due to their good empirical behavior. As shown in Appendix A, there is no hope to get any performance guarantee relative to the global optimal policy: We can build pathological GMDPs instances for which the optimal local policy performance is arbitrarily far from that of the optimal global policy. One question which remains open is whether we can provide bounds on the quality of the policies computed by the MF-API algorithm, relative to optimal local policies.[5] While this question has no answer yet, the experiments we performed on realistic problems (Section 5) show that the MF-API approach performs well. For the considered problems, even though we could not compute the exact value of the global optimal policy, we could compute an upper bound on this value, and we checked empirically, by Monte-Carlo simulation, that the expected value of the MF-API policies remained close to this upper-bound, thus giving empirical evidence of the performance of the algorithm.

[5] Since there may not exist a local policy which dominates all other local policies in every starting states, the notion of optimality would be defined with respect to a single starting state.

Table 5
Local transition probabilities $p(x_i'|x_{N(i)}, a_i)$, for the problem of disease management in a crop field.

	$x_i = 1$	$x_i = 2$
$a_i = 1$		
$x_i' = 1$	$1 - P$	0
$x_i' = 2$	P	1
$a_i = 2$		
$x_i' = 1$	1	q
$x_i' = 2$	0	$1 - q$

Table 6
Local rewards $r(x_i, a_i)$, for the problem of disease management in a crop field.

	$a_i = 1$	$a_i = 2$
$x_i = 1$	r	0
$x_i = 2$	$r/2$	0

4. Applying FMDP algorithms to GMDPs

The question we deal with in this section is the feasibility of designing FMDP-inspired exact and approximate solution algorithms for GMDPs. There are two main families of algorithms for solving Factored MDPs[6]:

- Exact or approximate approaches, based on dynamic programming algorithms manipulating factored representations of value functions and policies, either as trees [2] or as *Algebraic Decision Diagrams* [3,4].
- *Approximate Linear Programming* approaches, based on the use of parameterised value functions in MDP solution algorithms [5,17]. In this second family [18,9] have extended the approach to the case of factored action spaces.

In general, the usual solution methods for FMDPs fail for solving GMDPs because of their assumption of flat action space. In the following, we first describe an exact solution approach for GMDPs, based on a *Algebraic Decision Diagrams* (ADD) and implemented in the software SPUDD [3]. This approach is the state-of-the art method from the first family. We show that this approach is only applicable to GMDPs of very small size (less than 10 nodes). We then describe a second approach, proposed by Forsell and Sabbadin [9], based on *Approximate Linear Programming* (ALP). This second approach, which provides approximate solution policies, is applicable to GMDP problems of very large size. In Section 5, we will propose an empirical comparison of the ALP approach with the MF-API approach described in Section 3. Finally, in Section 6 we will specifically discuss the multidimensional action space hypothesis.

4.1. Stochastic programming using decision diagrams

The first family of approaches we consider use concise tree-structured (or ADD-structured) representations of the transition and reward functions of the FMDP for computing more compact policies and value functions (these can still be of exponential size, though). *Structured policy iteration* or *structured value iteration algorithms* [2] can be used to compute tree-structured solutions of a FMDP. The SPUDD [3] and APPRICOD [4] approaches implement respectively the ADD exact and approximate solution approaches. A limit of these approaches is that they handle only flat action space, while the action space of a GMDP is multidimensional. However, it remains possible to apply SPUDD or APPRICOD to GMDPs by using, for each action, a decision diagram description of the transitions and rewards. The SPUDD/APPRICOD representation grows exponentially in the number of vertices, while the corresponding GMDP description is concise.

In order to evaluate this approach, we applied SPUDD to a simplified version of the disease management example, with $\mathcal{A}_i = \{1, 2\}$ and $\mathcal{X}_i = \{1, 2\}$, where $x_i = 1$ means “uninfected" and $x_i = 2$ means “infected". The corresponding local transitions and rewards are expressed in Tables 5 and 6. They can be expressed in decision diagram form (Fig. 4).

Table 7 shows the size of the optimal (tree- and ADD-structured) policies computed by SPUDD for grids of size 2×1 to 3×3, as well as the corresponding computation times.

Note that for a grid problem of size 3×3, the action space size limit of SPUDD (256 actions) is exceeded. Even when this limit is increased, the size of the computed decision trees exceeds the available memory after few iterations of the algorithm. We therefore had to limit the action space to 256 actions (by arbitrarily fixing $a_1 = 1$) to obtain a problem which could be solved using SPUDD. So, clearly, the (exact) structured dynamic programming approach does not scale to the size of problems we wish to tackle.

One could argue that approximate solution methods based on similar principles, such as the APPRICOD approach [4] could allow to solve larger problems, albeit approximately. This is only partially true, however. The APRICODD algorithm handles *ADD approximations* of the value function of a factored MDP, within a value iteration algorithm. These ADD approximations have a reasonable size, which allows the application of the approach to MDPs with factored state spaces. However, the value

[6] Here, we only mention references to Factored MDPs solution methods, and deliberately omit *multiagent* resolution approaches developped for the close framework of *DEC-POMDPs* (see [16] for a recent reference on this family of approaches).

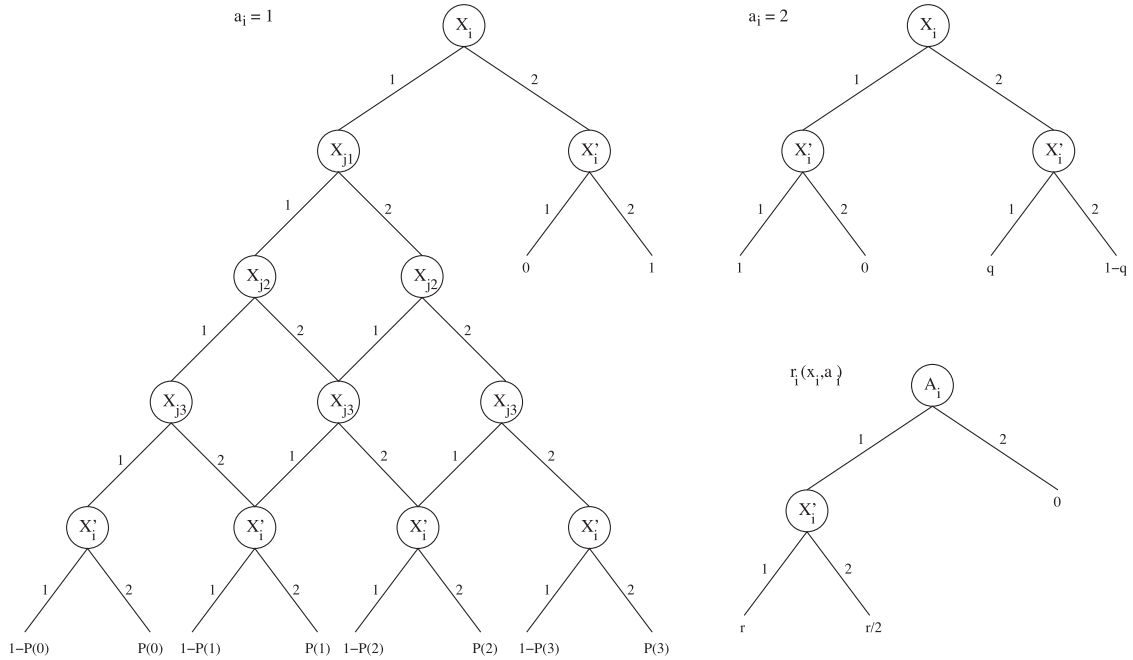

Fig. 4. Transitions and rewards in decision diagram form for a vertex i with three neighbours, j_1, j_2 and j_3. $P(k)$ is the probability that uninfected vertex i becomes infected when it has k infected neighbours.

Table 7
Optimal ADD policy representation size and computation time for problems with up to 9 vertices.

Grid size	2×1	2×2	3×2	3×3
|Action tree|	5	17	73	521
|Action ADD|	4	13	47	205
|Value tree|	31	511	8191	524287
|Value ADD|	25	281	2407	36836
Time (in s)	0.03	0.71	58.4	262.01

iteration algorithm updates involve a *maximum* operation over *all* elementary actions. In the GMDP framework, the action space is factored, meaning that the set $\mathcal{A}$ has exponential size (in n). Thus, the time complexity of APPRICOD would remain exponential in n, even though its space complexity could be kept polynomial, at the price of an approximation of the optimal policy. We therefore turned to a second approach, based on linear programming.

4.2. An approximate linear programming approach for GMDPs

In the ALP approach, the optimal value function of the problem is approximated by a linear combination of basis functions, each of which is of smaller scope than the optimal value function. The manipulation of a value function has thereby been transformed into the problem of computing weights of the different basis functions. Guestrin et al. [5] built on this idea and presented two approximate solution algorithms: one algorithm is based on linear programming and the other is based on dynamic programming. However, the running time of the algorithms and the quality of the computed solution are dependent on the basis functions, which are fixed and predefined by the designer. Poupart et al. [19] proposed an approach that automatically selects and modifies the basis functions. These algorithms, however, mainly focus on exploiting the factorisation of the state space of the MDP and do not consider the case where the action space is factored. Some algorithms based on the ALP approach have been suggested for the computation of policies for *collaborative multiagent MDPs*, in which the action space is also factored. Guestrin et al. [20] devised a model-dependent ALP approach, based on the same principles as [5] for computing approximate policies. The number and structure of the basis functions are selected by the user, and the solution and running time of the algorithm is thereby dependent on this selection. de Farias and Van Roy [17] proposed a constraint-sampling approach for handling problems with a large action space. However, they noted that for the algorithm to perform efficiently, it might need quite a large number of basis functions.

Forsell and Sabbadin [9] have shown how the structure of GMDPs can be used to design an efficient Approximate Linear Programming (ALP) algorithm. The model is inspired from and very close to that of [20,5], the GMDP structure determining the choice of the features of the ALP model. The chosen approach approximates the value function as a sum of n local value functions, one for each node of the graph G. The local value function v_i depends on x_i only, when it depends on $x_{N(i)}$ in the MF-API approach. Each local value function is computed independently by solving a LP system of small size. In the following section we briefly describe the approach adopted by Forsell and Sabbadin [9]. In the experimental study section, it will be compared to the mean-field approach.

4.2.1. Parameterised value functions

The *Approximate Linear Programming* (ALP) approach [6,17] uses a *parametrisation* v_w of the value function to compute an approximation of the value function of the optimal policy v_{δ^*}. The underlying idea is to define v_w as a linear combination of a set of predefined basis functions $H = \{h_1, \ldots, h_k\}$:

$$v_w(x) = \sum_{i=1}^{k} w_i h_i(x), \quad \forall x \in \mathcal{X},$$

where $h_i : \mathcal{X} \to \mathbb{R}$ and $w_i \in \mathbb{R}$. Solving a MDP can be seen as the problem of selecting the value of the parameters $w = (w_1, \ldots, w_k)$ such that the parameterised value function v_w approximates the optimal value function as closely as possible:

$$\text{Find } w^* = \arg\min_{w \in W} ||v_{\delta^*} - v_w||_\infty,$$

where $w^* = (w_1^*, \ldots, w_k^*)$, $W = \mathbb{R}^k$. Now, let ε_w denote the worst-case distance in quality between v_{δ^*} and v_w:

$$\varepsilon_w = ||v_{\delta^*} - v_w||_\infty = \max_{x \in \mathcal{X}} |v_{\delta^*}(x) - v_w(x)|.$$

Solving a MDP comes down to the problem of selecting the value of w, such that ε_w is as small as possible. However, as v_{δ^*} is unknown, it is impossible to compute w^* in this way without first solving the MDP. Instead, an upper bound of ε_w can be used [21].

$$\varepsilon_w = \max_{x \in \mathcal{X}} |v_{\delta^*}(x) - v_w(x)| \leq \beta = \frac{2\gamma}{1-\gamma}\Big(\max_{x \in \mathcal{X}} |v_w(x) - \mathcal{B}(v_w)(x)|\Big).$$

where $\mathcal{B}$ is the *Bellman operator*, defined as:

$$\mathcal{B}(v)(x) = \max_{a \in \mathcal{A}} \Big\{ r(x,a) + \gamma \sum_{x' \in \mathcal{X}} p(x'|x,a) v(x') \Big\}, \quad \forall x \in \mathcal{X}, \quad \forall v : \mathcal{X} \to \mathbb{R}.$$

The problem of selecting the value of w that minimises ε_w can then be solved approximately by solving the following non-linear programming problem (NLP):

$$\begin{array}{lll} \min & \varepsilon & \\ \text{s.t.} & \varepsilon \geq v_w(x) - \mathcal{B}(v_w)(x) & \forall x \in \mathcal{X} \\ & \varepsilon \geq \mathcal{B}(v_w)(x) - v_w(x) & \forall x \in \mathcal{X} \\ & w = (w_1, \ldots, w_k) \text{ free}, \ \varepsilon \text{free}. & \end{array} \tag{6}$$

4.2.2. ALP model in the GMDP case

In [9], the ALP approach was applied in order to provide approximate solution policies for GMDPs. The method is based on the following choice of basis functions: $H = \{h_{11}, \ldots, h_{n|\mathcal{X}_n|}\}$, where:

$$h_{ij}(x_i) = \begin{cases} 1, & \text{if } x_i = j, \\ 0, & \text{otherwise,} \end{cases}$$

where $i = 1, \ldots, n$, and $j = 1, \ldots, |\mathcal{X}_i|$. There are thus $\sum_{i=1}^{n} |\mathcal{X}_i|$ basis functions.

With this choice of basis functions, the Bellman operator can be decomposed as follows:

$$\mathcal{B}(v_w)(x) = \sum_{i=1}^{n} \mathcal{B}_i(v_w^i)(x_{N(i)}), \quad \forall x \in \mathcal{X}, \ \text{ where}$$

$$\mathcal{B}_i(v_w^i)(x_{N(i)}) = \max_{a_i \in \mathcal{A}_i} \Big\{ r_i(x_{N(i)}, a_i) + \gamma \sum_{x_i' \in \mathcal{X}_i} p_i(x_i'|x_{N(i)}, a_i) v_w^i(x_i') \Big\}, \quad \forall x_{N(i)} \in \mathcal{X}_{N(i)}.$$

However, even with this choice, problem (6) remains non-linear and still has a number of constraints which is exponential in the number of variables. Using two further relaxations enables to compute approximate local policies by solving n independent linear programs, one for each vertex, with $|\mathcal{X}_i| + 1$ variables and $|\mathcal{A}_i| \times |\mathcal{X}_{N(i)}|$ constraints. The author is referred to [9] for more details about the method. Note that the approximation is based on the fact that basis functions h_{ij}

have scopes of size one. The approach of [9] could not be applied with basis functions with larger scope. In particular, if it is theoretically possible to approximate the value function using features $h_i(x_{N(i)})$ and effectively build this approximation, the computation of a greedy policy with respect to this value function would be difficult (and it would certainly be global).

Finally, note also that even though the ALP approach comes with an upper bound on the error on the value function v_w, this bound is in general not very informative. For instance, in the upper bound β, note that $2\gamma/(1-\gamma)$ is high when γ is close to 1 and since computing exactly $\max_{x\in\mathcal{X}} |v_w(x) - \mathcal{B}(v_w)(x)|$ is too hard, this expression is upper-bounded in [9] by a sum of local upper bounds $\sum_{i=1}^{n} \max_{x_{N(i)}} |v_w^i(x_{N(i)}) - \mathcal{B}_i(v_w^i)(x_{N(i)})|$. In most case, this upper bound is really loose, and even sometimes trivial.

5. Experimental comparison of the MF-API and ALP approaches

In this section, we present the performance of the MF-API and ALP algorithms on the two examples described in Section 2.2: forest management under risk of wind damage and disease management in crop fields. We compare their performance regarding three criteria:

(i) Running time of the algorithms for problems consisting of hundreds of variables.
(ii) Values (estimated by Monte-Carlo, MC) of the policies δ_{API} and δ_{ALP} returned by the MF-API and ALP algorithms: $v^{\text{MC}}(\delta_{\text{API}})$ and $v^{\text{MC}}(\delta_{\text{ALP}})$.
(iii) Quality of the approximations of the value functions of δ_{API} and δ_{ALP}, provided by the algorithms: $\tilde{v}^{\text{MF-API}}$ and $\tilde{v}^{\text{ALP}}$. These are respectively compared to the MC estimations, $v^{\text{MC}}(\delta_{\text{API}})$ and $v^{\text{MC}}(\delta_{\text{ALP}})$.

Monte-Carlo estimations were computed as the average over 40 starting points, where the value for a specific start point was averaged over a hundred 44-step runs. The start points were randomly generated without repetition, together with a constraint of even distribution in order to avoid over-representation of a specific state of the nodes. In all experiments, a periodic discount rate $\gamma = 0.9$ was used. The algorithms were implemented in Scilab and the tests were run on an Intel Pentium 3.6 GHz machine with 1 GB of internal memory.

5.1. Benchmark of GMDP problems

In order to explore the influence of the different characteristics of forest and disease management problems, the performances were tested for problems of different sizes, different graph structures and different short-distance contamination probabilities (for the disease management problem), as described below.

5.1.1. Forest management under risk of wind damage

The graph topology considered for the forest management problem was that of Fig. 3, with an increasing number of stands (up to 700). For each test, an upper bound of the value of the optimal policy was computed by replacing the actual protection provided by neighbouring stands with a fixed protection effect corresponding to the maximal protection that neighbouring stands could provide. The resulting problem is fully decomposable and an exact solution could be computed, whose value function overestimates that of the solution of the original one. This upper bound was thus termed *Utopic*.

5.1.2. Disease management in crop fields

Here, we considered a constant "wheel" graph topology: an even number of nodes distributed on a circle. Each node formed part of a group of four neighbours: itself, the two closest nodes on the circle and the node diametrically opposed to it. An example of this structure for eight nodes is shown in Fig. 5. For this problem, the utopic upper bound on the value of the optimal policy was defined as the expected value when short-distance contamination is not possible ($p = 0$). This, again, leads to a fully decomposable problem (which is thus easily solvable) and whose solution overestimates that of the original problem. The parameters of the GMDP model for the disease management problem were set as follows: $\varepsilon = 0.01, p = 0.2, q = 0.9, r = 100$, and we solved it for an increasing number of nodes (up to 1600 nodes). We then considered a graph with sixteen nodes and we tested the influence of the probability of short-distance contamination (parameter p) on the performances. This parameter is particularly worthy of study as it is strongly linked to the capacity for a disease to spread across the whole area.

Finally, in order to evaluate the influence of the graph connectivity, we considered an initial wheel graph with sixteen nodes and a neighbourhood of size three for each node (each node had three neighbours: itself and the two closest nodes on the circle), and then applied a sequential procedure to build graphs with larger neighbourhood size. The graph with neighbourhood size ν was obtained from the graph with neighbourhood size $\nu - 1$ by adding edges between randomly selected nodes. We considered increasing connectivity up to $\nu = 5$. The parameters of the model were set to $\varepsilon = 0.01$, $p = 0.2$, $q = 0.9$, $r = 100$.

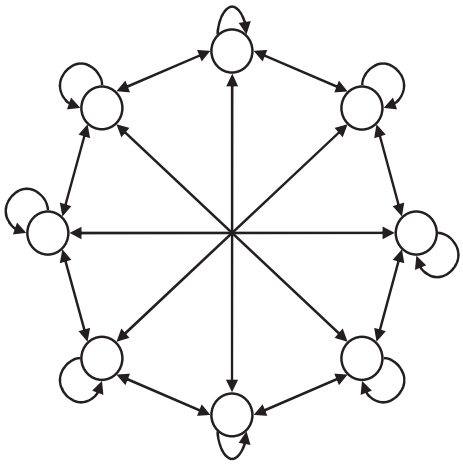

Fig. 5. Disease management problem: graph with wheel structure used to test the performance of ALP and MF-API for an increasing number of nodes.

5.2. Results of the experimental comparison

5.2.1. Forest management problems

We observed that the forest management problem was close to being decomposable and both MF-API and ALP policies had empirical values close to that of the utopic policy (Fig. 6(b)). However, this does not make the problem "easy", since random and greedy policies perform rather poorly in comparison (Fig. 6(a)). As expected, the running time for ALP increased linearly with the number of stands, while it increased polynomially for MF-API (Fig. 6(c)). A third-order polynomial function was fitted to the running time of MF-API: $time = 36.2258 + 1.1831n + 0.0727n^2 + 0.000007n^3$. The third-order coefficient was considerably smaller than the others. We also observed that the approximation of the value function computed using MF-API was considerably closer to the value of the optimal policy than that computed using ALP (Fig. 6(d)).

5.2.2. Disease management problems

For the first set of parameters of the disease management problem and the graph of Fig. 5, the conclusion was very similar to that of the forest management comparison: the quality of the policies returned by both methods where close to the "utopic" value, very close to each other and far better than that of random or greedy policies (Fig. 7 (a) and (b)). Once again, ALP ran in linear time, and MF-API in polynomial time: $time = 2.6980 + 0.7700n + 0.0152n^2 + 0.0n^3$ (Fig. 7 (c)). Again, this difference in time complexity corresponds to a difference in quality of the approximation of the expected values of the solution policies: the relative difference between $\tilde{v}^{\mathrm{MF-API}}$ and $v^{\mathrm{MC}}(\delta_{\mathrm{API}})$ was less than 5%, while it reached 60% between $\tilde{v}^{\mathrm{ALP}}$ and $v^{\mathrm{MC}}(\delta_{\mathrm{ALP}})$ (Fig. 7 (d)).

5.2.3. Influence of connectivity in the disease management problem

Graphs of increasing connectivity were generated (10 for each value of ν), for which policies were computed and their average expected values compared empirically (Fig. 8 (a)). Once again, MF-API and ALP policies were of similar quality and outperformed the greedy and random ones. The empirical value decreased when the connectivity increased, which is explained by the fact that a higher connectivity results in faster propagation of the disease and greater damages. Of course, this fact is not captured by the "utopic" approximation, which neglects propagation (Fig. 8(a)). We observed a non-linearity in the variation of the running time of ALP and MF-API with respect to the neighbourhood size (Fig. 8 (b)), coherent with the theoretical "exponential" increase (with ν) in time complexity. We also observed that the MF-API approximation of the value function, which was already good, increased with the connectivity of the graph (Fig. 8 (c)). This is due to the fact that the mean-field approximation of a spatial process is better when the graph is well connected.

5.2.4. Influence of the short-distance contamination parameter in the disease management problem

In order to check the variability of the results, we generated 50 graphs for a configuration with $\nu = 4$ and observed a variation in the average expected values of the policies of less than 1% (Fig. 9 (a)). More importantly, we observed a divergence in the value (evaluated empirically by Monte-Carlo simulations) of the policies provided by ALP and by MF-API when p increases (Fig. 9 (b)). Indeed, when $p = 0$ both algorithms compute exact policies. Then, when p starts to increase, the increasing spread of the disease makes the value of the optimal policy decrease, and of course MF-API and ALP policy values move further from the "utopic" value which increasingly overestimates the optimal policy value. There is then a threshold, around $p = 0.5$, where ALP performance continues to decrease while the MF-API performance starts to increase, becoming very close to optimal as p approaches 1. This behaviour of the mean field approximation is well known for uncontrolled stochastic processes: a phase transition is observed and, when close to the critical value the mean-field approximation is of poor quality while for low or high values of the parameter p the MF approximation improves.

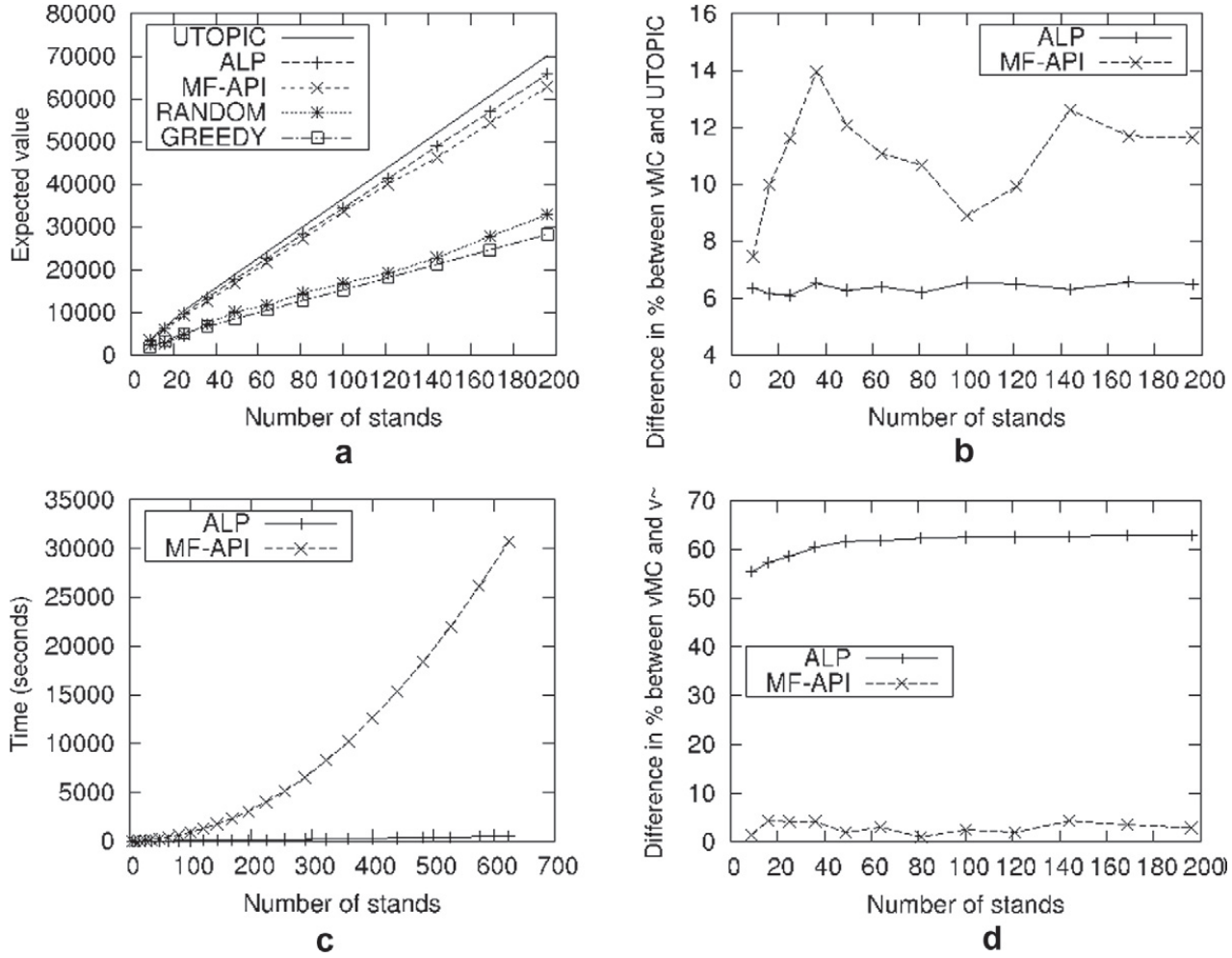

Fig. 6. Forest management problem: running times and expected values of the local policies for increasing sizes of forest. (a) Expected values of the computed policies and utopic upper bound. (b) Difference (in percentage) between $v^{MC}(\delta_{ALP})$ or $v^{MC}(\delta_{API})$ and the utopic upper bound. (c) Running times of ALP and MF-API. (d) Difference (in percentage) between $\tilde{v}^{ALP}$ and $v^{MC}(\delta_{ALP})$ or $\tilde{v}^{MF-API}$ and $v^{MC}(\delta_{API})$.

5.3. Comparison with a naive GMDP approximation by a system of independent MDPs

Throughout this paper, we have argued that eventhough the GMDP Mean Field approximate solution method decomposes the GMDP dynamics into independent unidimensional Markov chains, neighbourhood influence was retained, resulting into non-stationnarity of the Markov chains. In this section we propose a comparison with a naive approximation of the original GMDP problem by a system of n independent MDP where the transition probabilities $p_i(x'_i|x_{N(i)}, a_i)$ and reward functions $r_i(x_{N(i)}, a_i)$ are replaced with their simplest non spatial approximation,

$$\hat{p}_i(x'_i|x_i, a_i) = \frac{1}{|\mathcal{X}_{N(i)\setminus i}|} \sum_{x_{N(i)\setminus i}} p_i(x'_i|x_{N(i)}, a_i) \quad \text{and} \quad \hat{r}_i(x_i, a_i) = \frac{1}{|\mathcal{X}_{N(i)\setminus i}|} \sum_{x_{N(i)\setminus i}} r_i(x_{N(i)}, a_i).$$

We computed the optimal policy (named Naive Non Spatial policy, δ_{NNS}) corresponding to the union of the n optimal policies for each non spatial MDP problem and compared its (estimated value) to that of the MF-API policy δ_{API}. We performed this comparison on the disease management problem, for the graph structure of Fig. 5 with same parameters settings than above. When changing the value of the short-distance contamination parameter p from 0.2 to 0.6, i.e. from a problem with low to high spatial structure, the results confirm the superiority of the MF-API policy (see Fig. 10). This study confirms the interest of the MF approach, as a trade-off between a fully spatial problem of size n (which resolution is intractable) and n independent problems (which completely neglect spatial dependencies).

6. From flat to multidimensional action space

Throughout this article, we have emphasized the fact that, unlike existing factored MDP approaches, the GMDP framework together with the mean-field solution method, is able to handle multidimensional action spaces in Factored MDPs. We now come back to this point in order to point out where the GMDP framework and algorithms can be improved.

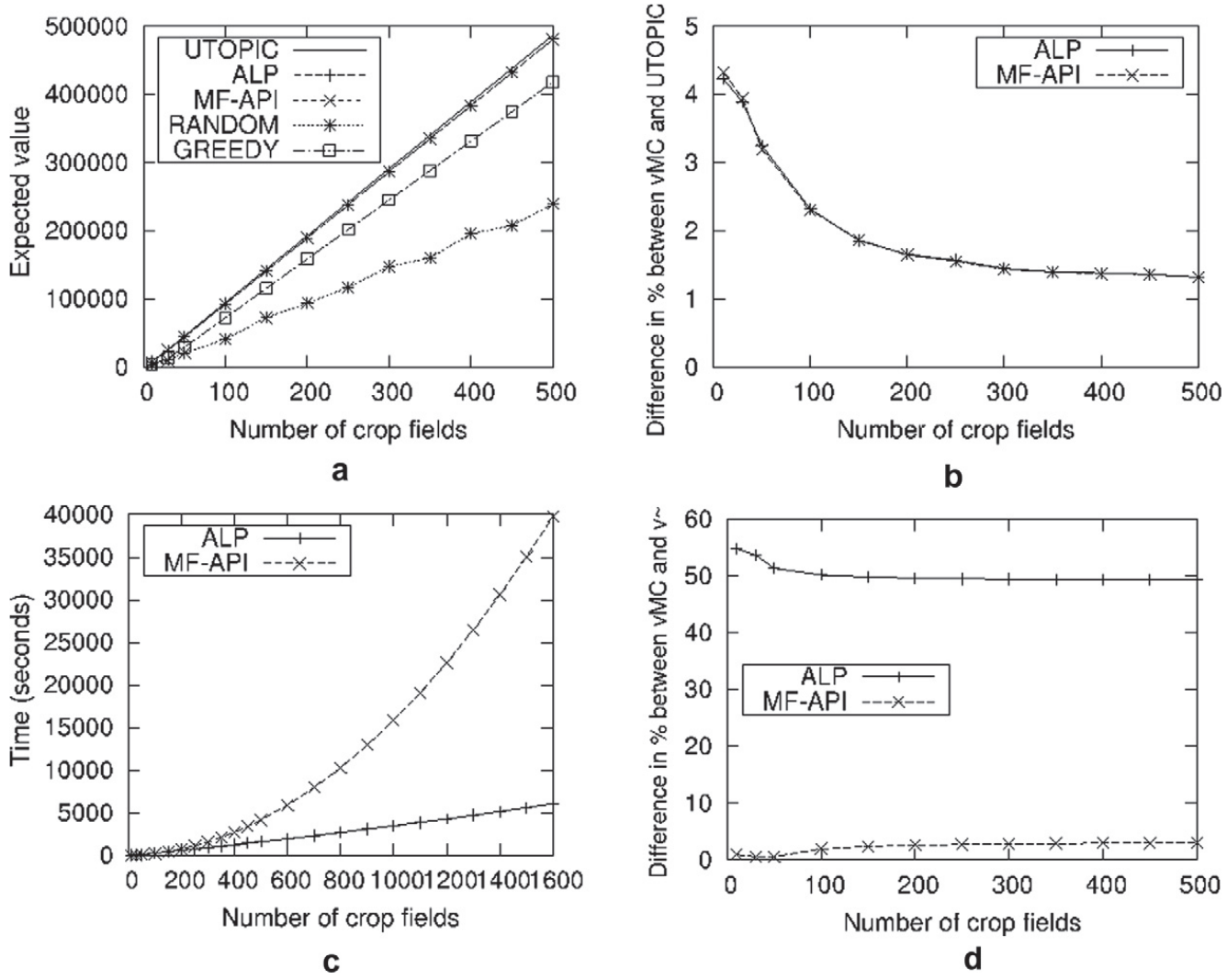

Fig. 7. Disease management problem: running times and expected values of the local policies for increasing number of fields. (a) Expected values of the computed policies and utopic upper bound. (b) Difference (in percentage) between $v^{MC}(\delta_{ALP})$ or $v^{MC}(\delta_{API})$ and the utopic upper bound. (c) Running times of ALP and MF-API. (d) Difference (in percentage) between $\tilde{v}^{ALP}$ and $v^{MC}(\delta_{ALP})$ or $\tilde{v}^{MF-API}$ and $v^{MC}(\delta_{API})$.

As far as the action space is concerned, we can distinguish three different cases, which we can illustrate on the famous *SysAdmin* problem [5], where actions correspond to rebooting any of n computers which can become "faulty" (depending on the status of neighbour computers, in the network):

P_1 In its original version [5], only one computer at a time can be rebooted, which amounts to define a *flat* action space of size $|\mathcal{A}| = n$.

P_2 In the GMDP framework, we would model the problem (certainly in a more realistic way if the time step considered is not too small) where we can reboot *any* subset of computers, thus leading to an action space $\mathcal{A} = \{0, 1\}^n$ of size 2^n.

P_3 The intermediate case is the one in which only k (k fixed) out of n computers at most can be rebooted at each time step, (for example because time constraints are involved).

These problems are of different nature. Problem P_1 cannot be modelled in the GMDP framework, since the GMDP framework assumes independent action choices for all machines. On the other hand, even though modelling problem P_2 in the factored MDP framework would be possible, FMDP solution methods would be impractical, as shown in Section 4. Finally, the intermediate case P_3, with the global constraint that exactly $k \leq n$ machines can be rebooted, can neither be dealt with practically in the framework of [5], when k becomes large, nor be modelled within the current GMDP framework: even though the action space has smaller size ($|\mathcal{A}| = C_n^k$) than in the independent case ($|\mathcal{A}| = 2^n$), the global constraint makes the local policies interdependent. We would have to consider only local policies which allow to satisfy the global constraint for all possible states of the world. The MF approximate policy improvement phase we suggest does not allow to compute policies satisfying these requirements. On the other hand, the MF approximate policy evaluation phase would not have to be modified to adapt to this case.

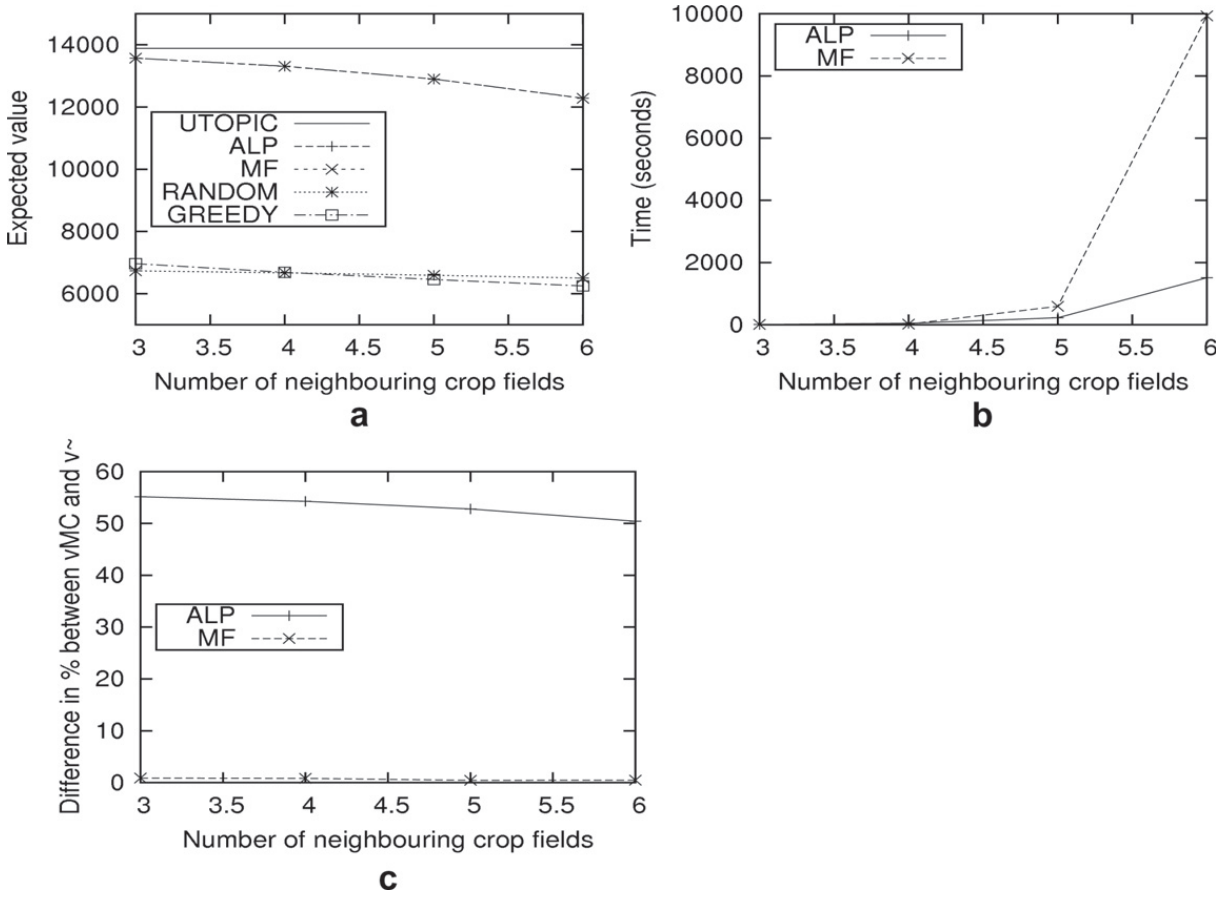

Fig. 8. Disease management problem: running times and expected values of the local policies for increasing connectivity. (a) Expected values of the computed policies and utopic upper bound. (b) Running times of ALP and MF-API. (c) Difference (in percentage) between $\tilde{v}^{\mathrm{ALP}}$ and $v^{\mathrm{MC}}(\delta_{\mathrm{ALP}})$ or $\tilde{v}^{\mathrm{MF-API}}$ and $v^{\mathrm{MC}}(\delta_{\mathrm{API}})$.

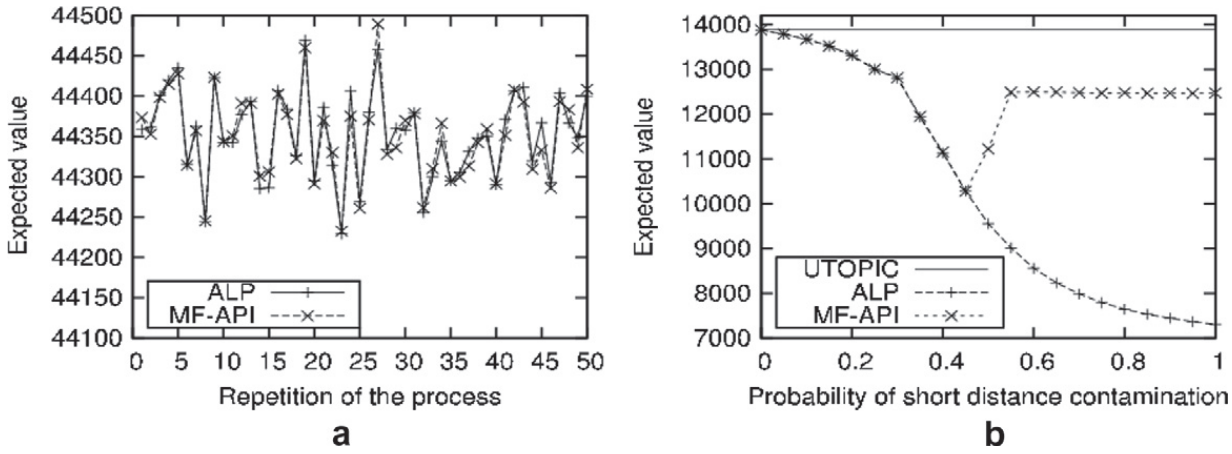

Fig. 9. Disease management problem: (a) Variability in the expected values of the local policy solutions of ALP and MF-API, for randomly generated graphs with a neighbourhood size of three. (b) Influence of the probability of short-distance contamination (p).

To conclude, available approximate solution algorithms for FMDPs enable only to deal with problems of type P_1. The contribution of the GMDP framework proposed in this article is to provide an approximate solution method for problems of type P_2. The challenge is now to propose methods which can solve problems of type P_3. In the GMDP framework, the deadlock is the policy update step. However, the mean-field evaluation can be applied to any local policy, in particular those satisfying global constraints. Thus, the MF principle could be extended to be applied to problem P_3.

7. Conclusion and further work

In this article, we have presented a *graph-based Markov Decision Process (GMDP)* framework for the control of spatio-temporal processes, which forms a special case of factored MDP. Optimal policies can only be computed for GMDPs of very

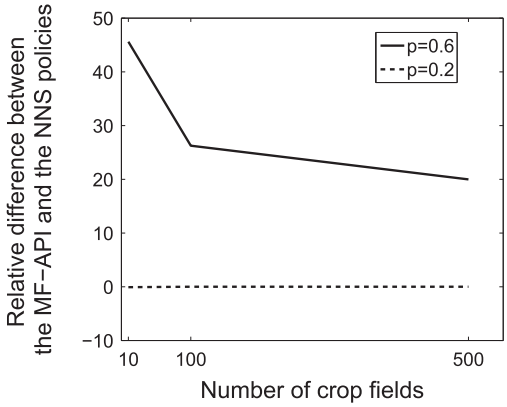

Fig. 10. Disease management problem: relative error ε between $\delta_{\mathrm{MF-API}}$ and δ_{NNS}. $\varepsilon = 100 * \frac{v^{\mathrm{MC}}(\delta_{\mathrm{MF-API}}) - v^{\mathrm{MC}}(\delta_{NNS})}{\delta_{NNS}}$ for low and high short-distance contamination rate.

small size. Therefore, we have proposed an approximate solution algorithm for larger problems, based on the mean-field approximation of a structured Markov process. The performance of the proposed algorithm was evaluated on two agricultural problems and was compared to an *Approximate Linear Programming* algorithm adapted to the GMDP framework. Experiments highlight the efficiency of the algorithm for solving large-scale GMDPs, as well as its ability to compute close-to-optimal policies for these problems. While this algorithm takes quadratic time to run when the ALP algorithm takes linear time in practice, this is balanced by a far better estimation of the value function of the returned policy. Furthermore, even though in most problems the policies returned by the ALP and MF-API approaches are of similar quality, we were able to find situations where the policies returned by the MF-API approach become significantly better. The work presented here has been applied to two different settings. First, in forestry where [13] considered a real forest estate in southern Sweden, and in plant disease management where [12] studied the case of phoma stem canker disease propagation in canola crops.

Among the hypotheses that were made in this work, the more questionable is perhaps that we assumed that there were no constraints on the admissible set of factored actions. It would certainly be an important gain if it were possible to extend the GMDP framework so as to be able to handle constraints on the action space. Note that the MF approximation applies to *evaluate* policies satisfying constraints over the action set. The *policy improvement* phase is the crucial difficulty that must be tackled in order to extend our approach.

Other methodological developments may concern the construction of solution algorithms for GMDPs based on *Reinforcement Learning* (RL) [7]. The advantage of RL algorithms is that they do not require explicit knowledge of the transition model, but can do with a simulation of it. They can thereby be used together with the large existing body of agricultural or environmental simulation-based models, for example [22,23]. In [24] we showed how a set of simple RL algorithms, suggested for solving factored MDPs and collaborative multiagent MDPs, could be adapted for solving a GMDP. This preliminary work seemed promising but needs further work in order to be evaluated.

Supplementary material

Supplementary data associated with this article can be found, in the online version, at doi:10.1016/j.ijar.2011.09.007.

References

[1] M.L. Puterman, Markov Decision Processes, JohnWiley and Sons, New York, 1994, pp.
[2] C. Boutilier, R. Dearden, M. Goldszmidt, Stochastic dynamic programming with factored representations, Artificial Intelligence 121 (1) (2000) 49–107.
[3] J. Hoey, R. St-Aubin, A. Hu, C. Boutilier, SPUDD: Stochastic planning using decision diagrams, in: Proceedings of UAI'99, Stockholm, Sweden, 1999.
[4] R. St-Aubin, J. Hoey, A. Hu, C. Boutilier, APRICODD: Approximate policy construction using decision diagrams, in: Proceedings of NIPS'00, Denver, CO, 2000.
[5] C. Guestrin, D. Koller, R. Parr, S. Venkataraman, Efficient solution algorithms for factored MDPs, Journal of Artificial Intelligence Research 19 (2003) 399–468.
[6] D.-P. de Farias, B. Van Roy, The linear programming approach to approximate dynamic programming, Operations Research 51 (6) (2003) 850–865.
[7] D.P. Bertsekas, J.N. Tsitsiklis, Neuro-Dynamic Programming, Massachussetts, Athena Scientific, Belmont, 1996.
[8] M. Opper, D. Saad (Eds.), Advanced Mean-Field Methods: Theory and Practice, Massachussetts Institute of Technology, 2001.
[9] N. Forsell, R. Sabbadin, Approximate linear-programming algorithms for graph-based Markov decision processes, in: Proceedings of ECAI06, Riva Del Garda, Italy, 2006, pp. 590–594.
[10] N.L. Zhang, R. Qi, D. Poole, A computational theory of decision networks, International Journal of Approximate Reasoning 11 (2) (1994) 83–158.
[11] R.K. Chornei, H. Daduna, P.S. Knopov, Control of Spatially Structured Random Processes and Random Fields with Applications, Springer, 2006.
[12] N. Peyrard, R. Sabbadin, E. Lô-Pelzer, J.N. Aubertot, A graph-based Markov decision process framework for optimising integrated management of diseases in agriculture, in: Proceedings of MODSIM'07, Christchurch, New-Zealand, 2007, pp. 2175–2181.
[13] N. Forsell, L.O. Eriksson, F. Garcia, R. Sabbadin, P. Wikström, Management of the risk of wind damage in forestry: a graph-based markov decision process approach, Annals of Operations Research, Published online: 09 February 2009, 2009.
[14] N. Peyrard, R. Sabbadin, Mean field approximation of the policy iteration algorithm for graph-based Markov decision processes, in: Proceedings of ECAI06, Riva Del Garda, Italy, 2006, pp. 595–599.
[15] E. Altman, Constrained Markov Decision Processes, Chapman & Hall/CRC, 1999.

[16] Y. Xiang, F. Hanshar, Comparison of tightly and loosely coupled decision paradigms in multiagent expedition, International Journal of Approximate Reasoning 51 (2010) 600–613.
[17] D.P. de Farias, B. Van Roy, On constraint sampling in the linear programming approach to approximate dynamic programming, Mathematics of Operations Research 29 (3) (2004) 462–478.
[18] B. Kveton, M. Hauskrecht, C. Guestrin, Solving factored MDPs with hybrid state and action variables, Journal of Artificial Intelligence Research (27) (2006) 153–201.
[19] P. Poupart, C. Boutilier, R. Patrascu, D. Schuurmans, Piecewise linear value function approximation for factored MDPs, in: Proceedings of AAAI'02, 2002, pp. 292–299.
[20] C. Guestrin, D. Koller, R. Parr, Multiagent planning with factored MDPs, in: Proceedings of NIPS'01, 2001, pp. 1523–1530.
[21] R.J. Williams, L.C.I. Baird, Tight performance bounds on greedy policies based on imperfect value functions, Technical report, College of Computer Science, Northeastern University, Boston, MA, 1993.
[22] K. Blennow, O. Sallnäs, Winda – a system of models for assessing the probability of wind damage to forest stands within a landscape, Ecologival Modelling 175 (2004) 87–99.
[23] M.A. Finney, Modeling the spread and behavior of prescribed natural fires, in: Proceedings of the 12th Conference on Fire and Forest Meterology, 1994, pp. 138–143.
[24] N. Forsell, F. Garcia, R. Sabbadin, Reinforcement learning for spatial processes, in: R.S. Anderssen, R.D. Braddock, L.T.H. Newham (Eds.), Proceedings of MODSIM'09, July 2009, pp. 755–761.