ECAI 2012
Luc De Raedt et al. (Eds.)

doi:10.3233/978-1-61499-098-7-181

A Reinforcement-Learning Algorithm for Sampling Design in Markov Random Fields

Mathieu BONNEAU[1, 2] and **Nathalie Peyrard**[1] and **Régis Sabbadin**[1]

Abstract. Optimal sampling in spatial random fields is a complex problem, which mobilizes several research fields in spatial statistics and artificial intelligence. In this paper we consider the case where observations are discrete-valued and modelled by a Markov Random Field. Then we encode the sampling problem into the Markov Decision Process (MDP) framework. After exploring existing heuristic solutions as well as classical algorithms from the field of Reinforcement Learning (RL), we design an original algorithm, LSDP (Least Square Dynamic Programming), which uses simulated trajectories to solve approximately any finite-horizon MDP problem. Based on an empirical study of the behaviour of these different approaches on binary models, we derive the following conclusions: i) a naïve heuristic, consisting in sampling sites where marginals are the most uncertain, is already an efficient sampling approach; ii) LSDP outperforms all the classical RL approaches we have tested; iii) LSDP outperforms the heuristic in cases when reconstruction errors have a high cost, or sampling actions are constrained. In addition, LSDP readily handles action costs in the optimisation problem, as well as cases when some sites of the MRF can not be observed.

1 INTRODUCTION

Optimal sampling in spatial random fields is a complex problem, which mobilizes several research fields in spatial statistics [2, 10] and artificial intelligence [7, 6, 13]. It raises methodological issues in modelling, inference and algorithm design. An active stream of research about optimal spatial sampling is dedicated to the study of the case of real-valued observations (e.g. temperature or pollution monitoring). Models and efficient algorithms have been proposed [9, 7], mainly based on the geostatistical framework of Gaussian random fields and kriging. Much less attention has been paid to the case of discrete-valued observations. However, this problem is ubiquitous in many studies about biological systems. Discrete-valued observations can be species abundance classes, disease severity classes, presence/absence values...

Solving optimal sampling problems in discrete-valued random fields is a difficult question admitting no universally accepted solution, so far. One should look for approximate solution algorithms with reasonable/moderate complexity and with satisfying approximation quality. We propose, similarly to [6, 13, 14], to define the optimal sampling problem within the framework of Markov random fields (MRF, [4]), classically used in image analysis. We consider the case of adaptive sampling, where the set of sampled sites is chosen sequentially and observations from previous sampling steps are taken into account to select the next sites to explore [19]. Simple heuristics have been proposed [19, 2, 14] to design adaptive sampling strategies. However, it is difficult to evaluate their quality since there is no efficient exact method to compare to. In this paper, we design a new reinforcement-learning (RL, [17]) algorithm which improves classical heuristic and RL approaches, thus providing a reference algorithm. The algorithm, named LSDP (Least Square Dynamic Programing) uses an encoding of the optimal adaptive sampling problem as a finite-horizon Markov Decision Process (MDP, [15]) with factored state space.

The MRF formalization of the optimal adaptive spatial sampling problem is introduced in Section 2, together with a computational complexity study. We show how to model it as a finite-horizon factored MDP in Section 3 and we discuss classical RL solutions in Section 4. Then, we describe the LSDP algorithm in Section 5. We present an empirical comparison between heuristic approaches, classical RL algorithms and LSDP in Section 6. Some methodological and applied perspectives of this work are discussed in Section 7.

2 OPTIMAL ADAPTIVE SAMPLING IN MARKOV RANDOM FIELDS

2.1 Problem statement

Let $X = (X_1, \ldots, X_n)$ be a vector of discrete random variables taking values in $\Omega^n = \{1, \ldots, K\}^n$. $V = \{1, \ldots, n\}$ is the set of indices of the vector X and an element $i \in V$ will be called a *site*. The distribution $\mathbb{P}$ of X is that of a Markov Random Field (MRF) with associated graph $G = (V, E)$ where $E \subseteq V^2$ is a set of undirected edges. $x = (x_1, \ldots, x_n)$ is a realization of X and we adopt the following notation: $x_B = \{x_i\}_{i \in B}, \forall B \subseteq V$. Then we can write $\mathbb{P}(X = x) \propto \prod_{c \in \mathcal{C}} \Psi_c(x_c)$, where $\mathcal{C}$ is the set of cliques of V and the $\Psi_c, c \in \mathcal{C}$ are strictly positive potential functions [4].

The sampling problem we consider can be described intuitively. Our goal is to reconstruct the vector X on a specified subset $R \subseteq V$ of sites of interest. To do this, we can acquire a limited number of observations on a subset $O \subseteq V$ of observable sites. We will assume that $R \cup O = V$ and intersection between O and R can be non-empty. The sampling problem is to select a set of sites $A \subseteq O$, (a *sample*), where X will be observed. When sample A is chosen, a *sample output* x_A results, from which the MRF distribution $\mathbb{P}$ is updated. Intuitively, our objective is to choose A in a sequential way, so that the updated distribution $\mathbb{P}(\cdot|x_A)$ becomes as informative as possible (in expectation over all possible sample outputs).
In the following we describe the different elements allowing to formally define the sampling optimisation problem.

[1] INRA ; UBIA UR875 ; BP 52627 – 31326 Castanet-Tolosan, France.
[2] INRA ; UMR1347 Agroécologie ; BP 86510 –21065 Dijon, France.
Email: {mathieu.bonneau,nathalie.peyrard,regis.sabbadin}@toulouse.inra.fr

Reconstruction. When a sample output x_A is available, the Maximum Posterior Marginals (MPM) criterion, classically used in image analysis, is used to derive an estimator x_R^* of the hidden map x_R:

$$x_R^* = \left\{ x_i^* \quad | \quad i \in R, \quad x_i^* = \operatorname*{argmax}_{x_i \in \Omega} \mathbb{P}(x_i \mid x_A) \right\}.$$

Adaptive sampling policy. In adaptive sampling, the sample A is chosen sequentially. The sampling plan is divided into H steps. $A^h \subseteq O$ is the sample explored at step $h \in \{1, \ldots, H\}$ and x_{A^h} is the sample output at step h. The samples size is fixed ($|A^h| = L$) and Δ_L is the set of all policies satisfying $|A^h| = L, \forall h$. The choice of sample A^h depends on the previous samples and outputs. An adaptive sampling policy $\delta = (\delta^1, \ldots, \delta^H)$ is then defined by an initial sample A^1 and functions δ^h specifying the sample chosen at step $h \geq 2$, depending on the results of the previous steps: $\delta^h((A^1, x_{A^1}), \ldots, (A^{h-1}, x_{A^{h-1}})) = A^h$.

A *history* is a trajectory $(A^1, x_{A^1}), \ldots, (A^H, x_{A^H})$ followed when applying policy δ. The set of all histories which can be followed by policy δ is τ_δ. We will assume throughout the paper that observations are reliable. As a consequence, we will only consider policies visiting each site at most once ($A^h \cap A^{h'} = \emptyset, \forall h \neq h'$). Furthermore, since our definition of the quality of a policy is based on the MPM criterion, it does not depend on the order in which observations are received. Therefore, the relevant information in a history can be summarized by the pair (A, x_A), where $A = \cup_h A^h$.

Sample cost. The modeling of a sampling cost function is an issue as it stands. Here we illustrate this notion with the simplest definition, where sample costs are additive. For a given history $\big((A^1, x_{A^1}) \ldots, (A^H, x_{A^H})\big)$, the total cost is

$$\sum_{h=1}^{H} c(A^h) = c\left(\cup_h A^h\right), \text{with } c(A^h) = \sum_{i \in A_h} c_i,\ c_i \in \mathbb{R}^+.$$

Quality of a sampling policy. The quality of a policy δ is measured as the expected quality of the estimator x_R^* that can be obtained from δ. In practice, we first define the quality of a history $((A_h, x_{A_h}))_{h=1..H}$ as a function of (A, x_A), where $A = \cup_h A_h$:

$$U\big(A, x_A\big) = \sum_{i \in R} \left[\max_{x_i \in \Omega} \left\{ \mathbb{P}(x_i \mid x_A) \right\} \right] - c(A). \tag{1}$$

The quality of a sampling policy δ is then defined as an expectation over all possible histories:

$$V(\delta) = \sum_{((A_h, x_{A_h}))_h \in \tau_\delta} \mathbb{P}\big(x_A\big) U\big(A, x_A\big).$$

Optimal adaptive sampling in MRF (OASMRF)**.** The problem of optimal adaptive sampling is to find the policy of highest quality :

$$\delta^* = \arg\max_{\delta \in \Delta_L} V(\delta). \tag{2}$$

2.2 Computational complexity of optimal adaptive sampling in MRF

In this section we study the computational complexity of the OASMRF problem. More precisely, we will study the following, *generalised* OASMRF problem (GOASMRF), expressed in a decision form: *Does there exist δ of depth at most N, such that:*

$$\sum_{((A_h, x_{A_h}))_{h=1..H} \in \tau_\delta} \mathbb{P}\big(x_A\big) U\big(A, x_A\big) \geq G\,?$$

Where $G > 0$ is a fixed threshold, and

$$U\big(A, x_A\big) = \sum_{i \in R} f_i\big(x_i^*, \mathbb{P}(x_i^* \mid x_A)\big) - c(A),$$

where the functions f_i are non-decreasing functions in their second argument and $x_i^* = \arg\max_{x_i} \mathbb{P}(x_i \mid x_A)$.

This form of utility of a history generalises (1), which is recovered when f_i is a projection on his second argument. The extended form can represent criteria consisting in maximising a weighted expected number of well-restored variables (when some variables are more important than others), or the expected number of variables restored with confidence above a given threshold. The fact that x_i^* is involved and not only its probability, allows to bias restoration towards particular values of x_i. This can be useful, for instance, if we want to build an invasive species map, where we give more weight to restoring invaded sites than non-invaded ones. Finally, the fact that f_i is non-decreasing is not essential for proving the proposition, but reflects the fact that the more certain we are about x_i^*, the better.

Proposition 1. *The* GOASMRF *problem is* PSPACE*-complete.*

Proof. There is not much difficulty in proving that GOASMRF belongs to PSPACE. The difficult part is to establish the PSPACE-hardness of the GOASMRF problem. To prove this, we reduce the *State Disambiguation (SD)* problem, which is known to be PSPACE-hard [1] to it. A detailed proof is given in the Appendix. □

The consequence of Proposition 1 is that exact optimization of the sampling policy is intractable. So, we must turn to approximate solution methods for computing sample policies. In the next section we present a (factored) Markov Decision Process (MDP) model of the OASMRF problem[3]. Using an MDP model allows us to solve OASMRF problems approximately by applying simulation-based Reinforcement Learning (RL) algorithms [17].

3 Finite horizon MDP modelling of the OASMRF problem

A finite-horizon Markov Decision Process model $\langle S, D, T, p, r \rangle$ is a 5-tuple, where S is a finite set of system *states*, D is a finite set of available *decisions*, $T = \{1, \ldots, H\}$ is a finite set of decision steps, termed *horizon*. p is a set of *transition functions* $p^t, t = 1 \ldots H$, where $p^t(s^{t+1}|s^t, d^t)$ indicates the probability that state $s^{t+1} \in S$ results when the system is in state $s^t \in S$ and decision $d^t \in D$ is implemented at time $t \in \{1, \ldots, H\}$. A *terminal state* $s^{H+1} \in S$ results when the last action is applied, at decision step H. r is a set of *reward functions*: $r^t(s^t, d^t) \in \mathbb{R}$ is obtained when the system is in state s^t at time t and d^t is applied. A *terminal reward* $r^{H+1}(s^{H+1})$ is obtained when state s^{H+1} is reached at time $H + 1$.

A *decision policy* (or *policy*, for short) $\pi = \{\pi^1, \ldots, \pi^H\}$ is a set of decision functions $\pi^t : S \to D$. Once a decision policy is fixed, the MDP dynamics becomes that of a finite Markov chain over S, with transition probability $p^t(s^{t+1}|s^t, \pi^t(s^t))$. The *value function* $V^\pi : S \times T \to \mathbb{R}$ of a policy π is defined as the expectation of the sum of future rewards, obtained from the current state and time step when following the Markov chain defined by π:

$$V^\pi(s, t) = \mathbb{E}_\pi \left[\sum_{t'=t}^{H+1} r^{t'} \mid s \right], \forall (s, t) \in S \times T.$$

[3] Which can be easily extended to GOASMRF.

Solving an MDP amounts to finding an *optimal policy* π^* which value is maximal for all states and decision steps: $V^{\pi^*}(s,t) \geq V^{\pi}(s,t), \forall \pi, s, t$. We now show how to model the OASMRF problem in the MDP framework.

State space. An MDP state $s^t, t = 1, \ldots, H+1$ summarizes current information about variables indexed in O:

$$s^t = \left(\bigcup_{h=1}^{t-1} A^h, \bigcup_{h=1}^{t-1} x_{A^h} \right), \forall t = 2, \ldots, H+1 \text{ and } s^1 = (\emptyset, \emptyset).$$

The total number of possible states of the system is, of course, exponential in the OASMRF representation size.

Action space. An admissible decision d^t is a sample A^t such that $|A^t| = L$ and such that $A^t \cap A^{t'} = \emptyset, \forall t' < t$.

Horizon. Decision steps in the MDP correspond to decision steps in the OASMRF problem. Thus, $T = \{1, \ldots, H\}$.

Transition functions. If $s^t = (A, x_A)$ and $d^t = A^t$ the transition function of the MDP can be derived straightforwardly from the original MRF distribution $\mathbb{P}$:

$$p^t\big(s^{t+1} \mid s^t, d^t\big) = \mathbb{P}\big(x_{A^t} \mid x_A\big), \forall t \in T.$$

Reward functions. $\forall t$, (negative) rewards represent sampling costs:

$$r^t(s^t, d^t) = r^t(d^t) = -c(A^t), \forall t \in T, s^t, d^t.$$

After decision d^H has been applied at decision step H, and state $s^{H+1} = (A, x_A)$ has been reached, the final reward $r^{H+1}(s^{H+1})$ is obtained, which is defined as the quality of the MPM reconstruction:

$$r^{H+1}(s^{H+1}) = \sum_{i \in R} \left[\max_{x_i \in \Omega} \Big\{ \mathbb{P}(x_i \mid x_A) \Big\} \right].$$

The optimal policy for the above-defined MDP is a set of functions associating samples to unions of past samples outputs. It thus has the same structure as an OASMRF sampling policy. Furthermore, we can establish the following proposition:

Proposition 2. *An optimal policy for the MDP model of an* OASMRF *problem provides an optimal policy for the initial OASMRF problem (2).*

Proof. (Sketched). The proof follows three steps and uses the fact that the quality of a policy and cost function does not depend on the order in which observations are obtained:

(i) We define a function ϕ, transforming any MDP policy π into a valid OASMRF policy $\delta = \phi(\pi)$, which defines actions independently of the order in which past observations were received, and show that $V(\phi(\pi)) = V^{\pi}((\emptyset, \emptyset), 1)$.

(ii) We establish that, for any partial history (past observations), the value of an optimal OASMRF policy starting from these observations does not depend on the order in which they were received. As a consequence, we can limit the search for optimal policies of the OASMRF problem to policies prescribing actions which do not depend on the order of observations.

(iii) We show that any such OASMRF policy δ can be transformed into an MDP policy, through a transformation μ, and that $V(\delta) = V^{\mu(\delta)}((\emptyset, \emptyset), 1)$.

As a result of these three steps, if π^* is an optimal policy for the MDP encoding of the OASMRF problem, then $\phi(\pi^*)$ is optimal for the OASMRF problem. □

In the following we will use the same notation δ to represent both OASMRF and MDP policies.

4 APPROACHES FOR SOLVING OASMRF

4.1 Exact dynamic programming

The *backwards induction* algorithm [15] can be applied to compute the optimal policy of any finite-horizon MDP. It consists in solving iteratively the following equations: $\forall t = H, \ldots, 1, \forall s, d \in S \times D^t$,

$$\begin{aligned} V^*(s, H+1) &= r^{H+1}(s), \\ Q^*(s,d,t) &= r^t(s,d) + \sum_{s'} p^t(s'|s,d) V^*(s', t+1), \quad (3) \\ \pi^{*,t}(s) &= \pi^*(s,t) = \arg\max_d Q^*(s,d,t), \\ V^*(s,t) &= \max_d Q^*(s,d,t). \end{aligned}$$

However, since the OASMRF problem is PSPACE-complete, exact dynamic programming is inapplicable to large problems. Therefore, we have to look for sub-optimal policies. To do this, we can explore two families of approaches used for solving OASMRF: *heuristic approaches* and *simulation-based* approaches.

4.2 Heuristic approaches

Heuristic approaches are methods for sample selection which provide an arbitrary sample in short time. These methods either solve a simpler optimization problem, or provide simple arbitrary policies. Several heuristics have been proposed, either in Statistics or in AI, that can be applied to solve the OASMRF problem. In spatial sampling of natural resources, random and regular sampling are classic ones [2]. Another classical method to sample 0/1 variables is Adaptive Cluster Sampling (ACS, [19]). Recently, [14] proposed a heuristic (*BP-max heuristic*) which consists in sampling locations where the marginal probabilities are less informative, in order to solve (2). It has been shown to outperform random, regular and ACS heuristics. In [7], the authors proposed to optimize a mutual information (MI) criterion to design sampling strategies in Gaussian Processes.

4.3 Simulation based approaches: Reinforcement learning

The main idea of Reinforcement Learning approaches (RL, [18],[17]) is to use repeated simulated *experiences* (s^t, d^t, r^t, s^{t+1}), instead of dynamic programming, in order to estimate Q^* or a parametrized approximation $\widetilde{Q}$ of Q^* [17][4]. They can either estimate Q^* directly (*Q-learning* approach, for example), or interleave estimation steps of a current policy π ($TD(\lambda)$ can be used) with improvement steps, in a general *policy iteration* scheme [17].

In most cases where simulation is used to solve large, factored MDP such as in the OASMRF problem, functions Q^π are too expensive to store in tabular form. In this case, a parametric approximation of the Q-function is built as : $\widetilde{Q}(s,d,t) = w^\top \phi(s,d,t)$, where $w \in \mathbb{R}^b$ is a vector of parameters values and $\phi : (S^t, D^t, t) \to \mathbb{R}^b$ is

[4] For simplicity notation $\widetilde{Q}$ is used instead of $\widetilde{Q}^*$

a mapping from state-action pairs to real-valued b-dimensional vectors. Simulations are used to compute values w of parameters that give a good approximation of Q^*. Algorithms for computing $\widetilde{w}$ are, for example, *LSPI* [8], *Fitted Q-iteration* ([3],[11]), etc. Online algorithms, as UCT ([5]), for example, could be applied. However, the time needed to compute sample policies online seems incompatible with the real-time constraints we are facing, therefore we did not consider these approaches.

5 LEAST-SQUARES DYNAMIC PROGRAMMING (LSDP)

5.1 Approximate dynamic programming

The main idea of the algorithm we propose is to combine a parametrized representation of the Q-function with *dynamic programming* (DP) iterations and simulation in order to approximate Q^*. Namely, we consider an approximation $\widetilde{Q}$ of Q^* as a linear combination of n arbitrary *features* [17]:

$$\begin{aligned}\widetilde{Q}(s,d,t) &= \sum_{i=1..n} w_i^{(t)}\phi_i(s,d,t), \forall s,d,\forall t\in T \text{ and}\\ \widetilde{Q}(s,H+1) &= r^{H+1}(s^{H+1}), \forall s.\end{aligned}$$

The weights w_i^t are computed recursively for $t=H$ to 1, in such a way that equations (3) are approximately satisfied:

$$\begin{aligned}\sum_{i=1..n} w_i^t\phi_i(s,d,t) &\approx r^t(s,d)+\sum_{s'} p^t(s'|s,d)\widetilde{V}(s',t+1)\\ \text{where } \widetilde{V}(s,t) &= \max_d \sum_{i=1..n} w_i^t\phi_i(s,d,t). \qquad (4)\end{aligned}$$

Equations (4) form a set of $|S|\times|D|$ linear equations for each time step $t\in T$, with variables $w_i^t, i=1..n$. These systems are clearly over-constrained ($|S|\times|D| \gg n$), therefore we look for *least-squares* solutions, instead of exact ones. The dynamic programming part of the approach comes from the fact that the systems are solved separately for $t=H$ to 2, each solution vector w^{t+1} being plugged into the system obtained at time t.

5.2 LSDP Algorithm

Systems (4) are too large to build when S is factored, not to mention solving. Therefore, we suggest to consider only a subset of equations, corresponding to a subset of samples (called *batch* [16]) $\mathcal{B}=\{(s,d,t)\}\subseteq S\times D\times T$. We propose to build $\mathcal{B}$ from a finite set of simulated trajectories (length $H+1$) starting in s_1, obtained by simulating successive transitions. Actions are chosen randomly, either maximizing $\widetilde{Q}^w$ (with probability $1-\varepsilon$) or uniformly (with probability ε) at each time step.

We use these batches to define the *Least-Squares Dynamic Programming* (LSDP) algorithm, a variant of the *policy iteration* algorithm [15]. LSDP iterates updates of the current parameters values w from a current simulation batch, applying approximate dynamic programming and accepting the updated parameters values only if the value of the corresponding policy (estimated by simulation) improves the previous one. If the value is not improved, another batch $\mathcal{B}'$ is randomly built and used. A maximum number of batches to simulate is fixed, and when reached, the current policy is returned.

Of course, one can note that, for a given set of parameters values, different batches may be obtained by simulation, leading to different updated parameters values and thus to different updated policies. Furthermore, there is no guarantee that the updated policy improves the current policy in state s_1. This is why the value of the updated policy has to be estimated (by simulation) and compared to the value of the previous policy, before being accepted if it actually improves. This conditional acceptation allows to guarantee that the successive policies returned by the algorithm are of increasing value.

5.3 Application to the OASMRF problem

In order to apply the LSDP algorithm to the OASMRF problem, we take into account the problem structure (i) to define features ϕ_i and (ii) to propose an adapted batch construction method.

The BP-max heuristic (see [14] and Section 4) can be mimicked by a linear combination of the following features, with all weights equal to 1: $\forall i\in\{1,\dots,n\}$,

$$\begin{aligned}\phi_i(s,d) &= \big(1-1_{\{i=d\}}\big)\max_{x_i\in\Omega}\widetilde{\mathbb{P}}\big(x_i \mid x_A\big)+1_{\{i=d\}}, \text{ where}\\ \widetilde{\mathbb{P}}(x_i \mid x_A) &= \mathbb{P}^{BP}(x_i)+\sum_{j\in A}\Big[\mathbb{P}^{BP}(x_i \mid x_j)-\mathbb{P}^{BP}(x_i)\Big].\end{aligned}$$

$A\subseteq O$ is the set of indices of previously observed variables, and $\mathbb{P}^{BP}(x_i|x_j)$ are approximations of the marginal distributions computed by the *Belief Propagation* (BP) algorithm [12]. Starting the LSDP algorithm with weights all equal to 1, iterated updates will allow to improve the value of the BP-max heuristic. Since computing final reward r^{H+1} is too time consuming using BP algorithm, we use distribution $\widetilde{\mathbb{P}}$ instead, which provides good empirical results.

The second point is the construction of the batch of simulations. Simulating trajectories in the OASMRF problem is complex since, for each transition, one has to simulate observations x_A from the MRF distribution $\mathbb{P}$. This requires to apply the Gibbs Sampling algorithm, which is rather costly, thus severely limiting the size and number of batches that can be constructed. However, larger batches can be constructed if we divide the construction into two phases. First, we simulate, off-line, a batch of hidden maps, $\{x^1,\dots x^p\}$, which will be used for all iterations of the LSDP algorithm. The construction of this batch is done using Gibbs Sampling, and induces a single overhead cost for the whole algorithm. Then, trajectories are easy to simulate: (i) a hidden map is selected, (ii) actions are chosen randomly (ε-greedily with respect to the current policy) and (iii) successor states follow immediately by reading the value of the variables corresponding to the current observation. This second phase of trajectories simulation is fast. Furthermore, simulated trajectories do not have to be stored (only the batch of maps does), thus saving much memory space.

6 EXPERIMENTAL EVALUATION

We present simulated problems to illustrate the gain of using LSDP instead of classical heuristics or RL-based solution algorithms. We compared LSDP to the random heuristic, the BP-max policy, TD(λ) with tabular representation of the Q-function and LSPI. We also compared LSDP to a greedy algorithm based on the *Mutual Information* (MI) criterion [7], with exact computation of the MI.

The OASMRF problem considered is the following. The graph G is a regular grid and $R=O=V$. One variable is observed at each decision step ($L=1$) and sampling costs are null. We considered the following Potts model distribution: $\forall\, x\in\{1,2\}^n$

$$\mathbb{P}_\beta\big(x\big)\propto\exp\Big(\sum_{(i,j)\in E}\beta 1_{\{x_i=x_j\}}\Big)$$

All experiments were run with $\beta = \frac{1}{2}$.

4×4 grid. This small problem was used in the experiments since we were able to compute the corresponding optimal policy and the exact value of any policy. TD(λ) was run with $\lambda = 0.1$, using an ϵ-greedy method for action choice ($\epsilon = 0.1$). The LSDP and LSPI algorithms were run with $\epsilon = 0.9$. For all RL algorithms we used the same batch size. The TD(λ) algorithm was run using 675000 simulated state-action trajectories. We ran LSDP and LSPI with a batch of 100 maps and 6750 iterations. For LSDP the value of the policy obtained at the last iteration of the algorithm was returned, while for LSPI the value of the best policy among all iterations was returned, since the latter algorithm oscillates.

The first conclusion is that the absolute difference between the values of all policies is small: an absolute increase of the percentages of 2.2 at most. We also compared the policies in terms of normalised gain compared to the random one δ_R (Figure 1): the score of a given policy δ is defined as $score1(\delta) = \frac{V(\delta)-V(\delta_R)}{V(\delta^*)-V(\delta_R)}$.

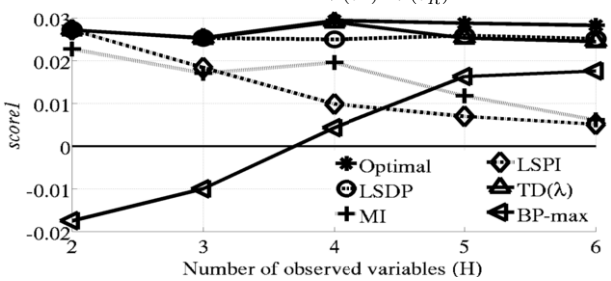

Figure 1. OASMRF problem with 16 variables: $score1$ of LSPD and classical RL-based and heuristic policies.

Among RL algorithms, TD(λ) is the best and LSDP gives very similar results. In comparaison, LSPI shows a poor behaviour, always returning dominated policies. Surprisingly the relative value of the MI policy decreases with the number of observed variables, while the opposite behavior is observed for the BP-max heuristic. The poor performance of the BP-max heuristic with small sample size is explained by the fact that with few observed sites, all sites have similar marginal probabilities, leading to a purely random choice of samples.

10×10 grid. For this problem size, only LSDP, LSPI, BP-max and random policy can be computed. For LSDP and LSPI we used a batch size of 1000 maps and 1000 iterations. The value of a policy was estimated by Monte Carlo approximation. We modified $score1$ into $score2(\delta) = \frac{V(\delta)-V(\delta_R)}{|V(\delta_{BP-max})-V(\delta_R)|}$: since the value of an optimal policy cannot be computed, δ_{BP-max} serves as a reference. Results are displayed on Figure 2.

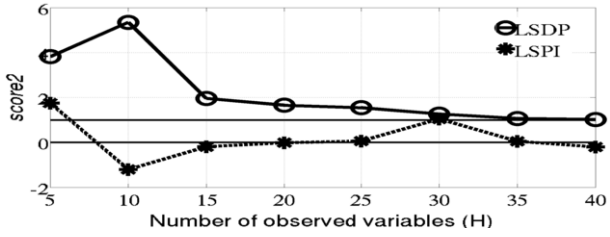

Figure 2. OASMRF problem with 100 variables: $score2$ of LSDP and LSPI policies.

We observed again the poor performance of the LSPI algorithm (even dominated by the random policies, for $H = 10$ to 20). On the contrary, LSDP performs quite better than the BP-max heuristic for small sample sizes. LSDP also performs better than LSPI, in terms of computation time: for $H = 40$, an iteration takes about 7 seconds for LSDP, 77 seconds for LSPI.

Constrained moves problem. We also compared LSDP, BP-max and random policies on a more realistic sampling problem, involving constrained moves on the grid for observing sites. After having observed a site, the agent can only move to distance-2 sites for the following observation.

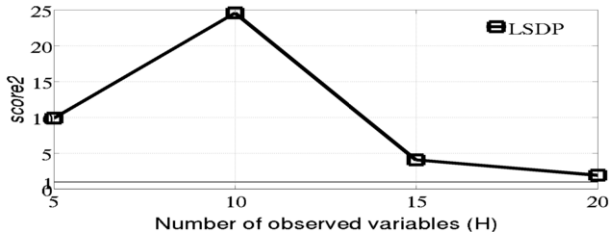

Figure 3. Constrained moves problem with 100 variables: $score2$ of LSDP policy.

We again observed that the absolute difference between all policies remained small (for $H = 10$, the value of the LSDP policy is 61.7 while the value of the heuristic policy is 59.4). LSPI showed the same poor behaviour than in the previous experiment. As we expected, the gain provided by LSDP in terms of relative improvement of the random policy ($H \leq 20$, see Figure 3) is significant when the sample size is small (Figure 3).

7 CONCLUSION

In this article, we have provided a factored MDP model to represent problems of optimal adaptive sampling of spatial processes expressed in the Markov random field framework. We have proved the PSPACE completeness of this problem. Then the MDP model has allowed us to propose an adapted simulation-based solution algorithm, LSDP, combination of a parametrized representation of the Q-function and Dynamic Programming principles.

Comparison of the LSDP algorithm with heuristic algorithms and classical RL algorithms enables us to draw the following conclusions. First, in small problems where the optimal policy can be computed, we notice that the performance of a purely random strategy is quite close to that of the optimal one. This seems to also hold for larger problems, where the estimated value of the random policy remains close to that of the LSDP policy. However, in real-life applications of sampling for mapping, small errors in the reconstruction of maps can lead to significant increases in management costs (think of imperfect mapping and eradication of invasive species, leading to future outbreaks).

Second, for large problems, TD(λ) or exact mutual information are too computationally intensive to apply, and the adaptation of the LSPI approach does not perform well. On the contrary, both BP-max heuristic and the LSDP algorithm provide good results. BP-max is less costly to apply than LSDP. However, it is an ad-hoc method and its performance depends on which form of sampling costs are considered. We can also predict poor performances when the set of observable variables differs from the set of variables of interest in the reconstruction. This limits the applicability of BP-max. In contrast, LSDP can handle different cost functions. It can also easily be

adapted to other definitions of policy value, provided that they can be estimated efficiently from a batch of trajectories. Furthermore, the LSDP algorithm can be applied to general factored finite-horizon MDP, and not only to spatial sampling problems.

LSDP is currently being validated on a real problem of sampling in crop fields for weeds mapping. We also plan to use it to design policies for controlling spatio-temporal systems (e.g. weeds control) and not only for building maps.

8 Appendix

We establish that the GOASMRF problem is PSPACE-complete. Let us define the *state-disambiguation* (SD) problem. We have:

- A set $\Theta = \{\theta_1, \ldots, \theta_l\}$ of possible states of the world and a probability distribution p over Θ.
- A utility function $u : \Theta \to [0; +\infty[$: $u(\theta_i)$ is the utility of discovering that the state of the world is θ_i.
- A set $\mathcal{Q} = \{Q_1, \ldots, Q_r\}$ of queries. $Q_j = \{q_{j1}, \ldots, q_{jm_j}\}$ is a set of subsets of Θ, such that $\bigcup_{1 \leq k \leq m_j} q_{jk} = \Theta$. If the true state of the world is θ_i and Q_j is asked, an answer is chosen (uniformly) randomly among the answers q_{jk} containing θ_i.
- A maximum number N of queries that can be asked and a target real value $G > 0$.

The SD problem consists in deciding whether there exists an adaptive policy, asking at most N queries, that gives expected utility at least G. If $p_\delta(\theta_i)$ denotes the probability of identifying θ_i by using policy δ, the SD problem amounts to deciding whether there exists δ such that $\sum_{1 \leq i \leq l} p(\theta_i) p_\delta(\theta_i) u(\theta_i) \geq G$. It has been shown that SD is PSPACE-hard, even when $N \leq l$ [1].

In order to prove that the GOASMRF problem is PSPACE-complete, we propose a reduction from a SD problem to a GOASMRF problem. Let SD = $(\Theta, u, \mathcal{Q}, N, G)$ be given.

- We build a GOASMRF over variables $X = (\theta, q_1, \ldots, q_r)$. Variables in the GOASMRF problem correspond to the sets in the SD problem: θ takes values in Θ and q_j in Q_j.
- The considered graphical model is a MRF with distribution:

$$\mathbb{P}(X) = \mathbb{P}(\theta) \prod_{j=1}^{r} \mathbb{P}(q_j|\theta),$$

  where $\mathbb{P}(\theta = \theta_i) = p(\theta_i), \forall i = 1..n$ and the conditional probabilities are $\mathbb{P}(q_j = q_{jk}|\theta = \theta_i) = \frac{1}{|\{q_{jk'} \in Q_j, \theta_i \in q_{jk'}\}|}$ if $\theta_i \in q_{jk}$ and $\mathbb{P}(q_j = q_{jk}|\theta = \theta_i) = 0$ else.
- Then, we set $R = \{\theta\}$ and $O = \{q_1, \ldots, q_r\}$: we want to restore the value of variable θ, but can only sample variables q_j.
- Only one site (variable) can be sampled at each of N time steps, and $H = N$.
- Cost function c is set uniformly null ($c(A) = 0, \forall A \subseteq O$).
- Function f_θ is defined as: $f_\theta(\theta_i, 1) = p(\theta_i) u(\theta_i)$ and $f_\theta(\theta_i, \nu) = 0, \forall \theta_i \in \theta, 0 \leq \nu < 1$. We get a reward only when the value of θ_i is known with certainty.

In order to prove that solving the GOASMRF problem we have just defined also solves the SD problem, it is enough to prove that: (i) any policy δ^{SD} in the SD problem has an equivalent policy $\delta^{GOASMRF}$ in the GOASMRF problem, and vice-versa, (ii) any two corresponding policies δ^{SD} and δ^{OASMRF} have identical values in their respective problems.

Point (i) holds since available actions in both frameworks correspond to the same q_j's (queries in SD and variables allowed for sampling in GOASMRF). Then, since allowed observations are the same in both cases and since the depth of both query trees are equal (to N), the set of policies are the same, and these are in direct correspondence in both problems.

For point (ii) note that the two values of a policy δ are defined by:

$$v^{GOASMRF}(\delta) = \sum_{(A, x_A) \in \tau_\delta} \mathbb{P}(x_A) U(A, x_A),$$

$$v^{SD}(\delta) = \sum_{1 \leq i \leq l} p(\theta_i) p_\delta(\theta_i) u(\theta_i).$$

For any strategy δ, let $\tau_\delta^{\theta_i}$ denote the set of branches which, in the SD case, allow to disambiguate set Θ in θ_i. Then it is easy to see that

$$v^{SD}(\delta) = v^{GOASMRF}(\delta) = \sum_{1 \leq i \leq l} \sum_{(A, x_A) \in \tau_\delta^{\theta_i}} p(\theta_i) \mathbb{P}(x_A) u(\theta_i).$$

ACKNOWLEDGEMENTS

Thanks to Bruno SCHERRER and Alain DUTECH for fruitful discussions and to the French National Research Agency which supported this work (LARDONS, ANR-2010-BLAN-0215-04).

REFERENCES

[1] V. Conitzer and T. Sandholm, 'Definition and complexity of some basic metareasoning problems', in *Proc. of the 18th International Joint Conference on Artificial Intelligence (IJCAI'03)*, pp. 1099–1106, (2003).
[2] J. de Gruijter, D. Brus, M. Bierkens, and K. Knotters, *Sampling for Natural Resource Monitoring*, Springer, 2006.
[3] D. Ernst, P. Geurts, and L. Wehenkel, 'Tree-based batch mode reinforcement learning', *J. of Mach. Learn. Research*, **6**, 503–556, (2005).
[4] S. Geman and D. Geman, 'Stochastic relaxation, Gibbs distribution, and the Bayesian restoration of images', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **6**, 721–741, (1984).
[5] L. Kocsis and C. Szepesvári, 'Bandit based monte-carlo planning', in *ECML'06*, pp. 282–293, (2006).
[6] A. Krause and C. Guestrin, 'Optimal value of information in graphical models', *J. of Artificial Intelligence Research*, **35**, 557–591, (2009).
[7] A. Krause, A. Singh, and C. Guestrin, 'Near-optimal sensor placements in Gaussian processes: theory, efficient algorithms and empirical studies', *J. of Machine Learning Research*, **9**, 235–284, (2008).
[8] M. Lagoudakis and R. Parr, 'Least-squares policy iteration', *Journal of Machine Learning Research*, (2003).
[9] A. Chaudhuri M. Fuentes and D. Holland, 'Bayesian entropy for spatial sampling design of environmental data', *Environmental and Ecological Statistics*, **14**, 323–340, (2007).
[10] WG Müller, *Collecting spatial Data*, Springer Verlag, 2007.
[11] D. Ormoneit and S. Sen, 'Kernel-based reinforcement learning', *Machine Learning*, **49**, 161–178, (2002).
[12] J. Pearl, *Probabilistic Reasonning in Intelligent Systems*, Morgan Kaufmann, 1988.
[13] N. Peyrard, R. Sabbadin, and U. F. Niaz, 'Decision-theoretic optimal sampling with hidden Markov random fields', in *ECAI 2010*, (2010).
[14] N. Peyrard, R. Sabbadin, D. Spring, R. Mac Nally, and B. Brook, 'Model-based adaptive spatial sampling for occurrence map construction', *Statistics and Computing*, (2012).
[15] M. Puterman, *Markov Decision Processes : Discrete Stochastic Dynamic Programming*, John Wiley & Sons, Inc, 1994.
[16] E. Rachelson, F. Schnitzler, and L. Wehenkel ans D. Ernst, 'Optimal sample selection for batch-mode reinforcement learning', in *Proc. of the 3rd Int. Conf. on Agent and AI (ICAART'11)*, Rome, Italy, (2011).
[17] R. S. Sutton and A.G. Barto, *Reinforcement Learning : An Introduction*, MIT Press, 1998.
[18] CS. Szepesvári, *Algorithms for Reinforcement Learning*, Morgan and Claypool, 2010.
[19] S. Thompson and G. Seber, *Adaptive sampling*, Series in Probability and Statistics, Wiley, New York, 1996.