Delft University of Technology

Delft Center for Systems and Control

Technical report 11-007

Optimistic planning for sparsely stochastic systems*

L. Buşoniu, R. Munos, B. De Schutter, and R. Babuška

If you want to cite this report, please use the following reference instead:

L. Buşoniu, R. Munos, B. De Schutter, and R. Babuška, "Optimistic planning for sparsely stochastic systems," *Proceedings of the 2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL 2011)*, Paris, France, pp. 48–55, Apr. 2011.

Delft Center for Systems and Control
Delft University of Technology
Mekelweg 2, 2628 CD Delft
The Netherlands
phone: +31-15-278.51.19 (secretary)
fax: +31-15-278.66.79
URL: `http://www.dcsc.tudelft.nl`

*This report can also be downloaded via `http://pub.deschutter.info/abs/11_007.html`

Optimistic Planning for Sparsely Stochastic Systems

Lucian Buşoniu*, Rémi Munos†, Bart De Schutter*, Robert Babuška*
*Delft Center for Systems and Control, Delft University of Technology, the Netherlands
Email: {i.l.busoniu,b.deschutter,r.babuska}@tudelft.nl
†SequeL team, INRIA Lille–Nord Europe, France. Email: remi.munos@inria.fr

***Abstract*—We propose an online planning algorithm for finite-action, sparsely stochastic Markov decision processes, in which the random state transitions can only end up in a small number of possible next states. The algorithm builds a planning tree by iteratively expanding states, where each expansion exploits sparsity to add all possible successor states. Each state to expand is actively chosen to improve the knowledge about action quality, and this allows the algorithm to return a good action after a strictly limited number of expansions. More specifically, the active selection method is *optimistic* in that it chooses the most promising states first, so the novel algorithm is called *optimistic planning for sparsely stochastic systems*. We note that the new algorithm can also be seen as model-predictive (receding-horizon) control. The algorithm obtains promising numerical results, including the successful online control of a simulated HIV infection with stochastic drug effectiveness.**

***Index Terms*—online planning, optimistic planning, Markov decision processes, stochastic systems, model-predictive control.**

I. Introduction

This paper concerns problems in which a nonlinear stochastic system must be optimally controlled in discrete time, so that a cumulative reward signal (the return) is maximized. Such problems arise in many fields, including artificial intelligence, automatic control, computer science, operations research, economics, medicine, etc. They can be modeled as Markov decision processes (MDPs), and due to the generality of this formulation, algorithms that solve MDPs are an extremely important field of research.

In particular, we consider a class of online model-based algorithms that, at each step, look at the current system state and employ the model to predict the system's response to various sequences of actions. Exploiting these predictions, an action that is as good as possible is applied, which results in a new state. The entire cycle then repeats. In computer science such algorithms belong to the planning class [17] and are known as online planning [16], [21] or sometimes lazy planning [12]. While in this paper we use the name 'online planning' and mainly refer to the computer science literature, it must be emphasized that such algorithms are also widely studied in systems and control, where they are known as model-predictive or receding-horizon control [9], [20].

We propose an online planning algorithm that works in finite-action, sparsely stochastic MDPs, in which the random state transitions can only end up in a small number N of possible next states. The algorithm builds a planning (lookahead) tree by taking the current state as the root node and expanding a new state at each iteration. Each state expansion exploits the sparsity of the transitions to add all the possible next states, for all the M discrete actions, as children to the tree. Because the algorithm works online, strict limits on its computational expense are imposed: it must perform at most n expansions. Therefore, each state to expand must be chosen in an active way, so as to improve the knowledge about action quality. An *optimistic* active planning procedure is adopted that expands the most promising states first – i.e., states corresponding to larger *upper bounds* on the possible returns. We call the resulting algorithm *optimistic planning for sparsely stochastic systems* (OPSS).

OPSS is evaluated numerically in two problems: a sparsely stochastic variant of the classical inverted pendulum, where OPSS is also compared to alternative planning algorithms; and a highly challenging problem involving the control of an HIV infection, for the case when the effectiveness of the applied drugs is stochastic.

Many other systems of interest are sparsely stochastic. Such systems usually arise by combining deterministic dynamics with discrete random variables, which could be an intrinsic part of the system, such as failure modes, job arrivals into a resource management system (e.g., elevator scheduling, traffic signal control), etc., or could represent external conditions (disturbances) such as user input, discrete actions of other agents in a multiagent system, etc.

The optimistic principle at the core of our approach is widely used in so-called 'bandit' methods, where it has strong theoretical foundations [2], [5], [11]. Bandits can be understood as a way of solving the exploration-exploitation dilemma, or alternatively as optimizing a stochastic function by taking as few suboptimal samples as possible. The application of bandits to tree exploration is especially relevant to online planning [11]. Such methods could be used directly to plan in MDPs, but they would not exploit the specific structure arising in this context. Instead, planning methods have been developed specifically for MDPs [4], [10], [12], [13], [15], [16], [18], see [21] for a review. Among these, [18] exploits a different notion of sparse stochasticity, in which only some of the states lead to stochastic outcomes. Our novel OPSS method is closest to the optimistic planning (OP) algorithms of [4], [15]. The OP algorithm of [15] is geared for deterministic systems, and in fact OPSS reduces to this algorithm in the deterministic case. Of course, OPSS is more general as it is also applicable to stochastic systems. Compared to the open-loop OP (OLOP) of [4], which is designed for nonsparsely stochastic systems, our approach takes advantage of the sparse stochasticity to find deterministic upper bounds on the returns, rather than upper confidence bounds in high probability, as is done in OLOP.

The online planning approach is different from the value-function and policy search methods usually considered in dynamic programming and reinforcement learning [3], [6], [22]–[24]; the latter methods usually seek a global solution, whereas online planning finds actions on demand, locally for each state where they are needed. Online planning is therefore much less dependent on the state space size. It is also generally suboptimal, while global methods do achieve optimality in some restricted settings. In realistic problems, however, global methods must also use approximation, thereby sacrificing optimality [6].

Next, Section II introduces MDPs and online planning algorithms in more detail. Section III introduces the novel OPSS algorithm, and Section IV presents numerical experiments validating it. Section V summarizes the paper and outlines our future plans for the algorithm.

II. Preliminaries

This section briefly introduces the optimal control problem and the class of algorithms considered in this paper, in the framework of Markov decision processes (MDPs). More details about MDPs and methods to solve them can be found in [3], [6], [22]–[24].

A. Markov decision processes

Consider an MDP with state space X and action space U. Assume for the simplicity of notation that X is countable. The probability that next state x' is reached after action u is taken in state x is $f(x,u,x')$, where $f : X \times U \times X \to [0,1]$ is the transition probability function. After the transition to x', a reward $r' = \rho(x,u,x')$ is received, where $\rho : X \times U \times X \to \mathbb{R}$ is the reward function.

A control policy $h : X \to U$ indicates how actions should be chosen given the state. Denoting by k the discrete time index, the expected infinite-horizon discounted return (for short, value) of state x under a policy h is:

$$V^h(x) = \mathrm{E}_{x_{k+1}\sim f(x_k,h(x_k),\cdot)} \left\{ \sum_{k=0}^{\infty} \gamma^k r_{k+1} \right\} \tag{1}$$

where $x_0 = x$, $r_{k+1} = \rho(x_k, h(x_k), x_{k+1})$, $\gamma \in (0,1)$ is the discount factor, and the notation $x_{k+1} \sim f(x_k, h(x_k), \cdot)$ means that x_{k+1} is drawn from the distribution $f(x_k, h(x_k), \cdot)$. Other types of return can also be used, such as finite-horizon or averaged over time. We call $V^h : X \to \mathbb{R}$ a 'value function'. The goal is to control the system using an optimal policy h^*, so that the value function is maximized for every $x \in X$. This maximal (optimal) value function, denoted by V^*, is unique, so it does not depend on the particular optimal policy.

It is also helpful to consider the values of state-action pairs, rather than just states. For instance, the Q-function of a policy h is the expected value of starting in a given state, applying a given action, and following h thereafter:

$$Q^h(x,u) = \mathrm{E}_{x'\sim f(x,u,\cdot)} \left\{ \rho(x,u,x') + \gamma V^h(x') \right\}$$

If the optimal Q-function Q^* (defined as the Q-function of any optimal policy) is available, an optimal policy h^* can immediately be found by the so-called *greedy* action selection:

$$h^*(x) \in \arg\max_{u\in U} Q^*(x,u) \tag{2}$$

Our algorithm is geared towards problems that satisfy the following assumptions.

Assumption 1: There is a small finite number M of actions, i.e., $|U| = M$. Moreover, after applying any action in any state, the number of reachable next states is at most a small finite N, i.e., $|\{x' \mid f(x,u,x') > 0\}| \leq N$ for any x, u, where $|\cdot|$ denotes set cardinality.

Assumption 2: The rewards are bounded in the interval $[0,1]$, i.e., $\rho(x,u,x') \in [0,1]$ for any x, u, x'.

The first, finite-action part of Assumption 1, while restrictive, is commonly used when solving MDPs, so our algorithm is not unusually restrictive in this way. We call systems that satisfy the second-part of Assumption 1 'sparsely stochastic', since if we were to represent for any fixed x, u the transition probabilities $f(x,u,\cdot)$ as a vector of length $|X|$, this vector would be generally be sparse (because X will generally be large, leading to $|X| \gg N$). Assumption 2 is not restrictive, as any bounded reward function can be normalized to $[0,1]$ by translation and scaling, without changing the optimal policies.

B. Online planning control

The planning algorithms considered in this paper work online and employ a model of the MDP, in the form of the functions f and ρ. At each step k, the model is employed to predict the possible behavior of the system starting from the current state x_k and responding to various sequences of actions. Using these predictions, the algorithm returns an action u_k that is as close to optimal as possible. This action is applied, the system transits to x_{k+1}, and the cycle repeats. The planning algorithm can be identified with a policy $h(x_k) = u_k$ (assuming it chooses actions deterministically, otherwise a stochastic policy must be used).

An important concern in these algorithms is the computational expense at each step, especially if the algorithm must be applied in a real-time fashion. Therefore, following [4], [15], we consider a setting in which this expense must be at most n units, where the exact units will be specified later for our algorithm, but can generally be number of evaluations of the model functions, computation time, number of basic arithmetic operations, etc.

To measure the quality of a planning algorithm h, the so-called 'simple regret' can be used, which at every state x is defined by:

$$\begin{aligned} \mathcal{R}^h(x) &= Q^*(x,h^*(x)) - Q^*(x,h(x)) \\ &= \max_{u\in U} Q^*(x,u) - Q^*(x,h(x)) \end{aligned} \tag{3}$$

i.e., the loss incurred by choosing $h(x)$ and then acting optimally, with respect to acting optimally from the first step. Note the regret is always nonnegative, and an optimal policy

achieves a regret of 0. Using $\mathcal{R}$ is motivated by the following result [15]:

$$\left\|V^* - V^h\right\|_\infty \leq \frac{\left\|\mathcal{R}^h\right\|_\infty}{1-\gamma}$$

which says that if h has a small regret $\mathcal{R}^h$, then the actual values V^h it obtains are close to the optimal values V^*.

III. Optimistic planning for sparsely stochastic systems

In this paper, we introduce a novel algorithm called *optimistic planning for sparsely stochastic systems* (OPSS). OPSS builds a planning (lookahead) tree starting from a root node that contains the state where an action must be chosen. At each iteration, the algorithm actively selects a leaf node (a state) and expands it, by exploiting the sparsity of the dynamics to generate all the one-step successor states for all possible actions. The computational unit consists of using the model functions f, ρ to generate these successors for a single state. Due to Assumption 1, there are at most NM successors. The algorithm stops growing the tree after n expansions and returns an action chosen on the basis of the final tree.

The procedure to select nodes for expansion is crucial: it should efficiently exploit the available computational budget n to obtain a regret (3) that is as small as possible. To this end, we design an selection procedure that is *optimistic*, in the sense of assuming the best possible optimal values compatible with the planning tree generated so far.

To formalize the criteria by which nodes are expanded and the final action is chosen, let us first introduce some notation:

- The entire tree is denoted by $\mathcal{T}$, and the set of leaf (unexpanded) nodes by $\mathcal{S}$.
- A node of the tree is identified with its associated state x. A child node is denoted x', and also has the meaning of next state. When leaf nodes must be distinguished, they are denoted by s. In the remainder of this section, we will prefer saying nodes rather than states.
- Because everything happens at the current time step, the time index k is dropped and the subscript of x and u is reused to indicate the depth d of a node in the tree, whenever this depth must be considered explicitly. So, x_0 is the root node, where an action must eventually be chosen, and x_d is a node at depth d. Of course, d still retains the meaning of time, since a node at depth d occurs after d transitions along a simulated trajectory starting from the root node. A function $D(x)$ is introduced that finds the depth of a node x.

Figure 1 shows an example of a planning tree.

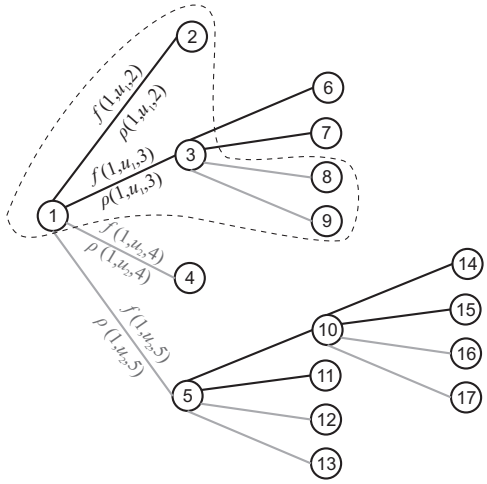

Fig. 1. A planning tree example for the case $N = M = 2$, after 4 expansions. Each node (state) is represented by an encircled number, black edges correspond to transitions resulting from the first action u_1, and gray edges to u_2. It is useful to label each edge by the transition probability from the parent to the child and by the associated reward (exemplified for the first level only). The dashed line encloses an optimistic subtree example (see Section III-A below), for the case in which $b(1, u_1) > b(1, u_2)$ and $b(3, u_1) < b(3, u_2)$.

A. Expansion criterion

For each $x \in \mathcal{T}$ and $u \in U$, define the *b-values* $b(x, u)$ recursively, starting from the leaf nodes, as follows:

$$b(s,u) = \frac{1}{1-\gamma}, \quad \forall s \in \mathcal{S}, u \in U$$

$$b(x,u) = \sum_{x'} f(x,u,x') \left[\rho(x,u,x') + \gamma \max_{u' \in U} b(x',u')\right], \quad \forall s \in \mathcal{T} \setminus \mathcal{S}, u \in U$$

where x' ranges through all the children of x which are reachable by taking action u. Each b-value $b(x, u)$ is an upper bound for the optimal Q-value $Q^*(x, u)$. This is immediately clear at the leaves, where the value $\frac{1}{1-\gamma}$ is an upper bound for any Q-value, because the rewards are in $[0, 1]$ and a discount factor $\gamma < 1$ is used. By backward induction, it is true at any inner node: since the b-values of the node's children are upper bounds on their Q-values, the resulting b-values of this node are upper bounds on its own Q-values. Note that although the bounds are loose at the leaves, they improve higher in the tree, which is what will matter for the algorithm's performance.

To obtain a set of candidate nodes for expansion, first an *optimistic subtree* is recursively built by starting from the root and selecting at each node x only its children associated to an optimistic action, i.e., a greedy action (2) in the b-values:

$$u^\dagger(x) \in \arg\max_{u \in U} b(x,u)$$

Ties can be broken in any way, but to make the algorithm predictable, they should preferably be broken deterministically, e.g., always in favor of the first node that was added to the tree. This procedure is optimistic because it uses b-values (upper bounds) as if they were optimal Q-values. Denote the

optimistic subtree by $\mathcal{T}^\dagger$, and its leaves by $\mathcal{S}^\dagger$. All these leaves are candidates for expansion. See Figure 1 for an example of an optimistic subtree.

To choose one leaf node to expand among the candidates $\mathcal{S}^\dagger$, we propose to maximize the *potential decrease* of the b-value $b(x_0, u^\dagger(x_0))$, i.e., of the upper bound on the optimal value of the root state. So, the criterion strives to maximally improve the knowledge about this optimal value.

The b-value considered can be written more explicitly as an expected optimistic return obtained along the paths from the root to all the leaf nodes in the optimistic subtree:

$$b(x_0, u^\dagger(x_0)) = \sum_{s \in \mathcal{S}^\dagger} \mathrm{P}(s) \left[\bar{R}(s) + \frac{\gamma^{D(s)}}{1-\gamma} \right] \tag{4}$$

where $\mathrm{P}(s)$ is the probability to reach s and $\bar{R}(s)$ is the discounted sum of rewards accumulated along the path. Denote the path by $x_0^s, x_1^s, \ldots, x_{D(s)}^s$ for a given s; of course, x_0^s is always x_0 and $x_{D(s)}^s$ is s itself. Then:

$$\mathrm{P}(s) = \prod_{d=0}^{D(s)-1} f(x_d^s, u^\dagger(x_d^s), x_{d+1}^s)$$

$$\bar{R}(s) = \sum_{d=0}^{D(s)-1} \gamma^d \rho(x_d^s, u^\dagger(x_d^s), x_{d+1}^s)$$

Consider the contribution of a single leaf node s to (4): $\mathrm{P}(s)\,[\bar{R}(s)+\gamma^{D(s)}/(1-\gamma)]$. If this leaf node were expanded, its contribution would decrease the most if the rewards along the transitions to all the new children nodes were 0. In that case, its updated contribution would be $\mathrm{P}(s)\,[\bar{R}(s)+\gamma^{D(s)+1}/(1-\gamma)]$, and its contribution would have decreased by:

$$\mathrm{P}(s) \left[\bar{R}(s) + \frac{\gamma^{D(s)}}{1-\gamma} - \bar{R}(s) - \frac{\gamma^{D(s)+1}}{1-\gamma} \right] = \mathrm{P}(s)\,\gamma^{D(s)}$$

So, finally, the rule for selecting a node to expand maximizes this potential decrease over the optimistic leaves:

$$\arg\max_{s \in \mathcal{S}^\dagger} \mathrm{P}(s)\,\gamma^{D(s)} \tag{5}$$

preferably breaking ties deterministically, for the same reason as above.

B. Action selection at the root

Similarly to the b-values, define the *ν-values* $\nu(x, u)$:

$$\nu(s, u) = 0, \quad \forall s \in \mathcal{S}, u \in U$$

$$\nu(x, u) = \sum_{x'} f(x, u, x') \left[\rho(x, u, x') + \gamma \max_{u' \in U} \nu(x', u') \right], \quad \forall s \in \mathcal{T} \setminus \mathcal{S}, u \in U$$

The difference from the b-values is that ν-values start with 0 at the leaves. Then, the root action, which is the final result of the algorithm, is selected with:

$$u_0 \in \arg\max_{u \in U} \nu(x_0, u) \tag{6}$$

breaking ties deterministically as before.

C. OPSS algorithm

The complete OPSS algorithm is shown in high-level pseudocode as Algorithm 1. Note that b-values, ν-values, path probabilities $\mathrm{P}(s)$, and partial returns $\bar{R}(s)$ do not have to be recomputed from scratch at each iteration, but can all be efficiently updated as new nodes are added.

Algorithm 1 OP for sparsely stochastic systems

Input: state x_0, model f, ρ, computational budget n

1: $\mathcal{T}_1 = \{x_0\}$
2: **for** $\ell = 1, \ldots, n$ **do**
3: build $\mathcal{T}_\ell^\dagger$, the optimistic subtree of $\mathcal{T}_\ell$
4: select node to expand: $s_\ell \in \arg\max_{s \in \mathcal{S}_\ell^\dagger} \mathrm{P}(s)\,\gamma^{D(s)}$
5: expand s_ℓ, obtaining $\mathcal{T}_{\ell+1}$
6: **end for**

Output: $u_0 \in \arg\max_{u \in U} \nu(x_0, u)$

As previously mentioned, optimistic algorithms for MDPs have been developed before. The most closely related algorithms are OP for deterministic systems [15] and open-loop OP (OLOP) [4], which works in stochastic systems. When the system is deterministic, OPSS reduces to OP for deterministic systems. Indeed, in that case the optimistic subtree reduces to a single path, since at every node x there is a single, deterministic successor for the optimistic action $u^\dagger(x)$. The node at the end of this path is the one selected for expansion by both OPSS and OP for deterministic systems, and the problem of selecting between candidate optimistic nodes does not arise in the deterministic case. In the stochastic case, we solve this problem using the expansion criterion (5). Note that the b-values and ν-values in OPSS have counterparts with similar meanings in OP for deterministic systems.

OLOP works for general, *nonsparsely* stochastic systems with finitely many actions, but plans in 'open loop', using only sequences of actions. At each iteration, such a sequence is applied in simulation, using a generative model (i.e., one that only generates random transitions without offering access to their distribution). The resulting random rewards are used to update upper confidence bounds *in high probability* on the returns, for every subsequence belonging to the chosen sequence. At the next iteration, the bounds are used to choose a promising next sequence to simulate, see [4] for details. The actual underlying sequences of states are never explicitly considered.[1] In contrast, OPSS does consider the state transitions. In fact, taking advantage of the sparse nature of the MDP, OPSS uses all possible transitions from each expanded node to find *deterministic* (exact) upper bounds: the b-values. Of course, to compute all the transitions OPSS requires access to their probability distribution.

OPSS can also be seen as a type of branch-and-bound optimization over action sequences; more precisely, over a space $\mathcal{H}$ consisting of all the action assignments to states

[1] Note the actual interaction with the system happens in closed loop for OLOP as well as OPSS, since both algorithms take into account the state at each time step.

along all possible random trajectories starting in x_0. Each node expansion corresponds to splitting a subset of $\mathcal{H}$ with the largest upper bound into M sets, and the criterion (5) selects for splitting the 'longest edge' of this set.

D. A uniform planning algorithm

As a baseline algorithm against which to compare, we will use the strategy of always expanding a leaf node with the smallest depth [15]. The resulting uniform planning procedure is shown in Algorithm 2.

Algorithm 2 Uniform planning for sparsely stochastic systems

Input: state x_0, model f, ρ, computational budget n

1: $\mathcal{T}_1 = \{x_0\}$
2: **for** $\ell = 1, \ldots, n$ **do**
3: select node to expand: $s_\ell \in \arg\min_{s \in \mathcal{S}_\ell} D(s)$
4: expand s_ℓ, obtaining $\mathcal{T}_{\ell+1}$
5: **end for**

Output: $u_0 \in \arg\max_{u \in U} \nu(x_0, u)$

IV. Experimental study

We report the results of numerical experiments validating OPSS. First, the behavior of OPSS is studied in a relatively simple inverted pendulum problem. Then, OPSS is applied to the highly challenging problem of controlling an infection with the human immunodeficiency virus (HIV).

A. Inverted pendulum swingup

Using the problem of swinging up an underactuated inverted pendulum, the behavior of OPSS is studied as a function of the computational budget n provided. OPSS is compared to the baseline, uniform planning algorithm, and to OLOP.

Inverted pendulum problem

The inverted pendulum consists of a weight of mass m attached to an actuated link that rotates in a vertical plane (see Figure 2). The available power is taken insufficient to push the pendulum up in a single rotation from every initial state. Instead, from certain states (e.g., pointing down), the pendulum needs to be swung back and forth to gather energy, prior to being pushed up and stabilized.

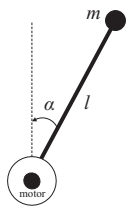

Fig. 2. Inverted pendulum schematic.

A continuous-time model of the pendulum dynamics is:

$$\ddot{\alpha} = 1/J \cdot [mgl \sin(\alpha) - b\dot{\alpha} - K^2\dot{\alpha}/R + Ku/R]$$

where $J = 1.91 \cdot 10^{-4}\,\mathrm{kgm}^2$, $m = 0.055\,\mathrm{kg}$, $g = 9.81\,\mathrm{m/s}^2$, $l = 0.042\,\mathrm{m}$, $b = 3 \cdot 10^{-6}\,\mathrm{Nms/rad}$, $K = 0.0536\,\mathrm{Nm/A}$, $R = 9.5\,\Omega$. The angle α varies in the interval $[-\pi, \pi]\,\mathrm{rad}$, with $\alpha = 0$ pointing up, and 'wraps around' so that e.g. a rotation of $3\pi/2$ corresponds to $\alpha = -\pi/2$. The state is $x = [\alpha, \dot{\alpha}]^\top$. The velocity $\dot{\alpha}$ is restricted to $[-15\pi, 15\pi]\,\mathrm{rad/s}$, using saturation. The sampling time is $T_\mathrm{s} = 0.05\,\mathrm{s}$, and the discrete-time transitions are obtained by numerically integrating the continuous-time dynamics between consecutive time steps.

The control action is limited to $[-3, 3]\,\mathrm{V}$ (insufficient to push up the pendulum in one go), and additionally an unreliable actuator is modeled that only applies the intended action u with probability 0.6, and applies an action with smaller magnitude, $0.7u$, with probability 0.4 (when the intended action is 0 it remains 0 with probability 1). This corresponds to a sparsely stochastic MDP with $N = 2$. The actions are discretized into the set $U = \{-3, 0, 3\}$, so that $M = 3$.

The goal is to stabilize the pendulum in the unstable equilibrium $x = 0$ (pointing up), and is expressed by the *unnormalized* rewards:

$$r = \rho_{\mathrm{unnorm}}(x, u, x') = -x^\top Q_{\mathrm{rew}} x - R_{\mathrm{rew}} u^2$$
$$\text{where: } Q_{\mathrm{rew}} = \mathrm{diag}[5, 0.1],\ R_{\mathrm{rew}} = 1$$

Here, Q_{rew} is chosen to penalize nonzero values of the two state variables to a similar extent, given their relative magnitudes; and R_{rew} penalizes energy consumption, to a smaller extent than the state deviations. Using the known bounds on the state and action variables, this reward function is normalized to the interval $[0, 1]$. The discount factor is $\gamma = 0.95$, sufficiently large to lead to a good control policy.

This problem is challenging for a planning algorithm such as OPSS, because the necessary swing-up trajectories must be planned over a relatively long horizon, and solutions that seem optimal over a short horizon will not work (instead, they will just push the pendulum in one direction without being able to swing it up).

Results and discussion

The performance of OPSS (Algorithm 1) is studied and compared to uniform planning (Algorithm 2) and OLOP [4]. For OPSS and uniform planning, the computational budget n varies in the set $\{100, 200, \ldots, 1000\}$. OLOP has a different computational unit, consisting of simulating a single random transition instead of NM such transitions, so for fairness it is allowed $NM = 6n$ transitions.[2]

To obtain a global performance measure, all algorithms are applied in an offline fashion, to find actions for the states on the grid:

$$X_0 = \left\{-\pi, \tfrac{-150\pi}{180}, \tfrac{-120\pi}{180}, \ldots, \pi\right\} \times \{-15\pi, -14\pi, \ldots, 15\pi\}$$

Since an exact optimal solution for the inverted pendulum problem is not known, in order to approximate the regret

[2]Note that instead of the theoretical OLOP algorithm of [4], we use a variant more amenable to practical implementation, which like OPSS relies on developing planning trees.

(3), a near-optimal solution is computed instead. To this end, the fuzzy Q-iteration algorithm [7] is modified to work for the sparsely stochastic systems considered in this paper, and applied to the inverted pendulum using a very accurate approximator over the state space.

Figure 3, top reports the (approximate) regret of the three algorithms, averaged over the set X_0. As expected, OPSS is better than uniform planning, since it expands the planning trees in a smart way. As Figure 3, middle shows, this results in much deeper trees than for uniform planning. Less expected is that, despite its strong theoretical guarantees, OLOP works poorly, similarly to uniform planning. This happens because the computational budgets considered do not allow OLOP to sufficiently decrease the upper confidence bounds on the returns; any advantage OLOP may have can only manifest for larger budgets. Because the algorithms simulate a similar number of transitions, their execution times are similar (Figure 3, bottom). Note that with these execution times the algorithms would not yet be applicable in real-time; a faster implementation than our proof-of-concept Matlab program is needed for that.

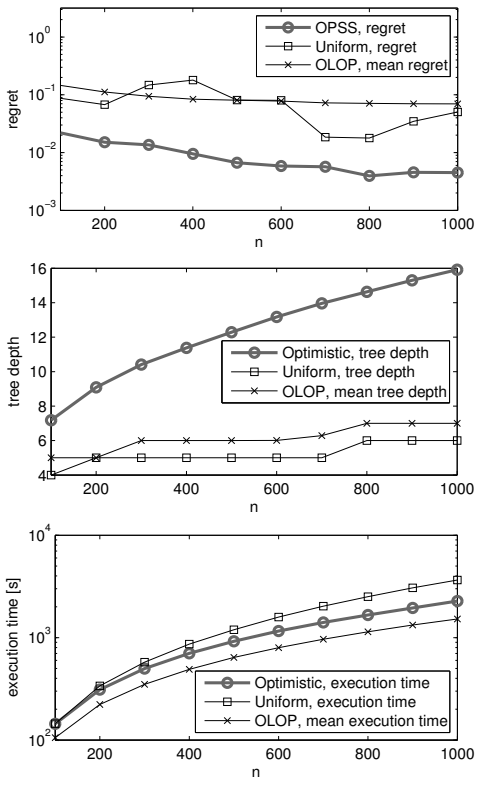

Fig. 3. Comparison between OPSS and uniform planning: average regret over X_0 (top), average tree depth over X_0 (middle), execution time (bottom). As the results of OLOP depend on particular realizations of stochastic trajectories, this algorithm is run 10 times and mean results are reported (the 95% confidence regions are too tight to be visible at this scale).

Figure 4 shows the actions found by the OPSS and OLOP for X_0, when n is 600. For comparison, the near-optimal policy found by fuzzy Q-iteration is also shown. OPSS provides a much better approximation of the optimal policy than OLOP, which e.g. shows no trace of the destabilizing actions required for a successful swingup (these actions are visible in the fuzzy Q-iteration policy as 'inverted' patches in the center-left and center-right regions of the figure, for $\alpha \approx -\pi, \pi$ and $\dot{\alpha} \approx 0$).

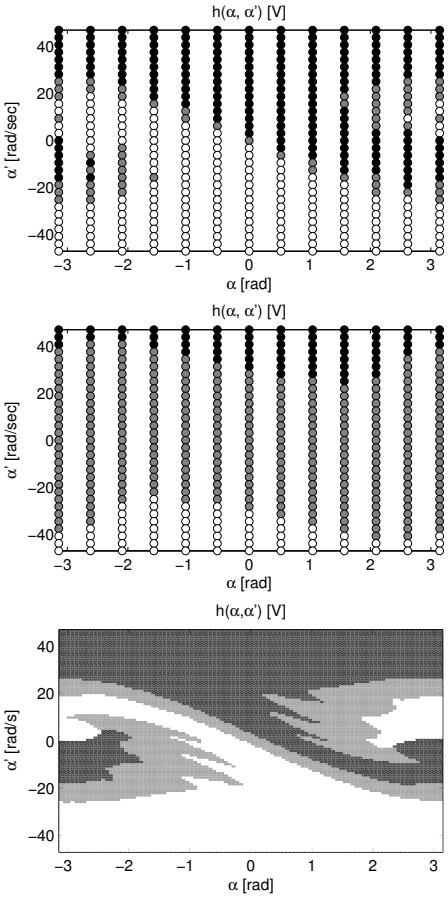

Fig. 4. Actions found by OPSS (top) and OLOP (middle) for $n = 600$, compared to a near-optimal policy (bottom). Black represents -3 V, gray 0 V, and white 3 V.

B. HIV infection control

Next, the highly challenging problem of controlling the treatment of a simulated HIV infection is considered.

HIV infection control problem

Prevalent HIV treatment strategies involve two types of drugs that will generically be called here 'drug 1' and 'drug 2', without going into details. The negative side effects of these drugs in the long term motivate the investigation of optimal strategies for their use. One such strategy involves so-called structured treatment interruptions (STI), where the patient is cycled on and off drugs, see e.g. [1]. In some remarkable cases, STI strategies eventually allowed the patients to control the infection in the absence of treatment [19].

The HIV infection dynamics are described by a six-dimensional nonlinear model with the state vector $x = [T_1, T_2, T_1^{\mathrm{t}}, T_2^{\mathrm{t}}, V, E]^\top$, where:[3]

[3]For the model equations and parameters, see [1].

- $T_1 \geq 0$ and $T_2 \geq 0$ are the counts of healthy type 1 and type 2 target cells [cells/ml].
- $T_1^{\mathrm{t}} \geq 0$ and $T_2^{\mathrm{t}} \geq 0$ are the counts of infected type 1 and type 2 target cells [cells/ml].
- $V \geq 0$ is the number of free virus copies [copies/ml].
- $E \geq 0$ is the number of immune response cells [cells/ml].

In STI, the two drugs are independently either fully administered (they are 'on'), or not at all (they are 'off'); thus there are two binary control variables u_1 and u_2, leading to $M = 4$. Two additional variables ϵ_1 and ϵ_2 represent the effectiveness of the two drugs, and are algebraically related to $u = [u_1, u_2]^\top$ (so they do not enter the state signal). Because it is not clinically feasible to change the treatment daily, the state is measured and the drugs are switched on or off once every 5 days [1]. So, the system is controlled in discrete time with a sampling time of 5 days – which means that plenty of time is available to optimize each control decision, an ideal setting for the online planning type of algorithms considered here.

Previous works using this model to derive near-optimal STI control [1], [8], [14] assumed a one-to-one mapping between drug application and effectiveness, so that whenever a drug is fully applied, its effectiveness is equal to some maximum value. This is not a realistic assumption, and here we relax it by introducing a stochastic relationship between u and ϵ:

$$\epsilon_1 = \begin{cases} 0 & \text{with probability } 1, \text{ if } u_1 = 0 \\ 0.77 & \text{with probability } 0.5, \text{ if } u_1 = 1 \\ 0.63 & \text{with probability } 0.5, \text{ if } u_1 = 1 \end{cases}$$

$$\epsilon_2 = \begin{cases} 0 & \text{with probability } 1, \text{ if } u_2 = 0 \\ 0.33 & \text{with probability } 0.5, \text{ if } u_2 = 1 \\ 0.27 & \text{with probability } 0.5, \text{ if } u_2 = 1 \end{cases}$$

So, depending on the action u, there can be up to $N = 4$ possible outcomes. Note that the expected values of ϵ_1 and ϵ_2 when the drugs are applied are, respectively, 0.7 and 0.3, equal to their deterministic values in [1], [8], [14].

The system has three uncontrolled equilibria. The *uninfected* equilibrium $x_{\mathrm{n}} = [1000000, 3198, 0, 0, 0, 10]^\top$ is unstable: as soon as V becomes nonzero due to the introduction of virus copies, the patient becomes infected and the state drifts away from x_{n}. More interesting are the *unhealthy* equilibrium $x_{\mathrm{u}} = [163573, 5, 11945, 46, 63919, 24]^\top$, which is stable and represents a patient with a very low immune response, for whom the infection has reached dangerous levels; and the *healthy* equilibrium $x_{\mathrm{h}} = [967839, 621, 76, 6, 415, 353108]^\top$, which represents a patient whose immune system controls the infection without the need of drugs. This latter equilibrium, although stable, has a very small basin of attraction. Ideally, an STI control strategy would drive the state into this basin of attraction so that the patient's immune system can take over.

We consider the problem of using STI from the initial state x_{u} such that the immune response of the patient is maximized and the number of virus copies is minimized, while also penalizing the quantity of drugs administered, to account for their side effects. The unnormalized reward function is [1]:

$$\rho_{\mathrm{unnorm}}(x, u, x') = -QV - R_1\epsilon_1^2 - R_2\epsilon_2^2 + SE \qquad (7)$$

where $Q = 0.1, R_1 = R_2 = 20000, S = 1000$. The term $-QV$ penalizes the amount of virus copies, $-R_1\epsilon_1^2$ and $-R_2\epsilon_2^2$ penalize drug use, while SE rewards the amount of immune response. Using some conservative bound estimates on the state variables, this reward is normalized to the interval $[0, 1]$.

Results and discussion

The results of applying OPSS to control the system online starting from x_{u}, with a computational budget of $n = 3000$ at each time step, are shown in Figure 5. As it was hoped for, the algorithm eventually stops administering drugs ($u_1 = u_2 = 0$), and the state slowly converges to the healthy equilibrium x_{h}, associated with a very strong immune response (E large). This solution is better than our previous one in [8], which keeps one drug on in steady state. It is similar in nature to the solutions in [1], [14], but addresses the more challenging case of stochastic drug effectiveness. We also applied uniform planning and OLOP to this problem, with poorer results than OPSS; graphs are not provided here due to space limitations.

The CPU time required by OPSS to plan an action for each state was around $350\,\mathrm{s}$ in our Matlab implementation – significantly smaller than the decision interval of 5 days, which means that the algorithm would easily satisfy real-time constraints for this problem.

V. Conclusions

This paper has introduced a novel online planning algorithm for MDPs with sparsely stochastic transitions. The algorithm builds a planning tree of states by actively choosing at each iteration which state to expand, so that good knowledge about the quality of actions is obtained after at most n expansions. The active state selection method exploits the optimistic planning principle [4], [15], so the novel algorithm is called *optimistic planning for sparsely stochastic systems*. The new algorithm has obtained very promising numerical results, including the successful online control of a (simulated) HIV infection under stochastic drug effectiveness.

The most important next step is the theoretical analysis of the algorithm, in particular, deriving bounds on the simple regret (3) at any state as a function of the computational budget n, the sparsity N, and the number of discrete actions M. On the empirical side, a comparison with other online planning techniques for stochastic systems would be very interesting. One important practical point of improvement is reusing the information derived at previous time steps; this can be done e.g. by reusing subtrees, or by developing a global 'approximate b-function' that compactly represents the existing knowledge about the upper bounds. Information reuse should allow the decrease of the computational budget n without sacrificing performance.

References

[1] B. Adams, H. Banks, H.-D. Kwon, and H. Tran, "Dynamic multidrug therapies for HIV: Optimal and STI control approaches," *Mathematical Biosciences and Engineering*, vol. 1, no. 2, pp. 223–241, 2004.

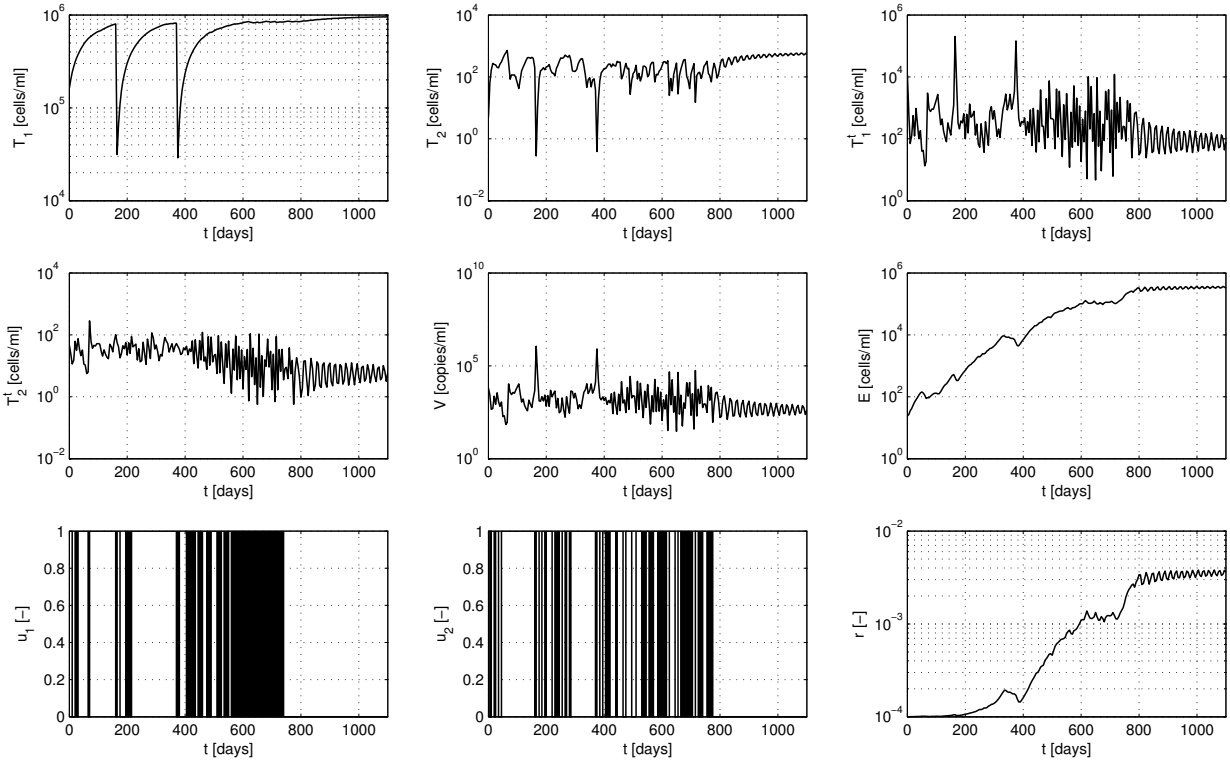

Fig. 5. Trajectory of HIV system controlled online with OPSS. The trajectories of the six system states are shown on the top and middle rows, while the two applied actions and the obtained rewards are shown on the bottom row. (Note that, because the bounds used to normalize the rewards are conservative, the normalized rewards are small in magnitude.)

[2] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, vol. 47, no. 2-3, pp. 235–256, 2002.

[3] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, 3rd ed. Athena Scientific, 2007, vol. 2.

[4] S. Bubeck and R. Munos, "Open loop optimistic planning," in *Proceedings 23rd Annual Conference on Learning Theory (COLT-10)*, Haifa, Israel, 27–29 June 2010, pp. 477–489.

[5] S. Bubeck, R. Munos, G. Stoltz, and C. Szepesvári, "Online optimization in X-armed bandits," in *Advances in Neural Information Processing Systems 21*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds. MIT Press, 2009, pp. 201–208.

[6] L. Buşoniu, R. Babuška, B. De Schutter, and D. Ernst, *Reinforcement Learning and Dynamic Programming Using Function Approximators*, ser. Automation and Control Engineering. Taylor & Francis CRC Press, 2010.

[7] L. Buşoniu, D. Ernst, B. De Schutter, and R. Babuška, "Approximate dynamic programming with a fuzzy parameterization," *Automatica*, vol. 46, no. 5, pp. 804–814, 2010.

[8] ——, "Cross-entropy optimization of control policies with adaptive basis functions," *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics*, vol. 41, no. 1, 2011, accepted for publication, available online.

[9] E. F. Camacho and C. Bordons, *Model Predictive Control*. Springer-Verlag, 2004.

[10] H. S. Chang, M. C. Fu, J. Hu, and S. I. Marcus, *Simulation-Based Algorithms for Markov Decision Processes*. Springer, 2007.

[11] P.-A. Coquelin and R. Munos, "Bandit algorithms for tree search," in *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence (UAI-07)*, Vancouver, Canada, 19–22 July 2007, pp. 67–74.

[12] B. Defourny, D. Ernst, and L. Wehenkel, "Lazy planning under uncertainties by optimizing decisions on an ensemble of incomplete disturbance trees," in *Recent Advances in Reinforcement Learning*, ser. Lecture Notes in Computer Science, S. Girgin, M. Loth, R. Munos, P. Preux, and D. Ryabko, Eds. Springer, 2008, vol. 5323, pp. 1–14.

[13] ——, "Planning under uncertainty, ensembles of disturbance trees and kernelized discrete action spaces," in *Proceedings 2009 IEEE International Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL-09)*, Nashville, US, 30 March – 2 April 2009, pp. 145–152.

[14] D. Ernst, G.-B. Stan, J. Gonçalves, and L. Wehenkel, "Clinical data based optimal STI strategies for HIV: A reinforcement learning approach," in *Proceedings 45th IEEE Conference on Decision & Control*, San Diego, US, 13–15 December 2006, pp. 667–672.

[15] J.-F. Hren and R. Munos, "Optimistic planning of deterministic systems," in *Proceedings 8th European Workshop on Reinforcement Learning (EWRL-08)*, Villeneuve d'Ascq, France, 30 June – 3 July 2008, pp. 151–164.

[16] M. J. Kearns, Y. Mansour, and A. Y. Ng, "A sparse sampling algorithm for near-optimal planning in large Markov decision processes," *Machine Learning*, vol. 49, no. 2-3, pp. 193–208, 2002.

[17] S. M. La Valle, *Planning Algorithms*. Cambridge University Press, 2006.

[18] M. Likhachev, G. J. Gordon, and S. Thrun, "Planning for Markov decision processes with sparse stochasticity," in *Advances in Neural Information Processing Systems 17*. MIT Press, 2004.

[19] J. Lisziewicz, E. Rosenberg, and J. Liebermann, "Control of HIV despite the discontinuation of antiretroviral therapy," *New England Journal of Medicine*, vol. 340, pp. 1683–1684, 1999.

[20] J. M. Maciejowski, *Predictive Control with Constraints*. Prentice Hall, 2002.

[21] L. Péret and F. Garcia, "Online resolution techniques," in *Markov Decision Processes in Artificial Intelligence*, O. Sigaud and O. Buffet, Eds. Wiley, 2010, ch. 6, pp. 153–183.

[22] O. Sigaud and O. Buffet, Eds., *Markov Decision Processes in Artificial Intelligence*. Wiley, 2010.

[23] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[24] Cs. Szepesvári, *Algorithms for Reinforcement Learning*. Morgan & Claypool Publishers, 2010.