Optimistic Planning for Sparsely Stochastic Systems

Lucian Buşoniu, Rémi Munos
Team SequeL
INRIA Lille-Nord Europe, France

Bart De Schutter, Robert Babuška
Delft Center for Systems and Control
Delft University of Technology, the Netherlands

Abstract

We describe 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 the most promising states are expanded first, in an *optimistic* procedure aiming to return a good action after a strictly limited number of expansions. The novel algorithm is called *optimistic planning for sparsely stochastic systems*.

Introduction

We desribe an online planning algorithm for Markov decision processes (MDPs). In particular, we consider finite-action, sparsely stochastic MDPs, in which the random state transitions can only end up in a small number N of possible next states. Many systems of interest are sparsely stochastic, e.g. when deterministic dynamics are combined with discrete random variables such as user input.

At each step, the algorithm builds a planning tree by taking the current state as the root node and then expanding new states. Each expansion exploits the sparsity of the transitions to add all the possible next states, for all the M discrete actions, as children to the planning tree. The algorithm works with a strict limit of n expansions, so each state to expand must be carefully chosen. An *optimistic* 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). We evaluate OPSS empirically for the inverted pendulum, comparing it to alternative planning algorithms.

To formalize the class of problems addressed, we denote by $x \in X$ the (countable) states, by $u \in U$ the actions, by $f(x, u, x')$ the stochastic dynamics (probability of reaching x' by applying u in x), and by r the rewards, computed with the reward function $\rho(x, u, x')$. The goal is to maximize the expected discounted sum of future rewards (return), with discount factor $\gamma \in (0, 1)$.

The following are required. (i) There is a small finite number M of actions. (ii) After applying any action in any state, the number of next states reachable with positive probability is at most a small finite N. (iii) The rewards are bounded in the interval $[0, 1]$. The crucial point here is (ii), i.e., that the system is 'sparsely stochastic': if we were to represent for some x, u the transition probabilities $f(x, u, \cdot)$ as a vector of length $|X|$, this vector would be sparse because $|X| \gg N$. Part (i) is quite standard, while (iii) can always be ensured by translating and scaling the (bounded) reward function.

We measure algorithm quality by the 'simple regret', that is, the loss incurred by choosing the action u_{algo} given by the algorithm and then acting optimally, with respect to acting optimally from the first step:

$$\mathcal{R}(x) = \max_{u \in U} Q^*(x, u) - Q^*(x, u_{\text{algo}}) \tag{1}$$

where Q^* is the optimal Q-function.

Algorithm

The algorithm builds a tree $\mathcal{T}$ starting from a root containing the state from which we must plan, x_0, and then expanding at each iteration a node from the set of leaves $\mathcal{S}$. Each node is identified with its state x; a child node is denoted x' and has the meaning of next state; leaves are denoted s. Node depth is $d(x)$. Each expansion consists of generating and adding all the N one-step successor states of the node-to-expand, for all M actions. The algorithm stops growing the tree after n expansions and returns an action chosen on the basis of the final tree.

The procedure for expanding nodes employs upper bounds b and lower bounds ν on the values of each node x (the value of x is the best return achievable over paths starting at x_0 and passing through x). These bounds are found starting from the leaves, e.g. for the b-values:

$$b(s) = R(s) + \frac{\gamma^{d}(s)}{1-\gamma}, \quad b(x) = \max_{u \in U} \sum_{x'} f(x, u, x') b(x')$$

where $R(s)$ is the partial return accumulated along the path to s, and x' ranges through all the children of x reachable by taking action u. To get ν, simply remove the term $\frac{\gamma^{d}(s)}{1-\gamma}$.

To obtain a set of candidate nodes for expansion, an *optimistic subtree* $\mathcal{T}^\dagger$ is recursively built, by starting from the root and selecting at each node x only children associated to a greedy action in the b-values: $u^\dagger(x) \in \arg\max_{u \in U} \sum_{x'} f(x, u, x') b(x')$. This procedure is optimistic because it uses upper bounds as if they were optimal

values. Among the leaves of this subtree, the node to expand is selected using: $\arg\max_{s\in\mathcal{S}^\dagger} \mathrm{P}(s)\,\frac{\gamma^{d(s)}}{1-\gamma}$, where $\mathrm{P}(s)$ is the probability to reach s.

To motivate this choice, note that both the optimal value and the value of the optimistic action $u^\dagger(x_0)$ lie between the lower bound $\sum_{s\in\mathcal{S}^\dagger} \mathrm{P}(s)\,\nu(s)$ and the upper bound $\sum_{s\in\mathcal{S}^\dagger} \mathrm{P}(s)\,b(s)$. This implies the regret of $u^\dagger(x_0)$ is at most the difference between the two: $\sum_{s\in\mathcal{S}^\dagger} \mathrm{P}(s)\,[b(s) - \nu(s)] = \sum_{s\in\mathcal{S}^\dagger} \mathrm{P}(s)\,\frac{\gamma^{d(s)}}{1-\gamma}$. Thus, by targeting the leaf with the largest contribution to this range, the expansion rule aims to maximally improve the knowledge about the regret of $u^\dagger(x_0)$, allowing the algorithm to make more informed choices later.

After n expansions, the algorithm selects at the root an action $u_0 \in \arg\max_{u\in U} \sum_{x'} f(x_0,u,x')\nu(x')$. Algorithm 1 summarizes OPSS in high-level pseudocode. Note that, since its updates are independent of the budget n, OPSS can just as well be used as an anytime algorithm.

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: $s_\ell \in \arg\max_{s\in\mathcal{S}_\ell^\dagger} \mathrm{P}(s)\,\frac{\gamma^{d(s)}}{1-\gamma}$
5: expand s_ℓ, obtaining $\mathcal{T}_{\ell+1}$
6: **end for**

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

Results and discussion

Using a sparsely stochastic variant of the inverted pendulum swingup (Buşoniu et al. 2011), the behavior of OPSS is studied for a computational budget n varying in $\{100, 200, \ldots, 1000\}$. OPSS is compared to a uniform planning algorithm, which always expands a node having the smallest depth, and to OLOP. Since OLOP has a different computational unit, consisting of simulating a single random transition instead of NM such transitions, it is allowed $NM = 6n$ transitions for fairness. To obtain a global performance measure, the algorithms find actions for the states on the grid $X_0 = \{-\pi, \frac{-150\pi}{180}, \frac{-120\pi}{180}, \ldots, \pi\} \times \{-15\pi, -14\pi, \ldots, 15\pi\}$, and the average simple regret on this grid is computed.

Figure 1, top reports the regret of the three algorithms. As expected, OPSS is better than uniform planning, since it expands the planning trees in a smart way. As Figure 1, bottom 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. Note that, because the algorithms simulate a similar number of transitions, their execution times are similar.

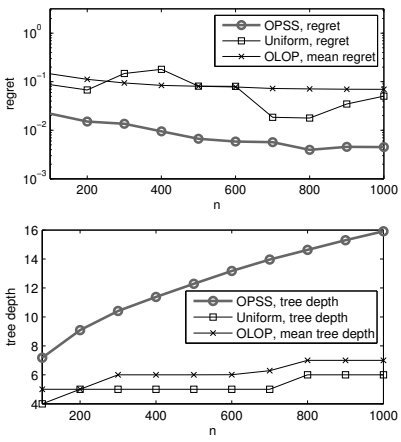

Figure 1: Comparison between OPSS, uniform planning, and OLOP: average regret over X_0 (top), average tree depth over X_0 (bottom). Mean results over 10 runs are reported for OLOP, as this algorithm works with random realizations of trajectories.

Related work: OPSS is most related to the OP algorithms of (Hren and Munos 2008; Bubeck and Munos 2010). The algorithm of (Hren and Munos 2008) is geared towards deterministic systems; OPSS reduces to it in the deterministic case, and generalizes it to stochastic systems. Compared to the open-loop OP (OLOP) of (Bubeck and Munos 2010), which is designed for nonsparsely stochastic systems, OPSS takes advantage of the sparse stochasticity to find deterministic upper bounds on the returns, rather than upper confidence bounds in high probability, as in OLOP. The idea of using upper and lower bounds to explore a planning tree was also used in (Walsh, Goschin, and Littman 2010).

A more extensive description and study of OPSS can be found in (Buşoniu et al. 2011).

Future work: The most important next step is the theoretical analysis of the regret as a function of the computational budget n. Also, while expanding the optimistic leaf with the largest contribution to the regret bound is intuitively a good choice, other options might be better for the algorithm in the long run. Finally, it will be interesting to empirically compare OPSS with other planning algorithms such as UCT (Kocsis and Szepesvári 2006).

References

Bubeck, S., and Munos, R. 2010. Open loop optimistic planning. In *COLT*, 477–489.

Buşoniu, L.; Munos, R.; De Schutter, B.; and Babuška, R. 2011. Optimistic planning for sparsely stochastic systems. In *ADPRL*, 48–55.

Hren, J.-F., and Munos, R. 2008. Optimistic planning of deterministic systems. In *EWRL*, 151–164.

Kocsis, L., and Szepesvári, C. 2006. Bandit based Monte-Carlo planning. In *ECML*, 282–293.

Walsh, T. J.; Goschin, S.; and Littman, M. L. 2010. Integrating sample-based planning and model-based reinforcement learning. In *AAAI*.