Optimistic planning for Markov decision processes

Lucian Buşoniu **Rémi Munos**
Team SequeL
INRIA Lille – Nord Europe
40 avenue Halley, Villeneuve d'Ascq, France[1]

Abstract

The reinforcement learning community has recently intensified its interest in online planning methods, due to their relative independence on the state space size. However, tight near-optimality guarantees are not yet available for the general case of stochastic Markov decision processes and closed-loop, state-dependent planning policies. We therefore consider an algorithm related to AO* that optimistically explores a tree representation of the space of closed-loop policies, and we analyze the near-optimality of the action it returns after n tree node expansions. While this *optimistic planning* requires a finite number of actions and possible next states for each transition, its asymptotic performance does not depend directly on these numbers, but only on the subset of nodes that significantly impact near-optimal policies. We characterize this set by introducing a novel measure of problem complexity, called the near-optimality exponent. Specializing the exponent and performance bound for some interesting classes of MDPs illustrates the algorithm works better when there are fewer near-optimal policies and less uniform transition probabilities.

1 Introduction

Stochastic optimal control problems arise in many fields, including artificial intelligence, automatic control, operations research, economics, etc. They can be modeled as Markov decision processes (MDPs) (Puterman, 1994), in which optimality is measured by a cumulative reward signal that must be maximized: the return. In this paper, we analyze an online, optimistic planning (OP) algorithm for discounted MDPs that have a finite number K of actions and N of possible random next states for every transition. This includes finite MDPs, as well as infinite (e.g. continuous-state) MDPs that satisfy the conditions.

At a given step of interaction with the system, OP develops a tree starting from a node containing the current system state and then iteratively expanding well-chosen nodes, where each expansion adds all NK children nodes containing all states reachable from the chosen node, see Figure 1. To choose which node to expand, first an *optimistic* subtree is constructed, by recursively navigating the tree along actions having the largest upper bound on their optimal return. Then, among the leaves of this subtree, a node is selected that maximizes the contribution of the node to the upper bound. After n such iterations, the algorithm returns an action which is applied to the system. A certain state may appear many times in the tree – as many as the number of ways it can be reached from the root; the simple variant of OP studied here does not merge information from these duplicates.

The main contribution of this paper is a *near-optimality guarantee* for OP as a function of the number of expansions n, in terms of the simple regret: the loss in performance incurred by taking the action returned instead of the optimal action. We show that OP adapts to the complexity of the planning problem, by only expanding nodes with significant contributions to near-optimal policies. This notion is formalized so that the quantity of nodes with ε contribution to ε-optimal policies is of order $\varepsilon^{-\beta}$, with β a positive *near-optimality exponent*. Then, we show that the simple regret is of order $n^{-1/\beta}$ for large n. When there are few near-optimal policies, as well as when the transition

Appearing in Proceedings of the 15^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2012, La Palma, Canary Islands. Volume XX of JMLR: W&CP XX.

[1]Lucian Buşoniu is now with the Research Center for Automatic Control (CRAN), University of Lorraine, 2 avenue Forêt de Haye, Vandoeuvre-les-Nancy, France. This work was performed while he was with INRIA Lille.

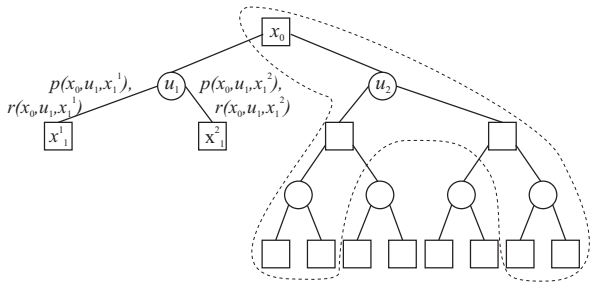

Figure 1: Illustration of an OP tree after three expansions, for $N = K = 2$. The squares are state nodes labeled by states x, and the actions u are explicitly included as circle, choice nodes. Transition arcs to next states are labeled by probabilities p and rewards r. The dashed outline encloses a possible optimistic subtree.

probabilities are nonuniform — both corresponding to having more structure in the MDP — β is small and the regret bound is better. To our knowledge, this is the first simple regret bound available for closed-loop planning in stochastic MDPs (closed-loop policies are state-dependent, rather than open-loop action sequences; e.g., the dashed line in Figure 1 encloses part of the subtree of a closed-loop policy).

The exponent β is related to other measures of complexity used in the literature on bandits with many arms (Kleinberg et al., 2008; Bubeck et al., 2009b; Wang et al., 2008). However, a direct application of those results would look at the closed-loop policies as the elementary entity (arm), and would thus not take into account essential problem structure: the fact that a node belongs to many policies, and expanding it improves knowledge about all these policies. A more refined notion of complexity is needed to capture this global structure, and β serves this purpose.

It must be noted that we consider the simple regret (Bubeck et al., 2009a) which is different from the usual cumulative regret in bandit theory: we look at the suboptimality of the action returned after a given number of calls to the model, whereas the cumulative regret is the loss of reward incurred while exploring the environment (e.g., Auer et al., 2002). Our measure corresponds to a numerical exploration-exploitation tradeoff, rather than an experimental one: we are interested in making the best possible use of a given numerical budget. Therefore, the simple regret is the appropriate measure of performance, and is directly related to the numerical complexity (i.e., number of calls to the model needed to return a near-optimal action). Note also that if each action returned by the algorithm is near-optimal, then the resulting policy is near-optimal.

While the bound does not directly depend on N and K, in practice they should not be too large, e.g. the probability mass should be concentrated on a few discrete next states. Fortunately this is true in many problems of interest. For example, combining continuous-state deterministic dynamics with random variables that only take a few discrete values leads to a small N. Such variables could be failure modes, job arrivals into a resource management system (e.g., elevator scheduling, traffic signal control), user input, etc.

Algorithmically, OP is rather simple, and can be seen as an extension of classical AO* heuristic search (Nilsson, 1980) to infinite-horizon discounted MDPs, similar to the AO* variant from (Hansen and Zilberstein, 1999). AO* builds a complete plan, which can only be done in finite MDPs with goal states and may require arbitrarily long computation. OP finds general near-optimal solutions while the expansion budget is limited to n. It can actually be applied in an any-time fashion, without fixing n in advance, which gives it potential for online real-time control.

OP was proposed and empirically studied by Buşoniu et al. (2011). It extends the earlier optimistic planning for deterministic MDPs (OPD) of Hren and Munos (2008), and in fact, in the deterministic case, OP and the regret bounds we derive here reduce to OPD and its bounds, respectively.

OP requires the full MDP model, including probability distributions over next states, whereas the class of sample-based planning methods only need to generate next states according to this distribution. Kearns et al. (2002) first proposed a "sparse-sampling" method that builds a uniform planning tree by sampling a fixed number of states for each action, up to some horizon, without adapting to the structure of the MDP. An adaptive-horizon extension was given by Péret and Garcia (2004). An optimistic sample-based algorithm is "upper-confidence-trees" (Kocsis and Szepesvári, 2006), which travels an optimistic path along the planning tree by choosing actions with maximal upper confidence bounds on the returns, and sampling states independently. UCT was extended to continuous actions by Mansley et al. (2011). UCT often works well in practice (Wang and Gelly, 2007) but good performance cannot be guaranteed in general since it may exhibit pathological behavior (Coquelin and Munos, 2007). Walsh et al. (2010) avoid this problem with an optimistic extension of sparse sampling that comes with so-called probably-approximately-correct guarantees, of a different nature from the regret bounds we introduce here. Bubeck and Munos (2010) do provide regret bounds for optimistic planning in stochastic problems, but only for open-loop sequences of actions, which are generally suboptimal.

Next, after providing in Section 2 the necessary notions concerning MDPs, we formalize the OP method in Section 3. In Section 4 we introduce the general regret bound and the insight behind it, and in Section 5 we specialize the analysis for some interesting types of MDPs. Section 6 concludes the paper. Proofs are collected in the supplementary material.

2 Background

Consider an MDP with state space X (taken countable for simplicity) and action space U. The probability that next state x' is reached after action u is taken in x is $p(x,u,x')$, where $p : X \times U \times X \to [0,1]$. After each transition a reward $r' = r(x,u,x')$ is received, where $r : X \times U \times X \to \mathbb{R}$. We assume that there is a finite number K of actions; that after applying any action in any state, the number of reachable next states is at most a finite N; and that rewards are in $[0,1]$. Denoting by k the discrete time index, the expected infinite-horizon discounted return (for short, value) of state x under policy $\pi : X \to U$ is:

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

where $x_0 = x$, $r_{k+1} = r(x_k, \pi(x_k), x_{k+1})$, and $\gamma \in (0,1)$ is the discount factor. The goal is to control the system using an optimal policy π^*, so that the value function $V^\pi : X \to \mathbb{R}$ is maximized for every $x \in X$. The resulting, maximal value function is denoted by V^* and is unique. It is also useful to consider the optimal Q-function: $Q^*(x,u) = \mathrm{E}_{x'\sim p(x,u,\cdot)}\{r(x,u,x') + \gamma V^*(x')\}$.

The quality of action u_k returned at state x_k by the planning algorithm is measured by the simple regret:

$$\mathcal{R}(x_k) = \max_{u\in U} Q^*(x_k,u) - Q^*(x_k,u_k) \quad (2)$$

i.e., the loss incurred by choosing u_k and only then acting optimally. A smaller regret is better, and optimal actions have 0 regret. If an algorithm achieves a simple regret of $\mathcal{R}$ for every state, the overall discounted return is $\frac{\mathcal{R}}{1-\gamma}$-optimal (Hren and Munos, 2008). Note that throughout the remainder of the paper, we focus on the algorithm when applied from some particular state; we leave this dependence implicit most of the time.

3 Optimistic planning algorithm

To introduce OP, we first formalize the notions of tree and node illustrated in Figure 1. A state node is denoted s and is labeled by an actual state x. Many nodes s may have the same state x. Such duplicates could be merged which would close the tree into a graph; however here we restrict ourselves to the simpler variant of OP that ignores duplicates. By a slight abuse of notation, introduce also a function $x(s)$ that returns the state label of a node. Define now the infinite planning tree $\mathcal{T}_\infty$—of which Figure 1 only shows the first few nodes—recursively as follows. Initialize $\mathcal{T}_\infty$ with root s_0 labeled by x_0, and recursively expand each node $s \in \mathcal{T}_\infty$ that does not yet have children. A node s is expanded by adding to it, for each $u \in U$, and then for each $x' \in X$ with $p(x(s),u,x') > 0$, a child node s' labeled by x'. Thus the branching factor of each node is NK (smaller if there are fewer than N successors for some actions). Denote by $\mathcal{C}(s,u)$ the set of children s' corresponding to action u; and by $\mathcal{C}(s) = \bigcup_{u\in U}\mathcal{C}(s,u)$. Note that $\mathcal{T}_\infty$ only includes the state nodes from Figure 1, and does not explicitly consider the actions. Nevertheless the actions are implicitly present in the connections between adjacent state nodes.

Algorithm 1 Optimistic planning

initialize tree: $\mathcal{T}_0 \leftarrow \{s_0\}$
for each iteration $t = 0, \ldots, n-1$ **do**
  starting from s_0, build optimistic subtree $\mathcal{T}_t^\dagger$:
  while $\mathcal{L}(\mathcal{T}_t^\dagger) \not\subseteq \mathcal{L}(\mathcal{T}_t)$ **do**
    retrieve a node $s \in \mathcal{L}(\mathcal{T}_t^\dagger) \setminus \mathcal{L}(\mathcal{T}_t)$
    find optimistic action at s:
    $u^\dagger = \arg\max_u \sum_{s'\in\mathcal{C}(s,u)} p(x(s),u,x(s'))b(s')$
    add $\mathcal{C}(s,u^\dagger)$ to $\mathcal{T}_t^\dagger$
  end while
  select leaf to expand: $s_t \leftarrow \arg\max_{s\in\mathcal{L}(\mathcal{T}_t^\dagger)} P(s)\,\frac{\gamma^{d(s)}}{1-\gamma}$
  create $\mathcal{C}(s_t)$ and add them to $\mathcal{T}_t$, obtaining $\mathcal{T}_{t+1}$
  update b-values upwards in the new tree $\mathcal{T}_{t+1}$:
  $b(s) = \begin{cases} R(s) + \frac{\gamma^{d(s)}}{1-\gamma}, & \text{if } s \text{ is leaf} \\ \max_u \sum_{s'\in\mathcal{C}(s,u)} p(x(s),u,x(s'))b(s'), & \text{else} \end{cases}$
end for
output u_0 maximizing lower-bound at s_0

Algorithm 1 gives OP as described in the introduction, which iteratively builds a finite tree contained at the top of $\mathcal{T}_\infty$. The algorithm formalizes the notions of optimistic subtree and leaf contribution. The mapping $\mathcal{L}(\cdot)$ provides the leaf nodes of its argument tree, and $d(s)$ gives the depth of a node. The b-value $b(s)$ is an upper bound on $V^*(x(s))$. Moreover, if a leaf s at depth d is reached from the root via the path: $s_0, u_0, s_1, u_1, \ldots, s_d = s$, where the implicit actions are mentioned, then s has a probability $P(s)$ and a partial

return $R(s)$ as follows:

$$P(s) = \prod_{d'=0}^{d-1} p(x(s_{d'}), u_{d'}, x(s_{d'+1}))$$
$$R(s) = \sum_{d'=0}^{d-1} \gamma^{d'} r(x(s_{d'}), u_{d'}, x(s_{d'+1}))$$

To choose the final action at the root, lower bounds are computed similarly to the b-values, but starting from 0 at the leaves.

The OP form given above is directly suitable for implementation. Next, we reinterpret OP as a search method in the space of *tree policies*, a form that is easier to analyze. A tree policy h is an assignment of actions to a subtree $\mathcal{T}_h$ of $\mathcal{T}_\infty$, $h : \mathcal{T}_h \to U$, recursively taking into account only the nodes reached under the action choices made so far:

$$\mathcal{T}_h = \{ s \in \mathcal{T}_\infty \mid s = s_0 \text{ or } \exists s' \in \mathcal{T}_h, s \in \mathcal{C}(s', h(s')) \}$$

where actions $h(s)$ are assigned as desired. The branching factor of $\mathcal{T}_h$ is N. Denote the expected return (value) of one tree policy h by $v(h)$, and the optimal, maximal value by v^*.

A *class* of tree policies, $H : \mathcal{T}_H \to U$, is obtained similarly but restricting the procedure to nodes in some finite tree $\mathcal{T}_t$ considered by OP, so that all action assignments below $\mathcal{T}_t$ are free. So H is a set of tree policies, where one such policy $h \in H$ is obtained by initializing the free actions. Note that $\mathcal{T}_H = \mathcal{T}_t \cap \mathcal{T}_h$ for any $h \in H$.

Figure 1 shows a subtree corresponding to such a policy class, enclosed by the dashed line; a full tree policy would be obtained by continuing with the action assignments until infinity. Tree policies h are more general than the usual MDP policies π, which would be stationary, i.e., would always take the same action in a given state x. Tree policies h are allowed to take different actions for different nodes s, even if they are labeled by the same state x. While it is known an optimal stationary policy always exists, searching in a larger class does not prevent the algorithm from achieving (near-)optimality. We will typically deal only with tree policies and call them simply "policies", except where confusion with MDP policies is possible.

The value of any policy h belonging to some class H is lower-bounded by:

$$\nu(H) = \sum_{s \in \mathcal{L}(\mathcal{T}_H)} P(s)\, R(s)$$

because the rewards that h can obtain below the leaves of $\mathcal{L}(\mathcal{T}_H)$ are lower-bounded by 0. Note that the probabilities $P(s)$ form a distribution over the leaves. Since rewards are also upper-bounded by 1, an upper bound on the value of $h \in H$ is:

$$\begin{aligned} b(H) &= \sum_{s \in \mathcal{L}(\mathcal{T}_H)} P(s) \left[R(s) + \frac{\gamma^{d(s)}}{1-\gamma} \right] \\ &= \nu(H) + \sum_{s \in \mathcal{L}(\mathcal{T}_H)} c(s) = \nu(H) + \mathrm{diam}(H) \end{aligned} \tag{3}$$

where we introduced the notations $c(s) = P(s)\, \frac{\gamma^{d}(s)}{1-\gamma}$, the *contribution* of leaf s to the difference between the upper and lower bounds, and $\mathrm{diam}(H) = \sum_{s \in \mathcal{L}(\mathcal{T}_H)} c(s)$, the *diameter* of H. This is indeed a diameter, $\mathrm{diam}(H) = \sup_{h,h' \in H} \ell(h, h')$, under the following metric on the space of policies:

$$\ell(h, h') = \sum_{s \in \mathcal{L}(\mathcal{T}_h \cap \mathcal{T}_{h'})} c(s)$$

The sum is taken over the shallowest nodes where h and h' are different. The diameter formula follows by noticing that for any s and u, $\sum_{s' \in \mathcal{C}(s,u)} c(s') = \gamma c(s)$ and so the contribution decreases monotonically with the depth, which implies that to maximize the distance between h and h', we should make these policies different at the shallowest possible places in the tree. Since the policies must be the same at inner nodes of $\mathcal{T}_H$, the leaves of this tree are the shallowest nodes where we can make h and h' different. A useful interpretation of $\mathrm{diam}(H)$ is that it represents the uncertainty on the value of policies in the class.

With these notations, OP can be restated more concisely and intuitively, as follows. The algorithm selects at each iteration an *optimistic* policy class, which maximizes the upper bound among all classes compatible with the current tree $\mathcal{T}_t$:

$$H_t^\dagger \in \arg\max_{H \in \mathcal{T}_t} b(H)$$

where the shorthand notation $H \in \mathcal{T}_t$ indicates that class H is compatible with $\mathcal{T}_t$, i.e., that $\mathcal{T}_H \subseteq \mathcal{T}_t$. The optimistic class is explored deeper, by expanding one of its leaf nodes (making the action choices for that node definite). The chosen leaf is the one maximizing the contribution:

$$s_t \in \arg\max_{s \in \mathcal{L}(\mathcal{T}_{H_t^\dagger})} c(s)$$

Under the metric ℓ, this can also be seen as splitting the set of policies H along the longest edge, where H is a hyperbox with $\left|\mathcal{L}(\mathcal{T}_{H_t^\dagger})\right|$ dimensions, having a length of $c(s)$ along dimension s. Note a crucial property of the algorithm: expanding s_t *also refines many additional policy classes*: all that reach s_t with nonzero probability. The algorithm continues at the

next iteration with the new, resulting tree $\mathcal{T}_{t+1}$. After n iterations, a *near-optimal* policy class is chosen, by maximizing this time the *lower* bound:

$$H_n^* \in \arg\max_{H \in \mathcal{T}_n} \nu(H) \tag{4}$$

intuitively seen as making a safe choice. The action returned by the algorithm at the root is the first action chosen by H_n^*. Of course, the algorithm may also return the multi-step policy H_n^*, useful for taking decisions along several steps of interaction.

We recall that the equivalent OP form in Algorithm 1 is more suitable for implementation than this abstract form.

4 Analysis of optimistic planning

The complexity of the tree policy space for a given MDP will be characterized in terms of a constant called *near-optimality exponent*. The dependence of the regret on this exponent and the number of expansions n will be studied.

Consider any node s on the complete, infinite planning tree $\mathcal{T}_\infty$. Define $n(s)$ to be the largest number, for any policy h whose subtree contains s, of *leaves* of the subtree $\mathcal{T}_{hs} \in \mathcal{T}_h$ containing only nodes with larger contributions than s:

$$n(s) = \sup_{\mathcal{T}_h \ni s} |\mathcal{L}(\mathcal{T}_{hs})|\,,\ \mathcal{T}_{hs} = \{s' \in \mathcal{T}_h \mid c(s') \geq c(s)\}$$

$\mathcal{T}_{hs}$ is indeed a proper subtree since $c(s)$ decreases monotonically along paths in $\mathcal{T}_h$. A policy subtree is schematically represented in Figure 2, together with an example of subtree $\mathcal{T}_{hs}$. Define using $n(s)$ also the quantity $\alpha(s) = n(s)c(s)$. An essential property of α is that it relates s to the diameter of some policy classes that have it as a leaf. Specifically, the diameter of any class among whose leaves $c(s)$ is largest, is upper-bounded by $\frac{N}{\gamma}\alpha(s)$.

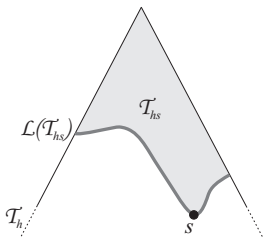

Figure 2: An infinite policy tree $\mathcal{T}_h$, and a subtree $\mathcal{T}_{hs}$ for some s.

Finally, define for each ε the set of nodes:

$$S_\varepsilon = \big\{s \in \mathcal{T}_\infty \big| \text{(i) } \alpha(s) \geq \varepsilon \text{ and } \text{(ii) } \exists h \ni s, v^* - v(h) \leq \frac{N}{\gamma}\alpha(s)\big\} \tag{5}$$

Condition (i) requires the node to have a sizable contribution in terms of α, and (ii) that the node belongs to a near-optimal policy. The following main result holds.

Theorem 1. *Let $\beta \geq 0$ be any constant so that:*

$$|S_\varepsilon| = \tilde{O}(\varepsilon^{-\beta}),\ \textit{i.e. } |S_\varepsilon| \leq a\left(\log\frac{1}{\varepsilon}\right)^b \varepsilon^{-\beta} \tag{6}$$

where $a, b > 0$ are some constants.[2] *The simple regret $\mathcal{R}_n$ of the action chosen by OP after n node expansions satisfies for large n:*

$$\mathcal{R}_n = \begin{cases} \tilde{O}(n^{-\frac{1}{\beta}}) & \text{if } \beta > 0 \\ O(\exp[-(\frac{n}{a})^{\frac{1}{b}}]) & \text{if } \beta = 0 \end{cases}$$

The constant β is the near-optimality exponent. It is important to note that OP does not require knowledge of the value of β, and yet, as the result shows, it automatically adapts to this value.

The measure β is connected to other complexity measures from optimistic optimization (bandits) and planning, such as the zooming dimension (Kleinberg et al., 2008), the near-optimality dimension (Bubeck et al., 2009b), or the branching factor of near-optimal action sequences for OPD (Hren and Munos, 2008) and open-loop optimistic planning (OLOP) (Bubeck and Munos, 2010). Characterizing the complexity of the planning problem by using these measures would essentially look at the individual tree policies as the elementary entity, and the analysis could only interpret the algorithm as a hierarchical search in the space of such policies. OP has this hierarchical component, which is realized by refining the optimistic policy class $H_t^\dagger$. However, OP does more than just this: since the chosen node s_t belongs to all the policy classes that reach it with a positive probability, expanding it refines all these classes, not just the optimistic one. This subtler, global effect must be captured in the definition of problem complexity, and this is achieved by introducing the quantity $\alpha(s)$ to describe the *global impact* of a node on the policies it belongs to, and by defining S_ε and β in terms of individual nodes and not directly policies.

[2] More generally, $f(x)$ is $\tilde{O}(g(x))$ if $f(x) \leq a(\log g(x))^b g(x)$ for some $a, b > 0$. This is usually understood to hold for all x, so when it only holds for large/small values of x, we explicitly mention this restriction.

Therefore, the analysis intuitively says that OP finds a near-optimal policy by only expanding those nodes that have a large impact on near-optimal policies. As the number of such nodes decreases, the sets S_ε are smaller, which is characterized by a smaller β, and the problem becomes easier; in particular, $\beta = 0$ means S_ε grows logarithmically instead of polynomially. The definition (5) of S_ε highlights two ways in which a problem can be easier: the transition probabilities are less uniform, leading to fewer nodes with large α; or the rewards are concentrated on a few actions, leading to fewer near-optimal policies. To formalize these intuitions, the next section will provide values of β for several representative types of MDPs, exhibiting varying degrees of structure in the rewards and probabilities.

Proof sketch of Theorem 1. We outline here how the result is obtained (see the supplementary material for the complete proof). The first big step is to show that the regret is related to the smallest α among expanded nodes, $\alpha^* = \min_{t=0,\dots,n-1} \alpha(s_t)$. This is true because (a) the suboptimality of the near-optimal policy class H_t^* (and hence the regret of the root action it would choose) can be bounded at every iteration by the diameter of the optimistic policy: $v^* - \nu(H_t^*) \leq \mathrm{diam}(H_t^\dagger)$, and (b) using the essential property of α, this diameter is in turn at most $\frac{N}{\gamma}\alpha(s_t)$. The near-optimal policy will not decrease in quality over iterations, which means the overall regret is bounded by $\frac{N}{\gamma}\alpha^*$.

The second major part of the proof is that the algorithm always usefully works to decrease the value of α^* – formally, that all the nodes it expands are in S_α^*, so that $n \leq |S_\alpha^*|$. Connecting all this with the definition of β, which implies $|S_\alpha^*| = \tilde{O}(\alpha^{*-\beta})$, the overall regret bound is obtained. □

Regarding the time complexity of OP, at the expense of some extra memory, each iteration can be brought down to $O(d(s_t))$, the depth of the expanded node. Depths generally depend on β, but to obtain a range notice they are between $O(\log n)$ when the tree is developed uniformly and $O(n)$ when a single path is developed (see also next section). So the overall complexity is between $O(n \log n)$ and $O(n^2)$. Furthermore, while here we focus on the near-optimality perspective, our result can also be interpreted the other way around: to achieve ε regret, a budget n on the order of $\varepsilon^{-\beta}$ should be spent.

5 Some interesting values of β

To add meaning to the near-optimality exponent β, in this section we provide its value for several interesting special cases. We obtain smaller values when the MDP has "more structure", namely when there are non-uniform probabilities or rewards. The earlier OPD regret bounds (Hren and Munos, 2008) are recovered in the deterministic case, showing that the OP guarantees encompass those of OPD as a special case.

5.1 Uniform rewards and probabilities

In this case, the rewards of all transitions are equal, and for any action, the probability of reaching one of the N next states is $\frac{1}{N}$.

Proposition 2. *In the uniform case,* $\beta_{unif} = \frac{\log NK}{\log 1/\gamma}$ *and* $\mathcal{R}_n = O(n^{-\frac{\log 1/\gamma}{\log NK}})$.

One interpretation of β is that the argument of the logarithm at the numerator is an equivalent branching factor of the tree that OP must explore. A branching factor of NK means that OP will have to explore the whole planning tree in a uniform fashion, expanding nodes in the order of their depth. So the uniform MDP is an interesting worst case, where β is the largest possible. (Notice also that in this case the bounds do not have a logarithmic component, so O is used instead of $\tilde{O}$.)

In fact, the regret bound of OP in uniform MDPs is the same as that of a *uniform planning* algorithm, which always expands the nodes in the order of their depth. However, the uniform algorithm can guarantee only this regret bound for *any* problem, whereas in non-uniform problems, OP adapts to the value of β to obtain better guarantees.

Proving the regret of the uniform algorithm helps in understanding the upcoming discussion on worst-case regret, so it is done here. Consider the uniform algorithm has expanded all nodes at depth $D-1$, after n expansions. Then:

$$\begin{aligned} v^* &\leq \max_{H\in\mathcal{T}_n} b(H) \\ &= \max_{H\in\mathcal{T}_n} [\nu(H) + \textstyle\sum_{s\in\mathcal{L}(\mathcal{T}_H)} P(s)\, \frac{\gamma^{d(s)}}{1-\gamma}] \\ &\leq \max_{H\in\mathcal{T}_n} [\nu(H) + \textstyle\sum_{s\in\mathcal{L}(\mathcal{T}_H)} P(s)\, \frac{\gamma^{D}}{1-\gamma}] \\ &= \max_{H\in\mathcal{T}_n} \nu(H) + \tfrac{\gamma^D}{1-\gamma} = \nu(H_n^*) + \tfrac{\gamma^D}{1-\gamma} \end{aligned}$$

where H_n^* is the policy class returned by (4). So $\mathcal{R}_n \leq v^* - \nu(H_n^*) \leq \frac{\gamma^D}{1-\gamma}$, and from $n \geq \frac{(NK)^D-1}{NK-1}$ after some calculation $\mathcal{R}_n = O(n^{-\frac{\log 1/\gamma}{\log NK}})$.

This regret is also the smallest achievable in a worst-case sense, which means that for any planning algorithm and value of n, one can construct a problem for which the regret is $\mathcal{R}_n = \Omega(n^{-\frac{\log 1/\gamma}{\log NK}})$. To see this, choose the largest D so that $n \geq \frac{(NK)^D-1}{NK-1}$, assign uniform probabilities everywhere, rewards of 1 for some

arbitrary policy h^* but only starting from level $D+1$ onward, and rewards of 0 everywhere else. Then, both OP and uniform planning have uniformly expanded all nodes up to $D-1$ but none at $D+1$, so they have no information and must make an arbitrary action choice, which may not be optimal, leading to a regret of $\frac{\gamma^{D+1}}{1-\gamma} = \Omega(n^{-\frac{\log 1/\gamma}{\log NK}})$. An algorithm that does not expand uniformly may miss the optimal policy for an even larger number of expansions n, so their regret is at least as large. This fact also shows that OP *behaves correctly* in the uniform case: as long as only uniform rewards and probabilities are observed, the tree must be expanded uniformly, and this behavior is reflected in the regret bound.

Finally, note that in the deterministic case, when $N = 1$, the regret bound of OPD for the uniform-reward case is recovered: $\mathcal{R}_n = O(n^{-\frac{\log 1/\gamma}{\log K}})$.

5.2 Structured rewards

In this case, probabilities are uniform but a single policy has maximal rewards (equal to 1) for all transitions, and all other transitions have a reward of 0, see Figure 3. So, there is a "maximal" amount of structure in the reward function.

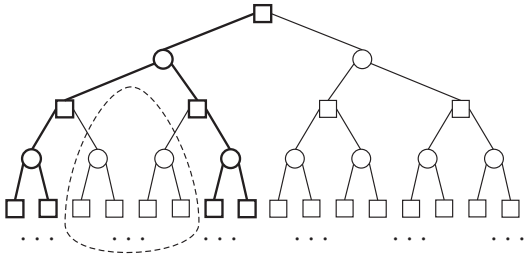

Figure 3: Illustration of a planning tree for structured rewards, up to depth 2, for $N = K = 2$. Thick lines: subtree of optimal policy, where each transition is associated with a reward of 1. Thin lines: the rest of the tree, associated with 0 rewards.

Proposition 3. *In the case of structured rewards, if* $N > 1$, $\beta_{\text{rew}} = \frac{\log N}{\log 1/\gamma}(1 + \frac{\log K}{\log N/\gamma})$, *whereas if* $N = 1$, $\beta_{\text{rew}} = 0$ *and* $\mathcal{R}_n = O(\exp(-\frac{n}{a}))$ *for some constant* a.

The values of β_{rew} are smaller than β_{unif}, so the guarantee takes advantage of the additional structure introduced in the problem by the reward function.

In the deterministic case, $\beta_{\text{rew}} = 0$, the problem becomes easy and the regret is exponential in n, having the form in the second branch of Theorem 1 for $b = 1$. This is the same bound that OPD obtains when a single policy is optimal (Hren and Munos, 2008).

Examining the algorithm reveals that it will only explore the optimal policy's subtree, with branching factor N, so the ideal value for β is $\frac{\log N}{\log 1/\gamma}$. Improving the analysis to avoid this dependency is a topic for future work.

5.3 Structured probabilities

Finally, we consider problems where the rewards for all transitions are equal, but the transitions have significantly different probabilities. Take for simplicity identical Bernoulli transitions: $N = 2$ and the two successors of any state (and thus state node) have probabilities p and $1-p$, so that p is close to 1.

Proposition 4. *In the Bernoulli case, for* p *close to* 1, $\beta_{\text{prob}} = \frac{\log K\eta'}{\log 1/(p\gamma\eta')}$, *where* $\eta' = \left(\frac{e}{\eta}\right)^{\eta}$ *and* $\eta = \frac{\log 1/(p\gamma)}{\log 1/(\gamma(1-p))}$

In the deterministic case, when $p \to 1$, $\eta \to \infty$ leading to $\eta' \to 1$, and $\beta_{\text{prob}} \to \frac{\log K}{\log 1/\gamma}$, recovering the uniform-case bound for $N = 1$ (i.e., the uniform-reward case). Nearby, when p is large, $\beta \approx \frac{\log K}{\log 1/\gamma}$, and OP expands "almost" only a tree with branching factor K. When the probabilities are uniform, $\eta = 1$ would lead to $\beta_{\text{prob}} = \frac{\log eK}{\log 2/e\gamma}$, and β_{unif}, which was developed specifically for that case, is better.

6 Conclusions

We have analyzed the regret of a planning algorithm for stochastic MDPs, which optimistically explores the space of closed-loop planning policies. The core feature of this method is that it adapts to the complexity of the planning problem, encoded in the near-optimality exponent β. An immediate topic for further research is the form of this exponent for an MDP with arbitrary transition probabilities. Based on the special cases for β studied, we expect it to decrease as the distribution becomes less uniform.

The basic form of OP studied here can benefit from many algorithmic improvements, and analyzing their effect on the regret would be very interesting. For example, since OP is similar to AO*, improvements originally developed for classical AO* can almost directly be applied, such as closing the tree into a graph upon encountering duplicate states (Hansen and Zilberstein, 1999). In online control, an essential improvement is reusing information across interaction steps. This could be done by maintaining an approximate value function (for the b- and/or ν-values), used to initialize leaf values and updated with tree data. In this context, near-optimality guarantees for approximate value iteration could be useful to improve the regret bounds under smoothness assumptions on the MDP (Szepesvári, 2001; Rust, 1996).

OP does not assume any knowledge about the complexity of the problem, e.g. it does not require to know β. Deriving algorithms that, when such prior knowledge is available, exploit it to obtain better performance guarantees is an important open issue.

Acknowledgements

This research was financially supported by the French National Research Agency (ANR) under project EXPLO-RA, and by the European Community's Seventh Framework Programme under grant no. 231495 (Complacs project). We also wish to thank the reviewers for their comments, which greatly helped to improve the paper.

References

Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256.

Bubeck, S. and Munos, R. (2010). Open loop optimistic planning. In *Proceedings 23rd Annual Conference on Learning Theory (COLT-10)*, pages 477–489, Haifa, Israel.

Bubeck, S., Munos, R., and Stoltz, G. (2009a). Pure exploration in multi-armed bandits problems. In *Proceedings 20th International Conference on Algorithmic Learning Theory (ALT-09)*, pages 23–37, University of Porto, Portugal.

Bubeck, S., Munos, R., Stoltz, G., and Szepesvári, C. (2009b). Online optimization in X-armed bandits. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 21*, pages 201–208. MIT Press.

Buşoniu, L., Munos, R., De Schutter, B., and Babuška, R. (2011). Optimistic planning for sparsely stochastic systems. In *Proceedings 2011 IEEE International Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL-11)*, pages 48–55, Paris, France.

Coquelin, P.-A. and Munos, R. (2007). Bandit algorithms for tree search. In *Proceedings 23rd Conference on Uncertainty in Artificial Intelligence (UAI-07)*, pages 67–74, Vancouver, Canada.

Hansen, E. A. and Zilberstein, S. (1999). A heuristic search algorithm for Markov decision problems. In *Proceedings Bar-Ilan Symposium on the Foundation of Artificial Intelligence*, Ramat Gan, Israel.

Hren, J.-F. and Munos, R. (2008). Optimistic planning of deterministic systems. In *Proceedings 8th European Workshop on Reinforcement Learning (EWRL-08)*, pages 151–164, Villeneuve d'Ascq, France.

Kearns, M. J., Mansour, Y., and Ng, A. Y. (2002). A sparse sampling algorithm for near-optimal planning in large Markov decision processes. *Machine Learning*, 49(2-3):193–208.

Kleinberg, R., Slivkins, A., and Upfal, E. (2008). Multi-armed bandits in metric spaces. In *Proceedings 40th Annual ACM Symposium on Theory of Computing (STOC-08)*, pages 681–690, Victoria, Canada.

Kocsis, L. and Szepesvári, C. (2006). Bandit based Monte-Carlo planning. In *Proceedings 17th European Conference on Machine Learning (ECML-06)*, pages 282–293, Berlin, Germany.

Mansley, C., Weinstein, A., and Littman, M. L. (2011). Sample-based planning for continuous action Markov decision processes. In *Proceedings 21st International Conference on Automated Planning and Scheduling*, pages 335–338, Freiburg, Germany.

Nilsson, N. (1980). *Principles of Artificial Intelligence*. Tioga Publishing.

Péret, L. and Garcia, F. (2004). On-line search for solving Markov decision processes via heuristic sampling. In *Proceedings 16th European Conference on Artificial Intelligence, ECAI'2004*, pages 530–534, Valencia, Spain.

Puterman, M. L. (1994). *Markov Decision Processes—Discrete Stochastic Dynamic Programming*. Wiley.

Rust, J. (1996). Numerical dynamic programming in economics. In Amman, H. M., Kendrick, D. A., and Rust, J., editors, *Handbook of Computational Economics*, volume 1, chapter 14, pages 619–729. Elsevier.

Szepesvári, Cs. (2001). Efficient approximate planning in continuous space Markovian decision problems. *AI Communications*, 13(3):163–176.

Walsh, T. J., Goschin, S., and Littman, M. L. (2010). Integrating sample-based planning and model-based reinforcement learning. In *Proceedings 24th AAAI Conference on Artificial Intelligence (AAAI-10)*, Atlanta, US.

Wang, Y., Audibert, J.-Y., and Munos, R. (2008). Algorithms for infinitely many-armed bandits. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 21*, pages 1729–1736. MIT Press.

Wang, Y. and Gelly, S. (2007). Modifications of UCT and sequence-like simulations for Monte-Carlo Go. In *Proceedings 2007 IEEE Symposium on Computational Intelligence and Games (CIG-07) USA, 1-5 April, 2007*, pages 175–182, Honolulu, Hawaii.