Optimistic planning of deterministic systems

Jean-Francois Hren, Rémi Munos

Optimistic planning of deterministic systems

Jean-François Hren and Rémi Munos

SequeL project, INRIA Lille - Nord Europe
40 avenue Halley, 59650 Villeneuve d'Ascq, France
jean-francois.hren@inria.fr and remi.munos@inria.fr

Abstract. If one possesses a model of a controlled deterministic system, then from any state, one may consider the set of all possible reachable states starting from that state and using any sequence of actions. This forms a tree whose size is exponential in the planning time horizon. Here we ask the question: given finite computational resources (e.g. CPU time), which may not be known ahead of time, what is the best way to explore this tree, such that once all resources have been used, the algorithm would be able to propose an action (or a sequence of actions) whose performance is as close as possible to optimality? The performance with respect to optimality is assessed in terms of the regret (with respect to the sum of discounted future rewards) resulting from choosing the action returned by the algorithm instead of an optimal action. In this paper we investigate an optimistic exploration of the tree, where the most promising states are explored first, and compare this approach to a naive uniform exploration. Bounds on the regret are derived both for uniform and optimistic exploration strategies. Numerical simulations illustrate the benefit of optimistic planning.

1 Introduction

This paper is concerned with the problem of making the best possible use of available numerical resources (such as CPU time, memory, number of calls to a black-box model, ...) in order to solve a sequential decision making problem in deterministic domains. We aim at designing *anytime algorithms*, i.e. which return higher accuracy solutions whenever additional resources are provided. To fix the setting, we are interested in generating a near-optimal policy for a deterministic system with discounted rewards, under finite action space but large state space (possibly infinite). We assume that we have a generative model of the state dynamics and rewards.

The approach consists in considering at each time-step t, the look-ahead tree (which may be constructed from our model) of all possible reachable states when using any sequence of actions, starting from x_t. Then a search is performed in this tree by using a specific exploration strategy. We address the question of how should one explore this tree such that after a finite number of numerical resources (here we will consider the number of node expansions or node transitions, which is directly related to the CPU time, or the number of calls to our model), we would be able to make the best possible decision about the action (or sequence

of actions) to choose from state x_t. This action (or sequence) is subsequently executed in the real world, and the overall process is repeated from the next state x_{t+1}. This is close in spirit to the sparse sampling algorithm of [KMN02], where a sampling device is used to explore the look-ahead tree and generate a near optimal action with high probability. Their setting is more general than ours since they consider general Markov Decision Processes, whereas we restrict ourself to deterministic dynamics and rewards (we will consider the stochastic case in future work). However the purpose of our study is the possible clever exploration of the tree, whereas they only consider uniform exploration. Empirical works, such as [PG04], suggest that non-uniform exploration may greatly improve the performance of the resulting policy, given a fixed amount of computation.

The bounds on the performance we obtain here do not depend on the dimension of the state space, contrarily to usual (possibly approximate) Dynamic Programming and Reinforcement Learning approaches (where a close-to-optimal policy results from the approximation of a value function over the whole domain), see [Put94,BT96,SB98], which are subject to the curse of dimensionality. However our bounds scale with the branching factor of some related trees and the time horizon $1/(\log 1/\gamma)$ (where γ is the discount factor). The performance measure we consider here is the regret $R(n)$, which is the performance loss (with respect to optimality) of the decision returned by the algorithm after having explored the tree using n units of computational resources. This notion will be made precise in the next section. If we consider a uniform exploration of the tree, one obtains the (upper and lower) bounds on the regret: $R(n) = \Theta(n^{-\frac{\log 1/\gamma}{\log K}})$, where K is the number of actions (the branching factor of the full look-ahead tree). We thus expect such approaches to be interesting (compared to value function-based approaches) when the state space is huge but the number of actions is relatively small.

In this paper we investigate an **optimistic exploration** of the tree, where the most promising nodes are explored first. The idea is to explore at each round the node that has a possibility of being the best (which motivates the term "optimistic"), in the sense of having the highest upper-bound. The idea of selecting actions based on upper confidence bounds (UCB) dates to early work in multi-armed bandit problems, see e.g. [Rob52,LR85], and more recently [ACBF02]. Planning under uncertainty using UCB has been considered in [KS06] and the resulting algorithm UCT (UCB applied to Trees) has been successfully applied to the large scale tree search problem of computer-go, see [GWMT06]. However, regret analysis shows that UCT may perform very poorly because of overly-optimistic assumptions of the bounds, see [CM07]. Our work is close in spirit to the BAST (Bandit Algorithm for Smooth Trees) algorithm of [CM07], where we use the inherent smoothness of the look-ahead tree (which comes from the fact that we consider the sum of discounted rewards along the paths) to settle true upper-bounds on the nodes value. Using this optimistic exploration, our main result is an upper-bound on the regret $R(n) = O(n^{-\frac{\log 1/\gamma}{\log \kappa}})$, where $\kappa \in (1, K]$ is the branching factor of a related subtree, composed of all the nodes that will eventually have to be evaluated in order to decide whether they belong

to an optimal path or not. In particular, the optimistic exploration is never worst than the uniform one (in some sense made precise later), and much better whenever κ is smaller than K and close to 1. We show that κ is also related to the proportion of near-optimal paths in the full look-ahead tree, and that in hard instances of search problems, κ is small compared to K, which increases the benefit of using optimistic rather than uniform exploration for complex problems. We further show that in some non-trivial cases, $\kappa = 1$, and exponential rates are derived.

In the next section, we introduce the notations and motivations for the tree exploration problem. Then we consider the uniform and optimistic exploration strategies. We conclude the paper by numerical experiments illustrating the benefit of the optimistic exploration.

2 Planning under finite numerical resources

We are interested in making the best possible use of available numerical resources for decision making. For that purpose, we assume that we possess a generative model that can be used to generate simulated transitions and rewards. The action-selection procedure (call it $\mathcal{A}$) takes as input the current state of the system and outputs an action $\mathcal{A}(n)$ (or a sequence of actions, but we will focus in this paper on the single output action case) using a finite number n of available numerical resources. The term resource refers to a piece of computational effort which may be measured e.g. in terms of CPU time or number of calls to the generative model. The amount n of available resources may not be known before they are all used, so we wish to design anytime algorithms. Our goal is that the proposed action $\mathcal{A}(n)$ be as close as possible to the optimal action in that state, in the sense that the regret $R_{\mathcal{A}}(n)$ (in the cumulative discounted sum of rewards to come) of choosing this action instead of the optimal one should be small. Let us now introduce some notations to define more precisely this notion of regret.

We consider here a deterministic controlled problem defined by the tuple (X, A, f, r), where X is the state space, A the action space, $f : X \times A \to X$ the transition dynamics, and $r : X \times A \to \mathbb{R}$ the reward function. If at time t, the system is in state $x_t \in X$ and the action a_t is chosen, then the system jumps to the next state $x_{t+1} = f(x_t, a_t)$ and a reward $r(x_t, a_t)$ is received. In this paper we will assume that all rewards are in the interval $[0, 1]$. We assume that the state space is large (possibly infinite), and the action space is finite, with K possible actions. We consider an infinite time horizon problem with discounted rewards ($0 \leq \gamma < 1$ is the discount factor). For any policy $\pi : X \to A$ we define the value function $V^\pi : X \to \mathbb{R}$ associated to that policy:

$$V^\pi(x) \stackrel{\text{def}}{=} \sum_{t \geq 0} \gamma^t r(x_t, \pi(x_t)),$$

where x_t is the state of the system at time t when starting from x (i.e. $x_0 = x$) and following policy π.

We also define the Q-value function $Q^\pi : X \times A \to \mathbb{R}$ associated to a policy π, in state-action (x, a), as:

$$Q^\pi(x,a) \stackrel{\text{def}}{=} r(x,a) + \gamma V^\pi(f(x,a)).$$

We have the property that $V^\pi(x) = Q^\pi(x, \pi(x))$. Now the optimal value function (respectively Q-value function) is defined as: $V^*(x) \stackrel{\text{def}}{=} \sup_\pi V^\pi(x)$ (respectively $Q^*(x,a) \stackrel{\text{def}}{=} \sup_\pi Q^\pi(x,a)$). From the dynamic programming principle, we have the Bellman equations:

$$V^*(x) = \max_{a \in A} \left[r(x,a) + \gamma V^*(f(x,a)) \right]$$

$$Q^*(x,a) = r(x,a) + \gamma \max_{b \in A} Q^*(f(x,a), b).$$

Now, let us return to the action-selection algorithm $\mathcal{A}$. After using n units of numerical ressources the algorithm $\mathcal{A}$ returns the action $\mathcal{A}(n)$. The regret resulting from choosing this action instead of the optimal one is:

$$R_{\mathcal{A}}(n) \stackrel{\text{def}}{=} \max_{a \in A} Q^*(x,a) - Q^*(x, \mathcal{A}(n)). \tag{1}$$

From an action-selection algorithm $\mathcal{A}$ one may define a policy $\pi_{\mathcal{A}}$ which would select in each state encountered along a trajectory the action $\mathcal{A}(n)$ proposed by the algorithm $\mathcal{A}$ using n resources. The following result motivates our choice of the previous definition for the regret in the sense that an algorithm with small regret will generate a close-to-optimal policy.

Proposition 1. *Consider a control algorithm using an action-selection procedure with regret ϵ (i.e. for each state x, the action-selection procedure returns an action a such that $Q^*(x,a) \geq V^*(x) - \epsilon$). Then the performance of the resulting policy $\pi_{\mathcal{A}}$ is $\frac{\epsilon}{1-\gamma}$-optimal, i.e. for all x,*

$$V^*(x) - V^{\pi_{\mathcal{A}}}(x) \leq \frac{\epsilon}{1-\gamma}$$

Proof. Let T and T^π be operators over bounded functions on X, defined as follows: for any bounded $V : X \to \mathbb{R}$, $TV(x) \stackrel{\text{def}}{=} \max_{a \in A} \left[r(x,a) + \gamma V(f(x,a)) \right]$, $T^\pi V(x) \stackrel{\text{def}}{=} r(x, \pi(x)) + \gamma V(f(x, \pi(x)))$.

We have the properties that T and T^π are contraction operators in sup norm, and that V^* is the fixed point of T (i.e. $V^* = TV^*$) and V^π is the fixed point of T^π (i.e. $V^\pi = T^\pi V^\pi$), see e.g. [Put94].

Using these notations, the assumption that $\mathcal{A}$ is an action selection procedure with regret ϵ writes that for all x, $T^{\pi_{\mathcal{A}}} V^*(x) \geq TV^*(x) - \epsilon$. Thus we have: $||V^* - V^{\pi_{\mathcal{A}}}||_\infty \leq ||TV^* - T^{\pi_{\mathcal{A}}} V^*||_\infty + ||T^{\pi_{\mathcal{A}}} V^* - T^{\pi_{\mathcal{A}}} V^{\pi_{\mathcal{A}}}||_\infty \leq \epsilon + \gamma ||V^* - V^{\pi_{\mathcal{A}}}||_\infty$, from which we deduce the proposition. □

Our goal is thus to define clever action-selection algorithms $\mathcal{A}$ such that, if provided with n units of computational resource, would return an action with minimal regret $R_{\mathcal{A}}(n)$.

The next section describes a method based on the construction of a uniform look-ahead tree.

3 Uniform planning

3.1 Look-ahead tree search

Given a state x, we describe a way to select an almost-optimal action, based on the construction of a uniform look-ahead tree. We consider the (infinite) tree $\mathcal{T}$ composed of all possible reachable states from x: the root corresponds to x and each node of depth d correspond to a state that is reachable from x after a sequence of d transitions. Each node i (associated to some state y), has K children $\mathcal{C}(i)$ (associated with the states $\{f(y,a)\}_{a\in A}$). Write 0 the root node, and $1 \dots K$ its K children (nodes of depth 1).

We call a path of $\mathcal{T}$ a (finite or infinite) sequence of connected nodes starting from the root. We define the value v_i of a node i as the supremum, over all infinite paths going through node i, of the sum of discounted rewards obtained along such paths. We have the property that $v_i = \max_{j\in\mathcal{C}(i)} v_j$, and the optimal value (value of the root) $v^* = v_0 = \max_{i\in\mathcal{T}} v_i = \max\{v_1, \dots, v_K\}$.

We consider numerical resources expressed in terms of the number of expanded nodes. This is directly related to the CPU time required to explore the corresponding part of the tree, or the amount of memory required to store information about the expanded nodes. This is also equivalent to the number of calls to the generative-model, providing the next-state $f(x,a)$ and the reward $r(x,a)$.

We say that a node is expanded when some numerical resources are allocated to this node and to the computation of the transitions to its children (by a call to the generative model for each actions). At any round n, the expanded tree $\mathcal{T}_n$ denotes the set of nodes already expanded. The set of nodes in $\mathcal{T}$ that are not in $\mathcal{T}_n$ but whose parents are in $\mathcal{T}_n$ is written $\mathcal{S}_n$: this represents the set of possible nodes that may be expanded at the next round (see fig. 1).

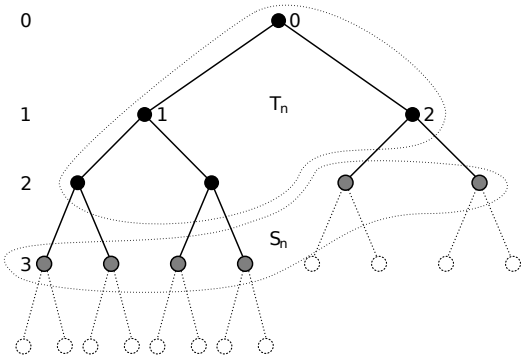

Fig. 1. Set of expanded nodes $\mathcal{T}_n$ (black dots) at round $n = 5$ and set of nodes $\mathcal{S}_n$ (gray dots) that may be expanded next. Here $K = 2$.

For any node $i \in \mathcal{S}_n$, we define the value u_i to be the sum of discounted rewards obtained along the (finite) path from the root to node i (this information is available at round n since the parent of i has been expanded and thus the transition to node i has been computed). Now, for any node $i \in \mathcal{T}_n$, we define in a recursive way $u_i = \max_{j \in \mathcal{C}(i)} u_j$. Since these u-values are defined in $\mathcal{S}_n$, they are also well defined in $\mathcal{T}_n$.

Note that since the u-values depend on $\mathcal{T}_n$, we will write them $u_i(n)$ whenever the dependency with respect to (w.r.t.) n is relevant. From their definition, we have the property that $u_i(n)$ is an increasing function of n.

Since the sum of discounted rewards from a node of depth d is at most $\gamma^d + \gamma^{d+1} + \cdots = \frac{\gamma^d}{1-\gamma}$ (recall that all rewards are in $[0,1]$), we have: for all $i \in \mathcal{T}_t \cup \mathcal{S}_t$ and $n \geq t \geq 1$, $u_i(n) \leq v_i \leq u_i(n) + \frac{\gamma^d}{1-\gamma}$.

3.2 The algorithm

Here we consider a uniform exploration policy, defined as follows. We first expand the root. Then, at each round n, we expand a node in $\mathcal{S}_n$ having the smallest depth. See Algorithm 1.

Algorithm 1 Uniform planning algorithm $\mathcal{A}_U$

Set $n = 0$. Expand the root.
while Numerical resource available **do**
  Expand a node $i \in \mathcal{S}_n$ with the smallest depth.
  $n = n + 1$
end while
return Action $\arg \max_{k \in \{1,\ldots,K\}} u_k(n)$

Thus, at all rounds, a uniform tree is expanded: Hence, at round $n = 1 + K + K^2 + \cdots + K^d = \frac{K^{d+1}-1}{K-1}$, all nodes of depth d or less have been expanded.

The uniform action-selection algorithm, written $\mathcal{A}_U$, returns, after n rounds, the action which corresponds to the children of the root $k \in \{1 \ldots K\}$ that has the highest u_k, i.e.:

$$\mathcal{A}_U(n) \stackrel{\text{def}}{=} \arg \max_{k \in \{1,\ldots,K\}} u_k(n),$$

(ties broken arbitrarily). In words, after n rounds, where we expanded the tree uniformly, $\mathcal{A}_U$ selects the action corresponding to the branch where we found the (finite) path with highest sum of rewards.

3.3 Analysis

Theorem 1. *Consider the uniform planning algorithm described above. Then for any reward function, the regret of the uniform algorithm is bounded by*

$$R_{\mathcal{A}_U}(n) \leq \frac{1}{\gamma(1-\gamma)} \big[n(K-1)+1\big]^{-\frac{\log 1/\gamma}{\log K}}. \tag{2}$$

Moreover, for all $n \geq 2$, there exists a reward function, such that the regret of this algorithm on this problem is at least

$$R_{\mathcal{A}_U}(n) \geq \frac{\gamma}{1-\gamma}\big[n(K-1)+1\big]^{-\frac{\log 1/\gamma}{\log K}}. \tag{3}$$

We deduce the dependency (in a worst-case sense): $R_{\mathcal{A}_U}(n) = \Theta(n^{-\frac{\log 1/\gamma}{\log K}})$.

Proof. For any $n \geq 2$, let d be the largest integer such that

$$n \geq \frac{K^{d+1}-1}{K-1}. \tag{4}$$

Thus all nodes of depth d have been expanded. Thus for all $k \in \{1,\ldots,K\}$, we have $v_k \leq u_k + \frac{\gamma^{d+1}}{1-\gamma}$, since all rewards up to depth d have been seen. Thus:

$$v^* = \max_{k\in\{1,\ldots,K\}} v_k \leq \max_{k\in\{1,\ldots,K\}} u_k + \frac{\gamma^{d+1}}{1-\gamma} = u_{\mathcal{A}_U(n)} + \frac{\gamma^{d+1}}{1-\gamma} \leq v_{\mathcal{A}_U(n)} + \frac{\gamma^{d+1}}{1-\gamma}.$$

Now, from (4), we have $d \geq \log_K[n(K-1)+1]-2$, from which we deduce that

$$v^* - v_{\mathcal{A}_U(n)} \leq \frac{\gamma^{d+1}}{1-\gamma} \leq \frac{1}{\gamma(1-\gamma)}\big[n(K-1)+1\big]^{-\log_K 1/\gamma}.$$

For the second part of the Theorem, consider a fixed n. Define d as the largest integer such that (4) holds. Thus, from (4), we have $d \leq \log_K[n(K-1)+1]-1$. Define the following reward function: all rewards are 0 except in a branch (say branch 1) where the rewards of all transitions from nodes of depth p to depth $p+1$, where $p > d+1$, is 1. Thus $v^* = v_1 = \frac{\gamma^{d+2}}{1-\gamma}$ and $v_2 = 0$.

Thus at the time of the decision, all u_k (for $k \in \{1,\ldots,K\}$) equal zero (only 0 rewards have been observed), and an arbitrary action (say action 2) is selected by the algorithm.

Thus $v^* - v_{\mathcal{A}_U(n)} = v_1 - v_2 = \frac{\gamma^{d+2}}{1-\gamma}$, and (3) follows from the bound on d. □

As a consequence of Theorem 1 and Proposition 1, we deduce that in order to guarantee that the performance of the uniform planning policy $\pi_{\mathcal{A}(n)}$ is ϵ-optimal (i.e. $||V^* - V^{\pi_{\mathcal{A}(n)}}||_\infty \leq \epsilon$), we need to devote $n = \Theta\big(\frac{1}{\epsilon(1-\gamma)^2}\big)^{\frac{\log K}{\log 1/\gamma}}$ units of resource per decision-step.

In this paper, we sought for other action-selection algorithms $\mathcal{A}$ that could make better use of available resources in the sense of minimizing the regret $R_{\mathcal{A}}(n)$ for a fixed amount n of computational resources. Note that the worst-case analysis considered in the second part of the proof of the previous Theorem may discourage us for searching for better bounds, since a similar analysis could be pursued on any specific algorithm. However we would like to define relevant classes of problems for which one would expect to obtain better convergence rates (for the regret) when using other action-selection algorithms than uniform: One cannot hope to achieve better rates for all problems but this may be possible for specific classes of problems.

4 Optimistic planning

In this section, we present an algorithm building an asymmetric look-ahead tree aiming at exploring first the most promising parts of the tree.

4.1 The algorithm

We define the b-value of each node $i \in \mathcal{S}_n$ of depth d by $b_i \stackrel{\text{def}}{=} u_i + \frac{\gamma^d}{1-\gamma}$, and for any node $i \in \mathcal{T}_n$, its b-value is defined recursively by: $b_i \stackrel{\text{def}}{=} \max_{j \in \mathcal{C}(i)} b_j$.

Note that like the u-values, the b-values are also well-defined and also depend on n, so we will write them $b_i(n)$ whenever the explicit dependency w.r.t. n is relevant. A property is that $b_i(n)$ is a decreasing function of n. Now, since all rewards are assumed to be in $[0,1]$, we have the immediate property that these b-values are upper-bounds on the values of the nodes: for all $i \in \mathcal{T}_t \cup \mathcal{S}_t$, for all $n \geq t$,

$$u_i(n) \leq v_i \leq b_i(n).$$

The optimistic exploration policy consists in expanding in each round a node $i \in \mathcal{S}_n$ which possesses the highest b-value. See Algorithm 2. The returned action corresponds to the child of the root with highest u-value: $\mathcal{A}_O(n) \stackrel{\text{def}}{=} \arg\max_{k \in \{1,\ldots,K\}} u_k(n)$ (ties broken arbitrarily).

Algorithm 2 Optimistic planning algorithm $\mathcal{A}_O$

Set $n = 0$. Expand the root.
while Numerical resource available **do**
  Expand a node $i \in \mathcal{S}_n$ s.t. $\forall j \in \mathcal{S}_n$, $b_i(n) \geq b_j(n)$.
  $n = n + 1$
end while
return Action $\arg\max_{k \in \{1,\ldots,K\}} u_k(n)$

4.2 Analysis

Theorem 2. *Consider the optimistic planning algorithm described above. At round n, the regret is bounded by*

$$R_{\mathcal{A}_O}(n) \leq \frac{\gamma^{d_n}}{1-\gamma}, \tag{5}$$

where d_n is the depth of the expanded tree $\mathcal{T}_n$ (maximal depth of nodes in $\mathcal{T}_n$).

As a consequence, for any reward function, the upper bound on the regret for the optimistic planning is never larger than that of the uniform planning (since the uniform exploration is the exploration strategy which minimizes the depth d_n for a given n, the depth obtained when using an optimistic algorithm is at least as high as that of the uniform one).

Proof. First let us notice that the action returned by the optimistic algorithm corresponds to a deepest explored branch. Indeed, if this was not the case, this would mean that if we write $i \in \mathcal{T}_n$ a node of maximal depth d_n, there exists a node $j \in \mathcal{T}_n$ of depth $d < d_n$ such that $u_j(n) \geq u_i(n)$. Thus there would exist a round $t \leq n$ such that node i has been expanded at round t, which would mean that $b_i(t) \geq b_j(t)$. But this is impossible since $b_i(t) = u_i(t) + \frac{\gamma^{d_n}}{1-\gamma} \leq u_i(n) + \frac{\gamma^{d_n}}{1-\gamma} \leq u_j(n) + \frac{\gamma^{d_n}}{1-\gamma} < u_j(n) + \frac{\gamma^{d}}{1-\gamma} = b_j(n) \leq b_j(t)$.

Thus the action returned by the algorithm corresponds to a branch that has been the most deeply explored. Let $i \in \mathcal{T}_n$ be a node of maximal depth d_n. Node i belongs to one of the K branches connected to the root, say branch 1. If the optimal action is 1 then the loss is 0 and the upper bound holds. Otherwise, the optimal action is not 1, say it is 2. Let $t \leq n$ be the round at which the node i was expanded. This means that $b_i(t) \geq b_j(t)$ for all nodes $j \in \mathcal{S}_t$, thus also for all nodes $j \in \mathcal{T}_t$. In particular, $b_1(t) = b_i(t) \geq b_2(t)$. But $b_2(t) \geq b_2(n)$. Now, the u-values are always lower bounds on the v-values, thus $u_1(t) \leq v_1$, and from the definition of u-values, $u_i(t) \leq u_1(t)$. We thus have:

$$\begin{aligned} v^* - v_{\mathcal{A}_O(n)} = v_2 - v_1 &\leq b_2(n) - v_1 \leq b_2(t) - v_1 \\ &\leq b_1(t) - u_1(t) \leq b_i(t) - u_i(t) = \frac{\gamma^{d_n}}{1-\gamma}, \end{aligned}$$

which concludes the proof. □

Remark 1 *This result shows that the upper bound for optimistic planning cannot be worst than the upper bound for uniform planning. This does not mean that for any problem, the optimistic algorithm will perform at least as well as a uniform algorithm. Indeed, this is not true since, if we consider the example mentioned in the proof of Theorem 1, if at time n all observed rewards are* 0*, any algorithm (such as the optimistic one) would deliver an arbitrary decision, which may be worst than another arbitrary decision (made for example by the uniform algorithm). However, if we consider equivalent classes of problems, where classes are defined by trees having the same reward function up to possible permutations of branches, then we conjecture that we have the stronger result that for any problem, the optimistic planning is never worse than the uniform planning performed on a problem of the same class. However we will not pursue this research further in this paper.*

Note that the lower bound obtained for the uniform planning also holds for the optimistic planning (the proof is the same: since, up to round n all rewards are 0, the optimistic planning will also build a uniform tree). This shows that no improvement (over uniform planning) may be expected in a worst-case setting, as already mentioned. In order to quantify possible improvement over uniform planning, one thus needs to define specific classes of problems.

For any $\epsilon \in [0, 1]$, define the proportion $p_d(\epsilon)$ of ϵ-optimal nodes of depth d:

$$p_d(\epsilon) \stackrel{\text{def}}{=} |\{\text{node } i \text{ of depth } d \text{ s.t. } v_i \geq v^* - \epsilon\}| K^{-d},$$

and write $p(\epsilon) \stackrel{\text{def}}{=} \lim_{d\to\infty} p_d(\epsilon)$ the proportion of ϵ-optimal paths in the tree (note that this limit is well defined since $p_d(\epsilon)$ is a decreasing function of d).

Theorem 3. *Let $\beta \in [0, \frac{\log K}{\log 1/\gamma}]$ be such that the proportion of ϵ-optimal nodes of depth $d \geq d_0$ (for some depth d_0) in the tree $\mathcal{T}$ is $O(\epsilon^\beta)$. More precisely, we assume that there exists positive constants d_0 and c, such that $\forall d \geq d_0$ $\forall \epsilon \geq 0$, $p_d(\epsilon) \leq c\epsilon^\beta$. Now let us define $\kappa \stackrel{\text{def}}{=} K\gamma^\beta$, which belongs to the interval $[1, K]$.*

If $\beta < \frac{\log K}{\log 1/\gamma}$ (i.e. $\kappa > 1$), then the regret of the optimistic algorithm is

$$R_{\mathcal{A}_O}(n) = O\big(n^{-\frac{\log 1/\gamma}{\log \kappa}}\big).$$

If $\beta = \frac{\log K}{\log 1/\gamma}$ (i.e. $\kappa = 1$), then we have the exponential rate:

$$R_{\mathcal{A}_O}(n) = O\big(\gamma^{\frac{(1-\gamma)^\beta}{c} n}\big).$$

Proof. Let us define the sub-tree $\mathcal{T}_\infty \subset \mathcal{T}$ of all the nodes i of depth d that are $\frac{\gamma^d}{1-\gamma}$-optimal, i.e.

$$\mathcal{T}_\infty \stackrel{\text{def}}{=} \bigcup_{d\geq 0} \big\{\text{node } i \text{ of depth } d \text{ s.t. } v_i + \frac{\gamma^d}{1-\gamma} \geq v^*\big\}.$$

Let us prove that all nodes expanded by the optimistic algorithm are in $\mathcal{T}_\infty$. Indeed, let i be a node of depth d expanded at round n. Then $b_i(n) \geq b_j$ for all $j \in \mathcal{S}_n \cup \mathcal{T}_n$, thus $b_i(n) = b_0(n)$ (b-value of the root). But $b_0(n) \geq v_0 = v^*$, thus $v_i \geq u_i(n) = b_i(n) - \frac{\gamma^d}{1-\gamma} \geq v^* - \frac{\gamma^d}{1-\gamma}$, thus $i \in \mathcal{T}_\infty$.

Now, from the definition of β, there exists d_0 such that the proportion of ϵ-optimal nodes of depth $d > d_0$ is at most $c\epsilon^\beta$, where c is a constant. We thus have that the number n_d of nodes of depth d in $\mathcal{T}_\infty$ is bounded by $c\big(\frac{\gamma^d}{1-\gamma}\big)^\beta K^d$.

Now, write d_n the depth of the expanded tree $\mathcal{T}_n$ at round n. Let $n_0 = \frac{K^{d_0+1}-1}{K-1}$ the number of nodes in $\mathcal{T}$ of depth less than d_0. We have:

$$n \leq n_0 + \sum_{d=d_0+1}^{d_n} n_d \leq n_0 + c \sum_{d=d_0+1}^{d_n} \big(\frac{\gamma^d}{1-\gamma}\big)^\beta K^d = n_0 + c' \sum_{d=d_0+1}^{d_n} \kappa^d$$

with $c' = c/(1-\gamma)^\beta$ and $\kappa = \gamma^\beta K$.

First, if $\kappa > 1$ then we have $n \leq n_0 + c'\kappa^{d_0+1}\frac{\kappa^{d_n-d_0}-1}{\kappa-1}$. Thus $d_n \geq d_0 + \log_K \frac{(n-n_0)(\kappa-1)}{c'\kappa^{d_0+1}}$. Now, from Theorem 2, we have the regret:

$$\begin{aligned} R_{\mathcal{A}_O}(n) &\leq \frac{\gamma^{d_n}}{1-\gamma} = \frac{1}{1-\gamma}\left[\frac{(n-n_0)(\kappa-1)}{c'\kappa^{d_0+1}}\right]^{\frac{\log\gamma}{\log\kappa}} \\ &= O\big(n^{-\frac{\log 1/\gamma}{\log\kappa}}\big). \end{aligned}$$

Now, if $\kappa = 1$, let $n_0 = \frac{K^{d_0+1}-1}{K-1}$. Following the same arguments as above, we deduce that $n \leq n_0 + \frac{c}{(1-\gamma)^\beta}(d_n - d_0)$, and the regret: $R_{\mathcal{A}_O}(n) \leq \frac{\gamma^{d_n}}{1-\gamma} = O\big(\gamma^{n\frac{(1-\gamma)^\beta}{c}}\big)$. □

Remark 2 *Let us notice that the proportion of ϵ-optimal paths is bounded by $c\epsilon^\beta$. β lies necessarily in the interval $[0, \log_{1/\gamma} K]$, and two extreme cases are:*

- *The case when all paths are optimal (i.e. all rewards are equal). This corresponds to $\beta = 0$, or $\kappa = K$.*
- *The case where there is only one path where rewards are 1, all other rewards being 0. Then, for any ϵ, the proportion of ϵ-optimal nodes of depth d is $1/K^d$ for $d \leq d_0$ for some depth d_0 for which $\frac{\gamma^{d_0}}{1-\gamma} \leq \epsilon$, and for all $d > d_0$ the proportion of ϵ-optimal nodes remains constant. Since $d_0 \geq \log_\gamma(1-\gamma)\epsilon$, the proportion of ϵ-optimal nodes of depth $d > d_0$ is at most $\left[(1-\gamma)\epsilon\right]^{\log_{1/\gamma} K}$, i.e. which corresponds to $\beta = \log_{1/\gamma} K$. This is the highest possible value for β. This corresponds to $\kappa = 1$.*

From this result, we see that when $\kappa > 1$, the decrease rate of the regret for the optimistic algorithm is $n^{-\frac{\log 1/\gamma}{\log \kappa}}$ instead of the $n^{-\frac{\log 1/\gamma}{\log K}}$ for the uniform one. By looking at the proof, we observe that κ plays the role of the **branching factor of the tree** $\mathcal{T}_\infty$ of the expandable nodes, similarly to K being the branching factor of the full tree $\mathcal{T}$. κ belongs to the interval $[1, K]$, thus the decrease rate for the regret of the optimistic planning is always at least as good as that of the uniform one. The bound for the optimistic planning is better than uniform whenever $\kappa < K$, and greatly better when κ is close to 1. Note that the tree $\mathcal{T}_\infty$ represents the set of nodes i such that given the observed rewards from the root to node i, one cannot decide whether i lies in an optimal path or not. This represents the set of nodes which would have to be expanded for finding an optimal path. Thus the performance of the optimistic algorithm is expressed in terms of the branching factor of the subtree $\mathcal{T}_\infty$ composed of all the nodes that have to be expanded eventually. We believe this is a strong optimality result, although we do not have lower bounds expressed in terms of β, yet.

Remark 3 *The case $\kappa = 1$ is also interesting because it provides rates that are exponential in n instead of polynomial ones. This case would hold for example if there exists d_0 such that for each node i_d of depth $d \geq d_0$ along an optimal path, if we write x_d the state corresponding to that node i_d, we require that the gap between the optimal value function at x_d and the Q-value of all suboptimal actions are larger than some constant value $\Delta > 0$, i.e., $\exists \Delta > 0$, for all $d \geq d_0$,*

$$V^*(x_d) - \max_{a \text{ s.t. } Q^*(x_d,a) < V^*(x_d)} Q^*(x_d, a) \geq \Delta. \tag{6}$$

Indeed, if this was true, we would have for all $d \geq d_0$, for any non-optimal child j of i_d, $v_j \leq v^* - \Delta\gamma^d$ (since $v_{i_d} = v^*$ because the nodes i_d are optimal). Now, the number of nodes in the branch j that belong to the expandable tree $\mathcal{T}_\infty$ is bounded by K^{h-d-1} where h is the maximal depth such that $v_j + \frac{\gamma^h}{1-\gamma} \geq v^*$, i.e., $\gamma^{h-d} \geq \Delta(1-\gamma)$. Thus $K^{h-d-1} = \frac{1}{K}\left[(1-\gamma)\Delta\right]^{-\beta}$ with $\beta = \log K / \log(1/\gamma)$.

Thus the number of nodes of depth $d \geq d_0$ in $\mathcal{T}_\infty$ is bounded by a constant independent of d, i.e. $\frac{1}{K}[\Delta(1-\gamma)]^{-\beta}$. Thus $p_d(\frac{\gamma^d}{1-\gamma}) \leq \frac{1}{K}\left[\Delta(1-\gamma)\right]^{-\beta}/K^d$. Thus

$p_n(\epsilon) \leq \frac{1}{K}[\epsilon/\Delta]^\beta$. *Thus* $p(\epsilon) \leq \frac{1}{K}[\epsilon/\Delta]^\beta$ *and we have* $\kappa = 1$ *and the constant* $c = \frac{1}{K}(1/\Delta)^\beta$.

Remark 4 *A natural extension of the previous case (for which we deduced* $\kappa = 1$*) is when from each state* x_d *corresponding to a node of depth* $d \geq d_0$*, for some* d_0*, there exist a small number* m *of* Δ*-optimal sequences of* h *actions, where* $\Delta > 0$ *is a fixed constant and* h *a fixed integer. This means that from* x_d *there exist at most* m *sequences of actions* $a_1, \ldots, a_h$*, leading to states* $(x_{d+i})_{1 \leq i \leq h}$*, such that*

$$\sum_{i=0}^{h-1} \gamma^i r(x_{d+i}, a_{i+1}) + \gamma^h V^*(x_{d+h}) \geq V^*(x_d) - \Delta.$$

Then one can prove that the branching factor κ *of the expandable tree is at most* $m^{1/h}$*. Notice that in the case presented in the previous remark we had* $m = 1$ *and* $h = 1$ *(and we deduced that* $\kappa = 1$*). However note that in the previous remark, we only assumed the property (6) along the optimal path (whereas we impose here that it holds for all nodes). The proof of this result is rather technical and not included here.*

5 Numerical experiments

We have done some numerical experiments to compare uniform and optimistic planning algorithms. We consider the system $\ddot{y}_t = a_t$, where a point defined by its position y_t and its velocity v_t is controlled by a force (action) $a_t \in \{-1, 1\}$. The dynamics are: $(y_{t+1}, v_{t+1})' = (y_t, v_t)' + (v_t, a_t)'\Delta t$, where $\Delta t = 0.1$ is the time discretization step. The reward of state (y_t, v_t) action a_t is defined by $\max(1 - y_{t+1}^2, 0)$ where y_{t+1} is the position of the resulting next state.

In figure 2 we show the trees resulting from the uniform and optimistic algorithms using the same number of numerical resources $n = 3000$. The initial state is $(y_0 = -1, v_0 = 0)$ and the discount factor $\gamma = 0.9$. The optimistic tree is not deeply explored on the left side of the initial state since this corresponds to states with low rewards, however it is more deeply expanded along some branches which provide high rewards, and in particular, the branch leading the states around the origin is very deeply (and sharply) expanded (maximal depth of 49).

We computed an average regret of both algorithms for $n = 2^{d+1} - 1$ with $d \in \{2, \ldots, 18\}$ (full trees of depth d), where the average is performed over 1000 trees where the initial state have been uniformly randomly sampled in the domain $[-1, 1] \times [-2, 2]$. Figure 3 shows the regret (in log scales) for both algorithms. The slope $\frac{\log R(n)}{\log n}$ of these curves indicate the exponent in the regret polynomial dependency. For the uniform curve, we calculate a numerical slope of about -1, thus the regret $R_{\mathcal{A}_U}(n) \simeq 1/n$. For the optimistic curve, the numerical slope is about $-3 = -\frac{\log 1/\gamma}{\kappa}$ which corresponds to the branching factor $\kappa = 1.04$. The regret of optimistic planning is thus of order $1/n^3$ which is significantly a better rate than the uniform one, and explains the great improvement of the optimistic

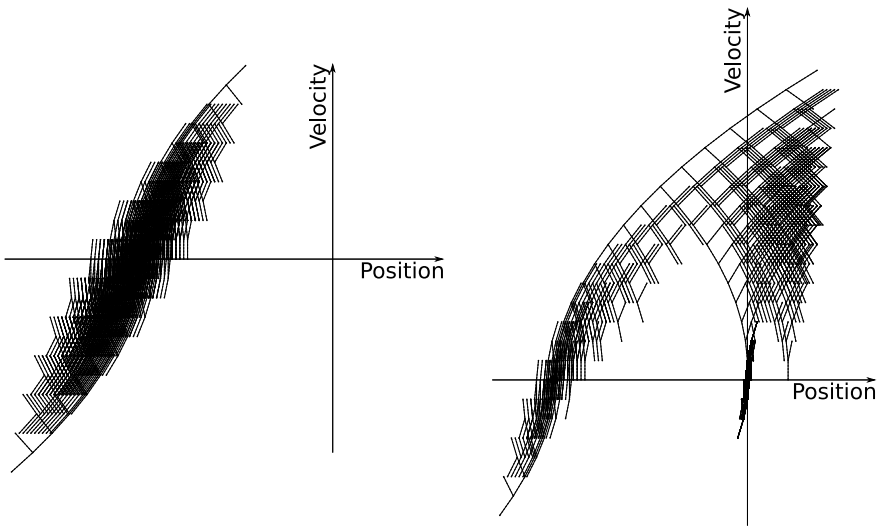

Fig. 2. Expanded trees with $n = 3000$ calls to the generative model using the uniform (left figure) and optimistic (right) planning algorithms. The depth of the trees is 11 for the uniform and 49 for the optimistic.

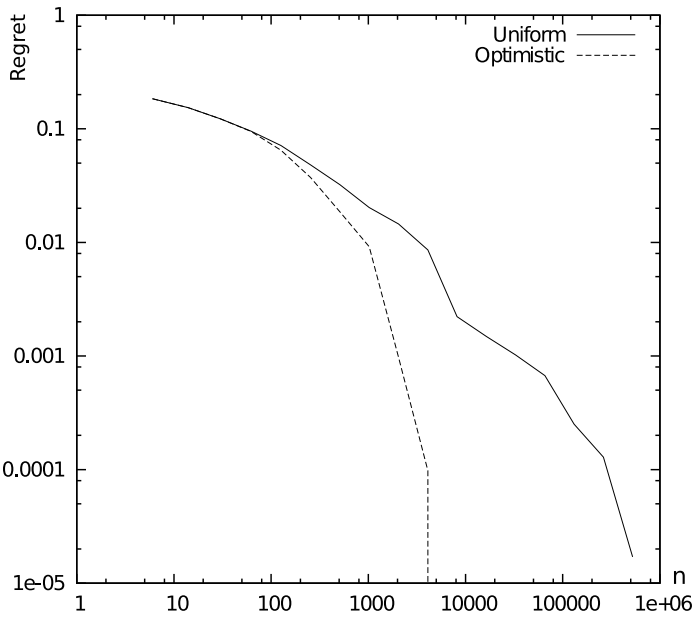

Fig. 3. Average regret $R(n)$ for both algorithms as a function of $n = 2^{d+1} - 1$ with $d \in \{2, \ldots, 18\}$, in logarithmic scales. The slope of each curve indicates the exponent in the actual polynomial dependency of the regret.

planning over the uniform approach. For illustration, achieving a regret of 0.0001 with uniform planning requires more than $n = 262142$ expanded nodes whereas optimistic planning only requires $n = 4094$ such nodes.

We do not have space to describe more challenging applications here but other simulations, including the double inverted pendulum linked by a spring, are described at the address: `http://sequel.futurs.inria.fr/hren/optimistic/`.

6 Conclusions and future works

An immediate remaining work is the derivation of β-dependant lower bounds for the optimistic planning. An extension of this work would consider the case of stochastic rewards, like in [CM07], where an additional term (coming from a Chernoff-Hoeffing bound) would be added to the b-values to define high probability upper-confidence bounds. Another, more challenging, extension would consider the stochastic transitions case. The possibility of combining this pure search approach with local approximation of the optimal value function is certainly worth investigating too.

References

[ACBF02] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning Journal*, 47(2-3):235–256, 2002.

[BT96] Dimitri P. Bertsekas and J. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

[CM07] P.-A. Coquelin and R. Munos. Bandit algorithms for tree search. *Uncertainty in Artificial Intelligence*, 2007.

[GWMT06] S. Gelly, Y. Wang, R. Munos, and O. Teytaud. Modification of UCT with patterns in Monte-Carlo go. *Technical Report INRIA RR-6062*, 2006.

[KMN02] M. Kearns, Y. Mansour, and A.Y. Ng. A sparse sampling algorithm for near-optimal planning in large Markovian decision processes. In *Machine Learning*, volume 49, pages 193–208, 2002.

[KS06] L. Kocsis and Cs. Szepesvari. Bandit based monte-carlo planning. *European Conference on Machine Learning*, pages 282–293, 2006.

[LR85] T. L. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6:4–22, 1985.

[PG04] L. Péret and F. Garcia. On-line search for solving large Markov decision processes. In *Proceedings of the 16th European Conference on Artificial Intelligence*, 2004.

[Put94] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley and Sons, 1994.

[Rob52] H. Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematics Society*, 58:527–535, 1952.

[SB98] R. Sutton and A. Barto. *Reinforcement Learning*. MIT Press, 1998.