Practical Open-Loop Optimistic Planning

Edouard Leurent[1,2](🖂) and Odalric-Ambrym Maillard[1]

[1] SequeL team, INRIA Lille - Nord Europe, France
{edouard.leurent,odalric.maillard}@inria.fr
[2] Renault Group, France

Abstract. We consider the problem of online planning in a Markov Decision Process when given only access to a generative model, restricted to open-loop policies - i.e. sequences of actions - and under budget constraint. In this setting, the *Open-Loop Optimistic Planning* (`OLOP`) algorithm enjoys good theoretical guarantees but is overly conservative in practice, as we show in numerical experiments. We propose a modified version of the algorithm with tighter upper-confidence bounds, `KL-OLOP`, that leads to better practical performances while retaining the sample complexity bound. Finally, we propose an efficient implementation that significantly improves the time complexity of both algorithms.

Keywords: Planning · Online learning · Tree search.

1 Introduction

In a *Markov Decision Process* (MDP), an agent observes its current state s from a state space S and picks an action a from an action space A, before transitioning to a next state s' drawn from a transition kernel $\mathbb{P}(s'|s,a)$ and receiving a bounded reward $r \in [0,1]$ drawn from a reward kernel $\mathbb{P}(r|s,a)$. The agent must act so as to optimise its expected cumulative discounted reward $\mathbb{E}\sum_t \gamma^t r_t$, also called expected *return*, where $\gamma \in [0,1)$ is the discount factor. In *Online Planning* [14], we do not consider that these transition and reward kernels are known as in *Dynamic Programming* [1], but rather only assume access to the MDP through a *generative model* (e.g. a simulator) which yields samples of the next state $s' \sim \mathbb{P}(s'|s,a)$ and reward $r \sim \mathbb{P}(r|s,a)$ when queried. Finally, we consider a *fixed-budget* setting where the generative model can only be called a maximum number of times, called the budget n.

Monte-Carlo Tree Search (`MCTS`) algorithms were historically motivated by the application of computer Go, and made a first appearance in the CrazyStone software [8]. They were later reformulated in the setting of Multi-Armed Bandits by [12] with their *Upper Confidence bounds applied to Trees* (`UCT`) algorithm. Despite its popularity [15,17,16], `UCT` has been shown to suffer from several limitations: its sample complexity can be at least doubly-exponential for some problems (e.g. when a narrow optimal path is hidden in a suboptimal branch), which is much worse than uniform planning [7]. The `Sparse Sampling` algorithm of [11] achieves better worst-case performance, but it is still non-polynomial and doesn't adapt to the structure of the MDP. In stark contrast, the *Optimistic Planning for Deterministic systems* (OPD) algorithm considered by [10] in the case of deterministic transitions and rewards exploits the structure of the cumulative discounted reward to achieve a problem-dependent polynomial bound on sample complexity. A similar line of work in a deterministic setting is that of SOOP and OPC by [3,4] though they focus on continuous action spaces. OPD was later extended to stochastic systems with the *Open-Loop Optimistic Planning* (OLOP) algorithm introduced by [2] in the open-loop setting: we only consider sequences of actions independently of the states that they lead to. This restriction in the space of policies causes a loss of optimality, but greatly simplifies the planning problem in the cases where the state space is large or infinite. More recent work such as StOp [18] and TrailBlazer [9] focus on the probably approximately correct (PAC) framework: rather than simply recommending an action to maximise the expected rewards, they return an ε-approximation of the value at the root that holds with high probability. This highly demanding framework puts a severe strain on these algorithms that were developed for theoretical analysis only and cannot be applied to real problems.

Contributions The goal of this paper is to study the practical performances of OLOP when applied to numerical problems. Indeed, OLOP was introduced along with a theoretical sample complexity analysis but no experiment was carried-out. Our contribution is threefold:

- First, we show that in our experiments OLOP is overly pessimistic, especially in the low-budget regime, and we provide an intuitive explanation by casting light on an unintended effect that alters the behaviour of OLOP.
- Second, we circumvent this issue by leveraging modern tools from the bandits literature to design and analyse a modified version with tighter upper-confidence bounds called KL-OLOP. We show that we retain the asymptotic regret bounds of OLOP while improving its performances by an order of magnitude in numerical experiments.
- Third, we provide a time and memory efficient implementation of OLOP and KL-OLOP, bringing an exponential speedup that allows to scale these algorithms to high sample budgets.

The paper is structured as follows: in section 2, we present OLOP, give some intuition on its limitations, and introduce KL-OLOP, whose sample complexity is further analysed in section 3. In section 4, we propose an efficient implementation of the two algorithms. Finally in section 6, we evaluate them in several numerical experiments.

Notations Throughout the paper, we follow the notations from [2] and use the standard notations over alphabets: a finite word $a \in A^*$ of length h represents a sequence of actions $(a_0, \cdots, a_h) \in A^h$. Its prefix of length $t \leq h$ is denoted $a_{1:t} = (a_0, \cdots, a_t) \in A^t$. A^∞ denotes the set of infinite sequences of actions. Two finite sequences $a \in A^*$ and $b \in A^*$ can be concatenated as $ab \in A^*$, the set of finite and infinite suffixes of a are respectively $aA^* = \{c \in \mathcal{A}^* : \exists b \in A^*$ such that $c = ab\}$ and aA^∞ defined likewise, and the empty sequence is $\emptyset$.

During the planning process, the agent iteratively selects sequences of actions until it reaches the allowed budget of n actions. More precisely, at time t during the m^{th} sequence, the agent played $a^m_{1:t} = a^m_1 \cdots a^m_t \in A^t$ and receives a reward Y^m_t. We denote the probability distribution of this reward as $\nu(a^m_{1:t}) = \mathbb{P}\left(Y^m_t | s_t, a^m_t\right) \prod_{k=1}^{t-1} \mathbb{P}\left(s_{k+1} | s_k, a^m_k\right)$, and its mean as $\mu(a^m_{1:t})$, where s_1 is the current state.

After this exploration phase, the agent selects an action $a(n)$ so as to minimise the *simple regret* $r_n = V - V(a(n))$, where $V = V(\emptyset)$ and $V(a)$ refers to the value of a sequence of actions $a \in A^h$, that is, the maximum expected discounted cumulative reward one may obtain after executing a:

$$V(a) = \sup_{b \in aA^\infty} \sum_{t=1}^{\infty} \gamma^t \mu(b_{1:t}), \tag{1}$$

2 Kullback-Leibler Open-Loop Optimistic Planning

In this section we present KL-OLOP, a combination of the OLOP algorithm of [2] with the tighter Kullback-Leibler upper confidence bounds from [5]. We first frame both algorithms in a common structure before specifying their implementations.

2.1 General structure

First, following OLOP, the total sample budget n is split in M trajectories of length L in the following way:

$$\begin{aligned} &M \text{ is the largest integer such that } M\lceil \log M/(2\log 1/\gamma)\rceil \le n;\\ &L = \lceil \log M/(2\log 1/\gamma)\rceil. \end{aligned}$$

The look-ahead tree of depth L is denoted $\mathcal{T} = \sum_{h=0}^{L} A^h$.

Then, we introduce some useful definitions. Consider episode $1 \le m \le M$. For any $1 \le h \le L$ and $a \in A^h$, let

$$T_a(m) \overset{\text{def}}{=} \sum_{s=1}^{m} \mathbb{1}\{a^s_{1:h} = a\}$$

be the number of times we played an action sequence starting with a, and $S_a(m)$ the sum of rewards collected at the last transition of the sequence a:

$$S_a(m) \overset{\text{def}}{=} \sum_{s=1}^{m} Y^s_h \mathbb{1}\{a^s_{1:h} = a\}$$

The empirical mean reward of a is $\hat{\mu}_a(m) \overset{\text{def}}{=} \dfrac{S_a(m)}{T_a(m)}$ if $T_a(m) > 0$, and $+\infty$ otherwise. Here, we provide a more general form for upper and lower confidence

Algorithm 1: General structure for Open-Loop Optimistic Planning

```
for each episode $m = 1, \cdots, M$ do
    Compute $U_a(m-1)$ from (4) for all $a \in \mathcal{T}$
    Compute $B_a(m-1)$ from (5) for all $a \in A^L$
    Sample a sequence with highest B-value: $a^m \in \arg\max_{a \in A^L} B_a(m-1)$
return the most played sequence $a(n) \in \arg\max_{a \in A^L} T_a(M)$
```

Table 1: Different implementations of Algorithm 1 in OLOP and KL-OLOP

Algorithm	OLOP	KL-OLOP
Interval I	$\mathbb{R}$	$[0, 1]$
Divergence d	$d_{\texttt{QUAD}}$	$d_{\texttt{BER}}$
$f(m)$	$4 \log M$	$2 \log M + 2 \log \log M$

bounds on these empirical means:

$$U_a^\mu(m) \stackrel{\text{def}}{=} \max \left\{ q \in I : T_a(m) d(\tfrac{S_a(m)}{T_a(m)}, q) \leq f(m) \right\} \tag{2}$$

$$L_a^\mu(m) \stackrel{\text{def}}{=} \min \left\{ q \in I : T_a(m) d(\tfrac{S_a(m)}{T_a(m)}, q) \leq f(m) \right\} \tag{3}$$

where I is an interval, d is a divergence on $I \times I \to \mathbb{R}^+$ and f is a non-decreasing function. They are left unspecified for now and their particular implementations and associated properties will be discussed in the following sections.

These upper-bounds U_a^μ for intermediate rewards finally enable us to define an upper bound U_a for the value $V(a)$ of the entire sequence of actions a:

$$U_a(m) \stackrel{\text{def}}{=} \sum_{t=1}^{h} \gamma^t U_{a_{1:t}}^\mu(m) + \frac{\gamma^{h+1}}{1-\gamma} \tag{4}$$

where $\frac{\gamma^{h+1}}{1-\gamma}$ comes from upper-bounding by one every reward-to-go in the sum (1), for $t \geq h+1$. In [2], there is an extra step to "sharpen the bounds" of sequences $a \in A^L$ by taking:

$$B_a(m) \stackrel{\text{def}}{=} \inf_{1 \leq t \leq L} U_{a_{1:t}}(m) \tag{5}$$

The general algorithm structure is shown in Algorithm 1. We now discuss two specific implementations that differ in their choice of divergence d and non-decreasing function f. They are compared in Table 1.

2.2 OLOP

To recover the original OLOP algorithm of [2] from Algorithm 1, we can use a quadratic divergence $d_{\texttt{QUAD}}$ on $I = \mathbb{R}$ and a constant function f_4 defined as follows:

$$d_{\texttt{QUAD}}(p, q) \stackrel{\text{def}}{=} 2(p-q)^2, \qquad f_4(m) \stackrel{\text{def}}{=} 4 \log M$$

Indeed, in this case $U^\mu_a(m)$ can then be explicitly computed as:

$$U^\mu_a(m) = \max\left\{q \in \mathbb{R} : 2(\frac{S_a(m)}{T_a(m)} - q)^2 \leq \frac{4\log M}{T_a(m)}\right\} = \hat{\mu}_a(m) + \sqrt{\frac{2\log M}{T_a(m)}}$$

which is the Chernoff-Hoeffding bound used originally in section 3.1 of [2].

2.3 An unintended behaviour

From the definition of $U_a(m)$ as an upper-bound of the value of the sequence a, we expect increasing sequences $(a_{1:t})_t$ to have non-increasing upper-bounds. Indeed, every new action a_t encountered along the sequence is a potential loss of optimality. However, this property is only true if the upper-bound defined in (2) belongs to the reward interval $[0, 1]$.

Lemma 1. *(Monotony of $U_a(m)$ along a sequence)*

- *If it holds that $U^\mu_b(m) \in [0, 1]$ for all $b \in A^*$, then for any $a \in A^L$ the sequence $(U_{a_{1:h}}(m))_{1\leq h\leq L}$ is non-increasing, and we simply have $B_a(m) = U_a(m)$.*
- *Conversely, if $U^\mu_b(m) > 1$ for all $b \in A^*$, then for any $a \in A^L$ the sequence $(U_{a_{1:h}}(m))_{1\leq h\leq L}$ is non-decreasing, and we have $B_a(m) = U_{a_{1:1}}(m)$.*

Proof. We prove the first proposition, and the same reasoning applies to the second. For $a \in A^L$ and $1 \leq h \leq L - 1$, we have by (4):

$$\begin{aligned} U_{a_{1:h+1}}(m) - U_{a_{1:h}}(m) &= \gamma^{h+1} U^\mu_{a_{1:h+1}}(m) + \frac{\gamma^{h+2}}{1-\gamma} - \frac{\gamma^{h+1}}{1-\gamma} \\ &= \gamma^{h+1}(\underbrace{U^\mu_{a_{1:h+1}}(m)}_{\in[0,1]} - 1) \leq 0 \end{aligned}$$

We can conclude that $(U_{a_{1:h}}(m))_{1\leq h\leq L}$ is non-increasing and that $B_a(m) = \inf_{1\leq h\leq L} U_{a_{1:h}}(m) = U_{a_{1:L}}(m) = U_a(m)$. □

Yet, the Chernoff-Hoeffding bounds used in `OLOP` start in the $U^\mu_a(m) > 1$ regime – initially $U^\mu_a(m) = \infty$ – and can remain in this regime for a long time especially in the near-optimal branches where $\hat{\mu}_a(m)$ is close to one.

Under these circumstances, the Lemma 1 has a drastic effect on the search behaviour. Indeed, as long as a subtree under the root verifies $U^\mu_a(m) > 1$ for every sequence a, then all these sequences share the same B-value $B_a(m) = U_{a_{1:1}(m)}$. This means that `OLOP` cannot differentiate them and exploit information from their shared history as intended, and behaves as uniform sampling instead. Once the early depths have been explored sufficiently, `OLOP` resumes its intended behaviour, but the problem is only shifted to deeper unexplored subtrees.

This consideration motivates us to leverage the recent developments in the Multi-Armed Bandits literature, and modify the upper-confidence bounds for the expected rewards $U^\mu_a(m)$ so that they respect the reward bounds.

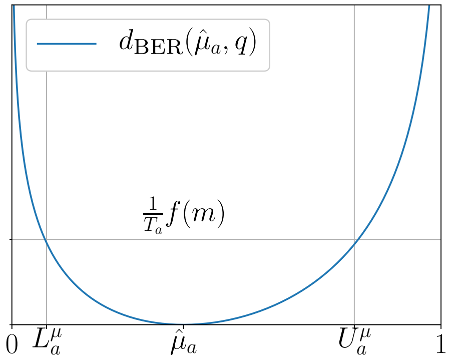

Fig. 1: The Bernoulli Kullback-Leibler divergence $d_{\mathtt{BER}}$, and the corresponding upper and lower confidence bounds U_a^μ and L_a^μ for the empirical average $\hat{\mu_a}$. Lower values of $f(m)$ give tighter confidence bounds that hold with lower probabilities.

2.4 KL-OLOP

We propose a novel implementation of Algorithm 1 where we leverage the analysis of the kl-UCB algorithm from [5] for multi-armed bandits with general bounded rewards. Likewise, we use the Bernoulli Kullback-Leibler divergence defined on the interval $I = [0, 1]$ by:

$$d_{\mathtt{BER}}(p, q) \overset{\text{def}}{=} p \log \frac{p}{q} + (1-p) \log \frac{1-p}{1-q}$$

with, by convention, $0 \log 0 = 0 \log 0/0 = 0$ and $x \log x/0 = +\infty$ for $x > 0$. This divergence and the corresponding bounds are illustrated in Figure 1.

$U_a^\mu(m)$ and $L_a^\mu(m)$ can be efficiently computed using Newton iterations, as for any $p \in [0, 1]$ the function $q \to d_{\mathtt{BER}}(p, q)$ is strictly convex and increasing (resp. decreasing) on the interval [p, 1] (resp. [0, p]).

Moreover, we use the constant function $f_2 : m \to 2 \log M + 2 \log \log M$. This choice is justified in the end of section 5. Because f_2 is lower than f_4, the Figure 1 shows that the bounds are tighter and hence less conservative than that of `OLOP`, which should increase the performance, provided that their associated probability of violation does not invalidate the regret bound of `OLOP`.

Remark 2 (Upper bounds sharpening). The introduction of the B-values $B_a(m)$ was made necessary in `OLOP` by the use of Chernoff-Hoeffding confidence bounds which are not guaranteed to belong to [0, 1]. On the contrary, we have in `KL-OLOP` that $U_a^\mu(m) \in I = [0, 1]$ by construction. By Lemma 1, the upper bounds sharpening step in line 3 of Algorithm 1 is now superfluous as we trivially have $B_a(m) = U_a(m)$ for all $a \in A^L$.

3 Sample complexity

We say that $u_n = \widetilde{O}(v_n)$ if there exist $\alpha, \beta > 0$ such that $u_n \leq \alpha \log(v_n)^\beta v_n$. Let us denote the proportion of near-optimal nodes κ_2 as:

$$\kappa_2 \overset{\text{def}}{=} \limsup_{h\to\infty} \left| \left\{ a \in a^H : V(a) \geq V - 2\frac{\gamma^{h+1}}{1-\gamma} \right\} \right|^{1/h}$$

Theorem 3 (Sample complexity). *We show that KL-OLOP enjoys the same asymptotic regret bounds as OLOP. More precisely, for any $\kappa' > \kappa_2$, KL-OLOP satisfies:*

$$\mathbb{E}\, r_n = \begin{cases} \widetilde{0}\left(n^{-\frac{\log 1/\gamma}{\log \kappa'}}\right), & \textit{if } \gamma\sqrt{\kappa'} > 1 \\ \widetilde{0}\left(n^{-\frac{1}{2}}\right), & \textit{if } \gamma\sqrt{\kappa'} \leq 1 \end{cases}$$

4 Time and memory complexity

After having considered the sample efficiency of OLOP and KL-OLOP, we now turn to study their time and memory complexities. We will only mention the case of KL-OLOP for ease of presentation, but all results easily extend to OLOP.

The Algorithm 1 requires, at each episode, to compute and store in memory of the reward upper-bounds and U-values of all nodes in the tree $\mathcal{T} = \sum_{h=0}^{L} A^h$. Hence, its time and memory complexities are

$$C(\texttt{KL-OLOP}) = O(M|\mathcal{T}|) = O(MK^L). \tag{6}$$

The curse of dimensionality brought by the branching factor K and horizon L makes it intractable in practice to actually run KL-OLOP in its original form even for small problems. However, most of this computation and memory usage is wasted, as with reasonable sample budgets n the vast majority of the tree $\mathcal{T}$ will not be actually explored and hence does not hold any valuable information.

We propose in Algorithm 2 a lazy version of KL-OLOP which only stores and processes the explored subtree, as shown in Figure 2, while preserving the inner workings of the original algorithm.

Theorem 4 (Consistency). *The set of sequences returned by Algorithm 2 is the same as the one returned by Algorithm 1. In particular, Algorithm 2 enjoys the same regret bounds as in Theorem 3.*

Property 5 (Time and memory complexity). Algorithm 2 has time and memory complexities of:

$$C(\texttt{Lazy KL-OLOP}) = O(KLM^2)$$

The corresponding complexity gain compared to the original Algorithm 1 is:

$$\frac{C(\texttt{Lazy KL-OLOP})}{C(\texttt{KL-OLOP})} = \frac{n}{K^{L-1}}$$

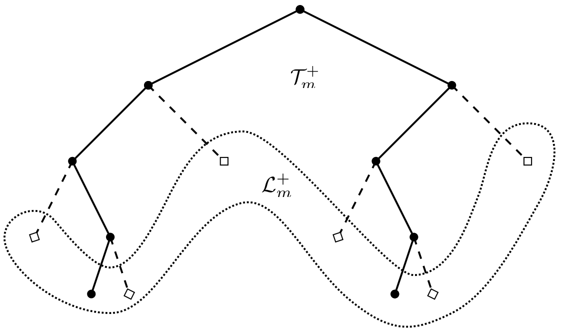

Fig. 2: A representation of the tree $\mathcal{T}_m^+$, with $K = 2$ actions and after episode $m = 2$, when two sequences have been sampled. They are represented with solid lines and dots $\bullet$, and they constitute the explored subtree $\mathcal{T}_m$. When extending $\mathcal{T}_m$ with the missing children of each node, represented with dashed lines and diamonds $\diamond$, we obtain the full extended subtree $\mathcal{T}_m^+$. The set of its leaves is denoted $\mathcal{L}_m^+$ and shown as a dotted set.

which highlights that only a subtree corresponding to the sample budget n is processed instead of the search whole tree $\mathcal{T}$.

Proof. At episode $m = 1, \cdots, M$, we compute and store in memory the reward upper-bounds and U-values of all nodes in the subtree $\mathcal{T}_m^+$. Moreover, the tree $\mathcal{T}_m^+$ is constructed iteratively by adding K nodes at most L times at each episode from 0 to m. Hence, $|\mathcal{T}_m^+| = O(mKL)$. This yields directly $C(\texttt{Lazy KL-OLOP}) = \sum_{m=1}^{M} O(mKL) = O(M^2KL)$. □

5 Proof of Theorem 3

We follow step-by step the pyramidal proof of [2], and adapt it to the Kullback-Leibler upper confidence bound. The adjustments resulting from the change of confidence bounds are highlighted. The proofs of lemmas which are not significantly altered are listed in the Supplementary Material.

We start by recalling their notations. Let $1 \leq H \leq L$ and $a^* \in A^L$ such that $V(a^*) = V$. Considering sequences of actions of length $1 \leq h \leq H$, we define the subset $\mathcal{I}_h$ of near-optimal sequences and the subset $\mathcal{J}$ of sub-optimal sequences that were near-optimal at depth $h-1$:

$$\mathcal{I}_h = \left\{ a \in A^h : V - V(a) \leq 2\frac{\gamma^{h+1}}{1-\gamma} \right\}, \mathcal{J}_h = \left\{ a \in A^h : a_{1:h-1} \in \mathcal{I}_{h-1} \text{ and } a \notin \mathcal{I}_h \right\}$$

By convention, $\mathcal{I}_0 = \{\emptyset\}$. From the definition of κ_2, we have that for any $\kappa' > \kappa_2$, there exists a constant C such that for any $h \geq 1$, $|\mathcal{I}_h| \leq C{\kappa'}^h$ Hence, we also have $|\mathcal{J}_h| \leq K|\mathcal{I}_{h-1}| = O({\kappa'}^h)$.

Algorithm 2: Lazy Open Loop Optimistic Planning

```
Let $M$ be the largest integer such that $M \log M/(2 \log 1/\gamma) \leq n$
Let $L = \log M/(2 \log 1/\gamma)$
Let $\mathcal{T}_0^+ = \mathcal{L}_0^+ = \{\emptyset\}$
for each episode $m = 1, \cdots, M$ do
    Compute $U_a(m-1)$ from (4) for all $a \in \mathcal{T}_{m-1}^+$
    Compute $B_a(m-1)$ from (5) for all $a \in \mathcal{L}_{m-1}^+$
    Sample a sequence with highest B-value: $a \in \arg\max_{a \in \mathcal{L}_{m-1}^+} B_a(m-1)$
    Choose an arbitrary continuation $a^m \in aA^{L-|a|}$        // e.g. uniformly
    Let $\mathcal{T}_m^+ = \mathcal{T}_{m-1}^+$ and $\mathcal{L}_m^+ = \mathcal{L}_{m-1}^+$
    for $t = 1, \cdots, L$ do
        if $a_{1:t}^m \notin \mathcal{T}_m^+$ then
            Add $a_{1:t-1}^m A$ to $\mathcal{T}_m^+$ and $\mathcal{L}_m^+$
            Remove $a_{1:t-1}^m$ from $\mathcal{L}_m^+$
return the most played sequence $a(n) \in \arg\max_{a \in \mathcal{L}_m^+} T_a(M)$
```

Now, for $1 \leq m \leq M$, $a \in A^t$ with $t \leq h$, $h' < h$, we define the set $\mathcal{P}_{h,h'}^a(m)$ of suffixes of a in $\mathcal{J}_h$ that have been played at least a certain number of times:

$$\mathcal{P}_{h,h'}^a(m) = \left\{ b \in aA^{h-t} \cap \mathcal{J}_h : T_b(m) \geq 2f(m)(h+1)^2\gamma^{2(h'-h+1)} + 1 \right\}$$

and the random variable:

$$\tau_{h,h'}^a(m) = \mathbb{1}\{T_a(m-1) < 2f(m)(h+1)^2\gamma^{2(h'-h+1)} + 1 \leq T_a(m)\}$$

Lemma 6 (Regret and sub-optimal pulls). *The following holds true:*

$$r_n \leq \frac{2K\gamma^{H+1}}{1-\gamma} + \frac{3K}{M} \sum_{h=1}^{H} \sum_{a \in \mathcal{J}_h} \frac{\gamma^h}{1-\gamma} T_a(M)$$

The rest of the proof is devoted to the analysis of the term $\mathbb{E} \sum_{a \in \mathcal{J}_h} T_a(M)$. The next lemma describes under which circumstances a suboptimal sequence of actions in $\mathcal{J}_h$ can be selected.

Lemma 7 (Conditions for sub-optimal pull). *Assume that at step $m+1$ we select a sub-optimal sequence a^{m+1}: there exist $0 \leq h \leq L, a \in \mathcal{J}_h$ such that $a^{m+1} \in aA^*$. Then, it implies that one of the following propositions is true:*

$$U_{a^*}(m) < V, \qquad \text{(UCB violation)}$$

or

$$\sum_{t=1}^{h} \gamma^t L_{a_{1:t}}^{\mu}(m) \geq V(a), \qquad \text{(LCB violation)}$$

or

$$\sum_{t=1}^{h} \gamma^t \left(U^\mu_{a_{1:t}}(m) - L^\mu_{a_{1:t}}(m)\right) > \frac{\gamma^{h+1}}{1-\gamma} \tag{Large CI}$$

Proof. As $a^{m+1}_{1:h} = a$ and because the U-values are monotonically increasing along sequences of actions (see Remark 2 and Lemma 1), we have $U_a(m) \geq U_{a^{m+1}}(m)$. Moreover, by Algorithm 1, we have $a^{m+1} = \arg\max_{a \in A^L} U_a(m)$ and $a^* \in A^L$, so $U_{a^{m+1}}(m) \geq U_{a^*}(m)$ and finally $U_a(m) \geq U_{a^*}(m)$.

Assume that (UCB violation) is false, then:

$$\sum_{t=1}^{h} \gamma^t U^\mu_{a_{1:t}}(m) + \frac{\gamma^{h+1}}{1-\gamma} = U_a(m) \geq U_{a^*}(m) \geq V \tag{7}$$

Assume that (LCB violation) is false, then:

$$\sum_{t=1}^{h} \gamma^t L^\mu_{a_{1:t}}(m) < V(a), \tag{8}$$

By taking the difference (7) - (8),

$$\sum_{t=1}^{h} \gamma^t \left(U^\mu_{a_{1:t}}(m) - L^\mu_{a_{1:t}}(m)\right) + \frac{\gamma^{h+1}}{1-\gamma} > V - V(a)$$

But $a \in \mathcal{J}_h$, so $V - V(a) \geq \frac{2\gamma^{h+1}}{1-\gamma}$, which yields (Large CI) and concludes the proof. □

In the following lemma, for each episode m we bound the probability of (UCB violation) or (LCB violation) by a desired confidence level δ_m, whose choice we postpone until the end of this proof. For now, we simply assume that we picked a function f that satisfies $f(m)\log(m)e^{-f(m)} = O(\delta_m)$. We also denote $\mathit{\Delta}_M = \sum_{m=1}^{M} \delta_m$.

Lemma 8 (Boundary crossing probability). *The following holds true, for any $1 \leq h \leq L$ and $m \leq M$,*

$$\mathbb{P}\left((\text{UCB violation}) \text{ } \textit{or} \text{ } (\text{LCB violation}) \text{ } \textit{is true}\right) = O((L+h)\delta_m)$$

Proof. Since $V \leq \sum_{t=1}^{h} \gamma^t \mu(a^*_{1:t}) + \frac{\gamma^{h+1}}{1-\gamma}$, we have,

$$\begin{aligned}
\mathbb{P}\left((\text{UCB violation})\right) &= \mathbb{P}\left(U_{a^*}(m) \leq V\right) \\
&= \mathbb{P}\left(\sum_{t=1}^{L} \gamma^t U^\mu_{a^*_{1:t}}(m) \leq \sum_{t=1}^{L} \gamma^t \mu(a^*_{1:t})\right) \\
&\leq \mathbb{P}\left(\exists 1 \leq t \leq L : U^\mu_{a^*_{1:t}}(m) \leq \mu(a^*_{1:t})\right) \\
&\leq \sum_{t=1}^{L} \mathbb{P}\left(U^\mu_{a^*_{1:t}}(m) \leq \mu(a^*_{1:t})\right)
\end{aligned}$$

In order to bound this quantity, we reduce the question to the application of a deviation inequality. For all $1 \leq t \leq L$, we have on the event $\{U^{\mu}_{a^*_{1:t}}(m) \leq \mu(a^*_{1:t})\}$ that $\hat{\mu}_{a^*_{1:t}}(m) \leq U^{\mu}_{a^*_{1:t}}(m) \leq \mu(a^*_{1:t}) < 1$. Therefore, for all $0 < \delta < 1 - \mu(a^*_{1:t})$, by definition of $U^{\mu}_{a^*_{1:t}}(m)$:

$$d(\hat{\mu}_{a^*_{1:t}}(m), U^{\mu}_{a^*_{1:t}}(m) + \delta) > \frac{f(m)}{T_{a^*_{1:t}}(m)}$$

As d is continuous on $(0,1) \times [0,1]$, we have by letting $\delta \to 0$ that:

$$d(\hat{\mu}_{a^*_{1:t}}(m), U^{\mu}_{a^*_{1:t}}(m)) \geq \frac{f(m)}{T_{a^*_{1:t}}(m)}$$

Since d is non-decreasing on $[\hat{\mu}_{a^*_{1:t}}(m), \mu(a^*_{1:t})]$,

$$d(\hat{\mu}_{a^*_{1:t}}(m), \mu(a^*_{1:t})) \geq d(\hat{\mu}_{a^*_{1:t}}(m), U^{\mu}_{a^*_{1:t}}(m)) \geq \frac{f(m)}{T_{a^*_{1:t}}(m)}$$

We have thus shown the following inclusion:

$$\{U^{\mu}_{a^*_{1:t}}(m) \leq \mu(a^*_{1:t})\} \subseteq \left\{\mu(a^*_{1:t}) > \hat{\mu}_{a^*_{1:t}}(m) \text{ and } d(\hat{\mu}_{a^*_{1:t}}(m), \mu(a^*_{1:t})) \geq \frac{f(m)}{T_{a^*_{1:t}}(m)}\right\}$$

Decomposing according to the values of $T_{a^*_{1:t}}(m)$ yields:

$$\{U^{\mu}_{a^*_{1:t}}(m) \leq \mu(a^*_{1:t})\} \subseteq \bigcup_{n=1}^{m} \left\{\mu(a^*_{1:t}) > \hat{\mu}_{a^*_{1:t},n} \text{ and } d(\hat{\mu}_{a^*_{1:t},n}, \mu(a^*_{1:t})) \geq \frac{f(m)}{n}\right\}$$

We now apply the deviation inequality provided in Lemma 2 of Appendix A in [5]: $\forall \varepsilon > 1$, provided that $0 < \mu(a^*_{1:t}) < 1$,

$$\mathbb{P}\left(\bigcup_{n=1}^{m} \left\{\mu(a^*_{1:t}) > \hat{\mu}_{a^*_{1:t},n} \text{ and } n d_{\mathtt{BER}}(\hat{\mu}_{a^*_{1:t},n}, \mu(a^*_{1:t})) \geq \varepsilon\right\}\right) \leq e\lceil \varepsilon \log m \rceil e^{-\varepsilon}\,.$$

By choosing $\varepsilon = f(m)$, it comes

$$\mathbb{P}\left((\text{UCB violation})\right) \leq \sum_{t=1}^{L} e\lceil f(m) \log m \rceil e^{-f(m)} = O(L\delta_m)$$

The same reasoning gives: $\mathbb{P}\left((\text{LCB violation})\right) = O(h\delta_m)$. □

Lemma 9 (Confidence interval length and number of plays). *Let* $1 \leq h \leq L$, $a \in \mathcal{J}_h$ *and* $0 \leq h' < h$. *Then* (Large CI) *is not satisfied if the following propositions are true:*

$$\forall 0 \leq t \leq h', T_{a_{1:t}}(m) \geq 2f(m)(h+1)^2\gamma^{2(t-h-1)} \tag{9}$$

and

$$T_a(m) \geq 2f(m)(h+1)^2\gamma^{2(h'-h-1)} \tag{10}$$

Proof. We start by providing an explicit upper-bound for the length of the confidence interval $U^\mu_{a_{1:t}} - L^\mu_{a_{1:t}}$. By Pinsker's inequality:

$$d_{\texttt{BER}}(p,q) > d_{\texttt{QUAD}}(p,q)$$

Hence for all $C > 0$,

$$d_{\texttt{BER}}(p,q) \leq C \implies 2(q-p)^2 < C \implies p - \sqrt{C/2} < q < p + \sqrt{C/2}$$

And thus, for all $b \in A^*$, by definition of U^μ and L^μ:

$$U^\mu_b(m) - L^\mu_b(m) \leq \frac{S_b(m)}{T_b(m)} + \sqrt{\frac{f(m)}{2T_b(m)}} - \left(\frac{S_b(m)}{T_b(m)} - \sqrt{\frac{f(m)}{2T_b(m)}}\right) = \sqrt{\frac{2f(m)}{T_b(m)}}$$

Now, assume that (9) and (10) are true. Then, we clearly have:

$$\begin{aligned}\sum_{t=1}^{h}\gamma^t\left(U^\mu_{a_{1:t}}(m) - L^\mu_{a_{1:t}}(m)\right) &\leq \sum_{t=1}^{h'}\gamma^t\sqrt{\frac{2f(m)}{T_{a_{1:t}}(m)}} + \sum_{t=h'+1}^{h}\gamma^t\sqrt{\frac{2f(m)}{T_{a_{1:t}}(m)}} \\ &\leq \frac{1}{(h+1)\gamma^{-h-1}}\sum_{t=1}^{h'}1 + \frac{1}{(h+1)\gamma^{-h-1}}\sum_{t=h'+1}^{h}\gamma^{t-h'} \\ &\leq \frac{\gamma^{h+1}}{h+1}\left(h' + \frac{\gamma}{1-\gamma}\right) \leq \frac{\gamma^{h+1}}{1-\gamma}\,. \qquad \square\end{aligned}$$

Lemma 10. *Let $1 \leq h \leq L, a \in \mathcal{J}_h$ and $0 \leq h' < h$. Then $\tau^a_{h,h'} = 1$ implies that either equation* (UCB violation) *or* (LCB violation) *is satisfied or the following proposition is true:*

$$\exists 1 \leq t \leq h' : |\mathcal{P}^{a_{1:t}}_{h,h'}(m)| < \gamma^{2(t-h')} \tag{11}$$

Lemma 11. *Let $1 \leq h \leq L$ and $0 \leq h' < h$. Then the following holds true,*

$$\mathbb{E}\,|\mathcal{P}^{\emptyset}_{h,h'}(M)| = \widetilde{O}\left(\gamma^{-2h'}\mathbb{1}_{h'>0}\sum_{t=0}^{h'}(\gamma^2\kappa')^t + (\kappa')^h\Delta_M\right).$$

Lemma 12. *Let $1 \leq h \leq L$. The following holds true,*

$$\mathbb{E}\sum_{a\in\mathcal{J}_h}T_a(M) = \widetilde{O}\left(\gamma^{-2h} + (\kappa')^h(1 + M\Delta_M + \Delta_M) + (\kappa'\gamma^{-2})^h\Delta_M\right)$$

Thus by combining Lemma 6 and 12 we obtain:

$$\mathbb{E}\,r_n = \widetilde{O}\left(\gamma^H + \gamma^{-H}M^{-1} + (\kappa'\gamma)^H M^{-1}(1 + M\Delta_M + \Delta_M) + (\kappa')^H\gamma^{-H}M^{-1}\Delta_M\right)$$

Finally,

- if $\kappa'\gamma^2 \leq 1$, we take $H = \lfloor \log M/(2\log 1/\gamma) \rfloor$ to obtain:

  $$\mathbb{E}\, r_n = \widetilde{O}\left(M^{-\frac{1}{2}} + M^{-\frac{1}{2}} + M^{-\frac{1}{2}} M^{\frac{\log \kappa'}{2\log 1/\gamma}} \varDelta_M\right)$$

  For the last term to be of the same order of the others, we need to have $\varDelta_M = O(M^{-\frac{\log \kappa'}{2\log 1/\gamma}})$. Since $\kappa'\gamma^2 \leq 1$, we achieve this by taking $\varDelta_M = O(M^{-1})$.
- if $\kappa'\gamma^2 > 1$, we take $H = \lfloor \log M/\log \kappa' \rfloor$ to obtain:

  $$\mathbb{E}\, r_n = \widetilde{O}\left(M^{\frac{\log \gamma}{\log \kappa'}} + M^{\frac{\log \gamma}{\log \kappa'}}(1 + M\varDelta_M + \varDelta_M) + M^{\frac{\log 1/\gamma}{\log \kappa'}} \varDelta_M\right)$$

  Since $\kappa'\gamma^2 > 1$, the dominant term in this sum is $M^{\frac{\log \gamma}{\log \kappa'}} M\varDelta_M$. Again, taking $\varDelta_M = O(M^{-1})$ yields the claimed bounds.

Thus, the claimed bounds are obtained in both cases as long as we can impose $\varDelta_M = O(M^{-1})$, that is, find a sequence $(\delta_m)_{1\leq m \leq M}$ and a function f verifying:

$$\sum_{m=1}^{M} \delta_m = O(M^{-1}) \quad \text{and} \quad f(m)\log(m)e^{-f(m)} = O(\delta_m) \tag{12}$$

By choosing $\delta_m = M^{-2}$ and $f(m) = 2\log M + 2\log\log M$, the corresponding `KL-OLOP` algorithm does achieve the regret bound claimed in Theorem 3.

6 Experiments

We have performed some numerical experiments to evaluate and compare the following planning algorithms[1]:

- `Random`: returns a random action, we use it as a minimal performance baseline.
- `OPD`: the *Optimistic Planning for Deterministic systems* from [10], used as a baseline of optimal performance. This planner is only suited for deterministic environments, and exploits this property to obtain faster rates. However, it is expected to fail in stochastic environments.
- `OLOP`: as described in section 2.2.[2]
- `KL-OLOP`: as described in section 2.4.[2]
- `KL-OLOP`(1): an aggressive version of `KL-OLOP` where we used $f_1(m) = \log M$ instead of $f_2(m)$. This threshold function makes the upper bounds even tighter, at the cost of an increased probability of violation. Hence, we expect this solution to be more efficient in close-to-deterministic environments. However, since we have no theoretical guarantee concerning its regret as we do with `KL-OLOP`, it might not be conservative enough and converge too early to a suboptimal sequence, especially in highly stochastic environments.

[1] The source code is available at `https://eleurent.github.io/kl-olop/`

[2] Note that we use the lazy version of `OLOP` and `KL-OLOP` presented in Section 4, otherwise the exponential running-time would have been prohibitive.

They are evaluated on the following tasks, using a discount factor of $\gamma = 0.8$:

- A highway driving environment [13]: a vehicle is driving on a road randomly populated with other slower drivers, and must make their way as fast as possible while avoiding collisions by choosing on the the following actions: `change-lane-left`, `change-lane-right`, `no-op`, `faster`, `slower`.
- A gridworld environment [6]: the agent navigates in a randomly-generated gridworld composed of either empty cells, terminal lava cells, and goal cells where a reward of 1 is collected at the first visit.
- A stochastic version of the gridworld environment with noisy rewards, where the noise is modelled as a Bernoulli distribution with a 15% probability of error, i.e. receiving a reward of 1 in an empty cell or 0 in a goal cell.

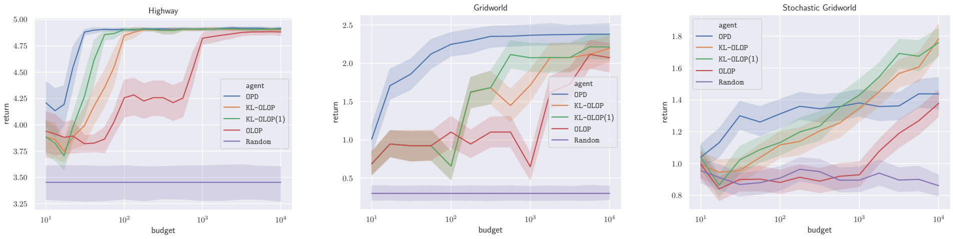

Fig. 3: Numerical experiments: for each environment-agent configuration, we compute the average return over 100 runs — along with its 95% confidence interval — with respect to the available budget n.

The results of our experiments are shown in Figure 3. The `ODP` algorithm converges very quickly to the optimal return in the two first environments, shown in Figure 3a and Figure 3b, because it exploits their deterministic nature: it needs neither to estimate the rewards through upper-confidence bounds nor to sample whole sequences all the way from the root when expanding a leaf, which provides a significant speedup. It can be seen as an oracle allowing to measure the conservativeness of stochastic planning algorithms. And indeed, even before introducing stochasticity, we can see that `OLOP` performs quite badly on the two environments, only managing to solve them with a budget in the order of $10^{3.5}$. In stark contrast, `KL-OLOP` makes a much better use of its samples and reaches the same performance an order of magnitude faster. This is illustrated by the expanded trees shown in Figure 4: `ODP` exploits the deterministic setting and produces a sparse tree densely concentrated around the optimal trajectory. Conversely, the tree developed by `OLOP` is evenly balanced, which suggests that `OLOP` behaves as uniform planning as hypothesised in Section 2.3. `KL-OLOP` is more efficient and expands a highly unbalanced tree, exploring the same regions as `ODP`. Furthermore, in the stochastic gridworld environment shown in Figure 3c, we observe that the deterministic `ODP` planner's performance saturates as

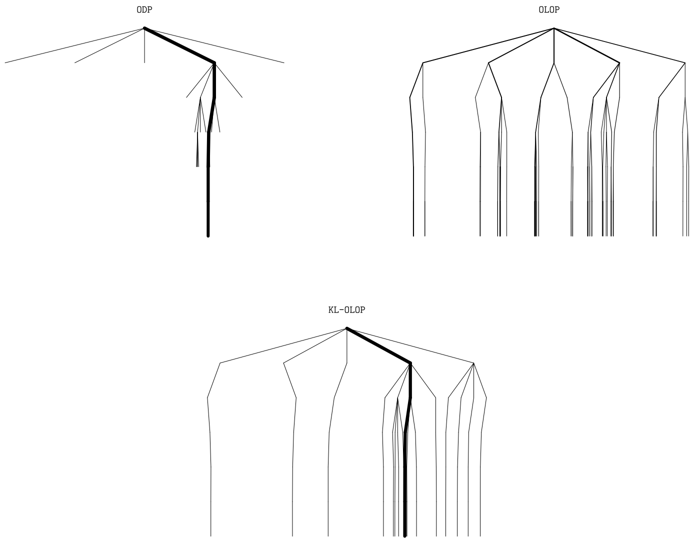

Fig. 4: The look-ahead trees (down to depth 6) expanded by the planning algorithms from the same initial state in the highway environment with the same budget $n = 10^3$. The width of edges represents the nodes visit count $T_a(M)$.

it settles to suboptimal trajectories, as expected. Conversely, the stochastic planners all find better-performing open-loop policies, which justifies the need for this framework. Again, `KL-OLOP` converges an order of magnitude faster than `OLOP`. Finally, `KL-OLOP`(1) enjoys good performance overall and displays the most satisfying trade-off between aggressiveness in deterministic environments and conservativeness in stochastic environments; hence we recommend this tuning for practical use.

7 Conclusion

We introduced an enhanced version of the `OLOP` algorithm for open-loop online planning, whose design was motivated by an investigation of the over-conservative search behaviours of `OLOP`. We analysed its sample complexity and showed that the original regret bounds are preserved, while its empirical performances are increased by an order of magnitude in several numerical experiments. Finally, we proposed an efficient implementation that benefits from a substantial speedup, facilitating its use for real-time planning applications.

Acknowledgments

This work has been supported by CPER Nord-Pas de Calais/FEDER DATA Advanced data science and technologies 2015-2020, the French Ministry of Higher Education and Research, INRIA, and the French Agence Nationale de la Recherche.

References

1. Bellman, R.: Dynamic Programming. Princeton University Press (1957)
2. Bubeck, S., Munos, R.: Open Loop Optimistic Planning. In: Proc. of COLT (2010)
3. Buşoniu, L., Daniels, A., Munos, R., Babuska, R.: Optimistic planning for continuous-action deterministic systems. IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning, ADPRL (2013)
4. Buşoniu, L., Páll, E., Munos, R.: Continuous-action planning for discounted infinite-horizon nonlinear optimal control with Lipschitz values. Automatica **92**(December), 100–108 (2018)
5. Cappé, O., Garivier, A., Maillard, O.A., Munos, R., Stoltz, G.: Kullback-Leibler Upper Confidence Bounds for Optimal Sequential Allocation. The Annals of Statistics **41**(3), 1516–1541 (2013)
6. Chevalier-Boisvert, M., Willems, L., Pal, S.: Minimalistic gridworld environment for openai gym. `https://github.com/maximecb/gym-minigrid` (2018)
7. Coquelin, P.A., Munos, R.: Bandit Algorithms for Tree Search. Proc. of UAI (2007)
8. Coulom, R.: Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search. In: Proc. of International Conference on Computer and Games (2006)
9. Grill, J.B., Valko, M., Munos, R.: Blazing the trails before beating the path: Sample-efficient Monte-Carlo planning. In: Proc. of NeurIPS (2016)
10. Hren, J.F., Munos, R.: Optimistic planning of deterministic systems. Lecture Notes in Computer Science (2008)
11. Kearns, M., Mansour, Y., Ng, A.Y.: A sparse sampling algorithm for near-optimal planning in large Markov decision processes. In: Proc. of IJCAI (2002)
12. Kocsis, L., Szepesvári, C.: Bandit based monte-carlo planning. In: Proc. of ECML (2006)
13. Leurent, E.: An environment for autonomous driving decision-making. `https://github.com/eleurent/highway-env` (2018)
14. Munos, R.: From Bandits to Monte-Carlo Tree Search: The Optimistic Principle Applied to Optimization. Foundations and Trends® in Machine Learning (2014)
15. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., Hassabis, D.: Mastering the game of go with deep neural networks and tree search. Nature **529**, 484–503 (2016)
16. Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., Hassabis, D.: A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. Science **362**(6419), 1140–1144 (2018)
17. Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al.: Mastering the game of go without human knowledge. Nature **550**(7676), 354 (2017)
18. Szorenyi, B., Kedenburg, G., Munos, R.: Optimistic planning in Markov decision processes using a generative model. In: Proc. of NeurIPS (2014)