Blazing the trails before beating the path: Sample-efficient Monte-Carlo planning

Jean-Bastien Grill **Michal Valko**
SequeL team, INRIA Lille - Nord Europe, France
jean-bastien.grill@inria.fr michal.valko@inria.fr

Rémi Munos
Google DeepMind, UK*
munos@google.com

Abstract

You are a robot and you live in a Markov decision process (MDP) with a finite or an infinite number of transitions from state-action to next states. You got brains and so you *plan* before you act. Luckily, your roboparents equipped you with a generative model to do some *Monte-Carlo planning*. The world is waiting for you and you have no time to waste. You want your planning to be efficient. *Sample-efficient.* Indeed, you want to exploit the possible structure of the MDP by exploring only a subset of states reachable by following near-optimal policies. You want guarantees on sample complexity that depend on a measure of the quantity of near-optimal states. You want something, that is an extension of Monte-Carlo sampling (for estimating an expectation) to problems that alternate maximization (over actions) and expectation (over next states). But you do not want to `StOP` with exponential running time, you want something simple to implement and computationally efficient. You want it all and you want it now. You want `TrailBlazer`.

1 Introduction

We consider the problem of sampling-based planning in a Markov decision process (MDP) when a *generative model* (oracle) is available. This approach, also called Monte-Carlo planning or Monte-Carlo tree search (see e.g., [12]), has been popularized in the game of computer Go [7, 8, 15] and shown impressive performance in many other high dimensional control and game problems [4]. In the present paper, we provide a sample complexity analysis of a new algorithm called `TrailBlazer`.

Our assumption about the MDP is that we possess a generative model which can be called from any state-action pair to generate rewards and transition samples. Since making a call to this generative model has a cost, be it a numerical cost expressed in CPU time (in simulated environments) or a financial cost (in real domains), our goal is to *use* this *model* as *parsimoniously* as possible.

Following *dynamic programming* [2], planning can be reduced to an approximation of the (optimal) value function, defined as the maximum of the expected sum of discounted rewards: $\mathbb{E}\left[\sum_{t\geq 0} \quad {}^t r_t\right]$; where $\quad \in [0; 1)$ is a known *discount factor*. Indeed, if an $''$-optimal approximation of the value function at any state-action pair is available, then the policy corresponding to selecting in each state the action with the highest approximated value will be $\mathcal{O}\left(''=(1-\quad)\right)$-optimal [3].

Consequently, in this paper, we focus on a near-optimal approximation of the value function *for a single given state* (or state-action pair). In order to assess the performance of our algorithm we measure its *sample complexity* defined as the number of oracle calls, given that we guarantee its *consistency*, i.e., that with probability at least $1-\quad$, `TrailBlazer` returns an $''$-approximation of the value function as required by the probably approximately correct (PAC) framework.

*on the leave from SequeL team, INRIA Lille - Nord Europe, France

29th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

We use a *tree representation* to represent the set of states that are reachable from any initial state. This tree alternates maximum (MAX) nodes (corresponding to actions) and average (AVG) nodes (corresponding to the random transition to next states). We assume the number K of actions is finite. However, the number N of possible next states is either *finite* or *infinite* (which may be the case when the state space is infinite), and we will report results in both the finite N and the infinite case. The root node of this planning tree represents the current state (or a state-action) of the MDP and its value is the maximum (over all policies defined at MAX nodes) of the corresponding expected sum of discounted rewards. Notice that by using a tree representation, we do not use the property that some state of the MDP can be reached by different paths (sequences of states-actions). Therefore, this state will be represented by different nodes in the tree. We could potentially merge such duplicates to form a graph instead. However, for simplicity, we choose not to merge these duplicates and keep a tree, which could make the planning problem harder. To sum up, our goal is to return, with probability $1 -$, an ε-accurate value of the root node of this planning tree while using as low number of calls to the oracle as possible. Our contribution is an algorithm called TrailBlazer whose sampling strategy depends on the specific structure of the MDP and for which we provide *sample complexity* bounds in terms of a new *problem-dependent measure of the quantity of near-optimal nodes*. Before describing our contribution in more detail we first relate our setting to what has been around.

1.1 Related work

Kocsis and Szepesvári [12] introduced the UCT algorithm (upper-confidence bounds for trees). UCT is efficient in computer Go [7, 8, 15] and a number of other control and game problems [4]. UCT is based on generating trajectories by selecting in each MAX node the action that has the highest upper-confidence bound, computed according to the UCB algorithm of Auer et al. [1]. UCT converges asymptotically to the optimal solution, but its sample complexity can be worse than doubly-exponential in[2] $(1/\varepsilon)$ for some MDPs [13]. One reason for this is that the algorithm can expand very deeply the apparently best branches but may lack sufficient exploration, especially when a narrow optimal path is hidden in a suboptimal branch. As a result, this approach works well in some problems with a specific structure but may be much worse than a uniform sampling in other problems.

On the other hand, a uniform planning approach is safe for all problems. Kearns et al. [11] generate a sparse look-ahead tree based on expanding all MAX nodes and sampling a finite number of children from AVG nodes up to a fixed depth that depends on the desired accuracy ε. Their sample complexity is[3] of the order of $(1/\varepsilon)^{\log(1/\varepsilon)}$, which is *non-polynomial* in $1/\varepsilon$. This bound is better than that for UCT in a worst-case sense. However, as their look-ahead tree is built in a *uniform* and *non-adaptive* way, this algorithm fails to benefit from a potentially favorable structure of the MDP.

An improved version of this sparse-sampling algorithm by Walsh et al. [17] cuts suboptimal branches in an adaptive way but unfortunately does not come with an improved bound and stays non-polynomial even in the simple Monte Carlo setting for which $K = 1$.

Although the sample complexity is certainly non-polynomial in the worst case, it *can be polynomial* in some specific problems. First, for the case of finite N, the sample complexity is polynomial and Szörényi et al. [16] show that a uniform sampling algorithm has complexity at most $(1/\varepsilon)^{2+\log(KN)/(\log(1/\gamma))}$. Notice that the product KN represents the *branching factor* of the look-ahead planning tree. This bound could be improved for problems with specific reward structure or transition smoothness. In order to do this, we need to design non-uniform, *adaptive* algorithm that captures the possible structure of the MDP when available, while making sure that in the worst case, we do not perform worse than a uniform sampling algorithm.

The case of deterministic dynamics ($N = 1$) and rewards considered by Hren and Munos [10] has a complexity of order $(1/\varepsilon)^{(\log \kappa)/(\log(1/\gamma))}$, where $\in [1; K]$ is the branching factor of the subset of near-optimal nodes.[4] The case of stochastic rewards has been considered by Bubeck and Munos [5] but with the difference that the goal was not to approximate the optimal value function but the value of the best *open-loop* policy which consists in a sequence of actions independent of states. Their sample complexity is $(1/\varepsilon)^{\max(2,(\log \kappa)/(\log 1/\gamma))}$.

[2] all guarantees in the form of $1/\varepsilon^c$ are up to a poly-logarithmic multiplicative factor

[3] neglecting exponential dependence in γ

[4] nodes that need to be considered in order to return a near-optimal approximation of the value at the root

In the case of general MDPs, Buşoniu and Munos [6] consider the case of a fully known model of the MDP. For any state-action, the model returns the expected reward and the set of all next states (assuming N is finite) with their corresponding transition probabilities. In that case, the complexity is $(1/\varepsilon)^{\log \kappa/(\log(1/\gamma))}$, where $\in [0; KN]$ can again be interpreted as a branching factor of the subset of near-optimal nodes. These approaches use the *optimism in the face of uncertainty* principle whose applications to planning have been studied by Munos [13]. `TrailBlazer` is different. It is *not optimistic* by design: To avoid voracious demand for samples it does not balance the upper-confidence bounds of all possible actions. This is crucial for polynomial sample complexity in the infinite case. The whole Section 3 shines many rays of intuitive light on this single and powerful idea.

The work that is most related to ours is `StOP` by Szörényi et al. [16] which considers the planning problem in MDPs with a generative model. Their complexity bound is of the order of $(1/\varepsilon)^{2+\log \kappa/(\log(1/\gamma))+o(1)}$, where $\in [0; KN]$ is a problem-dependent quantity. However, their defined as $\lim_{\varepsilon\to 0} \max(\ _1; \ _2)$ (in their Theorem 2) is somehow difficult to interpret as a measure of the quantity of *near-optimal nodes*. Moreover, `StOP` is not computationally efficient as it requires to identify the *optimistic policy* which requires computing an upper bound on the value of *any* possible policy, whose number is exponential in the number of `MAX` nodes, which itself is exponential in the planning horizon. Although they suggest (in their Appendix F) a computational improvement, this version is not analyzed. Finally, unlike in the present paper, `StOP` does not consider the case $N = \infty$.

1.2 Our contributions

Our main result is `TrailBlazer`, an algorithm with a bound on the number of samples required to return a high-probability ε-approximation of the root node whether the number of next states N is finite or infinite. The bounds use a problem-dependent quantity (or d) that measures the quantity of near-optimal nodes. We now summarize the results.

Finite number of next states ($N < \infty$): The sample complexity of `TrailBlazer` is of the order of[5] $(1/\varepsilon)^{\max(2,\log(N\kappa)/\log(1/\gamma)+o(1))}$; where $\in [1; K]$ is related to the branching factor of the set of near-optimal nodes (precisely defined later).

Infinite number of next states ($N = \infty$): The complexity of `TrailBlazer` is $(1/\varepsilon)^{2+d}$; where d is a measure of the difficulty to identify the near-optimal nodes. Notice that d can be finite even if the planning problem is very challenging.[6] We also state our contributions in specific settings in comparison to previous work.

- For the case $N < \infty$, we improve over the best-known previous worst-case bound with an exponent of $1/\varepsilon$) to $\max(2; \log(NK)/\log(1/\ ))$ instead of $2 + \log(NK)/\log(1/\ )$ reported by Szörényi et al. [16].
- For the case $N = \infty$, we identify properties of the MDP (when $d = 0$) under which the sample complexity is of order of $1/\varepsilon^2$. This is the case when there are non-vanishing action-gaps[7] from any state along near-optimal policies or when the probability of transitioning to nodes with gap Δ is upper bounded by Δ^2. This complexity bound is as good as Monte-Carlo sampling and for this reason **`TrailBlazer` is a natural extension of Monte-Carlo sampling** (where all nodes are `AVG`) **to stochastic control problems** (where `MAX` and `AVG` nodes alternate). Also, no previous algorithm reported a polynomial bound when $N = \infty$.
- In MDPs with deterministic transitions ($N = 1$) but stochastic rewards our bound is $(1/\varepsilon)^{\max(2,\log \kappa/(\log 1/\gamma))}$ which is similar to the bound achieved by Bubeck and Munos [5] in a similar setting (open-loop policies).
- In the evaluation case without control ($K = 1$) `TrailBlazer` behaves exactly as Monte-Carlo sampling (thus achieves a complexity of $1/\varepsilon^2$), even in the case $N = \infty$.
- Finally, `TrailBlazer` is easy to implement and is numerically efficient.

[5] neglecting logarithmic terms in ε and δ

[6] since when $N = \infty$ the actual branching factor of the set of reachable nodes is infinite

[7] defined as the difference in values of best and second-best actions

2 Monte-Carlo planning with a generative model

Setup We operate on a planning tree T: Each node of T from the root down is alternatively either an average (AVG) or a maximum (MAX) node. For any node s, $C[s]$ is the set of its children. We consider trees T for which the cardinality of $C[s]$ for any MAX node s is bounded by K. The cardinality N of $C[s]$ for any AVG node s can be either finite, $N < 1$, or infinite. We consider both cases. TrailBlazer applies to both situations. We provide performance guarantees for a general case and possibly tighter, N-dependent guarantees in the case of $N < 1$. We assume that we have a generative model of the transitions and rewards: Each AVG node s is associated with a transition, a random variable $_s$ 2 $C[s]$ and a reward, a random variable r_s 2 $[0; 1]$.

Input: , "
Set: 1= max(2 ;log(1 ="))
Set: $2\log(\text{"}(1 \quad))^2 \frac{\log \frac{\log(\ K\ )}{(1 \quad)}}{\log(\ =\ )}$
Set: m $(\log(1= \ ) + \ )=((1 \quad)^{2\text{"}2})$
Use: and as global parameters
Output:
call the root with parameters (m; "= 2)

Figure 1: TrailBlazer

Objective For any node s, we define the value function $V[s]$ as the optimum over policies (giving a successor to all MAX nodes) of the sum of discounted expected rewards playing policy ,

$$V[s] = \sup E \left[\sum_{t\ 0}^{\text{"}} {}^t r_{s_t} \ s_0 = s; \quad \# \right];$$

where 2 (0; 1) is the discount factor. If s is an AVG node, V satisfies the following Bellman equation,

$$V[s] = E[r_s] + \sum_{s^0 2 C[s]} p(s^0 j s) V[s^0]:$$

If s is a MAX node, then $V[s] = \max_{s^0 2 C[s]} V[s^0]$:

The planner has access to the oracle which can be called for any AVG node s to either get a reward r or a transition which are two independent random variables identically distributed as r_s and $_s$ respectively.

With the notation above, **our goal is to estimate the value $V[s_0]$ of the root node s_0** using the **smallest possible number of oracle calls**. More precisely, given any and ", we want to output a value "; such that $P[j \ \text{"}; \quad V \ [s_0]j > \text{"}\ ]$ using the smallest possible number of oracle calls $n_{\text{"}}$; . The number of calls is the sample complexity of the algorithm.

Input: m, "
Initialization: {Only executed on first call}
SampledNodes ; ,
r 0
Run:
if " 1=(1) **then**
  Output: 0
end if
if jSampledNodesj > m **then**
  ActiveNodes SampledNodes(1 : m)
else
  while jSampledNodesj < m **do**
    {new sample of next state}
    SampledNodes.append( )
    r r +[new sample of reward]
  end while
  ActiveNodes SampledNodes
end if {At this point, jActiveNodesj = m}
for all unique nodes s 2 ActiveNodes **do**
  k #occurrences of s in ActiveNodes
    call s with parameters (k; "=)
    + k=m
end for
Output: + r=jSampledNodesj

Figure 2: AVG node

2.1 Blazing the trails with TrailBlazer

To fulfill the above objective, our TrailBlazer constructs a planning tree T which is, at any time, a finite subset of the potentially infinite tree. Only the already visited nodes are in T and explicitly represented in memory. Taking the object-oriented paradigm, each node of T is a persistent object with its own memory which can receive and perform calls respectively from and to other nodes. A node can potentially be called several times (with different parameters) during the run of TrailBlazer and may reuse (some of) its stored (transition and reward) samples. In particular, after node s receives a call from its parent, node s may perform internal computation by calling its own children in order to return a real value to its parent.

Pseudocode of TrailBlazer is in Figure 1 along with the subroutines for MAX nodes in Figure 3 and AVG nodes in Figure 2. A node (MAX or AVG) is called with two parameters m and ", which represent some requested properties of the returned value: m controls the desired variance and " the desired maximum bias. We now describe the MAX and AVG node subroutines.

Definition 3. *We define $d \geq 0$ as the smallest d such that for all there exists $a > 0$ for which for all even $h > 0$,*

$$\mathbb{E}\left[K^{h/2}\mathbb{1}\left(S^h \in \mathcal{N}_h\right) \prod_{\substack{0 \leq h' < h \\ h' \equiv 0 (\mathrm{mod}\ 2)}} \left(\frac{}{h-h'}\right)^{\mathbb{1}\left(\Delta(S^h_{h'}) < 16 \frac{(h-h')=2}{(1-\ )}\right)}\right] \leq a^{\ -dh}$$

If no such d exists, we set $d = \infty$.

This definition of d takes into account the size of the near-optimality set (just like) but unlike it also takes into account the difficulty to identify the near-optimal paths.

Intuitively, the expected number of oracle calls performed by a given AVG node s is proportional to: $(1=\varepsilon^2) \times$ (the product of the inverted squared gaps of the set of MAX nodes in the path from the root to s) $\times$ (the probability of reaching s by following a policy which always tries to reach s).

Therefore, a near-optimal path with a larger number of small MAX node gaps can be considered *difficult*. By assigning a larger weight to difficult nodes, we are able to give a better characterization of the actual complexity of the problem and provide polynomial guarantees on the sample complexity for $N = \infty$ when d is finite.

Theorem 3. *If d is finite then there exists $C > 0$ such that for all $\varepsilon > 0$ and > 0, the expected sample complexity of* TrailBlazer *satisfies*

$$\mathbb{E}\left[n(\varepsilon;\ )\right] \leq C \frac{\left(\log(1=\ ) + \log(1=\varepsilon)\right)^3}{\varepsilon^{2+d}}.$$

Note that this result holds in expectation only, contrary to Theorem 2 which holds in high probability.

We now give an example for which $d = 0$, followed by a special case of it.

Lemma 1. *If there exists $c > 0$ and $b > 2$ such that for any near-optimal AVG node s,*

$$\mathbb{P}\left[\Delta(\ _s) \leq x\right] \leq c x^b;$$

where the random variable $_s$ is a successor state from s drawn from the MDP's transition probabilities, then $d = 0$ and consequently the sample complexity is of order $1=\varepsilon^2$.

Remark 2. *If there exists $\Delta_{\min}$ such that for any near-optimal MAX node s, $\Delta(s) \geq \Delta_{\min}$ then $d = 0$ and the sample complexity is of order $1=\varepsilon^2$. Indeed, in this case as $\mathbb{P}\left[\Delta_s \leq x\right] \leq \left(x=\Delta_{\min}\right)^b$ for any $b > 2$ for which $d = 0$ by Lemma 1.*

5 Conclusion

We provide a new Monte-Carlo planning algorithm TrailBlazer that works for MDPs where the number of next states N can be either finite or infinite. TrailBlazer is easy to implement and is numerically efficient. It comes packaged with a PAC consistency and two problem-dependent sample-complexity guarantees expressed in terms of a measure (defined by) of the quantity of near-optimal nodes or a measure (defined by d) of the difficulty to identify the near-optimal paths. The sample complexity of TrailBlazer improves over previous worst-case guarantees. What's more, TrailBlazer exploits MDPs with specific structure by exploring only a fraction of the whole search space when either or d is small. In particular, we showed that if the set of near-optimal nodes have non-vanishing action-gaps, then the sample complexity is $\widetilde{\mathcal{O}}(1=\varepsilon^2)$, which is the same rate as Monte-Carlo sampling. This is a pretty decent evidence that TrailBlazer is a natural extension of Monte-Carlo sampling to stochastic control problems.

Acknowledgements The research presented in this paper was supported by French Ministry of Higher Education and Research, Nord-Pas-de-Calais Regional Council, a doctoral grant of École Normale Supérieure in Paris, Inria and Carnegie Mellon University associated-team project EduBand, and French National Research Agency projects ExTra-Learn (n.ANR-14-CE24-0010-01) and BoB (n.ANR-16-CE23-0003)

References

[1] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002.

[2] Richard Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

[3] Dimitri Bertsekas and John Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA, 1996.

[4] Cameron B. Browne, Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, 2012.

[5] Sébastien Bubeck and Rémi Munos. Open-loop optimistic planning. In *Conference on Learning Theory*, 2010.

[6] Lucian Buşoniu and Rémi Munos. Optimistic planning for Markov decision processes. In *International Conference on Artificial Intelligence and Statistics*, 2012.

[7] Rémi Coulom. Efficient selectivity and backup operators in Monte-Carlo tree search. *Computers and games*, 4630:72–83, 2007.

[8] Sylvain Gelly, Wang Yizao, Rémi Munos, and Olivier Teytaud. Modification of UCT with patterns in Monte-Carlo Go. Technical report, Inria, 2006. URL https://hal.inria.fr/inria-00117266.

[9] Arthur Guez, David Silver, and Peter Dayan. Efficient Bayes-adaptive reinforcement learning using sample-based search. *Neural Information Processing Systems*, 2012.

[10] Jean-Francois Hren and Rémi Munos. Optimistic Planning of Deterministic Systems. In *European Workshop on Reinforcement Learning*, 2008.

[11] Michael Kearns, Yishay Mansour, and Andrew Y. Ng. A sparse sampling algorithm for near-optimal planning in large Markov decision processes. In *International Conference on Artificial Intelligence and Statistics*, 1999.

[12] Levente Kocsis and Csaba Szepesvári. Bandit-based Monte-Carlo planning. In *European Conference on Machine Learning*, 2006.

[13] Rémi Munos. From bandits to Monte-Carlo tree search: The optimistic principle applied to optimization and planning. *Foundations and Trends in Machine Learning*, 7(1):1–130, 2014.

[14] David Silver and Joel Veness. Monte-Carlo planning in large POMDPs. In *Neural Information Processing Systems*, 2010.

[15] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[16] Balázs Szörényi, Gunnar Kedenburg, and Rémi Munos. Optimistic planning in Markov decision processes using a generative model. In *Neural Information Processing Systems*, 2014.

[17] Thomas J Walsh, Sergiu Goschin, and Michael L Littman. Integrating sample-based planning and model-based reinforcement learning. *AAAI Conference on Artificial Intelligence*, 2010.

A Consistency

First, we comment on the behavior of the nodes when called multiple times.

Remark 3. *Notice that the output of the call to a node* s *with parameters* $(m; ")$ *and global* *does not depend on when this call was performed. That means if this call is performed again later in the execution with the same parameters* $(m; ")$ *and* *, the output would be the same. This is because, first, the* MAX *nodes does not store anything so that their behavior only depend on* $s; m; "$*; and* *. Second, the* AVG *nodes store the already sampled rewards, transitions and the order they were sampled but when called with parameters* $(m; ")$ *and* *, it returns the average of the* m *first sampled children only.*

From now we consider any ; ; and ; such that $0 <$ $<$ < 1 and $0 <$ < 1=2.

We define a random variable $_s(m; ")$ to be the output of the node s called with parameters $(m; ")$ and global parameters and . Notice that following Remark 3, $_s(m; ")$ does depend on $s; m; "$, , and all the transitions and rewards of itself and its descendants but does not depend on the current state of the algorithm.

We define the function $\mathsf{U}(m; ")$ such that the value $\mathsf{U}(m; ") + "$ is the width of a high-probability confidence interval for $_s(m; ")$ which is used in MAX node,

$$\mathsf{U}(m; ") \stackrel{\text{def}}{=} \frac{2}{1-} \sqrt{\frac{\log(K\,`=(\ ")) + \ =(\ -\ ) + 1}{m}}.$$

We define $`_{\max}(") \stackrel{\text{def}}{=} \max\{` : \mathsf{U}(`; ") \geq (1 - \ )"\}$. By definition of TrailBlazer, this is the maximum value that the children of a MAX node would be ever called with. We also define $\bar{`}(m; ") \stackrel{\text{def}}{=} \max(m; `_{\max}(")))$

We now construct event $\mathcal{B}_s$ for which Lemma 2 proves that it holds with high probability and that on $\mathcal{B}_s$, TrailBlazer works well.

Definition 4. *For any node* s, $" > 0$*; and* $m \in \mathbb{N}^+$*;*

- *If* $" \geq 1$=$(1 - \ )$*;*

$$\mathcal{B}_s(m; ") \stackrel{\text{def}}{=} \Omega:$$

- *Else, if* s *is a* MAX *node,*

$$\mathcal{B}_s(m; ") \stackrel{\text{def}}{=} \cap_{i \in \mathcal{C}[s]} \left(\cap_{\ell \leq \ell_{\max}(\varepsilon)} C_{i,\ell}(") \cap \mathcal{B}_{i,\ell_{\max}}\right)(")$$

$$\text{with} \begin{cases} C_{i,\ell}(") \stackrel{\text{def}}{=} \left\{ |\ _i\left(`; \frac{\eta}{1-\eta}\mathsf{U}(`; ")\right) - \mathcal{V}[i]| \leq \left(1 + \frac{\eta}{1-\eta}\right)\mathsf{U}(`; ")\right\} \\ \mathcal{B}_{i,\ell}(") \stackrel{\text{def}}{=} \mathcal{B}_i\left(`; \frac{\eta}{1-\eta}\mathsf{U}(`; ")\right): \end{cases}$$

- *Else, if* s *is an* AVG *node,*

$$\mathcal{B}_s(m; ") \stackrel{\text{def}}{=} \cap_{k \in \{1,..,m\}} \mathcal{B}_{\tau_{s;k}}(\ _s(k; m); "= \ );$$

with $_s(k; m) \stackrel{\text{def}}{=} |\{j : \ _{s,k} = \ _{s,j} \text{ and } j \leq m\}|$.

Intuitively, C is the event where the confidence bounds are satisfied. Using events C, Definition 4 recursively constructs the events $\mathcal{B}$ needed for Lemma 2.

We remark that there is no loop in this recursive definition as the definition of $\mathcal{B}_s$ only uses $\{\mathcal{B}_i\}_i$ where i is a child of s. Moreover, this way we define $\mathcal{B}_s$ in a finite number of steps as going down through MAX and AVG nodes increases $"$ to $" =$ and reaches 1=$(1 - \ )$ in a finite number of steps.

Moreover, note that $\mathcal{B}_s$ is *not* an intersection of a finite number of $C_{i,\ell}$ because $_{s,k}$ is a random variable and therefore in the AVG case, $\mathcal{B}_s$ is not just the intersection of m events.

Remark 4. *Notice that, by induction, for any* > 0*;* > 0*;* $"_1; "_2; m_1; m_2$*, and any node* s*,*

$$"_1 \leq "_2 \text{ and } m_1 \geq m_2 \implies \mathcal{B}_s(m_1; "_1) \subseteq \mathcal{B}_s(m_2; "_2):$$

Lemma 2. *For any $k \in \mathbb{N}^+$ and any node s, integers $m' \geq m$; and $" > 0$; such that $< \left(\bar{\ }(m;\ ")\right)^{-h_s(\varepsilon)}$ and*

- *either k is odd, s is an AVG node, and $" > \frac{(\gamma/\eta)^{(k-1)=2}}{1-\gamma}$,*
- *or k is even, s is a MAX node, and $" > \frac{(\gamma/\eta)^{(k=2)-1}}{\eta(1-\gamma)}$,*

then the two following properties are verified

Property A: $\mathbb{P}\left[\mathcal{B}_s(m;")\right] \geq 1 - \dfrac{\left(\bar{\ }(m;")\ K=\ ^2\right)^{h_s(\varepsilon)} - 1}{K - 1}$

Property B: $\forall\ ;\mathbb{E}\left[e^{\lambda(\mu_s(m,\varepsilon)-\mathcal{V}[s])}\middle|\mathcal{B}_s(m';")\right] \leq \exp\left(2\ \left(\dfrac{\bar{\ }(m;")}{2}\right)^{h_s(\varepsilon)} \times \dfrac{1-"\ (s)}{=\ -1} + "|\ \ | + \dfrac{2}{2(1-\ )^2 m}\right);$

where $(s) = \begin{cases} 1- & \text{if } s \text{ is a MAX node} \\ (1-\ )= & \text{if } s \text{ is an AVG node} \end{cases}$ *and* $h_s(") = \max\left(0; \frac{\log(\theta(s)\gamma\varepsilon)}{\log(\gamma/\eta)}\right)$

Lemma 2 claims that on a high-probability event $\mathcal{B}$, the sub-variance of $_s$ is of order at most $1{=}m$ and its bias of order at most $"$ up to the term independent of . We prove it by induction on k.

Proof. 1° Let $k = 1$. Then, k is odd and s is an AVG node. Since $" > 1{=}(1-\ )$, s outputs zero by the definition. For any $m' \geq m$ we have $\mathcal{B}_s(m;") = \mathcal{B}_s(m';") = \Omega$, since $" > 1{=}(1-\ )$.

Property A: By definition $h_s(") \geq 0$, therefore $\frac{\left(\bar{\ell}(m,\varepsilon)K/\eta^2\right)^{h_s(")}-1}{K-1} \geq 0$. Then,

$$\mathbb{P}\left[\Omega\right] = 1 \geq 1 - \frac{\left(\bar{\ }(m;")\ K=\ ^2\right)^{h_s(\varepsilon)} - 1}{K - 1}.$$

Property B: First, as the rewards are bounded from above by one, we have $\mathcal{V}\left[s\right] \leq \sum_h\ ^h \leq 1{=}(1-\ )$. Second, $\mathcal{V}\left[s\right] \geq 0$ as the rewards are bounded from below by zero. Finally, $"\ (s) \leq 1$, hence

$$2\ \left(\frac{\bar{\ }(m;")}{2}\right)^{h_s(\varepsilon)} \times \frac{1-"\ (s)}{=\ -1} \geq 0:$$

We combine these bounds to get

$$\forall\ ;\mathbb{E}\left[e^{\lambda(\mu_s(m,\varepsilon)-\mathcal{V}[s])}\middle|\mathcal{B}_s(m;")\right] = \mathbb{E}\left[e^{\lambda(\mu_s(m,\varepsilon)-\mathcal{V}[s])}\right] = \mathbb{E}\left[e^{\lambda(0-\mathcal{V}[s])}\right] \leq \exp\left(\frac{|\ |}{1-}\right)$$
$$\leq \exp\left("|\ \ |\right) \leq \exp\left(2\ \left(\frac{\bar{\ }(m;")}{2}\right)^{h_s(\varepsilon)} \times \frac{1-"\ (s)}{=\ -1} + "|\ \ | + \frac{2}{2(1-\ )^2 m}\right).$$

2° Let $k \in \mathbb{N}^+$. Assume that Lemma 2 is true for any $k' \leq k-1$ and let us prove it for k. We prove it separately for k odd and even.

k is even: Let s be a MAX node, m any integer and $"$ such that

$$" > \frac{(\ =\ )^{(k/2)-1}}{(1-\ )} \quad \text{and} \quad \frac{1}{\bar{\ }(m;")} > \ :$$

We deduce that

$$" > \frac{(\ =\ )^{(k/2)-1}}{(1-\ )} = \frac{(\ =\ )^{(k/2)-1}}{1-}.$$

Also, any child of s is an AVG node as nodes alternate between AVG and MAX. We can thus apply the induction assumption to any child of s with parameters $(m';"')$ with $"' <\ "$ and $m' \leq \bar{\ }(m;")$).

For any child $i \in \mathcal{C}[s]$ of s and any integer $\ell > 0$, we remind the reader the definition of $\mathcal{B}_{i,\ell}$,

$$\mathcal{B}_{i,\ell} = \mathcal{B}_i\left(\ell; \tfrac{\eta}{1-\eta}\mathsf{U}(\ell; \varepsilon)\right):$$

If $\mathsf{U}(\ell; \varepsilon) \geq (1 - \quad)\varepsilon$, then $\frac{\eta}{1-\eta}\mathsf{U}(\ell; \varepsilon) \geq \quad \varepsilon$ and $\ell \leq \ell_{\max}(\varepsilon)$.

We can thus apply our induction assumption to get that for any $\ell \leq \ell_{\max}$,

$$\text{A}\ \mathbb{P}[\mathcal{B}_{i,\ell}] \geq 1 - \frac{\left(\tau\left(\ell; \frac{\eta}{1-\eta}\mathsf{U}(\ell; \varepsilon)\right) K = \quad 2\right)^{h_i\left(\frac{\eta}{1-\eta}\mathsf{U}(\ell,\varepsilon)\right)} - 1}{K - 1} \tag{3}$$

$$\text{B}\ \forall \quad ; \mathbb{E}\left[e^{\lambda\left(\mu_i\left(\ell, \frac{}{1-}\mathsf{U}(\ell,\varepsilon)\right) - \mathcal{V}[i]\right)} \middle| \mathcal{B}_{i,\ell}\right] \tag{4}$$

$$\leq \exp\left(2 \quad \left(\frac{\tau\left(\ell; \frac{\eta}{1-\eta}\mathsf{U}(\ell; \varepsilon)\right)}{2}\right)^{h_i\left(\frac{\eta}{1-\eta}\mathsf{U}(\ell,\varepsilon)\right)} \times \frac{1 - \frac{\eta}{1-\eta}\mathsf{U}(\ell; \varepsilon) \quad (i)}{= \quad - 1} + \frac{\eta}{1-\eta}\mathsf{U}(\ell; \varepsilon)| \quad | + \frac{2}{2(1 - \quad)^2 \ell}\right)$$

We first prove Property A.

Combining Inequality 4 with Lemma 6, we get for all $i \in \mathcal{C}[s]$ and $\ell \leq \ell_{\max}$,

$$\begin{aligned}
\mathbb{P}[\mathcal{C}_{i,\ell}{}^{\mathsf{c}} | \mathcal{B}_{i,\ell}] &\leq 2\exp\left(2 \quad \ell_{\max}(\quad \varepsilon)^{h_i(\eta\varepsilon)} \times \frac{1 - \varepsilon \quad (i)}{= \quad - 1}\right) \exp\left(-\frac{\left(\mathsf{U}(\ell; \varepsilon)\left(1 + \frac{\eta}{1-\eta}\right) - \frac{\eta}{1-\eta}\mathsf{U}(\ell; \varepsilon)\right)^2}{\frac{2}{(1-\gamma)^2 \ell}}\right) \\
&\leq 2\exp\left(2\frac{1 - \varepsilon \quad (i)}{= \quad - 1}\right) \exp\left(-\frac{(1 - \quad)^2 \ell}{2}\mathsf{U}(\ell; \varepsilon)^2\right) \\
\text{as } (\quad < 1 = \ell) &\leq 2\exp\left(\frac{2}{-}\right) \exp\left(-\frac{(1 - \quad)^2 \ell}{2}\frac{1}{(1 - \quad)^2}\frac{4\log\left(K \ell = (\quad \varepsilon(1 - \quad))\right) + 4 \quad = (\quad - \quad) + 4}{\ell}\right) \\
&\leq \frac{1}{2}\left(\frac{\varepsilon(1 - \quad)}{K \ell}\right)^2 \leq \frac{1}{2}\frac{\varepsilon(1 - \quad)}{K \ell^2}.
\end{aligned}$$

Using a union bound over all $\ell \leq \ell_{\max}$ we get

$$\mathbb{P}[\cup_{\ell \leq \ell_{\max}} \mathcal{C}_{i,\ell}{}^{\mathsf{c}} | \mathcal{B}_{i,\ell}] \leq \sum_{l} \frac{1}{2}\frac{\varepsilon(1 - \quad)}{K \ell^2} \leq \frac{2}{12}\frac{\varepsilon(1 - \quad)}{K} \leq \frac{\varepsilon(1 - \quad)}{K}.$$

Now, we can finally bound the event for node s as

$$\begin{aligned}
\mathbb{P}[\mathcal{B}_s(m; \varepsilon)] &= \mathbb{P}\left[\cap_{i \in \mathcal{C}[s]}\left(\cap_{\ell \leq \ell_{\max}} \mathcal{C}_{i,\ell} \cap \mathcal{B}_{i,\ell_{\max}}\right)\right] \\
&= 1 - \mathbb{P}\left[\cup_{i \in \mathcal{C}[s]}\left(\cup_{\ell \leq \ell_{\max}} \mathcal{C}_{i,\ell}{}^{\mathsf{c}} \cup \mathcal{B}_{i,\ell_{\max}}{}^{\mathsf{c}}\right)\right] \\
&\geq 1 - \sum_{i \in \mathcal{C}[s]} \mathbb{P}\left[\cup_{\ell \leq \ell_{\max}} \mathcal{C}_{i,\ell}{}^{\mathsf{c}} \cup \mathcal{B}_{i,\ell_{\max}}{}^{\mathsf{c}}\right] \\
&= 1 - \sum_{i \in \mathcal{C}[s]}\left(1 - \mathbb{P}\left[\cap_{\ell \leq \ell_{\max}} \mathcal{C}_{i,\ell} \cap \mathcal{B}_{i,\ell_{\max}}\right]\right) \\
&= 1 - \sum_{i \in \mathcal{C}[s]}\left(1 - \mathbb{P}\left[\cap_{\ell \leq \ell_{\max}} \mathcal{C}_{i,\ell} | \mathcal{B}_{i,\ell_{\max}}\right] \mathbb{P}\left[\mathcal{B}_{i,\ell_{\max}}\right]\right):
\end{aligned}$$

We use Inequality 3 to get

$$\mathbb{P}[\mathcal{B}_{i,\ell_{\max}}] \geq 1 - \frac{\left(\tau(\ell_{\max}; \quad \varepsilon) K = \quad 2\right)^{h_i(\eta\varepsilon)} - 1}{K - 1}.$$

To relate $h_s(\text{"})$ to its child, we realize that i is an AVG node and hence,

$$\begin{aligned} h_i(\ \text{"}) &= \max\left(0; \frac{\log(\ (s)\ \ \text{"}(1-\ \ ))}{\log(\ =\ )}\right) \\ &= \frac{\log(\ (s)\ \ \text{"}(1-\ \ ))}{\log(\ =\ )} \\ &= \frac{\log(\ \text{"}(1-\ \ ))}{\log(\ =\ )} \\ &= \frac{\log(\ =\ ) + \log((1=\ )\text{"}(1-\ \ ))}{\log(\ =\ )} \\ &= h_s(\text{"}) - 1. \end{aligned}$$

We go back to bounding $\mathbb{P}[\mathcal{B}_s(m;\text{"})]$,

$$\begin{aligned} \mathbb{P}[\mathcal{B}_s(m;\text{"})] &\geq 1 - \sum_{i\in\mathcal{C}[s]}\left(1 - \left(\frac{\text{"}(1-\ \ )}{K}\right)\left(1 - \frac{\left(\bar{\ }(\grave{}_{\max};\ \text{"})\,K=\ ^2\right)^{h_i(\eta\varepsilon)} - 1}{K-1}\right)\right) \\ &\geq 1 - \sum_{i\in\mathcal{C}[s]}\left(\frac{}{K} + \frac{\left(\bar{\ }(\grave{}_{\max};\text{"})\,K\right)^{h_s(\varepsilon)-1} - 1}{K-1}\right) \qquad (\text{as } \text{"} \leq 1{=}(1-\ \ )) \\ &= 1 - \left(1 + K\frac{\left(\bar{\ }(\grave{}_{\max};\text{"})\,K\right)^{h_s(\varepsilon)-1} - 1}{K-1}\right) \\ &\geq 1 - \frac{\left(\bar{\ }(m;\text{"})\,K=\ ^2\right)^{h_s(\varepsilon)} - 1}{K-1}. \end{aligned}$$

To prove Property B, we introduce additional notation. We define

$$_{i,\ell} \stackrel{\text{def}}{=} \ _i\left(\grave{}; \tfrac{\eta}{1-\eta}\mathsf{U}(\grave{}; \text{"})\right).$$

We also define the empirically best and the best child as

$$\hat{\imath}_\ell \stackrel{\text{def}}{=} \arg\max_{i\in\mathcal{C}[s]} \ _{i,\ell} \quad \text{and} \quad i^\star \stackrel{\text{def}}{=} \arg\max_{i\in\mathcal{C}[s]} \mathcal{V}[i].$$

We define the random variable $\mathcal{L}_\ell$ as the set of all children that would be kept in Line 9 in Figure 3 if they are called with parameter $\grave{}$ as

$$\mathcal{L}_\ell \stackrel{\text{def}}{=} \left\{ i \in \mathcal{C}[s] : \ _{i,\ell} + \frac{2\mathsf{U}(\grave{}; \text{"})}{1-} \geq \sup_j \left[\ _{j,\ell} - \frac{2\mathsf{U}(\grave{}; \text{"})}{1-}\right]\right\}.$$

Notice that the MAX node is not constructing sets $\{\mathcal{L}_\ell\}_\ell$ iteratively: The reason is that some of the children might have been eliminated in the earlier rounds. However, at the end of the while loop in Figure 3, we get set $\mathcal{L}$ that is the intersection of all $\{\mathcal{L}_\ell\}_\ell$,

$$\mathcal{L} = \cap_{\ell\leq\ell_{\max}}\mathcal{L}_\ell.$$

We define $\mathcal{G}_\ell$ (which is *not* a random variable) as

$$\mathcal{G}_\ell \stackrel{\text{def}}{=} \left\{ i \in \mathcal{C}[s] : \mathcal{V}[i] + \frac{\mathsf{U}(\grave{}; \text{"})}{1-} \geq \mathcal{V}[i^\star] - \frac{\mathsf{U}(\grave{}; \text{"})}{1-}\right\}. \tag{5}$$

For simplicity, we also define $\hat{\imath} \stackrel{\text{def}}{=} \hat{\imath}_{\ell_{\max}}$; $\mathcal{G} \stackrel{\text{def}}{=} \mathcal{G}_{\ell_{\max}}$; and for all $i \in \mathcal{C}[s]$, $\ _i \stackrel{\text{def}}{=} \ _{i,\ell_{\max}}$.

We can now prove Property B. By definition of $C_{i,\ell}$ and induction assumption, we have

$$\left\{\cap_{i\in\mathcal{C}[s],\ell\leq\ell_{\max}} C_{i,\ell}\right\} \subseteq \{\forall i; \grave{} : |\ _{i,\ell} - \mathcal{V}[i]| \leq \mathsf{U}(\grave{}; \text{"}){=}(1-\ \ )\}$$

and we set $C \stackrel{\text{def}}{=} \cap_{i\in\mathcal{C}[s],\ell\leq\ell_{\max}} C_{i,\ell}$.

We distinguish two cases depending on the cardinality of the *deterministic* set $\mathcal{G}$ and assume C.

First case: $|\mathcal{G}| = 1$: If $|\mathcal{L}| > 1$ then for all $i \in \mathcal{L}$ such that $i \notin \mathcal{G} = \{i^\star\}$

$$\begin{aligned}
{}_i &\leq \mathcal{V}[i] + '' && \text{(because } C_{i,\ell_{\max}} \text{ holds)}\\
&< V(i^\star) - '' && \text{(because } i \in \mathcal{G}_s = \{i^\star\})\\
&\leq {}^{s,t}_{i^?} && \text{(because } C_{i^?,\ell_{\max}} \text{ holds)}
\end{aligned}$$

Therefore $\arg\max_{i\in\mathcal{L}} {}_i = i^\star$ which implies that $i^\star$ is called with parameters $(m'; '''')$ with

$$''''' \leq '' \leq '' \quad \text{and} \quad m' \geq m:$$

If $|\mathcal{L}| = 1$, then $\mathcal{L} = \{i^\star\}$ and consequently $i^\star$ is called with parameters $(m; '')$.

In both cases, on event C, ${}_s(m; '')$ is the output of a call to $i^\star$ with parameters $(m'; '''')$ with $m' \geq m$ and $''''' \leq ''$ and $m' \leq {}_{\max}$. We use Property B of induction assumption as ${}_{\max} \geq m$ and $\mathsf{U}({}_{\max}; '') = (1 - \ ) \leq ''$.

$$\begin{aligned}
\forall\ ; \mathbb{E}\left[e^{\lambda\left(\mu_{i^?}(m',\varepsilon') - \mathcal{V}[i^?]\right)} \middle| B_{i^?,\ell_{\max}}\right] &= \mathbb{E}\left[e^{\lambda(\mu_s(m,\varepsilon) - \mathcal{V}[s])} \middle| B_{i^?,\ell_{\max}}\right]\\
&\leq \exp\left(2\ \left(\frac{\bar{\ }(m'; '''')}{2}\right)^{h_{i^?}(\varepsilon')} \times \frac{1 - '''' \ (i^\star)}{= \ - 1} + ''''| \ \ | + \frac{2}{2(1 - \ )^2 m'}\right)\\
&\leq \exp\left(2\ \left(\frac{\bar{\ }(m; \ '')}{2}\right)^{h_s(\eta\varepsilon)} \times \frac{1 - \ '' \ (s)}{= \ - 1} + ''| \ \ | + \frac{2}{2(1 - \ )^2 m}\right)
\end{aligned}$$

Also, because for all $i; j \in \mathcal{C}[s]$ with $i \neq j$, B_i and ${}_j$ are independent, we can claim that

$$\forall\ ; \mathbb{E}\left[e^{\lambda(\mu_s(m,\varepsilon) - \mathcal{V}[s])} \middle| \cap_{i\in\mathcal{C}[s]} \mathcal{B}_{i,\ell_{\max}}\right] \leq \exp\left(2\ \left(\frac{\bar{\ }(m; \ '')}{2}\right)^{h_s(\eta\varepsilon)} \times \frac{1 - \ '' \ (s)}{= \ - 1} + ''| \ \ | + \frac{2}{2(1 - \ )^2 m}\right).$$

Using Lemma 7 we finaly get for all

$$\begin{aligned}
\mathbb{E}\left[e^{\lambda(\mu_s(m,\varepsilon) - \mathcal{V}[s])} \middle| \mathcal{B}_s(m; '')\right] &= \mathbb{E}\left[e^{\lambda(\mu_s(m,\varepsilon) - \mathcal{V}[s])} \middle| \cap_{i\in\mathcal{C}[s]} \left(\cap_{\ell\leq\ell_{\max}} C_{i,\ell} \cap \mathcal{B}_{i,\ell_{\max}}\right)\right]\\
&\leq \frac{\mathbb{E}\left[e^{\lambda(\mu_s(m,\varepsilon) - \mathcal{V}[s])} \middle| \cap_{i\in\mathcal{C}[s]} \mathcal{B}_{i,\ell_{\max}}\right]}{\mathbb{P}\left[\cap_{i\in\mathcal{C}[s],\ell\leq\ell_{\max}} C_{i,\ell} \middle| \cap_{i\in\mathcal{C}[s]} \mathcal{B}_{i,\ell_{\max}}\right]}\\
&\leq \frac{\exp\left(2\ \left(\frac{\bar{\ell}(m,\eta\varepsilon)}{\eta^2}\right)^{h_s(\eta\varepsilon)} \times \frac{1 - \eta\varepsilon\theta(s)}{\eta/\gamma - 1} + ''| \ \ | + \frac{2}{2(1 - \ )^2 m}\right)}{1 - \ ''(1 - \ )}
\end{aligned}$$

To simplify the expression, we note that for $0 < x \leq 1=2$, we have that $1=(1-x) \leq 1+2x \leq \exp(2x)$. We also use that $< 1=2$.

$$\begin{aligned}
\frac{\exp\left(2\ \left(\frac{\bar{\ell}(m,\eta\varepsilon)}{\eta^2}\right)^{h_s(\eta\varepsilon)} \times \frac{1 - \eta\varepsilon\theta(s)}{\eta/\gamma - 1}\right)}{1 - \ ''(1 - \ )} &\leq \exp\left(2\ \left(\frac{\bar{\ }(m; \ '')}{2}\right)^{h_s(\eta\varepsilon)} \times \frac{1 - \ '' \ (s)}{= \ - 1} + 2\ ''(1 - \ )\right)\\
&\leq \exp\left(2\ \left(\frac{\bar{\ }(m; ''')}{2}\right)^{h_s(\varepsilon)} \left(\frac{1 - \ ''(1 - \ )=}{= \ - 1} + ''(1 - \ )\right)\right)\\
&= \exp\left(2\ \left(\frac{\bar{\ }(m; ''')}{2}\right)^{h_s(\varepsilon)} \frac{1 - ''(1 - \ )}{= \ - 1}\right)\\
&= \exp\left(2\ \left(\frac{\bar{\ }(m; ''')}{2}\right)^{h_s(\varepsilon)} \frac{1 - \ '' \ (s)}{= \ - 1}\right)
\end{aligned}$$

Second case: $|\mathcal{G}| > 1$: For all $i \in \mathcal{G}_l$,

$$\begin{aligned} {}_{i,\ell} &\geq \mathcal{V}[i] - \mathsf{U}(\grave{}\,;\,'')=(1-\ ) && \text{(because } C \text{ holds)} \\ &\geq V(i^\star) - 3\mathsf{U}(\grave{}\,;\,'')=(1-\ ) && \text{(because } i \in \mathcal{G}_l\text{)} \\ &\geq {}^{s,t}_{i^?} - 4\mathsf{U}(\grave{}\,;\,'')=(1-\ ) && \text{(because } C \text{ holds)} \end{aligned}$$

Therefore for all $\grave{} \leq \grave{}_{\max}$,

$$\mathcal{G}_\ell \subset \mathcal{L}_\ell: \tag{6}$$

As a result $|\mathcal{L}| > |\mathcal{G}| > 1$. The output is then the maximum of the estimates in $\mathcal{L}$. The best estimate ${}_{\hat{\imath}}$ in $\mathcal{L}$ verifies

$$\begin{aligned} {}_{\hat{\imath}} &\geq {}_{i^?} && \text{(by definition of } \hat{\imath}\text{)} \\ &\geq V(i^\star) - \mathsf{U}(\grave{}_{\max};\,'')=(1-\ ) && \text{(because } C \text{ holds):} \end{aligned}$$

It also verifies that

$$\begin{aligned} {}_{\hat{\imath}} &\leq V(\hat{\imath}) + \mathsf{U}(\grave{}_{\max};\,'')=(1-\ ) && \text{(because } C_{\hat{\imath},\ell_{\max}} \text{ holds)} \\ &\leq V(i^\star) + \mathsf{U}(\grave{}_{\max};\,'')=(1-\ ) && \text{(by definition of } i^\star\text{):} \end{aligned}$$

Since ${}_s(m;\,'') = {}_{\hat{\imath}}$, we have

$$\left|\ {}_s(m;\,'') - \mathcal{V}[s]\right| \leq \frac{\mathsf{U}(\grave{}_{\max};\,'')}{1-} \leq '':$$

It follows that

$$\begin{aligned} \mathbb{E}\left[e^{\lambda(\mu_s(m,\varepsilon)-\mathcal{V}[s])}|\mathcal{B}_s(m;\,'')\right] &\leq \exp(|\ \ |'') \\ &\leq \exp\left(2\ \left(\frac{\bar{\ }(m;\,'';\,)}{2}\right)^{h_s(\varepsilon,)} \times \frac{1-\,'';\ \ (s)}{=\ -1} + |\ \ |'' + \frac{2}{2(1-\ )^2 m}\right). \end{aligned}$$

k is odd: Let s be a AVG node, $m \in \mathbb{N}_0^+$, and $''$ such that

$$'' > \frac{(\ =\ )^{(k-1)/2}}{(1-\ )} \quad \text{and} \quad \frac{1}{\max(m;\,\grave{}_{\max}('')) > \ :}$$

Hence, we can deduce

$$\frac{''}{-} > \frac{(\ =\ )^{(k-1)/2}}{(1-\ )} = \frac{(\ =\ )^{\frac{k-1}{2}-1}}{(1-\ )}.$$

Also, any child of s is a MAX node as nodes alternate between AVG and MAX. We can thus apply induction assumption to any child of s with parameters $(m';\,''')$ with $''' \geq ''=$ and $m' \leq \max(m;\,\grave{}_{\max}('')$.

For any child $i \in \mathcal{C}[s]$ of s and any integer $\grave{} \leq m$ we can thus apply our induction assumption to get

$$\mathbb{P}[\mathcal{B}_i(l;\,''=\ )] \geq 1 - \frac{\left(\bar{\ }(l;\,''=\ )\,K=\ ^2\right)^{h_i(\varepsilon/\gamma)} - 1}{K\ -1}$$

$$\forall\ ;\mathbb{E}\left[e^{\lambda(\mu_i(\ell,\varepsilon/\gamma))-\mathcal{V}[i])}\middle|\mathcal{B}_i(\grave{}\,;\,''=\ )\right] \leq \exp\left(2\ \left(\frac{\bar{\ }(\grave{}\,;\,''=\ )}{2}\right)^{h_i(\varepsilon/\gamma)} \times \frac{1-\,''=\ \ (i)}{=\ -1} + |\ \ |''=\ + \frac{2}{2(1-\ )^2\grave{}}\right)$$

We state again the definition of $\mathcal{B}_s$

$$\mathcal{B}_s(m;\,'') \stackrel{\text{def}}{=} \cap_{i\in\mathcal{C}[s]}\mathcal{B}_i(\ {}_i(m);\,''=\ )$$

with ${}_i(m) \stackrel{\text{def}}{=} |\{j:\ {}_{s,j} = i \text{ and } j \leq m\}|$.

We first prove property A.

$$
\begin{aligned}
\mathbb{P}\left[\cup_{i\in\mathcal{C}[s]}\mathcal{B}_i(\ _i(m);\text{"=}\ )^{\mathsf{c}}\right] &\leq \sum_{i\in\mathcal{C}[s]}\mathbb{P}\left[\mathcal{B}_i(\ _i(m);\text{"=}\ )^{\mathsf{c}}\right] \\
&= \int_k \sum_{i\in\mathcal{C}[s]}\mathbb{P}\left[\mathcal{B}_i(\ _i(m);\text{"=}\ )^{\mathsf{c}}\middle|\ _i(m)=k\right] d\ \ (k) \\
&= \int_k \sum_{i\in\mathcal{C}[s]}\mathbb{P}\left[\mathcal{B}_i(k_i;\text{"=}\ )^{\mathsf{c}}\middle|\ _i(m)=k_i\right] d\ \ (k) \\
&= \int_k \sum_{i\in\mathcal{C}[s]}\mathbb{P}\left[\mathcal{B}_i(k_i;\text{"=}\ )^{\mathsf{c}}\right] d\ \ (k) \\
&\leq \int_k \sum_{i\in\mathcal{C}[s];k_i>0} \frac{\left(\overline{\ }(k_i;\text{"=}\ )\ K\text{=}\ ^2\right)^{h_i(\varepsilon/\gamma)}-1}{K-1}\ \ (k) \\
&\leq \int \frac{\left(\overline{\ }(k_i;\text{"})\ K\text{=}\ ^2\right)^{h(\varepsilon)}-1}{K-1}\ \ (k) \\
&= \frac{\left(\overline{\ }(k_i;\text{"})\ K\text{=}\ ^2\right)^{h(\varepsilon)}-1}{K-1}
\end{aligned}
$$

We now prove property B.

$$
\begin{aligned}
&\mathbb{E}\left[\exp\left(\left(\ \text{=}m\right)\sum_{i\in\mathcal{C}[s]}\ _i(m)\ \ _i(\ _i(m);\text{"=}\ )\right)\middle|\mathcal{B}_s(m;\text{"})\right] \\
&= \int_k \mathbb{E}\left[\exp\left(\left(\ \text{=}m\right)\sum_{i\in\mathcal{C}[s]}\ _i(m)\ \ _i(\ _i(m);\text{"=}\ )\right)\middle|\mathcal{B}_s(m;\text{"});(\ _i(m))=(k_i)\right] d\ \ (k) \\
&= \int_k \mathbb{E}\left[\exp\left(\left(\ \text{=}m\right)\sum_{i\in\mathcal{C}[s]}k_i\ \ _i(k_i;\text{"=}\ )\right)\middle|\mathcal{B}_s(m;\text{"})\right] d\ \ (k) \\
&= \int_k \mathbb{E}\left[\prod_{i\in\mathcal{C}[s]}\exp\left(\left(\ \text{=}m\right)k_i\ \ _i(k_i;\text{"=}\ )\right)\middle|\mathcal{B}_s(m;\text{"})\right] d\ \ (k) \\
&\leq \int_k \exp\left(\sum_{i\in\mathcal{C}[s]}k_i\left(2\ \left(\frac{\overline{\ }(k_i;\text{"=}\ )}{2}\right)^{h_i(\varepsilon/\gamma)}\times\frac{1-\text{"=}\ \ (i)}{\text{=}\ -1}+|(\ \text{=}m)|\text{"=}\ +\frac{(\ \text{=}m)^2}{2(1-\ )^2k_i}\right)\right) d\ \ (k) \\
&= \int_k \exp\left(m2\ \left(\frac{\overline{\ }(m;\text{"=}\ )}{2}\right)^{h_i(\varepsilon/\gamma)}\times\frac{1-\text{"=}\ \ (i)}{\text{=}\ -1}+|\ \ |\text{"=}\ +\frac{^2}{2(1-\ )^2m}\right) d\ \ (k) \\
&\leq \exp\left(2\ \left(\frac{\overline{\ }(m;\text{"})}{2}\right)^{h_s(\varepsilon)}\times\frac{1-\text{"}\ \ (s)}{\text{=}\ -1}+|\ \ |\text{"=}\ +\frac{^2}{2(1-\ )^2m}\right)
\end{aligned}
$$

We can finally bound

$$
\begin{aligned}
&\mathbb{E}\left[\exp\left(r + \quad (\quad =m) \sum_{i\in\mathcal{C}[s]} \quad {}_i(m) \quad {}_i(\quad {}_i(m); \text{"}= \quad)\right) \middle| \mathcal{B}_s(m; \text{"})\right] \\
&\leq \exp\left(\frac{\quad {}^2}{2m} + 2 \quad \left(\frac{\text{"}(m; \text{"})}{\quad {}^2}\right)^{h_s(\varepsilon)} \times \frac{1 - \text{"} \quad (s)}{= \quad - 1} + | \quad | \text{"} + \frac{\quad {}^2 \quad {}^2}{2(1 - \quad)^2 m}\right) \\
&\leq \exp\left(2 \quad \left(\frac{\text{"}(m; \text{"})}{\quad {}^2}\right)^{h_s(\varepsilon)} \times \frac{1 - \text{"} \quad (s)}{= \quad - 1} + | \quad | \text{"} + \left(\frac{\quad}{\sqrt{2m}(1 - \quad)} + \frac{\quad}{\sqrt{2m}}\right)^2\right) \\
&= \exp\left(2 \quad \left(\frac{\text{"}(m; \text{"})}{\quad {}^2}\right)^{h_s(\varepsilon)} \times \frac{1 - \text{"} \quad (s)}{= \quad - 1} + | \quad | \text{"} + \frac{\quad {}^2}{2(1 - \quad)^2 m}\right).
\end{aligned}
$$

□

Now we stat and prove Theorem 4 that is equivalent to Theorem 1.

Theorem 4. *Let* $\quad$ *be the output of* `TrailBlazer`*.For all* $\quad ' > 0$*;* $\text{"} > 0$*, there exists an event* $\mathcal{B}$ *such that* $\mathbb{P}[\mathcal{B}] \geq 1 - \quad '$ *and on* $\mathcal{B}$*,* $| \quad - \mathcal{V}[s_0]| \leq \text{"}$*.*

Proof. We apply Lemma 2 to the root node s_0. Let us note the following:

$$
\begin{aligned}
m &= (\log(1= \quad) + \quad)=((1 - \quad)^2 \text{"}^2) \\
&= 2\log(\text{"}(1 - \quad))^2 \frac{\log\left(\frac{\log(K)}{(1-\eta)}\right)}{\log(\quad = \quad)}
\end{aligned}
$$

We set $\quad = \exp(\quad) \quad '$. We apply Lemma 2 on s_0 with $(m; \text{"}=2)$. We take

$$
\mathcal{B} = \mathcal{B}_{s_0}(\log(1= \quad) + \quad)=((1 - \quad)^2 \text{"}^2):
$$

We check that

$$
\frac{\left(\text{"}(m; \text{"}) \, K= \quad {}^2\right)^{h_s(\varepsilon)} - 1}{K \quad - 1} \leq \quad '
$$

and apply Lemma 6 with property B of Lemma 2 to get

$$
| \quad {}_{s_0}(m; \text{"}=2) - \mathcal{V}[s_0]| \leq \text{"}:
$$

□

B Sample complexity

Definition 5 (node classification)**.** *A node s of depth h is said to be*

- *suboptimal if there exists $h' < h$, such that $\Delta_{\to s}(s_{h'}) > 16\frac{\gamma^{(h-h')=2}}{\gamma(1-\gamma)}$,*
- *optimal if for all $h' < h$; $\Delta_{\to s}(s_{h'}) = 0$;*
- *near-optimal in all other cases.*

Lemma 3. *For any node s of depth h, integer m; and $" > 0$, a call to s with $(m; ")$ does not generate call to nodes of depth $h' > h + h_{\max}(")$ with*

$$h_{\max}(") \leq \left\lceil 2\frac{\log("(1-\ ))}{\log(\ =\ )} \right\rceil.$$

MAX and AVG nodes alternate. First, Any AVG node called with $"$ can only perform calls with $"' \geq "=\ $. Second, MAX nodes called with $"$ can only perform calls with $"' \geq\ "$. Finally, an AVG node called with $" = 1=(1-\ )$ *cannot* perform a call to deeper node.

For any node s called with different $(m_t; "_t)$ during the algorithm execution, we let $\Gamma_s = \max_t m_t$ be the maximum m_t that s has been called with.

Lemma 4. *We denote the parameters the root has been called with by $(m_0; "_0)$.*

If $\mathcal{B}_{s_0}(m; ")$ holds then for all nodes s of depth h and $h' < h$, the following holds,

$$s \text{ is suboptimal} \implies \Gamma_s = 0$$

$$\Delta(s_{h'}) > 16\frac{^{(h-h')/2}}{(1-\ )} \implies \text{any call to } s \text{ was generated by a call from } s_{h'} \text{ with } m \leq \Gamma_{s_{h'}}:$$

Proof. For any node s of depth h, we assume that $\Gamma_s > 0$ and we want to prove that s is not suboptimal. Let h' be some integer with $h' < h$. We note $(m_{s_{h'}}; "_{s_{h'}})$ the parameters $s_{h'}$ has been called with. If $\Gamma_s > 0$, then, by Lemma 3, that means that $s_{h'}$ performs a call to its child i_s in direction to s with parameters $(m; ")$ such that $h_{\max}(") \geq h - h'$.

For any $m'; "'$; if $s_{h'}$ performs a call to i_s with parameters $(m'; "')$ with $"' \leq "$ then

1. either $i_s \in \mathcal{L}_{\ell_{\max}(m_{s_{h'}}, \varepsilon_{s_{h'}})}$ and $"_{s_{h'}} \leq "$,
2. or $i_s \in \mathcal{L}_\ell$ for some $`$ such that $\frac{\eta}{1-\eta}\mathsf{U}_\ell \leq "$.

We use the definition of $\mathcal{G}$ (5) and the result of Equation 6.

Case 1. For all $`'$ we have $\mathcal{L}_{\ell'} \subset \mathcal{G}_{\ell'}$ therefore $\mathcal{V}[i^\star] - \mathcal{V}[i_s] \leq \frac{4\mathsf{U}_{`'}}{1-\eta} \leq 4\frac{\varepsilon}{\eta}$.

Case 2. By definition of $`_{\max}$, $\mathsf{U}_{\ell_{\max}+1} \leq (1-\ )"_{s_{h'}}$. We also have that

$$\forall `' \exists a \geq 0 \quad \frac{\mathsf{U}_{\ell'+1}}{\mathsf{U}_{\ell'}} = \sqrt{\frac{`'}{`'+1}\frac{\log(`'+1)+a}{\log(`')+a}} \geq \sqrt{\frac{`'}{`'+1}} \geq 1=\sqrt{2}:$$

We put it together using $i_s \in \mathcal{L}_{\ell_{\max}} \subset \mathcal{H}_{\ell_{\max}}$,

$$\mathcal{V}[i^\star] - \mathcal{V}[i_s] \leq \frac{4\mathsf{U}_{\ell_{\max}}}{1-} \leq \mathsf{U}_{\ell_{\max}} \leq \frac{4\sqrt{2}}{1-}\mathsf{U}_{\ell_{\max}+1} \leq 4\sqrt{2}"_s \leq 4\sqrt{2}"=\ :$$

Either $\mathcal{V}[i^\star] - \mathcal{V}[i_s] \leq \frac{4\varepsilon}{\eta}$ or $\mathcal{V}[i^\star] - \mathcal{V}[i_s] \leq \frac{4\varepsilon\sqrt{2}}{\eta}$ from which we deduce that

$$\mathcal{V}[i^\star] - \mathcal{V}[i_s] \leq \frac{4"\sqrt{2}}{}.$$

We use $h - h' \leq h_{\max}(\varepsilon) \leq 2\frac{\log(\varepsilon(1-\gamma))}{\log(\gamma/\eta)} + 1$ to get $(\ \ =\ \ )^{(h-h')/2} \geq \varepsilon(1-\ \ )\ \ =\ \ $ and using $\ \ \leq 1$

$$\varepsilon \leq \frac{1}{(1-\ \ )}(1=\ \ )^{(h-h')/2}\ \ \ \ ^{(h-h')/2}.$$

As $h - h' \leq h_{\max}(\varepsilon_0)$,

$$\log\left((1=\ \ )^{(h-h')/2}\right) \leq \log\left((1=\ \ )^{h_{\max}(\varepsilon_0)/2}\right) \leq \frac{\log 1=}{\log(1=\varepsilon)}\frac{\log(1=\varepsilon)}{\log(\ \ =\ \ )} \leq 1.$$

We combine

$$\varepsilon \leq \frac{e}{(1-\ \ )}\ \ \ \ ^{(h-h')/2} \text{ and } \mathcal{V}[i^\star] - \mathcal{V}[i_s] \leq \frac{4\varepsilon\sqrt{2}}{},$$

to finally get

$$\Delta_{\to s}(s_{h'}) = \mathcal{V}[i^\star] - \mathcal{V}[i_s] \leq \frac{4e\sqrt{2}}{(1-\ \ )}\ \ \ \ ^{(h-h')/2} \leq 16\frac{\ \ ^{(h-h')/2}}{(1-\ \ )}.$$

This is true for any $h' < h$ which proves that s is not suboptimal.

We now assume that $\Delta(s_{h'}) > 16\frac{\gamma^{(h-h')=2}}{\gamma(1-\gamma)}$ and prove that if a call to $i^\star$ later generated calls to s, then this call to $i^\star$ was done with $(m; \varepsilon)$ with $m \leq \Gamma_{s_{h'}}$.

By Lemma 3, $h_{\max}(\varepsilon) \geq h - h'$. We distinguish three cases,

1. either $s_{h'}$ calls $i^\star$ within the while loop,
2. or $s_{h'}$ calls $i^\star$ outside the while loop with $|\mathcal{L}| > 1$,
3. or $s_{h'}$ calls $i^\star$ outside the while loop with $|\mathcal{L}| = 1$.

Case 1. If this call is made to $i \neq i^\star$ with $\varepsilon = \frac{\eta}{1-\eta}U_\ell$, with the same reasoning as above we conclude that $\mathcal{V}[i^\star] - \mathcal{V}[i_s] \leq \frac{4e\sqrt{2}}{\gamma(1-\gamma)}\ \ \ \ ^{h-h'}$ which contradicts the assumption.

Case 2. We use again the same reasoning as above with $U_{\ell_{\max}}$ to conclude that if $\varepsilon_{s_{h'}} \geq \ \ \varepsilon$ then $\mathcal{V}[i^\star] - \mathcal{V}[i_s] \leq \frac{4e\sqrt{2}}{\gamma(1-\gamma)}\ \ \ \ ^{h-h'}$.

Case 3. In this last case, the call is done with $(m_{s_{h'}}; \varepsilon_{s_{h'}})$ and $m_{s_{h'}} \leq \Gamma_{s_{h'}}$.

□

B.1 Finite case: Proof of Theorem 2

Proof. Any near-optimal node of depth h called with parameters $(m'; \varepsilon')$ generates at most

$$\frac{\log(1=(\varepsilon\ \ ))}{(1-\ \ )^2(1-\ \ )^2\varepsilon'^2} \quad \text{samples.}$$

This follows from the same reasoning as in previous theorem. Then we can use that $\varepsilon' \leq \varepsilon(\ \ =\ \ )^{h/2}$. Therefore, the maximum number of samples T is

$$T \leq \sum_{h \leq h_{\max}} N^h\ \ \ \ ^h\frac{\log(1=(\varepsilon\ \ ))}{(1-\ \ )^2(1-\ \ )^2\varepsilon^2\ \ \ \ ^{h/2}=\ \ ^{h/2}}.$$

If $N \quad = \quad ^2 > 1$, then,

$$
\begin{aligned}
T &\leq \mathcal{O}\left(h_{\max}(N \quad = \quad)^{h_{\max}} \frac{\log(1=(" \quad))}{(1- \quad)^2(1- \quad)^2 "^2}\right) \\
&= \mathcal{O}\left(h_{\max}(1="^{(\log(N\kappa)+2\log(\eta/\gamma))/\log(\eta/\gamma)} \frac{\log(1=(" \quad))}{(1- \quad)^2(1- \quad)^2 "^2}\right) \\
&= \mathcal{O}\left(\frac{\log(1="}{\log(\quad = \quad)}(1="^{\log(N\kappa)/\log(\eta/\gamma)} \frac{\log(1=(" \quad))}{(1- \quad)^2(1- \quad)^2}\right) \\
&= \mathcal{O}\left((1="^{\log(N\kappa)/\log(1/\gamma)+o(1)} \frac{\log(1=(" \quad))\log(1="^4}{(1- \quad)^5}\right) \qquad \text{Setting} \quad = \quad ^{1/\max(2,\log(1/\varepsilon))} \\
&= \mathcal{O}\left((1="^{\log(N\kappa)/\log(1/\gamma)+o(1)} \left(\frac{\log(1= \quad)+\log(1="}{1- \quad}\right)^5\right)
\end{aligned}
$$

If $N \quad = \quad ^2 < 1$; then,

$$
\begin{aligned}
T &\leq \mathcal{O}\left((1="^2 \frac{1}{1-N \quad = \quad ^2} \frac{\log(1=(" \quad))}{(1- \quad)^2(1- \quad)^2}\right) \\
&= \mathcal{O}\left((1="^2 \frac{(\log(1= \quad)+\log(1="))^3}{(1-N \quad = \quad ^2)(1- \quad)^4}\right). \qquad \text{Setting} \quad = \quad ^{1/\max(2,\log(1/\varepsilon))}
\end{aligned}
$$

If $N \quad = \quad ^2 = 1$; then,

$$
\begin{aligned}
T &\leq \mathcal{O}\left(h_{\max}(1= \quad)^{h_{\max}} \frac{\log(1=(" \quad))}{(1- \quad)^2(1- \quad)^2 "^2}\right) \\
T &\leq \mathcal{O}\left(\frac{\log(1=(" \quad))\log(1="^4}{(1- \quad)^5 "^2}\right) \qquad (1= \quad)^{h_{\max}} = \mathcal{O}(1) \\
T &\leq \mathcal{O}\left((1="^2 \left(\frac{\log(1= \quad)+\log(1="}{1- \quad}\right)^5\right).
\end{aligned}
$$

□

B.2 Infinite case

For any MAX node s' of depth h' and any $h > h'$, we define $\Gamma^h_{s'}$ as the maximum m the node s' has been called with; such that this call generated calls up to depth h.

Lemma 5. *There exists such that for any node s of depth h and any ancestor $s_{h'}$ of s of depth $h' \leq h$,*

$$
\mathbb{E}\left[\Gamma_s\right] \leq \mathbb{1}\left(s \textit{ is not suboptimal}\right) p(s|s_0) \prod_{\substack{0\leq h'<h \\ h'\equiv 0 (\mathrm{mod}\ 2)}} \left(\frac{1}{h-h'}\right)^{\mathbb{1}\left(\Delta(s_{h'})<16\frac{(h-h')=2}{(1- \quad)}\right)} \Gamma_{s_0}.
$$

Proof. We prove that

$$
\mathbb{E}\left[\Gamma^h_{s_{h'+1}}\right] \leq \mathbb{1}\left(s_{h'+1} \text{ is not suboptimal}\right) \mathbb{E}\left[\Gamma_{s^h_{h'}}\right] p(s|s') \left(\frac{1}{h-h'}\right)^{\mathbb{1}\left(\Delta(s_{h'})<16\frac{(h-h')=2}{(1- \quad)}\right)}.
$$

If $\Delta_{s_{h'}} \geq 16\frac{\gamma^{(h-h')=2}}{\gamma(1-\gamma)}$ then by Lemma 4, either s is suboptimal and therefore $\Gamma_s = 0$ or the calls of $s_{h'}$ towards s which generate calls up to depth h are done with parameters $m \leq \mathbb{E}\left[\Gamma^h_s\right]$.

In the other case, a node called with $(m; ")$ generates at most $\mathcal{O}\left(1="^2\right)$ samples. Moreover, since $s_{h'}$ is of depth h', we know that it is called with $" \geq \mathcal{O}\left((\ = \ )^{(h-h')/2}\right) \geq \ ^{(h-h')/2}=K$ for some as $(1= \ )^{h_{\max}} = \mathcal{O}(1)$:

Also, if no sample from $s_{h'}$ reaches $s_{h'+1}$ then $\Gamma_s = 0$. Together, we get that

$$\Gamma_{s_{h'+1}} \leq \mathbb{1}\left(\text{One of the } \left(\frac{1}{h-h'}\right)^{\mathbb{1}\left(\Delta(s_{h'})<16\frac{(h-h')=2}{(1- \ )}\right)} \text{ samples from } s_{h'} \text{ goes to } s_{h'+1}\right)\Gamma_{s_{h'}}$$

$$\mathbb{E}\left[\Gamma_{s_{h'+1}}\right] \leq \mathbb{P}\left[\text{One of the } \left(\frac{1}{h-h'}\right)^{\mathbb{1}\left(\Delta(s_{h'})<16\frac{(h-h')=2}{(1- \ )}\right)} \text{ samples from } s_{h'} \text{ goes to } s_{h'+1}\right]\mathbb{E}\left[\Gamma_{s_{h'}}\right]$$

$$\leq \mathbb{E}\left[\Gamma_{s^h_{h'}}\right] p(s|s')\left(\frac{1}{h-h'}\right)^{\mathbb{1}\left(\Delta(s_{h'})<16\frac{(h-h')=2}{(1- \ )}\right)}.$$

Now $\mathbb{E}\left[\Gamma_{s_{h'+1}}\right] = 0$ and therefore $s_{h'+1}$ is h-suboptimal. Moreover, if one of the $s_{h'}$ is h-suboptimal then s is suboptimal. To finish the proof, we multiply all these inequalities over h'. □

B.2.1 Proof of Theorem 3

Proof. We bound the expected number of oracle calls as

$$\begin{aligned}
\mathbb{E}\left[n(";\ )\right] &\leq \mathbb{E}\left[\sum_{h=1}^{h_{\max}(\varepsilon)} \sum_{s \text{ of depth } h} \Gamma_s\right] \\
&= \sum_{h=1}^{h_{\max}(\varepsilon)} \sum_{s \text{ of depth } h} \mathbb{E}\left[\Gamma_s\right] \\
&\leq \sum_{h=1}^{h_{\max}(\varepsilon)} K^{h/2}\mathbb{E}\left[\mathbb{1}\left(S^h \in \mathcal{N}_h\right) \prod_{\substack{0\leq h'<h \\ h'\equiv 0 (\text{mod } 2)}} \left(\frac{1}{2(h-h')}\right)^{\mathbb{1}\left(\Delta(S^h_{h'})<16\frac{(h-h')=2}{(1- \ )}\right)}\right]\Gamma_{s_0} \\
&\leq \sum_{h=1}^{h_{\max}(\varepsilon)} a^{\ -dh}\Gamma_{s_0} \\
&\leq a h_{\max}(")\,\mathcal{O}\left(1="\right)^d \Gamma_{s_0} \\
&\leq \mathcal{O}\left(\log(1=")(1=")^d \frac{\left(\log(1= \ ) + \log(1="))\right)^2}{"^2}\right) \\
&\leq \mathcal{O}\left(\frac{\left(\log(1= \ ) + \log(1="))\right)^3}{"^{2+d}}\right).
\end{aligned}$$

This is true on $\mathcal{B}_s(m; ")$ which holds with probability $1 -$ but as $\ll \mathcal{O}\left(1="^3\right)$, the result also holds on the whole probability space. □

B.2.2 Proof of Lemma 1

Proof. We note $x_{h'} \stackrel{\text{def}}{=} \frac{1}{\xi\gamma^{h-h'}}$ and $y_{h'} \stackrel{\text{def}}{=} 16\frac{\gamma^{(h-h')=2}}{\gamma(1-\gamma)}$ so that

$$\mathbb{E}\left[\mathbb{1}\left(S^h \in \mathcal{N}_h\right) \prod_{\substack{0\leq h'<h \\ h'\equiv 0 (\text{mod } 2)}} \left(\frac{1}{h-h'}\right)^{\mathbb{1}\left(\Delta(S^h_{h'})<16\frac{(h-h')=2}{(1- \ )}\right)}\right] = \mathbb{E}\left[\mathbb{1}\left(S^h \in \mathcal{N}_h\right) \prod_{\substack{0\leq h'<h \\ h'\equiv 0 (\text{mod } 2)}} x_{h'}^{\mathbb{1}\left(\Delta(S^h_{h'})<y_{h'}\right)}\right]:$$

Since at all AVG nodes transitions are taken independently:

$$
\begin{aligned}
\mathbb{E}\left[\mathbb{1}\left(S^h \in \mathcal{N}_h\right) \prod_{\substack{0\le h'<h \\ h'\equiv 0 (\mathrm{mod}\ 2)}} x_{h'}^{\mathbb{1}\left(\Delta(S^h_{h'})<y_{h'}\right)}\right] &= \mathbb{E}\left[\mathbb{1}\left(S^h \in \mathcal{N}_h\right)\right] \prod_{\substack{0\le h'<h \\ h'\equiv 0 (\mathrm{mod}\ 2)}} \mathbb{E}\left[x_{h'}^{\mathbb{1}\left(\Delta(S^h_{h'})<y_{h'}\right)}\right] \\
&= \mathbb{P}\left[S^h \in \mathcal{N}_h\right] \prod_{\substack{0\le h'<h \\ h'\equiv 0 (\mathrm{mod}\ 2)}} \left(1 + x_{h'}\mathbb{P}\left[\left(\Delta(S^h_{h'}) < y_{h'}\right)\right]\right) \\
&\le \mathbb{P}\left[S^h \in \mathcal{N}_h\right] \prod_{\substack{0\le h'<h \\ h'\equiv 0 (\mathrm{mod}\ 2)}} \left(1 + a x_{h'} y^b_{h'}\right) \\
&\le \mathbb{P}\left[S^h \in \mathcal{N}_h\right] \exp\left(\sum_{0\le h'<h, h'\equiv 0 (\mathrm{mod}\ 2)} a x_{h'} y^b_{h'}\right) \\
&\le \mathbb{P}\left[S^h \in \mathcal{N}_h\right] \exp\left(\sum_{0\le h'<h, h'\equiv 0 (\mathrm{mod}\ 2)} {}_2{}^{(b-2)h'}\right) \\
&\le \mathbb{P}\left[S^h \in \mathcal{N}_h\right] \exp\left(\sum_{h'=0}^{\infty} {}_2{}^{(b-2)h'}\right) \\
&\le \mathbb{P}\left[S^h \in \mathcal{N}_h\right] \ {}_3
\end{aligned}
$$

We then conclude with

$$
\begin{aligned}
\mathbb{P}\left[S^h \in \mathcal{N}_h\right] &\le \prod_{\substack{0\le h'<h \\ h'\equiv 0 (\mathrm{mod}\ 2)}} a y^b_{h'} \\
&\le \prod_{\substack{0\le h'<h \\ h'\equiv 0 (\mathrm{mod}\ 2)}} a \left(16 \frac{{}^{(h-h')/2}}{(1- \quad)}\right)^b \\
&= (1 = K^{h/2}) \prod_{\substack{0\le h'<h \\ h'\equiv 0 (\mathrm{mod}\ 2)}} K a \left(16 \frac{{}^{(h-h')/2}}{(1- \quad)}\right)^b \\
&\le \ {}_4 = K^{h/2}:
\end{aligned}
$$

□

C On the choice of the near-optimality definition

In this part, we discuss the possible alternative choices to the near-optimality set $\mathcal{N}_h$ and explain the choice we made in Definition 1. From this definition, the knowledge of the rewards of the near-optimal nodes is enough to compute an optimal policy. Therefore, whatever the rewards associated with the non-near-optimal nodes are, they do not change the optimal policy. A good *adaptive* algorithm, would not exploring the whole tree, but would with high probability only explore a set not significantly larger than a set $\mathcal{N}_h$ of near-optimal nodes. There are other possible definitions of the near-optimality set satisfying this desired property that we could possibly consider for the definition of $\mathcal{N}_h$. Ideally, this set would be as small as possible.

Alternative definition 1 An idea could be to consider for any node s and an ancestor s', the value of the policy starting from s' choosing the action leading to s every time it can and playing optimally otherwise. Let $\mathcal{V}_s(s')$ be this value. A natural approach would be to define $\mathcal{N}_h$ only based on $\mathcal{V}_s(s_0)$

with s_0 being the root. To ensure that exploring the near-optimal nodes is enough to compute the optimal policy we need any near-optimal nodes s of depth h to verify

$$(\mathcal{V}[s_0] - \mathcal{V}_s(s_0))\, p(s|s_0) \leq \frac{\quad^{h}}{1-}.$$

When there is no AVG node, $p(s) = 1$ and this definition coincides with the near-optimal set defined in OLOP [5]. Nevertheless, in our case, $p(s)$ can be low and even 0 in the infinite case. When $p(s|s_0) = 0$ for all s, any node would be near-optimal which is bad because then the near-optimality set is large and the sample complexity of the resulting algorithm suffers.

Alternative definition 2 There is a smarter way to define a near-optimality set. We could define it as all the nodes s of depth h such that

$$\forall h' \leq h; \quad \sum_{i=h'}^{h} p(s_i|s_{h'})\ \ ^{i}\left(\mathcal{V}[s_i] - \mathcal{V}[c(s_i; s)]\right) \leq \frac{\quad^{h}}{1-}.$$

Taking $h' = 0$, we recover the alternative definition 1. Alternative definition 2 defines a smaller near-optimality set, as the condition needs to be verified for all h' and not just for $h' = 0$. This definition would also lead to a smaller near-optimality set than our Definition 1. Indeed, in Definition 1 we consider only the first term of the sum above. This enables the algorithm to compute its strategy from a node s only based on the value of its descendants. This is an ingredient leading to better computational efficiency of TrailBlazer. Indeed, the computational complexity of our algorithm is linear in the number of calls to the generative model. In addition, when the transition probabilities are low, both definitions are close and coincide in the infinite case, when the probability transitions are continuous and therefore $p(s|s') = 0$. In this case ($p(s|s') = 0$), for any set of nodes of not null probability mass, it is possible to change the value of the rewards on this subset such that the value of the root change. In this sense, this definition of near-optimal nodes is minimal when $p(s|s') = 0$.

D Auxiliary material

Lemma 6. *If for some* $\ > 0;\ \ ; M$*, a random variable* X *verifies that*

$$\mathbb{E}\left[e^{\lambda X}\right] \leq M \exp\left(|\ \ | \ \ + \frac{\ ^{2}\ ^{2}}{2}\right);$$

then for all $u > 0$

$$\mathbb{P}\left[|X| \geq u\right] \leq 2M \exp\left(-\frac{(u - \ \ )^2}{2\ \ ^{2}}\right).$$

Proof. Using Markov inequality for $u >$ and setting $= \frac{u-\alpha}{\beta^2}$, we get

$$\mathbb{P}\left[X \geq u\right] \leq \frac{\mathbb{E}\left[e^{\lambda X}\right]}{e^{\lambda u}} \leq M \exp\left(|\ \ | \ \ - \ \ u + \frac{\ ^{2}\ ^{2}}{2}\right) \leq M \exp\left(-\frac{(u - \ \ )^2}{2\ \ ^{2}}\right).$$

Similarly, for $u < -\ \ ,$

$$\mathbb{P}\left[X \leq u\right] \leq M \exp\left(-\frac{(u + \ \ )^2}{2\ \ ^{2}}\right).$$

Taking a union bound, we have that for all $u >\ \ ,$

$$\mathbb{P}\left[|X| \geq u\right] \leq 2M \exp\left(-\frac{(u - \ \ )^2}{2\ \ ^{2}}\right).$$

□

Lemma 7. *For any positive random variable* X *and event* A*,*

$$\mathbb{E}\left[X|A\right] \leq \frac{\mathbb{E}\left[X\right]}{\mathbb{P}\left[A\right]}.$$

Proof.

$$\mathbb{E}\left[X\right] = \mathbb{E}\left[X|A\right]\mathbb{P}\left[A\right] + \mathbb{E}\left[X|A^c\right]\mathbb{P}\left[A^c\right] \geq \mathbb{E}\left[X|A\right]\mathbb{P}\left[A\right]$$

□