Scalable Safe Policy Improvement via Monte Carlo Tree Search

Alberto Castellini [1] **Federico Bianchi** [* 1] **Edoardo Zorzi** [* 1 2]
Thiago D. Simão [3] **Alessandro Farinelli** [1] **Matthijs T. J. Spaan** [4]

Abstract

Algorithms for safely improving policies are important to deploy reinforcement learning approaches in real-world scenarios. In this work, we propose an algorithm, called MCTS-SPIBB, that computes safe policy improvement online using a Monte Carlo Tree Search based strategy. We theoretically prove that the policy generated by MCTS-SPIBB converges, as the number of simulations grows, to the optimal safely improved policy generated by Safe Policy Improvement with Baseline Bootstrapping (SPIBB), a popular algorithm based on policy iteration. Moreover, our empirical analysis performed on three standard benchmark domains shows that MCTS-SPIBB scales to significantly larger problems than SPIBB because it computes the policy online and locally, i.e., only in the states actually visited by the agent.

1. Introduction

Safety is a paramount requirement for the deployment of reinforcement learning (RL; Sutton & Barto, 2018). In environments where humans interact with robots or other kinds of autonomous agents (e.g., autonomous cars, drones, or industrial plants) safety, robustness, and reliability of control policies are crucial issues. Safe RL investigates how these issues can be addressed by learning policies that maximize expected return while ensuring minimal performance level or respecting safety constraints during learning (García & Fernández, 2015). In this work, we focus on Safe Policy Improvement (SPI; Thomas et al., 2015; Petrik et al., 2016) for Markov Decision Processes (MDPs; Puterman, 2014; Russell & Norvig, 2020) where the agent is provided with a baseline policy and data is collected running this policy in the real environment. The goal of SPI is to use this data and the baseline policy to compute a new policy with better performance than the baseline. The method is considered safe if it is guaranteed to return an improved policy with high probability.

One of the most popular algorithms for SPI is Safe Policy Improvement with Baseline Bootstrapping (SPIBB; Laroche et al., 2019). It is an extension of policy iteration which computes a policy that safely improves a baseline given a dataset of trajectories produced by executing such baseline in the environment. SPIBB considers a percentile criterion that optimizes the policy in the worst-case scenario (Petrik et al., 2016). The safe improvement is achieved by bootstrapping the policy trained from data with the baseline policy in the state-action pairs not executed enough times in the available trajectories. In practice, SPIBB searches the improved policy in the constrained space of policies that are equal to the baseline in the state-action pairs not observed enough times in the dataset of trajectories (*SPIBB constraint*). This strategy is proven to produce a safe improvement. SPIBB's derivation from policy iteration, however, limits its applicability to small domains, since the running time of this algorithm depends cubically (if the Bellman equation is solved by matrix inversion) or quadratically (if it is solved by dynamic programming) on the size of the state space (Sutton & Barto, 2018). We investigate how to develop SPI algorithms for large problems.

Several approaches have been proposed to scale reinforcement learning to large state spaces (Kearns et al., 2002). Among the most promising techniques, there is Monte Carlo Tree Search (MCTS; Coulom, 2007; Browne et al., 2012), an online sampling-based lookahead-search method that efficiently computes policies converging to optimality and having a small error probability if stopped before convergence. In MCTS, Upper Confidence Bound applied to Trees (UCT; Kocsis & Szepesvári, 2006) is used as an action selection strategy to deal with the exploration-exploitation dilemma. UCT extends the Upper Confidence Bound algorithm (UCB; Auer et al., 2002), originally defined for multi-armed bandit problems. The per-state running time of MCTS has no dependence on the number of states but only on the number of

[*] Equal contribution [1] Department of Computer Science, University of Verona, Verona, Italy [2] Department of Computer Science, ETH Zurich, 8092 Zurich, Switzerland [3] Department of Software Science, Radboud University, Nijmegen, Netherlands [4] Department of Software Technology, Delft University of Technology, Delft, Netherlands. Correspondence to: Alberto Castellini <alberto.castellini@univr.it>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

simulations, hence the approach is suitable for domains with large state spaces, e.g., the game of Go where the popular AlphaGo reached superhuman performance using MCTS (Silver et al., 2016). However, using MCTS in the context of SPI is not trivial because the approach is online, i.e., it does not compute the policy for all states. Furthermore, MCTS performs simulations to approximate action Q-values considering future rewards. Simulations addressed by UCT and rollout policies are used to search for the optimal policy. These elements are not present in SPIBB which is based on a different RL paradigm.

Against this background, we propose a novel algorithm that, given a baseline policy, guarantees SPI using MCTS as a solution strategy. This is, to the best of our knowledge, the first use of MCTS to improve the scalability on SPI. The proposed methodology, called MCTS-SPIBB, extends MCTS by considering the *SPIBB constraint*. The policies generated by MCTS-SPIBB converge to the improved policies generated by SPIBB as the number of MCTS simulations increases. This is achieved by suitably merging UCT with the baseline policy inside MCTS. We prove this result by a full theoretical analysis of our approach. We also empirically evaluate the proposed algorithm on three domains, i.e., *GridWorld* (Russell & Norvig, 2020), *SysAdmin* (Guestrin et al., 2003) and *WetChicken* (Scholl et al., 2022b) showing three key points: the policy generated by MCTS-SPIBB converges to an optimal policy satisfying the SPIBB constraint; MCTS-SPIBB produces a safe policy improvement; MCTS-SPIBB scales better than SPIBB to large domains.

In summary, this work proposes four main contributions to the state-of-the-art: *i)* a formalization of the problem of SPI in the MCTS framework; *ii)* an online sampling-based algorithm for SPI on MDPs, called MCTS-SPIBB; *iii)* a theoretical proof that the policy generated by MCTS-SPIBB converges to the SPIBB policy as the number of simulations increases; *iv)* an empirical evaluation of convergence, safety, and scalability of MCTS-SPIBB on benchmark domains.

2. Related Work

The main research topics related to our work are offline RL, safe policy improvement, and MCTS-based planning/RL.

Offline RL develops reinforcement learning algorithms that generate control policies using data previously collected by a behaviour policy, without collecting additional data (Levine et al., 2020). Ernst et al. (2005) use regression tree approximations to generate the Q-function from a batch of trajectories. Other methods using low dimensional or linear parametrizations of the Q-function are described by Lange et al. (2012). Fujimoto et al. (2019) use a policy similar to the baseline to create the dataset. Kumar et al. (2019) minimize the bootstrapping error using actions within the support of the dataset. Furthermore, Kumar et al. (2020); Yu et al. (2020); Kidambi et al. (2020) tackle the problem of the distributional shift between offline training data and the states visited by the learned policy. However, none of these methods considers the behaviour policy as an explicit input, and none of them provides guarantees on the improvement with respect to that policy.

Safe Policy Improvement is a specialization of offline RL for safety-critical applications in which a behaviour policy must be improved with guarantees on the performance of the new policy w.r.t. the performance of the behaviour policy (Levine et al., 2020). Delage & Mannor (2010) propose a percentile criterion which maximizes the expected return of the worst cases. Thomas et al. (2015) allow to compute policies with performance below a bound with probability higher than a predefined value. Some model-based methods provide improved policies with guarantees on the performance by searching in the estimated transition model with guarantees on their error. Petrik et al. (2016) use a percentile criterion defined by Delage & Mannor (2010) and Robust MDPs to find a lower bound on the policy performance in the worst-case scenario. Laroche et al. (2019) extend this method allowing a constraint on the minimum number of observations of a transition to be violated on some state-action pairs. Extensions (Simão & Spaan, 2019b;a; Nadjahi et al., 2019; Simão et al., 2020; Scholl et al., 2022a; Simão et al., 2023; Wienhöft et al., 2023) and alternative strategies (Abbasi-Yadkori et al., 2016; Cohen et al., 2018; Chandak et al., 2020; Sarafian et al., 2020) are also available, but none of them use MCTS to scale to large domains, making MCTS-SPIBB the first algorithm that employs this strategy.

MCTS-based Planning/RL aims to scale sequential decision-making based on MDP/POMDP to large domains, such as real-world applications, via efficient Monte Carlo sampling. Kearns et al. (2002) proposes a first online planning algorithm based on sparse sampling, which has no complexity dependence on the state-space dimension. MCTS (Chaslot et al., 2008; Browne et al., 2012) extends this approach using the UCT algorithm (Kocsis & Szepesvári, 2006; Coulom, 2007) for balancing exploration and exploitation. The technique converges to the optimal policy more efficiently than Monte Carlo sampling. UCT extends the UCB strategy (Auer et al., 2002) to the case of non-stationary bandit problems, in which the payoff sequences may drift as actions are taken. MCTS was also applied to POMDPs (Silver & Veness, 2010) with extensions in several directions, such as model learning (Katt et al., 2017; 2019) and prior knowledge exploitation (Castellini et al., 2019; Mazzi et al., 2021a;b; 2023). None of these algorithms, however, aim to solve the SPI problem. The challenge in this context is to consider the baseline policy inside the MCTS, which needs to extend the action selection strategy (UCT and rollout) to guarantee the safety of the improvement.

3. Background

We describe MDPs, SPIBB and MCTS, and introduce the mathematical notation used in the following sections.

3.1. Markov Decision Processes

A Markov Decision Process (MDP; Puterman, 2014) is a tuple $M = \langle S, A, T, R, \gamma\rangle$, where S is a finite set of *states*, A is a finite set of *actions* (we represent each action with its index, i.e., $A = \{1, \dots, |A|\}$), $T : S \times A \to \mathcal{P}(S)$ is a stochastic *transition function*, where $\mathcal{P}(E)$ denotes the space of probability distributions over the finite set E, therefore $T(s, a, s')$ indicates the probability of reaching the state $s' \in S$ after executing $a \in A$ in $s \in S$, $R : S \times A \to [-R_{max}, R_{max}]$ is a bounded stochastic *reward function*, and $\gamma \in [0, 1)$ is a *discount factor*. The set of stochastic policies for M is $\Pi = \{\pi : S \to \mathcal{P}(A)\}$.

Given an MDP M and a policy π we can compute state values $V^\pi_M(s), s \in S$, namely, the expected value acquired by π from s; and action values $Q_M(s, a), s \in S, a \in A$, namely, the expected value acquired by π when action a is performed from state s. To evaluate the performance of a policy π in an MDP M, i.e., $\rho(\pi, M)$, we compute its expected return (i.e., its value) in the initial state s_0, namely, $\rho(\pi, M) = V^\pi_M(s_0)$. The goal of MDP solvers, such as value iteration and policy iteration (Russell & Norvig, 2020; Sutton & Barto, 2018), is to compute optimal policies, namely, policies having maximal values (i.e., expected return) in all their states. Given a policy subset $\Pi' \subseteq \Pi$, a policy π' is said to be Π'-optimal for an MDP M when it has maximal performance among the policies in Π', that is $\rho(\pi', M) = \max_{\pi\in\Pi'} \rho(\pi, M)$. We use V_{max} to denote the known upper bound of the return's absolute value, i.e., $V_{max} \leq \frac{R_{max}}{1-\gamma}$.

3.2. Safe Policy Improvement with Baseline Bootstrapping

Let us represent the true environment by an MDP $M^* = \langle S, A, T^*, R, \gamma\rangle$ with unknown transition model T^* and known reward function R. This is common in real-world domains where the transition model must be estimated or inferred from small amounts of data. Safe policy improvement (SPI) aims to compute a new policy π_I that outperforms a baseline policy π_0 using a *dataset of trajectories* $\mathcal{D} = \langle s_j, a_j, r_j, s'_j\rangle_{j\in[1,N]}$ collected using π_0 in the true environment. The dataset is used to compute the Maximum Likelihood Estimation (MLE) MDP $M^{\mathcal{D}} = \langle S, A, T^{\mathcal{D}}, R, \gamma\rangle$ where $T^{\mathcal{D}}$ is the transition statistics observed in the dataset. The *safety* of the improvement must be guaranteed, namely, π_I must outperform π_0 with an admissible performance loss $\zeta \in \mathbb{R}^+$ and confidence level $1 - \delta$, with $0 \leq \delta \leq 1$ and loss $\rho(\pi_0, M) - \rho(\pi_I, M)$.

SPI was formalized on MDPs using different *percentile criteria* (Delage & Mannor, 2010) (notice that a percentile criterion was first proposed in 2007 for exploration (Delage & Mannor, 2007) and in 2008 for safety (Schneegass et al., 2008)). Safe Policy Improvement with Baseline Bootstrapping (SPIBB) considers the *worst-case scenario* (Petrik et al., 2016) namely, $\pi_I = \arg\max_{\pi\in\Pi} \min_{M\in\Xi}(\rho(\pi, M) - \rho(\pi_0, M))$, where Ξ is the set of admissible MDPs $\Xi(M^{\mathcal{D}}, e) = \{M = \langle S, A, R, T, \gamma\rangle \mid \forall(s, a) \in S \times A, \|T(s, a, \cdot) - T^{\mathcal{D}}(s, a, \cdot)\|_1 \leq e(s, a)\}$. The MDPs in this set have transition model T with L_1 distance from the transition model $T^{\mathcal{D}}$ estimated from data $\mathcal{D}$ smaller than the error $e(s, a)$. The error function $e : S \times A \to \mathbb{R}$ is an arbitrary function representing the uncertainty over the estimated transition model $T^{\mathcal{D}}$. The optimization of π_I is, however, NP-hard.

The main contribution of SPIBB (Laroche et al., 2019) is to reformulate the percentile criterion to make the search of an efficient and provably-safe policy tractable. Let $N_{\mathcal{D}}(s, a) \in \mathbb{N}$ be the number of occurrences of a state-action pair (s, a) in $\mathcal{D}$ and $N_\wedge \in \mathbb{N}$ a related threshold. SPIBB splits state-action pairs in two subsets: the *bootstrapped subset* $\mathcal{B} = \{(s, a) : N_{\mathcal{D}}(s, a) < N_\wedge\}$ is the set of state-action pairs that occur less than $N_\wedge$ times in $\mathcal{D}$; the non-bootstrapped set $\overline{\mathcal{B}} = \{(s, a) \colon N_{\mathcal{D}}(s, a) \geq N_\wedge\}$ is the set of state-action pairs that occur at least $N_\wedge$ times in $\mathcal{D}$. We also define bootstrapped and non-bootstrapped action sets, for each state s, as functions $\mathcal{B}_A(s) = \{a \in A \colon (s, a) \in \mathcal{B}\}$ and $\overline{\mathcal{B}}_A(s) = \{a \in A \colon (s, a) \in \overline{\mathcal{B}}\}$, respectively. SPIBB policies satisfy the *SPIBB constraint* $\pi^{spibb}(s, a) = \pi_0(s, a)$, if $(s, a) \in \mathcal{B}$. The approach guarantees that π^{spibb} is a ζ-approximate safe policy improvement of the baseline π_0 with high probability $1-\delta$, where ζ depends on $N_\wedge$ and δ. The search is done in the subspace of policies equal to the baseline in the state-action pairs not observed enough times in $\mathcal{D}$, namely, $\Pi_0 = \{\pi \in \Pi : \pi(s, a) = \pi_0(s, a) : \forall(s, a) \in \mathcal{B}\}$. The SPIBB algorithm (Algorithm 1; Laroche et al., 2019) performs policy iteration (Sutton & Barto, 2018) with transition model $T^{\mathcal{D}}$ and constraining the policy to Π_0 at each policy improvement step.

3.3. Monte Carlo Tree Search

Given the current state of the agent, MCTS (Browne et al., 2012) first generates a Monte Carlo tree rooted in the state to estimate in a sample-efficient way the Q-values for that state. Then, it uses these estimates to select the best action. We perform $m \in \mathbb{N}$ simulations using, at each step, Upper Confidence Bound applied to Trees (Kocsis & Szepesvári, 2006) (inside the tree) or a rollout policy (from a leaf to the end of the simulation) to select the action, and the known transition model (or an equivalent simulator) to perform the step from one state to the next. Simulations allow to update two node statistics, namely, the average discounted return $Q(s, a)$ obtained selecting action a and the number of times $N(s,a)$ action a was selected from node (state) s. UCT extends UCB1 (Auer et al., 2002) to sequential decisions and allows to balance exploration and exploitation in the simulation steps performed inside the tree, and to find the optimal action as m tends to infinity. Given the average return $\bar{X}_{a,T_a(t)}$ of each action $a \in A$ of a node, where $T_a(t)$ is the number of times action a has been selected up to simulation t from that node, UCT selects the action with the best upper confidence bound. In other words, the index of the action selected at the t-th visit of a node is $I_t = \operatorname{argmax}_{a \in 1,\dots,|A|} \ \bar{X}_{a,T_a(t)} + 2C_p\sqrt{\frac{\ln(t-1)}{T_a(t-1)}}$, with appropriate constant $C_p > 0$. When all m simulations are performed the action a with maximum average return $\bar{X}_{a,T_a(t)}$ in the root is executed in the real environment.

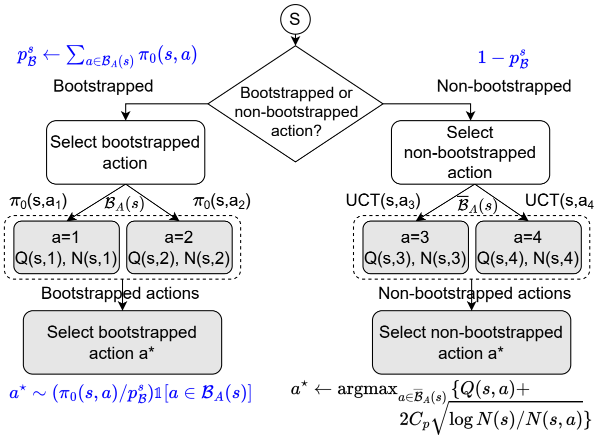
S
$p^s_{\mathcal{B}} \leftarrow \sum_{a \in \mathcal{B}_A(s)} \pi_0(s,a)$
Bootstrapped
Bootstrapped or non-bootstrapped action?
$1 - p^s_{\mathcal{B}}$
Non-bootstrapped
Select bootstrapped action
Select non-bootstrapped action
π_0(s,a$_1$)
$\mathcal{B}_A(s)$
π_0(s,a$_2$)
UCT(s,a$_3$)
$\overline{\mathcal{B}}_A(s)$
UCT(s,a$_4$)
a=1 Q(s,1), N(s,1)
a=2 Q(s,2), N(s,2)
a=3 Q(s,3), N(s,3)
a=4 Q(s,4), N(s,4)
Bootstrapped actions
Non-bootstrapped actions
Select bootstrapped action a*
Select non-bootstrapped action a*
$a^\star \sim (\pi_0(s,a)/p^s_{\mathcal{B}})\mathbb{1}[a \in \mathcal{B}_A(s)]$
$a^\star \leftarrow \operatorname{argmax}_{a \in \overline{\mathcal{B}}_A(s)} \{Q(s,a) + 2C_p\sqrt{\log N(s)/N(s,a)}\}$

Figure 1. Action selection strategy of MCTS-SPIBB: flow chart.

4. Method

The MCTS-SPIBB algorithm is first presented from a high-level perspective and then in detail.

4.1. MCTS-SPIBB: Overview

MCTS-SPIBB is a Monte Carlo Tree Search extension of SPIBB (Laroche et al., 2019). As MCTS approximates optimal policies generated by policy iteration, so MCTS-SPIBB approximates Π_0-optimal policies generated by SPIBB starting from a baseline π_0. The MCTS-SPIBB policy with infinite simulations is Π_0-optimal in the Maximum Likelihood Estimate (MLE) transition model $T^{\mathcal{D}}$ (Theorem 1 by Laroche et al., 2019) and a ζ-approximate safe policy improvement over π_0 (Theorem 2 by Laroche et al., 2019). Since MCTS-SPIBB computes the policy online and locally it can scale to larger problems than SPIBB. The convergence of MCTS-SPIBB to the optimal policy in Π_0 w.r.t. $M^{\mathcal{D}}$ and the capability of MCTS-SPIBB to scale better than SPIBB are two key contributions shown in Sections 5 (theoretical analysis) and Section 6 (empirical analysis), respectively.

The main idea behind MCTS-SPIBB is to extend UCT considering the constraint on bootstrapped actions. This is non-trivial for several reasons, e.g., UCT selects actions according to Q-values and the constraint is on action selection probabilities, and the effect of the constraint accumulates in the layers of the MC tree. Bootstrapped actions must be selected with probability $\pi_0(s,a)$ during the simulations to generate optimal policies in Π_0. Figure 1 shows a diagram that highlights the key idea behind this extension. Given a state s in the MC tree, we split the actions into bootstrapped state-action pairs $(s,a) \in \mathcal{B}$ and non-bootstrapped state-action pairs, $(s,a) \in \overline{\mathcal{B}}$ (i.e., respectively, actions a_1 and a_2, and actions a_3 and a_4 in Figure 1). When the simulation reaches state s, we select a bootstrapped action with probability $p^s_{\mathcal{B}} = \sum_{a \in \mathcal{B}_A(s)} \pi_0(s,a)$, where $\mathcal{B}_A(s)$ is the set of bootstrapped actions for state s, and a non-bootstrapped action with probability $p^s_{\overline{\mathcal{B}}} = 1 - p^s_{\mathcal{B}}$. In the first case, we choose the specific action according to the probability distribution $\pi_0(s,\cdot)$. In the second case, we choose the specific action according to the UCT strategy, which considers the current estimates of Q-values and visit counts (respectively, $Q_3(s,a)$, $Q_4(s,a)$ and $N_3(s,a)$, $N_4(s,a)$ in Figure 1) and guarantees to select the optimal action with enough simulations. In rollout, baseline probabilities are used for bootstrapped actions and uniform selection for non-bootstrapped actions. At the end of the simulations, the estimated Q-values $Q(s,a)$ of the root state s are used to compute the probabilities of the improved policy $\pi^\circ(s,a)$ as *i)* $\pi_0(s,a)$ if $a \in \mathcal{B}(s)$, *ii)* $1 - p^s_{\mathcal{B}}$ if $a = \underset{a' \in \overline{\mathcal{B}}_A(s)}{\operatorname{argmax}}\, Q(s,a')$, *iii)* 0 otherwise.

4.2. MCTS-SPIBB: Algorithm

Algorithms 1-4 show the pseudocode of MCTS-SPIBB. The differences w.r.t. standard MCTS are highlighted in blue. The agent is in a state s, and a baseline policy π_0 is available together with a dataset of trajectories $\mathcal{D}$ generated using π_0 on the real environment M^*. From dataset $\mathcal{D}$ we assume to have computed the state-action pair counts matrix $N_{\mathcal{D}}(s,a)$ and the MLE transition model $T^{\mathcal{D}}$. Other mathematical symbols present in the algorithms and already defined above are R, γ, $N_\wedge$, m, and the threshold ϵ used to end simulation steps. For the sake of compactness, these elements are used by the algorithms although not explicit input parameters.

MCTS-SPIBB (**Algorithm 1**) first generates the Monte Carlo tree *Tr* for state s performing m simulations (lines 3-5). Then it produces the improved policy $\pi^\circ(s,a)$ setting *i)* the probabilities of bootstrapped actions to the related baseline probabilities (lines 7-9), *ii)* the probability of the best non-bootstrapped action $a^\star$ to the total probability available for non-bootstrapped actions (lines 10-12), *iii)* the probability of other non-bootstrapped actions to zero (line 6).

Algorithm 1 MCTS-SPIBB

```
Input: $s$: current state; $\pi_0$: baseline policy; $N_\wedge$: minimum
    count; $N_\mathcal{D}$: counter; $m$: total number of simulations;
    $\mathcal{B}_A(s)$, $\overline{\mathcal{B}}_A(s)$: bootstrapped/non-bootstrapped actions
Tr $\leftarrow$ {} // Empty MC tree
// Build MC tree (i.e., compute $Q(s,a)$)
for $i = 1, \cdots, m$ do
    Simulate(Tr, $s, 0, \pi_0, \mathcal{B}_A(s), \overline{\mathcal{B}}_A(s)$)
end for
$\pi^\circ(s,\cdot) \leftarrow (0,\cdots,0)$ // Initialize MCTS-SPIBB policy
for $a \in \mathcal{B}_A(s)$ do
    $\pi^\circ(s,a) \leftarrow \pi_0(s,a)$
end for
$a^\star \leftarrow \text{argmax}_{a \in \overline{\mathcal{B}}_A(s)}$ {Tr.Q(s,a)}
$p^s_\mathcal{B} \leftarrow \sum_{a \in \mathcal{B}_A(s)} \pi_0(s,a)$ // $\mathcal{B}_A(s)$ total probability
$\pi^\circ(s,a^\star) \leftarrow 1 - p^s_\mathcal{B}$
return $a \sim \pi^\circ(s,\cdot)$
```

Algorithm 2 Simulate

```
Input: Tr: MC tree structure; $s$: state node; $d$: current depth; $\pi_0$: baseline policy; $\mathcal{B}_A(s)$, $\overline{\mathcal{B}}_A(s)$:
    bootstrapped/non-bootstrapped actions
if $\gamma^d < \varepsilon$ then
    return 0
end if
// Node expansion
if $s \notin$ Nodes then
    for $a \in \mathcal{A}$ do
        Nodes($sa$) $\leftarrow (N_\text{init}(s,a), Q_\text{init}(s,a), \emptyset)$
    end for
    return Rollout($s, d, \pi_0, \mathcal{B}_A(s), p^s_\mathcal{B}$)
end if
$p^s_\mathcal{B} \leftarrow \sum_{a \in \mathcal{B}_A(s)} \pi_0(s,a)$ // Tot prob bootstrapped act
$a^\star \leftarrow$ SelectAction($s, \mathcal{B}_A(s), \overline{\mathcal{B}}_A(s), \pi_0, p^s_\mathcal{B}, False$)
$s' \sim T^\mathcal{D}(s,a^\star,\cdot)$;  $r \leftarrow R(s,a^\star)$
$R \leftarrow r + \gamma \cdot$Simulate(Tr, $s', d+1, \pi_0, \mathcal{B}_A(s'), \overline{\mathcal{B}}_A(s')$)
$N(s) \leftarrow N(s) + 1$
$N(s,a^\star) \leftarrow N(s,a^\star) + 1$
$Q(s,a^\star) \leftarrow Q(s,a^\star) + \frac{(R - Q(s,a^\star))}{N(s,a^\star)}$
return $R$
```

Algorithm 3 SelectAction

```
Input: $s$: state node; $\mathcal{B}_A(s)$, $\overline{\mathcal{B}}_A(s)$: bootstrapped/non-
    bootstrapped action sets; $\pi_0$: baseline policy; $p^s_\mathcal{B}$: total
    probability of bootstrapped actions; $roll$: rollout flag
1:  $\theta \sim \mathcal{U}([0,1])$ // Uniform sampling from $[0,1]$
2:  if $\theta \le p^s_\mathcal{B}$ then
3:      $p(\cdot) \leftarrow (0,\cdots,0)$ // Init. bootstrapped probabilities
4:      for $a \in \mathcal{B}_A(s)$ do
5:          $p(a) \leftarrow \pi_0(s,a)/p^s_\mathcal{B}$
6:      end for
7:      $a^\star \sim p(\cdot)$ // Sample bootstrapped action
8:  else
9:      if $\neg roll$ then
10:         // Sample non-bootstrapped action using UCT
11:         $a^\star \leftarrow \text{argmax}_{a \in \overline{\mathcal{B}}_A(s)} \{Q(s,a) + 2C_p\sqrt{\frac{\log N(s)}{N(s,a)}}\}$
12:     else
13:         $a^\star \sim \pi_{rollout}(s,\cdot)$ // Sample uniformly
14:     end if
15: end if
16: return $a^\star$
```

Algorithm 4 Rollout

```
Input: $s$ : state node; $d$: current depth; $\pi_0$: baseline policy;
    $\mathcal{B}_A(s)$: bootstrapped action set; $p^s_\mathcal{B}$: total probability of
    bootstrapped actions
if $\gamma^d < \varepsilon$ then
    return 0
end if
$a^\star \leftarrow$ SelectAction($s, \mathcal{B}_A(s), \{\}, \pi_0, p^s_\mathcal{B}, True$)
$s' \sim T^\mathcal{D}(s,a^\star,\cdot)$;  $r \leftarrow R(s,a^\star)$
return $r + \gamma \cdot \text{Rollout}(s', d+1, \pi_0, \mathcal{B}_A(s), p^s_\mathcal{B})$
```

Finally, it randomly samples an action from $\pi^\circ(s,\cdot)$ and returns it. Simulations (**Algorithm 2**) are performed using almost a standard MCTS strategy. Steps are performed using the MLE transition model T^D and the simulator is set up as a standard MCTS simulator. **Algorithm 3** selects actions according to the *strategy described in Subsection 4.1*. It first decides whether to bootstrap or not considering the total probability of bootstrapped actions $p^s_\mathcal{B}$ (lines 1-2). If it decides to bootstrap, it samples the action according to the probability distribution of those actions (lines 3-7). Otherwise, it samples the action according to standard UCT if the step is performed inside the tree (lines 9-11), or according to the rollout policy (we used a uniform policy in our tests) if the step is performed outside the tree (line 13). Finally, **Algorithm 4** performs the standard MCTS rollout using the new function for action selecting (i.e., Algorithm 3). *What differentiates MCTS-SPIBB from the standard MCTS algorithm is the way in which actions are selected both inside the tree and during rollouts.*

There are two non-trivial parts in the integration of SPIBB with MCTS. The first concerns the design of the action selection strategy (Figure 1); the second is the theoretical proof that this strategy, which merges UCT with baseline policy, actually provides a safe improvement (see next section and Appendix A). The way in which, given a state s, we first decide if to select a bootstrapped or a non-bootstrapped action, and then we use baseline probabilities (bootstrapped case) or Q-value estimates (non-bootstrapped case) to select actions, guarantees MCTS-SPIBB convergence to SPIBB. This would not have been ensured by other strategies.

5. Theoretical Analysis

We prove that, given a baseline π_0, the improved policy π° generated by MCTS-SPIBB converges, as the number of simulations tends to infinity, to the improved policy π^{spibb} generated by SPIBB, which is optimal in Π_0 and a safe improvement of π_0. Using the notation of Kocsis et al. (2006) and Auer et al. (2002), which is derived from multi-armed bandit theory, we indicate with $X_{i,t}$ the (random) payoff (i.e., return) obtained by selecting action $i \in A$ in the t-th simulation passing from the current state. The average payoff of action i after m simulations passing from the current state is indicated as $\bar{X}_{i,m} := \frac{1}{m}\sum_{t=1}^{m} X_{i,t}$ and the expected average payoff (i.e., the expected average return) of the current state after m simulations passing through it is indicated as $\mathbb{E}\{\bar{X}_m\}$. Among m simulations, n are assumed to select a non-bootstrapped action and c a bootstrapped action. $T_i(m)$ denotes the number of times action i was selected after m simulations (notation in Appendix A.1.1).

The analysis is based on an assumption and three theorems. For the sake of compactness, the full assumption is reported in **Assumption A.5** (see Appendix A) and summarized here: without loss of generality, the expected value of the average payoff of each action $i \in A$ converges to some value $\mu_i \in \mathbb{R}$; payoffs $X_{i,t}$ are limited to range $[0, 1]$; the probability distributions over average payoffs concentrate quickly around their means, according to the Hoeffding inequality.

The first theorem provides a bound on the bias of the estimated value (i.e., expected average return) for the current state after m simulations $\mathbb{E}\{\bar{X}_m\}$, given the bias on the estimated value of the optimal non-bootstrapped action $\delta^*_n := \mathbb{E}\{\bar{X}_{i^*,n}\} - \mu^*$ (with μ^* expected value of the average payoff of the optimal non-bootstrapped action) and the biases on the estimated values of the bootstrapped actions $\delta_{i,T_i(c)} := \mathbb{E}\{\bar{X}_{i,T_i(c)}\} - \mu_i$ (with $i \in \mathcal{B}_A(s)$, μ_i expected value of the average payoff of bootstrapped action i). $N_0(\epsilon)$ is such that if $t \geq N_0(\epsilon)$ then $|\delta_{i,t}| \leq \epsilon\Delta_i/2$ and $|\delta_{j^*,t}| \leq \epsilon\Delta_i/2$, where $\Delta_i = \mu^* - \mu_i$, with i suboptimal action and j^* optimal action.

Theorem 5.1. *Let* $\bar{X}_m = \frac{1}{m}\sum_{i\in\mathcal{B}_A} T_i(c)\bar{X}_{i,T_i(c)} + \frac{1}{m}\sum_{i\in\overline{\mathcal{B}}_A} T_i(n)\bar{X}_{i,T_i(n)}$. *Under Assumption A.5 the following bound holds* $\left|\mathbb{E}\{\bar{X}_m\} - \sum_{i\in\mathcal{B}_A} \pi_0(i)\cdot\mu_i - p_{\overline{\mathcal{B}}}\cdot\mu^\star\right| \leq \sum_{i\in\mathcal{B}_A} \pi_0(i)\cdot|\delta_{i,T_i(c)}| + p_{\overline{\mathcal{B}}}\cdot|\delta^*_n| + O\left(\frac{K(C_p^2 \ln m + N_0)}{m}\right)$ *where* $N_0 = N_0(\epsilon)$.

Proof. (Sketch) The idea behind the proof is to split the bias on the expected payoff of the state (left side of the inequality) into the bias of the expected payoff of the optimal non-bootstrapped action and the bias of each bootstrapped action. The first is bounded by Theorem 3 of (Kocsis et al., 2006) and the second by $\sum_{i\in\mathcal{B}_A} \pi_0(i)\cdot|\delta_{i,T_i(c)}|$. Theorem A.6 (see Appendix A) provides the full derivation. □

The second theorem proves the convergence of the estimated state value $\mathbb{E}\{\bar{X}_m\}$ to the optimal value in Π_0, which is the value of the optimal policy in Π_0 computed by SPIBB.

Theorem 5.2. *Consider algorithm MCTS-SPIBB running on a tree of depth* D*, branching factor* $|A| = L + K$ *with* L *bootstrapped actions and* K *non-bootstrapped actions in each state, and stochastic payoffs at the leaves. Assume that payoffs lie in* $[0, 1]$*. Then the bias of the estimated expected payoff* $\bar{X}_m$ *is* $O\left(\frac{L^D KD \ln m + L^D K^D}{m}\right)$.

Proof. (Sketch) The proof is made by induction on D considering the bound of Theorem 5.1 to perform the inductive step from $L + K$ MC trees of depth $D - 1$ (one tree for each non-bootstrapped action and one tree for each bootstrapped action) to a MC tree of depth D. Theorem A.7 (see Appendix A) provides the full derivation. □

The third theorem proves that the estimated state value concentrates quickly around its mean.

Theorem 5.3. *Fix an arbitrary* $\delta > 0$ *and let* $\Delta_m = 9\sqrt{2p^s_{\overline{\mathcal{B}}} m \ln(4/\delta)} + C_p\sqrt{m\ln(2L/\delta)}\sum_{i\in\mathcal{B}_A}\sqrt{\pi_0(i)}$. *Let* $n_0 \in \mathbb{N}$ *be such that* $\sqrt{n_0} \geq O(K(C_p^2 \ln n_0 + N_0(1/2)))$, *if* $m \geq \frac{n_0}{p_{\overline{\mathcal{B}}}}$ *then under Under Assumption A.5 the following bounds hold:* $\mathbb{P}\left\{m\bar{X}_m \geq m\mathbb{E}\left\{\bar{X}_m\right\} + \Delta_m\right\} \leq \delta$ *and* $\mathbb{P}\left\{m\bar{X}_m \leq m\mathbb{E}\left\{\bar{X}_m\right\} - \Delta_m\right\} \leq \delta$.

Proof. (sketch) The proof is obtained by splitting each probability $\mathbb{P}\{\cdot\}$ into two terms, one for non-bootstrapped actions and one for bootstrapped actions. Bounds on the probability distribution of average payoffs in Assumption A.5 and the Hoeffding inequality allow to prove the theorem. Details are reported in Theorem A.8 (see Appendix A). □

This theoretical analysis shows the safety of MCTS-SPIBB by demonstrating the convergence of the policy generated by MCTS-SPIBB to the policy generated by SPIBB (which is proved to be safe in (Laroche et al., 2019)). Furthermore, SPIBB computes an optimal policy in Π_0 (Theorem 1, Laroche et al., 2019), namely, it solves in an optimal way the problem of SPI with baseline bootstrapping (satisfying the percentile criterion of Eq. 1 by Laroche et al., 2019).

6. Experiments

We first apply MCTS-SPIBB to two benchmark domains, Gridworld and SysAdmin, showing empirically that *i)* the performance of the improved policy generated by MCTS-SPIBB converges to that of the policy generated by SPIBB as the number of simulations increases, *ii)* MCTS-SPIBB guarantees the safety of the improvement, iii) MCTS-SPIBB can scale to larger domains than SPIBB. To further extend our analysis, we also compare MCTS-SPIBB with state-of-the-art SPI algorithms (i.e., SPIBB, some SPIBB extensions presented by Scholl et al. (2022b), Basic-RL, DUIPI (Schneegass et al., 2010), R-Min (the pessimistic adaptation of R-Max, Brafman & Tennenholtz, 2003), and RaMDP (Petrik et al., 2016)) on a public benchmark based on the WetChicken domain proposed by Scholl et al. (2022b). This analysis shows that MCTS-SPIBB reaches state-of-the-art performance on small domains and scales to large domains.

Domains. In *GridWorld* (Laroche et al., 2019) an agent moves in a $N \times N$ grid starting from the bottom-left corner and aiming to reach the top-right corner. The agent can select four actions, i.e., moving north, south, east, or west. Each action has a 75% chance of moving the agent in the desired direction, 5% in the opposite direction, and 10% chance of moving it in each of the other two directions. The reward is 1 if the agent reaches the target cell (top-right corner), 0 otherwise. The size of the state space is $|S| = N^2$ and that of the action space is $|A| = 4$, resulting in $4N^2$ possible state-action pairs. In *SysAdmin* (Guestrin et al., 2003) an agent has to administer a network of N machines. Each machine is connected to two other machines on either side of it to form a ring topology. A binary random variable represents whether each machine is working or has failed. At each time step, the agent can reboot one machine at a cost of -1 or do nothing with null cost, furthermore, it receives a reward of 1 for each working machine and a penalty of -1 for each failed machine. Every machine has a probability of 0.05 to fail at each time step. This probability increases by 0.3 for each neighboring machine that failed. If a machine is rebooted, then it works with probability 1. The size of the state space is $|S| = 2^N$, and the size of the action space $|A| = N + 1$, resulting in $2^N(N + 1)$ state-action pairs. We chose these two benchmark domains to evaluate MCTS-SPIBB because, *i)* they are well-known domains on which also other SPI algorithms have been tested (e.g., Gridworld, Laroche et al., 2019), *ii)* the state space of these domains can be enlarged at will (e.g., the largest SysAdmin instance tested has 35 machines, namely, about 35 billion states). The benchmark proposed by Scholl et al. (2022b) is executed on the *WetChicken* domain: an agent floats in a small boat on a river with a waterfall at one end. The goal for the agent is to stay as close as possible to the waterfall without falling down. The closer the agent is to the waterfall, the greater the reward. If it falls, the episode ends. The river is represented as a 5×5 grid in the benchmark (i.e., $|S| = 25$) and five actions can be chosen by the agent (Appendix C).

Software and Hardware. The original code of SPIBB[1] and our code of MCTS-SPIBB[2] are publicly available. Experiments were performed on a laptop with an 11th Gen Intel(R) Core(TM) i7-1165G7, 2.80 GHz with 10 GB RAM.

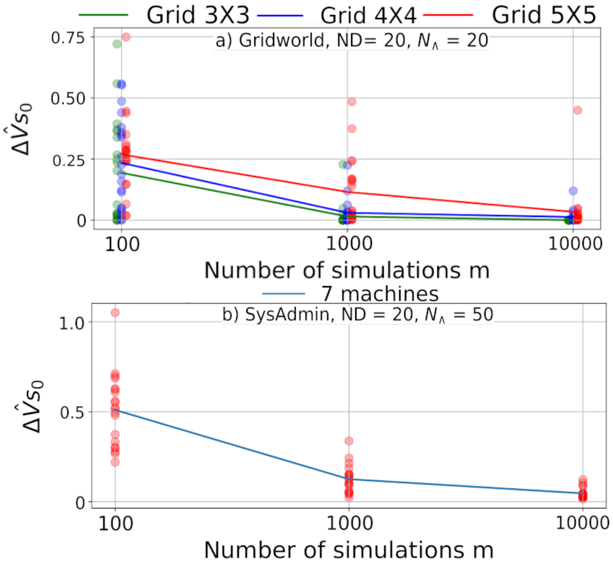

Figure 2. Results on convergence. X-axis: number of simulations m. Y-axis: absolute difference of values in s_0 between MCTS-SPIBB and SPIBB $\Delta V_{s_0} = |V_{M^*}^{\pi^\circ}(s_0) - V_{M^*}^{\pi^{spibb}}(s_0)|$.

Results on Convergence. To show that the performance of the policy generated by MCTS-SPIBB converges to that of the policy generated by SPIBB as the number of simulations m increases, we perform experiments on Gridworld 3x3, 4x4, 5x5, and SysAdmin with 7 machines. For each domain, we first generate a baseline policy (details in Appendix B). Then, we generate $ND = 20$ datasets each containing $|\mathcal{D}| = 10000$ trajectories for Gridworld (trajectory lengths are 15 steps for 3x3, 20 steps for 4x4 and 30 steps for 5x5) and $|\mathcal{D}| = 5000$ trajectories for SysAdmin (trajectory length is 15 steps). Then, for each dataset, we compute the MLE transition model $T^{\mathcal{D}}$, the state-action pair count matrix $N_{\mathcal{D}}(s, a)$ and the bootstrapped/non-bootstrapped action sets $\mathcal{B}_A(s)/\overline{\mathcal{B}}_A(s)$ using threshold $N_\wedge = 5$ for Gridworld (average % of safe actions is $|\overline{\mathcal{B}}_A(s)|/|A| \cdot 100 = 81\%$) and $N_\wedge = 50$ for SysAdmin (avg % of safe actions: 13.4%). Finally, for each dataset we generate the improved policy with both SPIBB and MCTS-SPIBB and compute the absolute difference between their values in the initial state s_0, that is $\Delta V_{s_0} = |V_{M^*}^{\pi^\circ}(s_0) - V_{M^*}^{\pi^{spibb}}(s_0)|$ (notice that in this test we compute the entire policy (all states) also with MCTS-SPIBB, and evaluate it using policy evaluation). Figure 2 shows the value of ΔV_{s_0} (y-axis) for each domain (Fig. 2.a for Gridworld and Fig. 2.b for SysAdmin) and for each dataset (each point is a dataset) with $m = 100, 1000, 10000$ simulations (x-axis). Lines connect average values. In both domains avg ΔV_{s_0} tends to zero showing the convergence.

[1] https://github.com/RomainLaroche/SPIBB
[2] https://github.com/Isla-lab/mctsspibb

Results on Safety. To show the practical impact in the safety of policies generated by MCTS-SPIBB we perform the same experiments performed by Laroche et al. (2019) on Gridworld 5x5 and SysAdmin with 7 machines. Namely, for each domain, we first generate a baseline policy. Then, for different dataset sizes $|\mathcal{D}|$ we generate $ND = 20$ datasets, each containing $|\mathcal{D}|$ trajectories. In particular, in Gridworld $|\mathcal{D}| \in \{2, 10^1, 10^2, 10^3, 10^4\}$ and each trajectory is 30 steps long, while in SysAdmin $|\mathcal{D}| \in \{5, 500, 5000\}$ and each trajectory is 15 steps long. Afterward, for each dataset, we compute $T^{\mathcal{D}}$, $N_{\mathcal{D}}(s, a)$ and sets $\mathcal{B}_A(s)/\overline{\mathcal{B}}_A(s)$ using threshold $N_\wedge = 20$ for Gridworld and $N_\wedge = 50$ for SysAdmin (average % of safe actions for each $|\mathcal{D}|$ are reported in Figure 3). Finally, for each dataset, we generate the improved policy using three algorithms, namely, MCTS-SPIBB (with 10000 simulations), SPIBB, and Basic RL (Basic RL is the vanilla Batch RL used as a non-safe baseline also by Laroche et al. (2019); it computes the optimal policy in the MLE MDP $M^{\mathcal{D}}$ even when too few samples are available for some state-action pairs) and we evaluate their performance on the real environment as $\rho(\pi_I, M^*) = V^{\pi_I}_{M^*}(s_0)$ (also in this test the policy based on MCTS-SPIBB is computed in all states to allow the usage of policy evaluation to compute values).

Figures 3.a,c show the results on Gridworld and SysAdmyn, respectively. The size of the dataset $|\mathcal{D}|$ is shown on the x-axis and the performance of the improved policy on the y-axis. Each point represents the performance of an improved policy generated using a specific dataset and a specific algorithm. The yellow line represents the performance of the baseline π_0 and the green line is the performance of the optimal policy, namely, the policy computed using policy iteration with the true transition model T^*. Basic RL performs better than MCTS-SPIBB in some cases but it is not safe (we use it only as an unsafe baseline in our tests), in fact, it achieves a performance decrease on small datasets (i.e., $|\mathcal{D}| = 2$ in Gridworld and $|\mathcal{D}| = 5$ in SysAdmin) as it has no safety guarantees. On the contrary, MCTS-SPIBB and SPIBB perform almost identically (differences in performance are not statistically significant) and their performance is always equal or higher than that of the baseline, i.e., they are safe. This behaviour is also shown in Figures 3.b,d considering 15% Conditional Value-at-Risk (15%-CVaR), i.e., the mean performance over the 15% worst runs. For each algorithm-$|\mathcal{D}|$ pair, we select the worst 3 points and draw lines among averages. Interestingly, MCTS-SPIBB and SPIBB are still safe and they perform very similarly. Experiments with different parameters (see Appendix C) confirm the result.

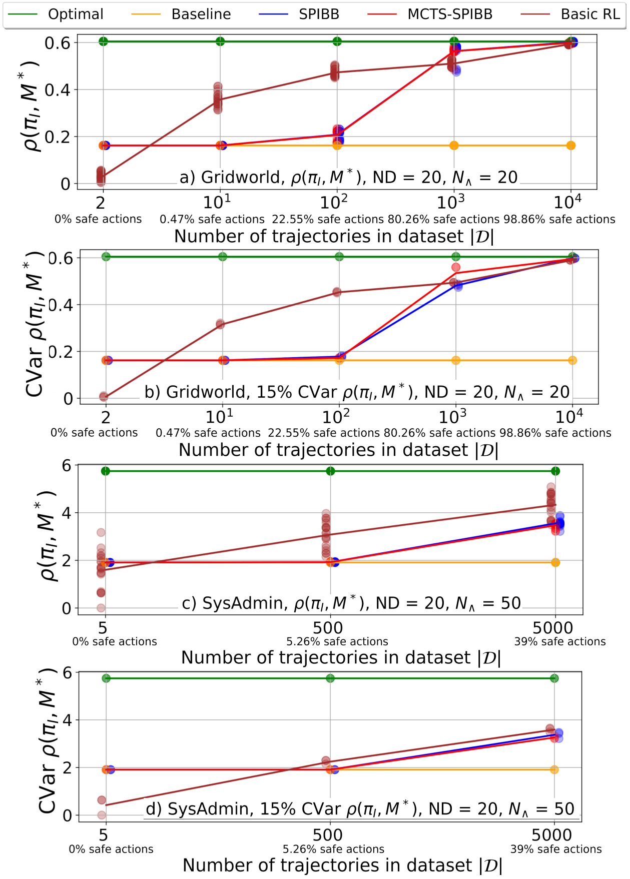

Figure 3. Results on safety. Performance $\rho(\pi_I, M^*)$ and 15%-CVaR $\rho(\pi_I, M^*)$ (y-axis) depending on dataset size $\mathcal{D}$ (x-axis) on Gridworld (a,b) and SysAdmin (c,d).

Results on Scalability. SPIBB complexity is $O(|S \cdot A|^3)$ if the Bellman equation is solved exactly or $O(|S|^2 \cdot |A|)$ if it is solved by dynamic programming. On the other hand, MCTS-SPIBB complexity is $O(m)$, with m number of simulations, and it does not directly depend on the size of the state/action space. This allows MCTS-SPIBB to scale to larger domains than SPIBB. In fact, in domains with a large number of states, each iteration of SPIBB improves all states, although many of them are not reached in real runs. This could require a large time or produce limited improvements. On the other hand, MCTS-SPIBB employs the time available to improve the policy only on the small number of states actually visited, producing larger improvements. To show this in practice, we perform a test on SysAdmin (we selected this domain because its state space scales exponentially with the number of machines) with an increasing number of machines, and compare the time required and the performance achieved by the two algorithms. In particular, we make tests with $|\mathcal{D}| = 5000$ and $N_\wedge = 5$, varying the number of machines (4, 7, 10, 12, 13, 20, 35), which also changes the percentage of safe actions (98%, 49%, 18%, 12%, 8.5%, 6.1%, and 4%, respectively).

Figure 4 shows the time needed by SPIBB (light blue line), SPIBB$_{DP}$ based on dynamic programming (dotted blue

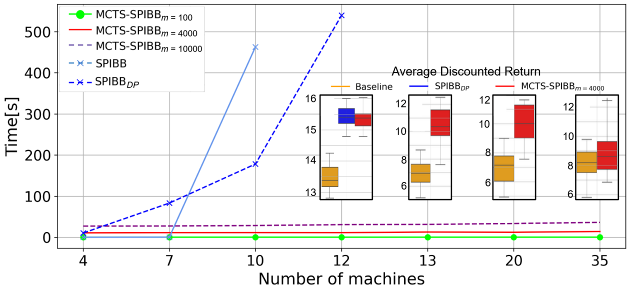

Figure 4. Results on scalability. Lines: computational time of SPIBB, SPIBB$_{DP}$ and MCTS-SPIBB on SysAdmin with 4, 7, 10, 12, 13, 20, and 35 machines. Box plots: performance of baseline policy, SPIBB$_{DP}$ and MCTS-SPIBB on 12, 13, 20, and 35 machines.

line) and MCTS-SPIBB (light green, red, and dotted purple lines). For SPIBB and SPIBB$_{DP}$ we evaluate the time for computing the complete policy (all states, because it is the only output the algorithm can provide), while for MCTS-SPIBB we evaluate the time for computing the policy in a single state (because the algorithm works online on the single states visited in the runs). We notice, again, that in standard applications with large state spaces, the agent only reaches a very small subset of states, hence the total time per episode is in general much lower than the time required to compute the entire policy with SPIBB. We consider $m = 100, 4000, 10000$ simulations. SPIBB manages to compute the policy only until 10 machines, SPIBB$_{DP}$ instead works until 12 machines, whereas MCTS-SPIBB can work even with a larger number of machines. We analyze the performance, in terms of average discounted return, of the improved policy for 13, 20, and 35 machines to see if it actually improves the baseline where SPIBB and SPIBB$_{DP}$ do not work. The box plots in Figure 4 show that the policies computed by MCTS-SPIBB with $m = 4000$ (which requires only about 12 seconds per step) for 13, 20, and 35 machines are actual improvements of the related baseline policies (larger m can be also used). Notice that the performance drop for problems with more machines is justified by the lower percentage of safe actions, which makes the policy more conservative. The average discounted return of each policy is computed performing 20 runs from s_0. Results on 15% CVaR (Appendix C) show that the improvements in all domain sizes are also safe.

Comparison with other state-of-the-art SPI Algorithms. We evaluate the performance of MCTS-SPIBB on the *WetChicken* domain in the context of the benchmark presented by Scholl et al. (2022b). In particular, we compare MCTS-SPIBB with SPIBB, some SPIBB extensions (Scholl et al., 2022b), Basic-RL, DUIPI, R-Min, RaMDP. We compute the performance of each algorithm using different dataset dimensions (i.e., $|\mathcal{D}| = 100, 200, 500, 1000, 2000, 5000$). For each algorithm and each dimension we perform 10000 runs and compute mean performance and 1%-CVaR (Scholl et al., 2022b).

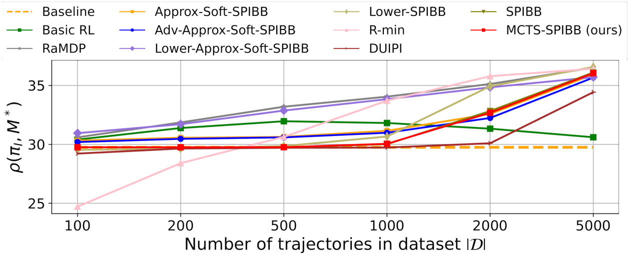

Figure 5. Comparison between MCTS-SPIBB and state-of-the-art algorithms on *WetChicken* (benchmark proposed in (Scholl et al., 2022b)).

Results, displayed in Figure 5 (further details in Figure 8, Appendix C), confirm that MCTS-SPIBB performs similarly to SPIBB if enough simulations are used (we use $m = 1000$ simulations in these tests), hence it also has state-of-the-art performance. Furthermore, all the state-of-the-art algorithms considered in this benchmark fail to scale to grids larger than 60×60 (i.e., $|S| = 3600$ and $|A| = 5$) on our computer for memory issues, while MCTS-SPIBB computes improved policies on larger grids. For instance, Figure 9 in Appendix C shows that MCTS-SPIBB can compute an improved policy on a 70×70 *WetChicken* grid.

7. Conclusions and Future Work

MCTS-SPIBB is a novel approach that uses MCTS to guarantee SPI with Baseline Bootstrapping. The use of MCTS allows to scale to domains significantly larger than the ones addressed by state-of-the-art approaches. The methodological contribution allowing this scaling is a non-trivial extension of UCT that satisfies safety constraints on state-action pairs with low coverage. We proved the convergence of the improved policy generated by MCTS-SPIBB to the policy generated by SPIBB, as the number of simulations grows. Furthermore, we successfully tested the approach on two benchmark domains showing that MCTS-SPIBB can compute the policy on environment sizes not manageable by SPIBB. Future work aims to extend this method to partially observable MDPs and multi-agent MDPs, and to apply it to other (Fu et al., 2020) (also real-world) problems. We will also investigate the possibility to substitute the offline strategy with an online one.

Acknowledgements

This research has been partially funded by NWO grant NWA.1160.18.238 (PrimaVera) and MUR project *Dipartimenti di Eccellenza 2018-2022*.

References

Abbasi-Yadkori, Y., Bartlett, P. L., and Wright, S. J. A fast and reliable policy improvement algorithm. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS), JMLR: W&CP volume 51*, pp. 1338–1346. PMLR, 2016.

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

Boucheron, S., Lugosi, G., and Massart, P. *Concentration Inequalities - A Nonasymptotic Theory of Independence*. Oxford University Press, 2013.

Brafman, R. I. and Tennenholtz, M. R-max - a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3: 213–231, 2003.

Browne, C., Powley, E. J., Whitehouse, D., Lucas, S. M., Cowling, P. I., Rohlfshagen, P., Tavener, S., Liebana, D. P., Samothrakis, S., and Colton, S. A survey of Monte Carlo Tree Search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, 2012.

Castellini, A., Chalkiadakis, G., and Farinelli, A. Influence of state-variable constraints on Partially Observable Monte Carlo planning. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 5540–5546. ijcai.org, 2019.

Chandak, Y., Jordan, S., Theocharous, G., White, M., and Thomas, P. S. Towards safe policy improvement for non-stationary MDPs. In *Proceedings of the 33th Conference on Neural Information Processing Systems (NeurIPS)*, pp. 9156–9168. Curran Associates, Inc., 2020.

Chaslot, G., Bakkes, S., Szita, I., and Spronck, P. Monte Carlo Tree Search: A new framework for game AI. In *Proceedings of the 4th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, (AIIDE), pp. 216–217. AAAI Press, 2008.

Cohen, A., Yu, L., and Wright, R. Diverse exploration for fast and safe policy improvement. In *Proceedings of the 32th AAAI Conference on Artificial Intelligence (AAAI)*. AAAI Press, 2018.

Coulom, R. Efficient selectivity and backup operators in Monte-Carlo Tree Search. In *Computers and Games*, pp. 72–83. Springer Berlin Heidelberg, 2007.

Delage, E. and Mannor, S. Percentile optimization in uncertain Markov decision processes with application to efficient exploration. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pp. 225–232, New York, NY, USA, 2007.

Delage, E. and Mannor, S. Percentile optimization for Markov Decision Processes with parameter uncertainty. *Operations Research*, 58(1):203–213, 2010.

Ernst, D., Geurts, P., and Wehenkel, L. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6:503–556, 2005.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4RL: Datasets for deep data-driven reinforcement learning. *CoRR*, abs/2004.07219, 2020.

Fujimoto, S., Meger, D., and Precup, D. Off-policy deep reinforcement learning without exploration. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pp. 2052–2062. PMLR, 2019.

García, J. and Fernández, F. A comprehensive survey on safe Reinforcement Learning. *Journal of Machine Learning Research*, 16:1437–1480, 2015.

Guestrin, C., Koller, D., Parr, R. E., and Venkataraman, S. Efficient solution algorithms for factored MDPs. *Journal of Artificial Intelligence Research*, 19:399–468, 2003.

Katt, S., Oliehoek, F. A., and Amato, C. Learning in POMDPs with Monte Carlo Tree Search. In *Proceeding of the 34th International Conference on Machine Learning (ICML)*, pp. 1819–1827. PMLR, 2017.

Katt, S., Oliehoek, F. A., and Amato, C. Bayesian reinforcement learning in factored POMDPs. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pp. 7–15. IFAAMAS, 2019.

Kearns, M., Mansour, Y., and Ng, A. Y. A sparse sampling algorithm for near-optimal planning in large Markov decision processes. In *Machine Learning*, 2002.

Kidambi, R., Rajeswaran, A., Netrapalli, P., and Joachims, T. MOReL: Model-based Offline Reinforcement Learning. In *Proceedings of the 33th Conference on Neural Information Processing Systems (NeurIPS)*, pp. 21810–21823. Curran Associates, Inc., 2020.

Kocsis, L. and Szepesvári, C. Bandit based Monte-Carlo planning. In *Machine Learning: ECML 2006. 17th European Conference on Machine Learning*, volume 4212 of *LNCS*, pp. 282–293. Springer-Verlag, 2006.

Kocsis, L., Szepesvari, C., and Willemson, J. Improved Monte-Carlo Search. *University of Tartu, Estonia, Technical Report*, 1, 2006.

Kumar, A., Fu, J., Soh, M., Tucker, G., and Levine, S. Stabilizing off-policy Q-learning via bootstrapping error reduction. In *Proceedings of the 32th Conference on*

Neural Information Processing Systems (NeurIPS), pp. 11761–11771. Curran Ass. Inc., 2019.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative Q-learning for offline reinforcement learning. In *Proceedings of the 33th Conference on Neural Information Processing Systems (NeurIPS)*, pp. 1179–1191. Curran Associates, Inc., 2020.

Lange, S., Gabel, T., and Riedmiller, M. Batch reinforcement learning. In *Reinforcement Learning: State-of-the-Art*, pp. 45–73. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

Laroche, R., Trichelair, P., and Tachet Des Combes, R. Safe policy improvement with baseline bootstrapping. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pp. 3652–3661. PMLR, 2019.

Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline Reinforcement Learning: Tutorial, review, and perspectives on open problems. *CoRR*, abs/2005.01643, 2020.

Mazzi, G., Castellini, A., and Farinelli, A. Identification of unexpected decisions in partially observable Monte-Carlo planning: A rule-based approach. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pp. 889–897. IFAAMAS, 2021a.

Mazzi, G., Castellini, A., and Farinelli, A. Rule-based shielding for partially observable Monte-Carlo planning. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 243–251. AAAI Press, 2021b.

Mazzi, G., Meli, D., Castellini, A., and Farinelli, A. Learning logic specifications for soft policy guidance in POMCP. In *Proceedings of the 22nd International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pp. 373–381. IFAAMAS, 2023.

Nadjahi, K., Laroche, R., and Tachet des Combes, R. Safe policy improvement with soft baseline bootstrapping. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*, pp. 53–68, 2019.

Petrik, M., Ghavamzadeh, M., and Chow, Y. Safe policy improvement by minimizing robust baseline regret. In *Proceedings of the 30th Conference on Neural Information Processing Systems (NeurIPS)*, pp. 2306–2314. Curran Associates Inc., 2016.

Puterman, M. L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 2014.

Russell, S. J. and Norvig, P. *Artificial Intelligence: A Modern Approach (4th Edition)*. Pearson, 2020.

Sarafian, E., Tamar, A., and Kraus, S. Constrained policy improvement for efficient reinforcement learning. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence, (IJCAI)*, pp. 2863–2871. ijcai.org, 2020.

Schneegass, D., Udluft, S., and Martinetz, T. Uncertainty propagation for quality assurance in reinforcement learning. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pp. 2588–2595, 2008.

Schneegass, D., Hans, A., and Udluf, S. *Uncertainty in Reinforcement Learning - Awareness, Quantisation, and Control*, pp. 65–90. InTech, 2010.

Scholl, P., Dietrich, F., Otte, C., and Udluft, S. Safe policy improvement approaches on discrete markov decision processes. *CoRR*, abs/2201.12175, 2022a.

Scholl, P., Dietrich, F., Otte, C., and Udluft, S. Safe policy improvement approaches and their limitations. In *Agents and Artificial Intelligence*, pp. 74–98, Cham, 2022b. Springer International Publishing.

Silver, D. and Veness, J. Monte-Carlo planning in large POMDPs. In *Proceedings of the 23th Conference on Neural Information Processing Systems (NeurIPS)*, pp. 2164–2172. Curran Ass., 2010.

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. P., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

Simão, T. D. and Spaan, M. T. J. Structure learning for safe policy improvement. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 3453–3459. ijcai.org, 2019a.

Simão, T. D. and Spaan, M. T. J. Safe policy improvement with baseline bootstrapping in factored environments. In *the 33th AAAI Conference on Artificial Intelligence (AAAI)*, pp. 4967–4974. AAAI Press, 2019b.

Simão, T. D., Laroche, R., and Tachet des Combes, R. Safe policy improvement with an estimated baseline policy. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pp. 1269–1277. IFAAMAS, 2020.

Simão, T. D., Suilen, M., and Jansen, N. Safe policy improvement for POMDPs via finite-state controllers. In *Proceedings of the 37th AAAI Conference on Artificial Intelligence (AAAI)*. AAAI Press, 2023.

Sutton, R. and Barto, A. *Reinforcement Learning, An Introduction*. MIT Press, 2nd edition, 2018.

Thomas, P. S., Theocharous, G., and Ghavamzadeh, M. High confidence policy improvement. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning (ICML)*, pp. 2380–2388. PMLR, 2015.

Wienhöft, P., Suilen, M., Simão, T. D., Dubslaff, C., Baier, C., and Jansen, N. More for less: Safe policy improvement with stronger performance guarantees. *CoRR*, abs/2305.07958, 2023.

Yu, T., Thomas, G., Yu, L., Ermon, S., Zou, J. Y., Levine, S., Finn, C., and Ma, T. MOPO: Model-based Offline Policy Optimization. In *Proceedings of the 33th Conference on Neural Information Processing Systems (NeurIPS)*. Curran Associates, Inc., 2020.

A. Theoretical Analysis: Full Version

In this appendix, we provide the full version of the theoretical analysis, with complete proofs.

A.1. Convergence of MCTS-SPIBB to SPIBB

The convergence of UCT to an optimal policy given an unconstrained policy space is here implemented in the context of safe policy improvement with baseline bootstrapping, in which the policy has to satisfy the *SPIBB constraint*, namely, the improved policy must belong to the subspace $\Pi_0 = \{\pi \in S \rightarrow \mathcal{P}(A) | \pi(s,a) = \pi_0(s,a) : \forall (s,a) \in \mathcal{B}\}$, with $\mathcal{B}$ set of state-action pairs that occur less than $N_\wedge$ times in dataset $\mathcal{D}$. This constraint requires that for each state encountered in simulations we consider in different ways bootstrapped and non-bootstrapped actions. In particular, non-bootstrapped actions can still be selected using the UCT strategy, which is proved to converge to optimality, while bootstrapped actions (i.e., actions that have not been observed enough times in dataset $\mathcal{D}$) must be selected according to the related baseline probabilities $\pi_0(s,a)$. The bias of the estimated state-value (i.e., expected average payoff) of the root node of a MCTS computed by UCT tends to zero as the number of simulations tends to infinity (Kocsis et al., 2006). In other words, the estimated state-value of the root node tends to its optimal value and the action with maximum Q-value tends to be the optimal one as the number of simulations tends to infinity. Our proof adds to the bias of the estimated state-value (related to UCT for non-bootstrapped actions) the bias due to baseline policy sampling (for bootstrapped actions), and shows that the composed bias still converges to zero as the number of simulations tends to infinity. We notice that the two contributions mix with each other in the lower levels of the MC tree, hence the values of all (bootstrapped and non-bootstrapped) actions of the root node are affected by contributions of bootstrapped and non-bootstrapped actions of the underlying nodes. This makes the analysis non-trivial and of interest.

A.1.1. Notation

We indicate: actions in the set $A = \{1, \ldots, |A|\}$ as $i \in A$ (considering each action as its index in A), bootstrapped actions for the current state as $i \in \mathcal{B}_A$ (for the sake of compactness, we omit from $\mathcal{B}_A(s)$ symbol s for the state when we refer to the current state), and non-bootstrapped actions for the current state as $i \in \overline{\mathcal{B}}_A$. We use L to denote the number of bootstrapped actions for the current state and K to denote the number of non-bootstrapped actions. Being i an action index and t the current simulation index, we indicate with $X_{i,t}$ the (random) payoff (i.e., return) obtained by selecting i in the t-th simulation. Formally, this payoff is a realization of a random variable with a non-stationary distribution (i.e., the distribution can change across epochs t). Given m total simulations, the number of times we have selected action i up to that point is $T_i(m) \coloneqq \sum_{t \in \{1,\ldots,m\}} \mathbb{1}[I_t = i]$, where I_t is a random variable representing the action taken in the t-th simulation. The average payoff after m simulations is indicated as $\bar{X}_{i,m} \coloneqq \frac{1}{m} \sum_{t=1}^{m} X_{i,t}$.

As mentioned above, payoff sequences are non-stationary, namely, taking the same action multiple times can result in payoffs taken from different distributions. Formally, if we fix an action $i \in A$, and a payoff sequence $X_{i,1}, X_{i,2}, \ldots, X_{i,m}$, then $X_{i,t} \sim \mathcal{P}_t$ and $X_{i,t'} \sim \mathcal{P}_{t'}$ with $\mathcal{P}_t$ possibly different from $\mathcal{P}_{t'}$. We indicate with $\mu_{i,m} \coloneqq \mathbb{E}\{\bar{X}_{i,m}\}$ the expected value of the average payoff of action i after m simulations, and with μ_i its value in the limit $\mu_i \coloneqq \lim_{m\to\infty} \mu_{i,m}$ (see Assumption A.5 for its existence). We also define $\delta_{i,m} \coloneqq \mu_{i,m} - \mu_i$ as the difference, at simulation m, between the expectation of the current mean and its value in the limit. For the non-bootstrapped actions, we assume there exists only a single optimal action $i^\star$, as in (Kocsis et al., 2006)[3]. For quantities related to this action we drop the i symbol and use the $\star$ symbol, e.g., $\mu^\star_m, \mu^\star, X^\star_t, \bar{X}^\star_m$. We define the difference between the expected value of the average payoff of the optimal action μ^* and the expected value of the average payoff of another action i, namely μ_i, as $\Delta_i \coloneqq \mu^\star - \mu_i$. This value is always positive. Since $\delta_{i,t}$ converges by assumption to zero, for all $\epsilon > 0$ there exists and index $N_0(\epsilon)$ such that if $t \geq N_0(\epsilon)$ then $|\delta_{i,t}| \leq \epsilon\Delta_i/2$ and $|\delta_{j^*,t}| \leq \epsilon\Delta_i/2$, where i is a suboptimal action and j^* is the optimal action. This says that if enough (i.e., $\geq N_0(\epsilon)$) simulations are run then the estimated expected average payoff of the optimal action will be far enough from the estimated expected average payoffs of all suboptimal actions, hence, in that case, the optimal action can be clearly identified as the action with the highest estimated expected average payoff. In the following, we will arbitrarily use $\epsilon = 1/2$, as in (Kocsis et al., 2006), for which it holds that if $t \geq N_0(1/2)$ then $|\delta_{i,t}| \leq \Delta_i/4$ and $|\delta_{j^*,t}| \leq \Delta_i/4$, therefore each estimated expected average payoff of suboptimal action is farther than $\Delta_i/2$ from the estimated expected average payoff of the optimal action.

[3] Notice that if all actions are bootstrapped then $n = 0$, $c = m$ and $\bar{X}_m = \bar{X}_c$. On the other hand, if all actions are non-bootstrapped then $c = 0$, $n = m$, and $\bar{X}_m = \bar{X}_n$.

Table 1. Summary of mathematical notation.

Symbol	Description
$\mathcal{B}_A$	Set of bootstrapped actions for the current state
$\overline{\mathcal{B}}_A$	Set of non-bootstrapped actions for the current state
L	Number of bootstrapped actions for the current state
K	Number of non-bootstrapped actions for the current state
$X_{i,t}$	Payoff (i.e., return) obtained by selecting action i in the t-th simulation
C_p	Constant regulating exploration-exploitation tradeoff in UCT
m	Number of simulations performed from the current state
c	Number of bootstrapped actions selected from the current state
n	Number of non-bootstrapped actions selected from the current state
$m = c + n$	Relationship between m, c and n
$T_i(m) \coloneqq \sum_{t \in \{1,\dots,m\}} \mathbb{1}[I_t = i]$	Number of times action i was selected in current state after m simulations
$\bar{X}_{i,m} \coloneqq \frac{1}{m} \sum_{t=1}^{m} X_{i,t}$	Average payoff of action i after m simulations from the current state
$\mu_{i,m} \coloneqq \mathbb{E}\{\bar{X}_{i,m}\}$	Expected value of the average payoff of action i after m simulations
$\mu_i \coloneqq \lim_{m \to \infty} \mu_{i,m}$	Expected value of the average payoff of action i in the the limit
$\lvert\delta_{i,m}\rvert \coloneqq \lvert\mu_{i,m} - \mu_i\rvert$	Bias of estimated expected average payoff at simulation m
$\mu_n^\star, \mu^\star, X_t^\star, \bar{X}_n^\star$	Symbols defined above but related to the optimal non-bootstrapped action
$\Delta_i \coloneqq \mu^\star - \mu_i$	Diff. btw exp. val. of avg payoff optimal and another non-bootstrapped act.
$N_0(\epsilon)$	if $t \geq N_0(\epsilon)$ then for non-bootstrapped actions $\lvert\delta_{i,t}\rvert \leq \epsilon\Delta_i/2$ and $\lvert\delta_{j^*,t}\rvert \leq \epsilon\Delta_i/2$
$p_{\mathcal{B}} \coloneqq \sum_{i \in \mathcal{B}_A} \pi_0(i)$	Probability of selecting a bootstrapped action from the current state
$p_{\overline{\mathcal{B}}} \coloneqq 1 - p_{\mathcal{B}}$	Probability to select a non-bootstrapped action from the current state
$\lim_{m \to \infty} \frac{c(m)}{m} = p_{\mathcal{B}}$	Value of c in the limit
$\lim_{m \to \infty} \frac{n(m)}{m} = p_{\overline{\mathcal{B}}}$	Value of n in the limit
$\lim_{m \to \infty} \frac{T_i(m)}{m} = \pi_0(i), \forall i \in \mathcal{B}_A$	Value of $\frac{T_i(m)}{m}$ in the limit
$\lim_{m \to \infty} \frac{T_i(c(m))}{m} = \pi_0(i), \forall i \in \mathcal{B}_A$	Value of $\frac{T_i(c(m))}{m}$ in the limit
$\lim_{c \to \infty} \frac{T_i(c)}{c} = \frac{\pi_0(i)}{p_{\mathcal{B}}}, \forall i \in \mathcal{B}_A$	Value of $\frac{T_i(c)}{c}$ in the limit

We then define the probability of selecting a bootstrapped action from the current state as $p_{\mathcal{B}} \coloneqq \sum_{i \in \mathcal{B}_A} \pi_0(i)$ and the probability to select a non-bootstrapped action as $p_{\overline{\mathcal{B}}} \coloneqq 1 - p_{\mathcal{B}}$. Also in this case we omit the symbol s since we refer to the current state. Over m simulations performed from the current state, we assume c simulations selected a bootstrapped action and n simulations selected a non-bootstrapped action, hence $m = c + n$. For m tending to ∞ we have that c/m tends to $p_{\mathcal{B}}$, hence we write the limit $\lim_{m \to \infty} \frac{c(m)}{m} = p_{\mathcal{B}}$ (symbol $c(m)$ is used only here to emphasize that c depends on m). Similarly, with non-bootstrapped actions, for m tending to ∞ we have that n/m tends to $p_{\overline{\mathcal{B}}}$, hence we write the limit $\lim_{m \to \infty} \frac{n(m)}{m} = p_{\overline{\mathcal{B}}}$. We then represent by $T_i(c)$ the number of times bootstrapped action $i \in \mathcal{B}_A$ has been selected over c selections of bootstrapped actions and by $T_i(n)$ the number of times non-bootstrapped action $i \in \overline{\mathcal{B}}_A$ has been selected over n selections of non-bootstrapped actions. Focusing on single bootstrapped actions we have that $\forall i \in \mathcal{B}_A, T_i(m)/m$ tends to $\pi_0(i)$ as m tends to ∞. Therefore we can also write the limit $\lim_{m \to \infty} \frac{T_i(m)}{m} = \pi_0(i), \forall i \in \mathcal{B}_A$. The limit can also be written as $\lim_{m \to \infty} \frac{T_i(c(m))}{m} = \pi_0(i), \forall i \in \mathcal{B}_A$ since $T_i(m) = T_i(c(m))$ (the first notation refers to the number of times action i is selected over m selections of bootstrapped or non-bootstrapped actions, while the second notation refers to the number of times action i is selected over $c(m)$ selections of bootstrapped actions only, but the number is the same). Finally, the same limit can be written also in terms of c in the following form $\lim_{c \to \infty} \frac{T_i(c)}{c} = \frac{\pi_0(i)}{p_{\mathcal{B}}}, \forall i \in \mathcal{B}_A$. Notice that, the ratio $\frac{\pi_0(i)}{p_{\mathcal{B}}}$ is used in line 5 of Algorithm 3 for selecting bootstrapped actions. Table 1 summarizes the mathematical notation introduced so far and used in the theoretical analysis.

A.1.2. Derivation

The average payoff of the root state of a MC tree considering only non-bootstrapped actions is (Kocsis et al., 2006)

$$\bar{X}_n \coloneqq \frac{1}{n} \sum_{i \in \overline{\mathcal{B}}_A} T_i(n) \bar{X}_{i,T_i(n)}. \tag{1}$$

This value is proved to converge to the value of the optimal policy when UCT is used as action selection strategy. We define in a similar way the average payoff of the bootstrapped actions, namely,

$$\bar{X}_c \coloneqq \frac{1}{c} \sum_{i \in \mathcal{B}_A} T_i(c) \bar{X}_{i,T_i(c)}. \tag{2}$$

We have therefore split in two parts the average payoff of the root node of our MC tree, corresponding to the current state of the agent. Putting together in a weighted way the two terms we obtain the total average payoff

$$\begin{aligned} \bar{X}_m &\coloneqq \frac{1}{m}(c \cdot \bar{X}_c + n \cdot \bar{X}_n) \\ &= \frac{c}{m} \cdot \bar{X}_c + \frac{n}{m} \cdot \bar{X}_n \\ &= \frac{1}{m} \sum_{i \in \mathcal{B}_A} T_i(c) \bar{X}_{i,T_i(c)} + \frac{1}{m} \sum_{i \in \overline{\mathcal{B}}_A} T_i(n) \bar{X}_{i,T_i(n)} \\ &= \frac{1}{m} (\sum_{i \in \mathcal{B}_A} T_i(c) \bar{X}_{i,T_i(c)} + \sum_{i \in \overline{\mathcal{B}}_A} T_i(n) \bar{X}_{i,T_i(n)}) \end{aligned} \tag{3}$$

Our proof aims to show that the difference between the estimated expected average payoff $\mathbb{E}\{\bar{X}_m\}$ (i.e., the expected value of the current state) computed using MCTS-SPIBB and the true expected average payoff of the optimal policy (i.e., the true optimal expected value of the current state) in Π_0 (i.e., the space of policies satisfying the SPIBB constraint) tends to zero as the number of simulations m tends to infinity. In the following, we call this difference *bias* of the estimated expected average payoff of the current state or *bias* of the expected value of the current state. In this derivation, we prove that UCT and the baseline policy can be used together to select optimal actions in the simulations achieving a MCTS-based version of SPIBB that computes optimal policies in Π_0.

We will make use of the following standard concentration inequalities:

Fact A.1 (Hoeffding's inequality (Boucheron et al., 2013)). Let $Y_1, \ldots, Y_n$ be independent random variables such that $a_p \leq Y_p \leq b_p$ almost surely for all $p < n$. Let $S_n = \sum_{p=1}^{n}(Y_p - \mathbb{E}(Y_p))$. Then for every $\lambda > 0$

$$\mathbb{P}\left\{S_n \geq \lambda\right\} \leq exp\left(-\frac{2\lambda^2}{\sum_{p=1}^{n}(b_p - a_p)^2}\right) \tag{4}$$

Fact A.2 (Hoeffding-Azuma inequality (Kocsis et al., 2006)). Let $Y_1, \ldots, Y_n$ be a martingale difference (i.e., $\mathbb{E}(Y_p|Y_1, \ldots, Y_{p-1}) = Y_{p-1}$) with $|Y_p| \leq C$ and $C > 0$. Let $S_n = \sum_{p=1}^{n} Y_p$, then for every $\epsilon > 0$

$$\mathbb{P}\left\{S_n \geq \epsilon n\right\} \leq \exp\left(-\frac{2n\epsilon^2}{C^2}\right) \tag{5}$$

$$\mathbb{P}\left\{S_n \leq -\epsilon n\right\} \leq \exp\left(-\frac{2n\epsilon^2}{C^2}\right) \tag{6}$$

Then, from (Kocsis et al., 2006) we get the following concentration inequality for $\bar{X}_{i,T_i(t)} - \mu_i$ on non-bootstrapped actions.

Fact A.3 (Inequality for bias $\bar{X}_{i,T_i(t)} - \mu_i$ on non-bootstrapped actions (Kocsis et al., 2006)). Let X_{it} be payoffs i.i.d. (or a form of martingale difference process shifted by a constant) and $c_{t,T_i(t)} = \sqrt{\frac{2 \ln t}{T_i(t)}}$ the bias sequence of UCT, then

$$\mathbb{P}\left\{\bar{X}_{i,T_i(t)} \geq \mu_i + c_{t,T_i(t)}\right\} \leq t^{-4} \tag{7}$$

$$\mathbb{P}\left\{\bar{X}_{i,T_i(t)} \leq \mu_i - c_{t,T_i(t)}\right\} \leq t^{-4} \tag{8}$$

This inequality follows from Hoeffding-Azuma inequality (Fact A.2), considering $Y_p = X_{i,p} - \mu_i$ with $p = 1, \ldots, T_i(t)$, which is a martingale difference with $|Y_p| \leq 1$, since $|X_{i,p} - \mu_i| \leq 1$ with $p > 0$. Hence, we use $C = 1$ and

$S_{T_i(t)} = \sum_{p=1}^{T_i(t)} Y_p = T_i(t)(\bar{X}_{i,T_i(t)} - \mu_i)$, with $\epsilon = c_{t,T_i(t)} = \sqrt{\frac{2\ln t}{T_i(t)}}$. Substituting these values in Equation 5 we obtain

$$\mathbb{P}\left\{T_i(t)(\bar{X}_{i,T_i(t)} - \mu_i) \geq T_i(t)\sqrt{\frac{2\ln t}{T_i(t)}}\right\} \leq \exp\left(-\frac{2T_i(t)\frac{2\ln t}{T_i(t)}}{1}\right)$$

from which is simply derived Equation 7. Equation 8 is obtained similarly from Equation 6.

We then extend Fact A.3 to bootstrapped actions obtaining the following concentration inequality for payoff bias sequence generated by bootstrapped actions.

Fact A.4 (Inequality for bias $\bar{X}_{i,T_i(t)} - \mu_i$ on bootstrapped actions)**.** Let X_{it} be payoffs i.i.d. (or a form of martingale difference process shifted by a constant) and $\pi_0(i), i \in \mathcal{B}_A$, the baseline probabilities for bootstrapped actions from the current state, then

$$\mathbb{P}\left\{\bar{X}_{i,T_i(t)} \geq \mu_i + \sqrt{\frac{2\ln t}{T_i(t)}}\right\} \leq t^{-4\pi_0(i)} \tag{9}$$

$$\mathbb{P}\left\{\bar{X}_{i,T_i(t)} \leq \mu_i - \sqrt{\frac{2\ln t}{T_i(t)}}\right\} \leq t^{-4\pi_0(i)} \tag{10}$$

Let us start now the actual derivation defining an assumption that characterizes the payoff sequences used in the rest of the proof.

Assumption A.5. Let I_t be a discrete action index set, namely a random variable representing the action taken at the t-th action selection from the current state. In MCTS-SPIBB I is computed in two steps (see Figure 1), first selecting between bootstrapped and non-bootstrapped actions according to probability $p_{\mathcal{B}}$ (see line 2 of Algorithm 3), then, in case of non-bootstrapped action, using the UCT strategy, i.e., $I_t = \text{argmax}_{i \in \overline{\mathcal{B}}_A(s)}\{\bar{X}_{i,T_i(t-1)} + 2C_p\sqrt{\frac{\ln(t-1)}{T_i(t-1)}}\}$ (see (Kocsis et al., 2006) and Algorithm 3, line 11, in this paper), and in case of bootstrapped action according to the probabilistic strategy $I \sim \pi_0(s,.)/p_{\mathcal{B}}$ (see Algorithm 3, line 3-7, in this paper), hence each bootstrapped action i is selected with probability $\pi_0(s,i)/p_{\mathcal{B}}$. Index set I_t defines a filtration $\{\mathcal{F}_{i,t}\}_t$ such that $\{X_{i,t}\}_t$ is $\mathcal{F}_{i,t}$-adapted and $X_{i,t}$ is conditionally independent of $\mathcal{F}_{i,t+1}, \mathcal{F}_{i,t+2}, \ldots$ given $\mathcal{F}_{i,t-1}$ (Kocsis et al., 2006). Then we assume $0 \leq X_{i,t} \leq 1$ and that the limit of $\mu_{i,m} = \mathbb{E}\{\bar{X}_{i,m}\}$ exists both for non-bootstrapped and bootstrapped actions. Furthermore, we assume there exist a constant $C_p > 0$ and an integer N_p such that for $n > N_p$ and for any $\delta > 0$, with $\Delta_n(\delta) := C_p\sqrt{n\ln(1/\delta)}$, the following concentration bounds hold for all actions $i \in A$:

$$\mathbb{P}\left\{n\bar{X}_{i,n} \geq n\mathbb{E}\left\{\bar{X}_{i,n}\right\} + \Delta_n(\delta)\right\} \leq \delta \tag{11}$$

$$\mathbb{P}\left\{n\bar{X}_{i,n} \leq n\mathbb{E}\left\{\bar{X}_{i,n}\right\} - \Delta_n(\delta)\right\} \leq \delta \tag{12}$$

Concentration bounds of Equations 11 and 12 hold due to Hoeffding inequality (see Fact A.1) considering the payoffs $X_{i,t}$ as independent random variables. Their values are $0 \leq X_{i,t} \leq 1$, hence $a_t = 0$ and $b_t = 1$ for $1 \leq t \leq T_i(t)$. Then we define $S_{T_i(t)} = \sum_{t=1}^{T_i(t)}(X_{i,t} - \mathbb{E}\left\{X_{i,t}\right\})$ and $\lambda = \Delta_{T_i(t)}(\delta) = C_p\sqrt{T_i(t)\ln(1/\delta)}$. Substituting these values in Equation 4 we get

$$\mathbb{P}\left\{\sum_{t=1}^{T_i(t)}(X_{i,t} - \mathbb{E}\left\{X_{i,t}\right\}) \geq \Delta_{T_i(t)}(\delta)\right\} \leq \exp\left(-\frac{2C_p^2 T_i(t)\ln(1/\delta)}{T_i(t)}\right)$$

from which Equations 11 and 12 are simply derived with $C_p = \frac{1}{\sqrt{2}}$.

As a first result we provide a bound on the difference between the expected average payoff $\mathbb{E}\{\bar{X}_m\}$ estimated by MCTS-SPIBB after m simulations and the optimal average payoff satisfying the SPIBB constraint, namely, $\sum_{i \in \mathcal{B}_A} \pi_0(i)\mu_i + p_{\overline{\mathcal{B}}} \cdot \mu^\star$. This is, in practice, the bias of the value of the current state (i.e., root node) with respect to the optimal value in Π_0. The action selection strategy used in the following Theorem A.6 mixes UCT, for non-bootstrapped actions, and baseline policy, for bootstrapped actions. The mix is performed in a non-stationary context in which payoffs can drift since several actions must be selected in sequence considering also states reached in successive time instants. Notice that also the payoff of bootstrapped actions can drift because of the influence of UCT in the low levels of the sub-trees used for the estimation of these payoffs.

Theorem A.6 (Bias of the mean.). *Let*

$$\bar{X}_m = \frac{1}{m}\sum_{i\in\mathcal{B}_A} T_i(c)\bar{X}_{i,T_i(c)} + \frac{1}{m}\sum_{i\in\overline{\mathcal{B}}_A} T_i(n)\bar{X}_{i,T_i(n)} \tag{13}$$

Under Assumption A.5 the following bound holds

$$\left|\mathbb{E}\{\bar{X}_m\} - \sum_{i\in\mathcal{B}_A}\pi_0(i)\cdot\mu_i - p_{\overline{\mathcal{B}}}\cdot\mu^\star\right| \leq \sum_{i\in\mathcal{B}_A}\pi_0(i)\cdot|\delta_{i,T_i(c)}| + p_{\overline{\mathcal{B}}}\cdot|\delta^*_n| + O\left(\frac{K(C_p^2\ln m + N_0)}{m}\right) \tag{14}$$

where $N_0 = N_0(\epsilon)$.

Proof. First, we notice that the term $\sum_{i\in\mathcal{B}_A}\pi_0(i)\mu_i$ in Equation 14 represents the true optimum value of the weighted sum of the average payoffs obtained from bootstrapped actions, i.e., it is the equivalent (for bootstrapped actions) of value $p_{\overline{\mathcal{B}}}\cdot\mu^\star$ for the optimal non-bootstrapped action. Let $\bar{X}_c := \frac{1}{c}\sum_{i\in\mathcal{B}_A} T_i(c)\bar{X}_{i,T_i(c)}$ (see Equation 2), $\bar{X}_n := \frac{1}{n}\sum_{i\in\overline{\mathcal{B}}_A} T_i(n)\bar{X}_{i,T_i(n)}$ (see Equation 1) and $\bar{X}_m := \frac{c}{m}\cdot\bar{X}_c + \frac{n}{m}\cdot\bar{X}_n$ (see Equation 3). For the linearity of the expectation, we can rewrite the expected payoff $\mathbb{E}\{\bar{X}_m\}$ as follows

$$\mathbb{E}\{\bar{X}_m\} = \mathbb{E}\left\{\frac{c}{m}\cdot\bar{X}_c + \frac{n}{m}\cdot\bar{X}_n\right\} = \frac{c}{m}\mathbb{E}\{\bar{X}_c\} + \frac{n}{m}\mathbb{E}\{\bar{X}_n\} \tag{15}$$

Therefore, the left part of Inequality 14 can be written as

$$\left|\mathbb{E}\{\bar{X}_m\} - \sum_{i\in\mathcal{B}_A}\pi_0(i)\mu_i - p_{\overline{\mathcal{B}}}\cdot\mu^\star\right| = \left|(\frac{c}{m}\mathbb{E}\{\bar{X}_c\} - \sum_{i\in\mathcal{B}_A}\pi_0(i)\mu_i) + (\frac{n}{m}\mathbb{E}\{\bar{X}_n\} - p_{\overline{\mathcal{B}}}\cdot\mu^\star)\right| \tag{16}$$

$$\left[\textit{Since}\ \lim_{m\to\infty}\frac{c(m)}{m} = p_{\mathcal{B}}\ \textit{and}\ \lim_{m\to\infty}\frac{n(m)}{m} = p_{\overline{\mathcal{B}}}\right] \quad = \left|(p_{\mathcal{B}}\mathbb{E}\{\bar{X}_c\} - \sum_{i\in\mathcal{B}_A}\pi_0(i)\mu_i) + (p_{\overline{\mathcal{B}}}\mathbb{E}\{\bar{X}_n\} - p_{\overline{\mathcal{B}}}\cdot\mu^\star)\right| \tag{17}$$

$$= \left|(p_{\mathcal{B}}\mathbb{E}\{\bar{X}_c\} - \sum_{i\in\mathcal{B}_A}\pi_0(i)\mu_i) + p_{\overline{\mathcal{B}}}(\mathbb{E}\{\bar{X}_n\} - \mu^\star)\right| \tag{18}$$

$$\left[\textit{Triangle inequality}\right] \quad \leq \left|p_{\mathcal{B}}\mathbb{E}\{\bar{X}_c\} - \sum_{i\in\mathcal{B}_A}\pi_0(i)\mu_i\right| + p_{\overline{\mathcal{B}}}\left|\mathbb{E}\{\bar{X}_n\} - \mu^\star\right| \tag{19}$$

$$\left[\textit{Theorem 3 of (Kocsis et al., 2006)}\right] \quad \leq \left|p_{\mathcal{B}}\mathbb{E}\{\bar{X}_c\} - \sum_{i\in\mathcal{B}_A}\pi_0(i)\mu_i\right| + p_{\overline{\mathcal{B}}}\cdot|\delta^*_n| + O\left(p_{\overline{\mathcal{B}}}\frac{K(C_p^2\ln n + N_0)}{n}\right) \tag{20}$$

$$\left[\mathbb{E}\{\bar{X}_c\} = \sum_{i\in\mathcal{B}_A}\frac{T_i(c)}{c}\mathbb{E}\{\bar{X}_{i,T_i(c)}\}\right] \quad = \left|p_{\mathcal{B}}\sum_{i\in\mathcal{B}_A}\frac{T_i(c)}{c}\mathbb{E}\{\bar{X}_{i,T_i(c)}\} - \sum_{i\in\mathcal{B}_A}\pi_0(i)\mu_i\right| + p_{\overline{\mathcal{B}}}\cdot|\delta^*_n| \tag{21}$$

$$+ O\left(\frac{n}{m}\frac{K(C_p^2\ln n + N_0)}{n}\right) \tag{22}$$

$$\left[\lim_{c\to\infty}\frac{T_i(c)}{c} = \frac{\pi_0(i)}{p_{\mathcal{B}}};\ \mu_{i,T_i(c)} := \mathbb{E}\{\bar{X}_{i,T_i(c)}\}\right] \quad = \left|\sum_{i\in\mathcal{B}_A}\pi_0(i)\mu_{i,T_i(c)} - \sum_{i\in\mathcal{B}_A}\pi_0(i)\mu_i\right| + p_{\overline{\mathcal{B}}}\cdot|\delta^*_n| + O\left(\frac{K(C_p^2\ln n + N_0)}{m}\right) \tag{23}$$

$$\left[\lim_{m\to\infty}\frac{n(m)}{m} = p_{\overline{\mathcal{B}}}\right] \quad = \left|\sum_{i\in\mathcal{B}_A}\pi_0(i)(\mu_{i,T_i(c)} - \mu_i)\right| + p_{\overline{\mathcal{B}}}\cdot|\delta^*_n| + O\left(\frac{K(C_p^2\ln(p_{\overline{\mathcal{B}}}\cdot m) + N_0)}{m}\right) \tag{24}$$

$$\left[\delta_{i,T_i(c)} := \mu_{i,T_i(c)} - \mu_i\text{; Prop. of logarithms}\right] \quad = \left| \sum_{i \in \mathcal{B}_A} \pi_0(i) \cdot \delta_{i,T_i(c)} \right| + p_{\overline{\mathcal{B}}} \cdot |\delta_n^*| + O\left(\frac{K(C_p^2 \ln p_{\overline{\mathcal{B}}} + C_p^2 \ln m + N_0)}{m} \right) \tag{25}$$

$$\left[C_p^2 \ln p_{\overline{\mathcal{B}}} \leq 0 \to \text{removed}\right] \quad = \left| \sum_{i \in \mathcal{B}_A} \pi_0(i) \cdot \delta_{i,T_i(c)} \right| + p_{\overline{\mathcal{B}}} \cdot |\delta_n^*| + O\left(\frac{K(C_p^2 \ln m + N_0)}{m} \right) \tag{26}$$

$$\left[\text{Triangle inequality}\right] \quad \leq \sum_{i \in \mathcal{B}_A} \pi_0(i) \cdot |\delta_{i,T_i(c)}| + p_{\overline{\mathcal{B}}} \cdot |\delta_n^*| + O\left(\frac{K(C_p^2 \ln m + N_0)}{m} \right) \tag{27}$$

□

The bound provided by Theorem A.6 depends on the biases $\delta_{i,T_i(c)}$ of the values of bootstrapped actions in the current state and the bias δ_n^* of the optimal non-bootstrapped action in the current state. Namely, it considers only (action-)nodes of the MC tree one level below the root. The next theorem extends this result considering all the levels of the MC tree up to the leaves and providing a bound that depends only on the total number of simulations m, the depth of the tree D and the number of non-bootstrapped and bootstrapped actions, K and L, respectively. The bound converges to zero as m increases. This proves that the value generated by MCTS-SPIBB for the current state converges to the optimal value in Π_0 (as SPIBB).

Theorem A.7 (Convergence of the estimated expected payoff $\bar{X}_m$.)**.** *Consider algorithm MCTS-SPIBB running on a tree of depth D, branching factor $|A| = |\mathcal{B}_A| + |\overline{\mathcal{B}}_A| = L + K$ with $0 \leq L \leq |A|$ bootstrapped actions and $0 \leq K \leq |A|$ non-bootstrapped actions in each state, and stochastic payoffs at the leaves. Assume that the payoffs lie in the interval* $[0, 1]$*. Then the bias of the estimated expected payoff, $\bar{X}_m$, is*

$$O\left(\frac{L^D K D \ln m + L^D K^D}{m} \right). \tag{28}$$

Proof. The proof is made by induction on D.

Base. Consider the case with $D = 1$. The bias of $\mathbb{E}\left\{\bar{X}_m\right\}$ can be bounded, according to Theorem A.6, as

$$\left| \mathbb{E}\{\bar{X}_m\} - \sum_{i \in \mathcal{B}_A} \pi_0(i)\mu_i - p_{\overline{\mathcal{B}}} \cdot \mu^\star \right| \leq \sum_{i \in \mathcal{B}_A} \pi_0(i) \cdot |\delta_{i,T_i(c)}| + p_{\overline{\mathcal{B}}} \cdot |\delta_n^*| + O\left(\frac{K(\ln m + N_0)}{m} \right). \tag{29}$$

With $D = 1$ the biases of bootstrapped actions $|\delta_{i,T_i(c)}|$ and the optimal non-bootstrapped action $|\delta_n^*|$ refer to leaf nodes and UCT on non-bootstrapped actions correspond to UCB1. Since $0 \leq X_{i,t} \leq 1$ and μ_i exist $\forall i \in A$, for Assumption A.5, we have that $|\delta_{i,T_i(c)}| := |\mathbb{E}\left\{\bar{X}_{i,T_i(c)}\right\} - \mu_i| \leq 1, \forall i \in \mathcal{B}_A$ and $|\delta_n^*| := |\mathbb{E}\left\{\bar{X}_{i^*,n}\right\} - \mu^*| \leq 1$, therefore

$$\sum_{i \in \mathcal{B}_A} \pi_0(i) \cdot |\delta_{i,T_i(c)}| + p_{\overline{\mathcal{B}}} \cdot |\delta_n^*| \leq p_{\mathcal{B}} + p_{\overline{\mathcal{B}}} = 1. \tag{30}$$

Then, the term $O\left(\frac{K \ln m + K N_0}{m}\right)$ of Equation 29 is also an $O\left(\frac{LK \ln m + LK}{m}\right)$ (i.e., theorem thesis with $D = 1$) because *i)* $(K \ln m \leq LK \ln m)$ since $L \geq 1$ (otherwise we are in the case of only non-bootstrapped actions for which Th. 7 of (Kocsis et al., 2006) holds), *ii)* $(K N_0 \leq LK)$ since $L \geq 1$ and $N_0 = O(L^D K^{D-1})$, that is, $N_0 = O(L)$ with $D = 1$. Notice that with $D = 1$ N_0 is actually a constant that depends neither on L nor on K.

Inductive step. Let us assume that the theorem holds for all trees of depth $D - 1$, namely, that for each of these trees the bias of the estimated expected payoff is

$$O\left(\frac{L^{D-1} K (D-1) \ln m + L^{D-1} K^{D-1}}{m} \right). \tag{31}$$

then we show that also for all trees of depth D the theorem holds. Theorem A.6 allows us to connect the root node of a tree of depth D with its children nodes, which are root nodes of subtrees of depth $D - 1$. According to that theorem, the bias of the estimated expected payoff in the root (depth D) is

$$\left| \mathbb{E}\{\bar{X}_m\} - \sum_{i \in \mathcal{B}_A} \pi_0(i)\mu_i - p_{\overline{\mathcal{B}}} \cdot \mu^\star \right| \leq \sum_{i \in \mathcal{B}_A} \pi_0(i) \cdot |\delta_{i,T_i(c)}| + p_{\overline{\mathcal{B}}} \cdot |\delta_n^*| + O\left(\frac{K(\ln m + N_0)}{m} \right). \tag{32}$$

where $|\delta_{i,T_i(c)}| = |\mu_{i,T_i(c)} - \mu_i| = |\mathbb{E}\left\{\bar{X}_{i,T_i(c)}\right\} - \mu_i|, i \in \mathcal{B}_A$ are the biases of the estimated expected payoffs of nodes reachable performing a bootstrapped action from the root, and $|\delta_n^*| = |\mu_n^* - \mu^*| = |\mathbb{E}\left\{\bar{X}_{i^*,n}\right\} - \mu^*|$ is the bias of the estimated expected payoffs of the node reachable performing the optimal non-bootstrapped action (identified using the UCT strategy) from the root. Since the nodes reached performing a bootstrapped or the optimal non-bootstrapped action are roots of a sub-tree with depth reduced of 1 w.r.t. the root, biases $|\delta_{i,T_i(c)}|$ and $|\delta_n^*|$ refer to trees of depth $D-1$ for which the inductive hypothesis holds.

We consider now each of the three terms on the right of inequality (32) separately. Then, we put the results together to complete the proof. The first term is related to the sum of biases of bootstrapped actions, namely, $\sum_{i\in\mathcal{B}_A} \pi_0(i) \cdot |\delta_{i,T_i(c)}|$. We consider each action i separately and apply the inductive hypothesis on it, obtaining the following bound

$$\begin{aligned}
& |\delta_{i,T_i(c)}| = |\mathbb{E}\left\{\bar{X}_{i,T_i(c)}\right\} - \mu_i| \qquad (33)\\
& \left[\textit{By inductive hypothesis}\right] \quad = O\left(\frac{L^{D-1}K(D-1)\ln T_i(c) + L^{D-1}K^{D-1}}{T_i(c)}\right) \qquad (34)\\
& \left[\lim_{m\to\infty}\frac{T_i(c(m))}{m} = \pi_0(i), \forall i \in \mathcal{B}_A\right] \quad = O\left(\frac{L^{D-1}K(D-1)\ln\left(m\cdot\pi_0(i)\right) + L^{D-1}K^{D-1}}{m\cdot\pi_0(i)}\right) \qquad (35)\\
& \left[\textit{Properties of logarithms}\right] \quad = O\left(\frac{L^{D-1}K(D-1)\ln m + L^{D-1}K(D-1)\ln\pi_0(i) + L^{D-1}K^{D-1}}{m\cdot\pi_0(i)}\right) \qquad (36)\\
& \left[L^{D-1}K(D-1)\ln\pi_0(i) \le 0 \to \textit{removed}\right] \quad = O\left(\frac{L^{D-1}K(D-1)\ln m + L^{D-1}K^{D-1}}{m\cdot\pi_0(i)}\right) \qquad (37)\\
& = O\left(\frac{L^{D-1}KD\ln m - L^{D-1}K\ln m + L^{D-1}K^{D-1}}{m\cdot\pi_0(i)}\right) \qquad (38)
\end{aligned}$$

Summing up over all bootstrapped actions i according to $\sum_{i\in\mathcal{B}_A}\pi_0(i)\cdot|\delta_{i,T_i(c)}|$ we obtain

$$\begin{aligned}
& \sum_{i\in\mathcal{B}_A}\pi_0(i)\cdot|\delta_{i,T_i(c)}| = O\left(\sum_{i\in\mathcal{B}_A}\pi_0(i)\frac{L^{D-1}KD\ln m - L^{D-1}K\ln m + L^{D-1}K^{D-1}}{m\cdot\pi_0(i)}\right) \qquad (39)\\
& = O\left(\sum_{i\in\mathcal{B}_A}\frac{L^{D-1}KD\ln m - L^{D-1}K\ln m + L^{D-1}K^{D-1}}{m}\right) \qquad (40)\\
& \left[\textit{Sum over } L \textit{ bootstrapped actions, no dependence on } i \textit{ in the sum}\right] \quad = O\left(L\cdot\frac{L^{D-1}KD\ln m - L^{D-1}K\ln m + L^{D-1}K^{D-1}}{m}\right) \qquad (41)\\
& = O\left(\frac{L^{D}KD\ln m - L^{D}K\ln m + L^{D}K^{D-1}}{m}\right) \qquad (42)
\end{aligned}$$

The second term we consider is $p_{\overline{\mathcal{B}}}\cdot|\delta_n^*|$ which is related to the bias of the optimal non-bootstrapped action. We obtain the bound

$$\begin{aligned}
& p_{\overline{\mathcal{B}}}\cdot|\delta_n^*| = p_{\overline{\mathcal{B}}}|\mathbb{E}\left\{\bar{X}_{i^*,n}\right\} - \mu^*| \qquad (43)\\
& \left[\textit{By inductive hypothesis}\right] \quad = p_{\overline{\mathcal{B}}}\cdot O\left(\frac{L^{D-1}K(D-1)\ln n + L^{D-1}K^{D-1}}{n}\right) \qquad (44)\\
& \left[\lim_{m\to\infty}\frac{n(m)}{m} = p_{\overline{\mathcal{B}}}\right] \quad = p_{\overline{\mathcal{B}}}\cdot O\left(\frac{L^{D-1}K(D-1)\ln\left(p_{\overline{\mathcal{B}}}\cdot m\right) + L^{D-1}K^{D-1}}{p_{\overline{\mathcal{B}}}\cdot m}\right) \qquad (45)\\
& = O\left(\frac{L^{D-1}K(D-1)\ln\left(p_{\overline{\mathcal{B}}}\cdot m\right) + L^{D-1}K^{D-1}}{m}\right) \qquad (46)\\
& \left[\textit{Properties of logarithms}\right] \quad = O\left(\frac{L^{D-1}K(D-1)\ln p_{\overline{\mathcal{B}}} + L^{D-1}K(D-1)\ln\left(m\right) + L^{D-1}K^{D-1}}{m}\right) \qquad (47)\\
& \left[L^{D-1}K(D-1)\ln p_{\overline{\mathcal{B}}} \le 0 \to \textit{removed}\right] \quad = O\left(\frac{L^{D-1}K(D-1)\ln m + L^{D-1}K^{D-1}}{m}\right) \qquad (48)
\end{aligned}$$

$$= O\left(\frac{L^{D-1}KD\ln m - L^{D-1}K\ln m + L^{D-1}K^{D-1}}{m}\right) \tag{49}$$

The third term we consider is $O\left(\frac{K(\ln m+N_0)}{m}\right)$, namely, the second part of the bias related to non-bootstrapped actions

$$O\left(\frac{K(\ln m + N_0)}{m}\right) = O\left(\frac{K\ln m + KN_0}{m}\right) \tag{50}$$

$$\left[\text{With } N_0 = O(L^D K^{D-1})\right] \quad = O\left(\frac{K\ln m + L^D K^D}{m}\right) \tag{51}$$

Finally, we put together the three terms of Equations 42, 49, and 51 in Equation 32 obtaining a bound for the bias of the estimated expected payoff of the root of the tree having depth D, which is

$$\left|\mathbb{E}\{\bar{X}_m\} - \sum_{i\in\mathcal{B}_A}\pi_0(i)\mu_i - p_{\overline{\mathcal{B}}}\cdot\mu^\star\right| \le O\left(\frac{L^D KD\ln m - L^D K\ln m + L^D K^{D-1}}{m}\right) \tag{52}$$

$$+ O\left(\frac{L^{D-1}KD\ln m - L^{D-1}K\ln m + L^{D-1}K^{D-1}}{m}\right) \tag{53}$$

$$+ O\left(\frac{K\ln m + L^D K^D}{m}\right) \tag{54}$$

$$\left[\text{Rearranging terms}\right] \quad = O\left(\frac{L^{D-1}KD\ln m + L^D KD\ln m + K\ln m}{m}\right) \tag{55}$$

$$+ O\left(\frac{L^D K^D + L^{D-1}K^{D-1} + L^D K^{D-1}}{m}\right) \tag{56}$$

$$+ O\left(\frac{-L^{D-1}K\ln m - L^D K\ln m}{m}\right) \tag{57}$$

$$\left[\text{Removing (positive) terms with smaller } D \text{ order}\right] \quad = O\left(\frac{L^D KD\ln m + L^D K^D - L^{D-1}K\ln m - L^D K\ln m}{m}\right) \tag{58}$$

$$\left[-L^{D-1}K\ln m \le 0, -L^D K\ln m \le 0 \to \text{removed}\right] \quad = O\left(\frac{L^D KD\ln m + L^D K^D}{m}\right) \tag{59}$$

which proves the induction. □

The third theorem provides, finally, a concentration bound that shows that the estimated optimal payoff concentrates quickly around its mean.

Theorem A.8 (Concentration bound for the mean.). *Fix an arbitrary* $\delta > 0$ *and let* $\Delta_m = 9\sqrt{2p_{\overline{\mathcal{B}}}m\ln(4/\delta)} + C_p\sqrt{m\ln(2L/\delta)}\sum_{i\in\mathcal{B}_A}\sqrt{\pi_0(i)}$*. Let* $n_0 \in \mathbb{N}$ *be such that* $\sqrt{n_0} \ge O(K(C_p^2\ln n_0 + N_0(1/2)))$*, if* $m \ge \frac{n_0}{p_{\overline{\mathcal{B}}}}$ *then under Assumption A.5 the following bounds hold:*

$$\mathbb{P}\left\{m\bar{X}_m \ge m\mathbb{E}\left\{\bar{X}_m\right\} + \Delta_m\right\} \le \delta, \tag{60}$$

$$\mathbb{P}\left\{m\bar{X}_m \le m\mathbb{E}\left\{\bar{X}_m\right\} - \Delta_m\right\} \le \delta. \tag{61}$$

Proof. Let us consider the first bound, in Equation 60. We first split the probability in two parts, namely,

$$\mathbb{P}\left\{m\bar{X}_m \ge m\mathbb{E}\left\{\bar{X}_m\right\} + \Delta_m\right\} = \tag{62}$$

$$\text{Since } \bar{X}_m = \frac{c}{m}\cdot\bar{X}_c + \frac{n}{m}\cdot\bar{X}_n \quad = \mathbb{P}\left\{c\bar{X}_c + n\bar{X}_n \ge \mathbb{E}\left\{c\bar{X}_c + n\bar{X}_n\right\} + \Delta_m\right\} \tag{63}$$

$$\text{Linearity of expectation and } \Delta_m := \Delta_c + \Delta_n \quad = \mathbb{P}\left\{c\bar{X}_c + n\bar{X}_n \ge c\cdot\mathbb{E}\left\{\bar{X}_c\right\} + n\cdot\mathbb{E}\left\{\bar{X}_n\right\} + \Delta_c + \Delta_n\right\} \tag{64}$$

$$\text{Since } \mathbb{P}\left\{A+B \ge C+D\right\} \le \mathbb{P}\left\{A \ge C\right\} + \mathbb{P}\left\{B \ge D\right\} \quad \le \mathbb{P}\left\{c\bar{X}_c \ge c\cdot\mathbb{E}\left\{\bar{X}_c\right\} + \Delta_c\right\} + \mathbb{P}\left\{n\bar{X}_n \ge n\cdot\mathbb{E}\left\{\bar{X}_n\right\} + \Delta_n\right\} \tag{65}$$

$$= P_{\mathcal{B}} + P_{\overline{\mathcal{B}}}. \tag{66}$$

The part related to *non-bootstrapped* actions $P_{\overline{\mathcal{B}}} := \mathbb{P}\left\{n\bar{X}_n \geq n \cdot \mathbb{E}\left\{\bar{X}_n\right\} + \Delta_n\right\}$ can be bounded using Theorem 5 of (Kocsis et al., 2006). For an arbitrary $\delta_{\overline{\mathcal{B}}} = \delta/2$ and $\Delta_n = 9\sqrt{2n\ln(2/\delta_{\overline{\mathcal{B}}})} = 9\sqrt{2n\ln(4/\delta)}$ there exist a n_0 such that $\sqrt{n_0} \geq O(K(C_p^2 \ln n_0 + N_0(1/2)))$ and for any $n \geq n_0$, under Assumption A.5 the following bound holds:

$$P_{\overline{\mathcal{B}}} = \mathbb{P}\left\{n\bar{X}_n \geq n \cdot \mathbb{E}\left\{\bar{X}_n\right\} + \Delta_n\right\} \leq \delta_{\overline{\mathcal{B}}} = \delta/2 \tag{67}$$

We notice that Δ_n can be expressed also in terms of the total number of simulations m, since $\lim_{m\to\infty} \frac{n(m)}{m} = p_{\overline{\mathcal{B}}}$. In that case we have

$$\Delta_n = 9\sqrt{2p_{\overline{\mathcal{B}}} m \ln(4/\delta)}. \tag{68}$$

The probability related to *bootstrapped* actions $P_{\mathcal{B}} := \mathbb{P}\left\{c\bar{X}_c \geq c \cdot \mathbb{E}\left\{\bar{X}_c\right\} + \Delta_c\right\}$ can be decomposed into probabilities for the single actions, as follows:

$$\begin{aligned}
P_{\mathcal{B}} &= \mathbb{P}\left\{c\bar{X}_c \geq c \cdot \mathbb{E}\left\{\bar{X}_c\right\} + \Delta_c\right\} && (69)\\
\left[\textit{Since } \bar{X}_c := \frac{1}{c}\sum_{i\in\mathcal{B}_A} T_i(c)\bar{X}_{i,T_i(c)}\right] \quad &= \mathbb{P}\left\{\sum_{i\in\mathcal{B}_A} T_i(c)\bar{X}_{i,T_i(c)} \geq \sum_{i\in\mathcal{B}_A} T_i(c)\mathbb{E}\left\{\bar{X}_{i,T_i(c)}\right\} + \Delta_c\right\} && (70)\\
\left[\Delta_c := \sum_{i\in\mathcal{B}_A} \Delta_{c_i}\right] \quad &= \mathbb{P}\left\{\sum_{i\in\mathcal{B}_A} T_i(c)\bar{X}_{i,T_i(c)} \geq \sum_{i\in\mathcal{B}_A} \left(T_i(c)\mathbb{E}\left\{\bar{X}_{i,T_i(c)}\right\} + \Delta_{c_i}\right)\right\} && (71)\\
\left[\textit{Since } \mathbb{P}\left\{\sum_i A_i \geq \sum_i B_i\right\} \leq \sum_i \mathbb{P}\left\{A_i \geq B_i\right\}\right] \quad &\leq \sum_{i\in\mathcal{B}_A} \mathbb{P}\left\{T_i(c)\bar{X}_{i,T_i(c)} \geq \left(T_i(c)\mathbb{E}\left\{\bar{X}_{i,T_i(c)}\right\} + \Delta_{c_i}\right)\right\} && (72)
\end{aligned}$$

We consider each term $\mathbb{P}\left\{T_i(c)\bar{X}_{i,T_i(c)} \geq \left(T_i(c)\mathbb{E}\left\{\bar{X}_{i,T_i(c)}\right\} + \Delta_{c_i}\right)\right\}$ in the sum separately. We observe that it corresponds to Equation 11 of Assumption A.5, substituting $T_i(c)$ to n and $\Delta_{c_i} = C_p\sqrt{T_i(c)\ln(\frac{1}{\delta_{\mathcal{B},i}})}$ to Δ_n with $\delta_{\mathcal{B},i} > 0, \forall i \in \mathcal{B}_A$. Also in this case Δ_{c_i} can be expressed in terms of m since $\lim_{m\to\infty} \frac{T_i(c(m))}{m} = \pi_0(i), \forall i \in \mathcal{B}_A$. In that case we have

$$\Delta_{c_i} = C_p\sqrt{\pi_0(i) m \ln(\frac{1}{\delta_{\mathcal{B},i}})}. \tag{73}$$

Therefore, $\mathbb{P}\left\{T_i(c)\bar{X}_{i,T_i(c)} \geq \left(T_i(c)\mathbb{E}\left\{\bar{X}_{i,T_i(c)}\right\} + \Delta_{c_i}\right)\right\} \leq \delta_{\mathcal{B},i}$ by Assumption A.5 (proved above using the Hoeffding inequality). This guarantees that the estimated payoff of each bootstrapped action concentrates quickly around its mean, hence computing the sum over bootstrapped actions in Eq. 72 we obtain:

$$\begin{aligned}
P_{\mathcal{B}} &\leq \sum_{i\in\mathcal{B}_A} \mathbb{P}\left\{T_i(c)\bar{X}_{i,T_i(c)} \geq \left(T_i(c)\mathbb{E}\left\{\bar{X}_{i,T_i(c)}\right\} + \Delta_{c_i}\right)\right\} && (74)\\
&\leq \sum_{i\in\mathcal{B}_A} \delta_{\mathcal{B},i} && (75)\\
\left[\delta_{\mathcal{B},i} := \delta/(2L)\right] \quad &= \delta/2 && (76)
\end{aligned}$$

Notice that we selected $\delta_{\mathcal{B},i} := \delta/(2L)$, so that $\delta_{\mathcal{B}} = \sum_{i\in\mathcal{B}_A} \delta_{\mathcal{B},i} = L\delta/(2L) = \delta/2$. To conclude the proof we substitute $\delta_{\mathcal{B},i}$ in Δ_{c_i} obtaining $\Delta_{c_i} = C_p\sqrt{\pi_0(i) m \ln(\frac{2L}{\delta})}$ and we sum up Δ_{c_i} over bootstrapped actions obtaining

$$\begin{aligned}
\Delta_c &= \sum_{i\in\mathcal{B}_A} \Delta_{c_i} && (77)\\
&= \sum_{i\in\mathcal{B}_A} C_p\sqrt{\pi_0(i) m \ln(\frac{2L}{\delta})} && (78)\\
&= C_p\sqrt{m\ln(2L/\delta)} \sum_{i\in\mathcal{B}_A} \sqrt{\pi_0(i)} && (79)
\end{aligned}$$

In conclusion

$$\mathbb{P}\left\{m\bar{X}_m \geq m\mathbb{E}\left\{\bar{X}_m\right\} + \Delta_m\right\} \leq P_{\mathcal{B}} + P_{\overline{\mathcal{B}}} \tag{80}$$

$$\leq \delta/2 + \delta/2 \tag{81}$$

$$= \delta \tag{82}$$

with

$$\Delta_m = \Delta_n + \Delta_c \tag{83}$$

$$= 9\sqrt{2p_{\overline{\mathcal{B}}} m \ln(4/\delta)} + C_p\sqrt{m\ln(2L/\delta)} \sum_{i\in\mathcal{B}_A} \sqrt{\pi_0(i)} \tag{84}$$

Equation 61 can be proved similarly.

□

B. Supplementary Material About the Methodology

B.1. Generation of the Baseline Policy

Baseline policies are generated on a small version of GridWorld and SysAdmin by first creating an optimal policy π_{opt} with policy iteration, and then applying random noise to the probabilities to introduce suboptimal choices. On large SysAdmin domains, used to test MCTS-SPIBB scalability, the computation of the optimal policies by policy iteration is impossible, hence the baseline policies are generated from simple rules. For instance, given the configuration of machines turned on/off in a specific state, our rule says: if all machines are off, turn on a random machine; if at least one machine is on, turn on the closest machine that is off; if all machines are on, do nothing.

C. Supplementary Results

C.1. Results on Convergence

We also performed an empirical analysis of the number of simulations required for concentration by Theorem 5.3. We focused on two random states, one for GridWorld (i.e., third row, second column of the grid) and one for SysAdmin (i.e., 4 machines off out of 7). Substituting parameters C_p, γ, L, K, D, π_0 in the inequality containing term n_0, considering $N_0(1/2) = L^D K^{D-1}$ (as in Theorem 5.2), solving the inequality, and finally computing m from n_0 we get that the number of simulations required by the GridWorld state is $m = 5.03 \cdot 10^{63}$ and that required by the SysAdmin state is $m = 5.05 \cdot 10^{62}$. These numbers are much higher than those we used to get good results in our empirical tests (i.e., $m = 100, 1000, 10000$ for both GridWorld and SysAdmin). The motivation for this is that the assumption of $N_0(1/2) = O(L^D K^{D-1})$ made in the proof of Theorem 5.2 is an overestimation of the actual number of simulations needed to reach convergence. However, this assumption simplifies the mathematical computation and allows us to obtain the proof. We notice that a similar overestimation is made also in (Laroche et al., 2019) to compute parameter $N_\wedge$. More strict bounds could exist but their derivation is not the focus of this work. We will try to derive stricter bounds in future work.

C.2. Results on Safety

We perform supplementary experiments on safety on Gridworld 5x5 and SysAdmin with 7 machines. For each domain, we first generate a baseline; then, for different dataset sizes $|\mathcal{D}|$ we generate $ND = 20$ datasets each containing $|\mathcal{D}|$ trajectories. In particular, in Gridworld $|\mathcal{D}| \in \{2, 10^1, 10^2, 10^3, 10^4\}$ and each trajectory is 30 steps long, while in SysAdmin $|\mathcal{D}| \in \{5, 500, 5000\}$ and each trajectory is 15 steps long. Afterward, for each dataset, we compute the MLE transition model $T^{\mathcal{D}}$, the state-action pair count matrix $N_{\mathcal{D}}(s, a)$ and the bootstrapped/non-bootstrapped action sets $\mathcal{B}_A(s)/\overline{\mathcal{B}}_A(s)$ using threshold $N_\wedge = 5$. Finally, for each dataset, we generate the improved policy using MCTS-SPIBB (with 10000 simulations), SPIBB, and Basic RL and we evaluate their performance on the real environment as values in the initial state s_0, i.e., $\rho(\pi_I, M^*) = V^{\pi_I}_{M^*}(s_0)$ (the policy based on MCTS-SPIBB is computed in all states to allow the usage of policy evaluation to compute the values).

Figures 6.a,c show the results on Gridworld and SysAdmyn, respectively. We observe that also in this supplementary experiment, Basic RL achieves a performance decrease since it is not safe. MCTS-SPIBB and SPIBB perform almost

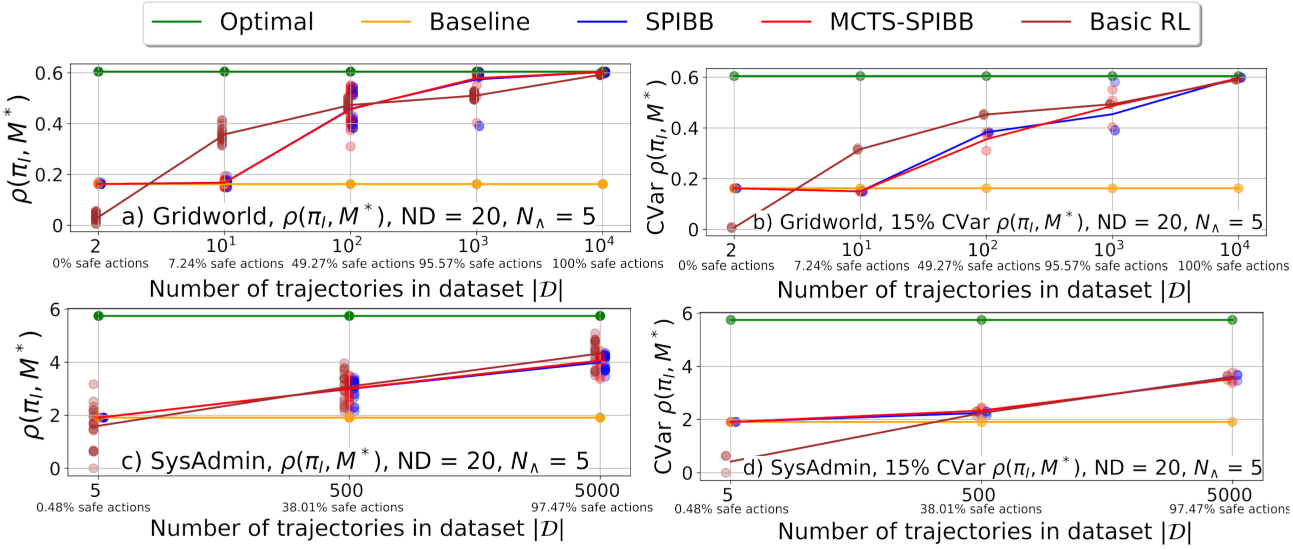

Figure 6. Supplementary results on safety. Performance $\rho(\pi_I, M^*)$ and 15%-CVaR $\rho(\pi_I, M^*)$ on Gridworld 5×5 (a,b) and SysAdmin with 7 machines (c,d).

identically and their performance is always equal or higher than that of the baseline since they are safe. In Figures 6.b,d which consider 15% Conditional Value-at-Risk (15%-CVaR), MCTS-SPIBB and SPIBB are still safe and they perform very similarly.

C.3. Results on Scalability

We perform supplementary experiments on scalability on SysAdmin. Figure 7 shows the time needed by SPIBB (Laroche et al., 2019) (light blue line), SPIBB$_{DP}$ based on dynamic programming (dotted blue line) and MCTS-SPIBB (light green, red and dotted purple lines). We analyze the time for computing the policy and performance of the improved policy with the worst 15%-CVar computed by MCTS-SPIBB on 13, 20, and 35 machines to see if it actually improves the baseline where SPIBB and SPIBB$_{DP}$ do not work.

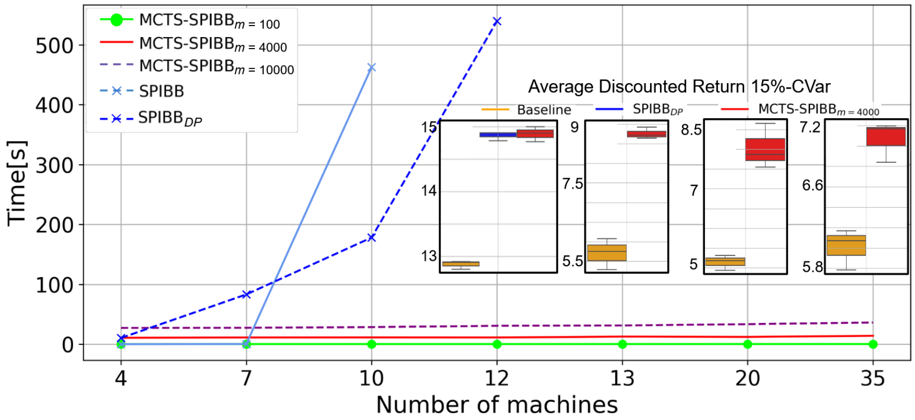

Figure 7. Results on scalability. Lines: computational time of SPIBB, SPIBB$_{DP}$ and MCTS-SPIBB on SysAdmin with 4, 7, 10, 12, 13, 20, and 35 machines. Box plots: 15%-CVar of baseline policy, SPIBB$_{DP}$ and MCTS-SPIBB on 12, 13, 20, and 35 machines.

C.4. Comparison with Other State-of-the-art SPI Algorithms

Full Description of the WetChicken Domain. An agent floats in a small boat on a river with a waterfall at one end. The goal for the agent is to stay as close as possible to the waterfall without falling down. The closer the agent is to the waterfall, the greater the reward, but if it falls down, the episode ends. The river is represented as a 5×5 grid in the benchmark (i.e., $|S| = 25$). Five actions can be chosen by the agent (*drift* - do nothing; *hold* - paddle back with half its power; *paddle back* - paddle back; *right* - go right parallel to the waterfall; *left* - go left parallel to the waterfall. The agent starts at position $(x, y) = (0, 0)$, and its position at time t is denoted by the pair (x_t, y_t). The river has turbulence equal to $b_t = 3.5 - v_t$, which is stronger for small y, and a stream towards the waterfall equal to $v_t = \frac{3}{5}y_t$, which is stronger for larger y. The effects of turbulence are stochastic and are defined by $\tau \sim U(-1, 1)$. Given a state $s_t = (x_t, y_t)$ and an action $a_t = (a_x, a_y)$, the next state is calculated as $s' = (x'_t, y'_t) = (round(x_t + a_x + v_t + \tau_t b_t), round(x_t, a_y))$.

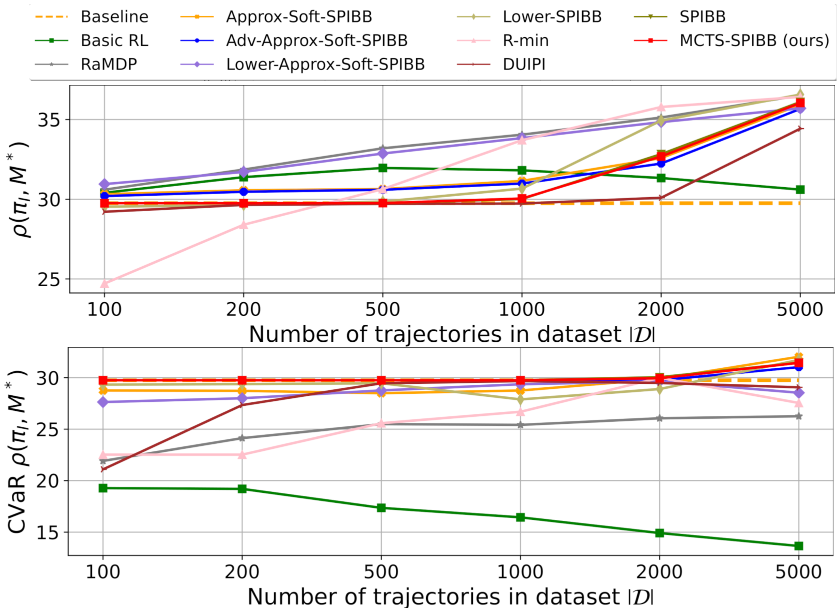

Figure 8. Performance comparison between MCTS-SPIBB and state-of-the-art SPI algorithms (SPIBB, SPIBB extensions, Basic-RL, DUIPI, R-Min, RaMDP) on the WetChicken domain (Scholl et al., 2022b).

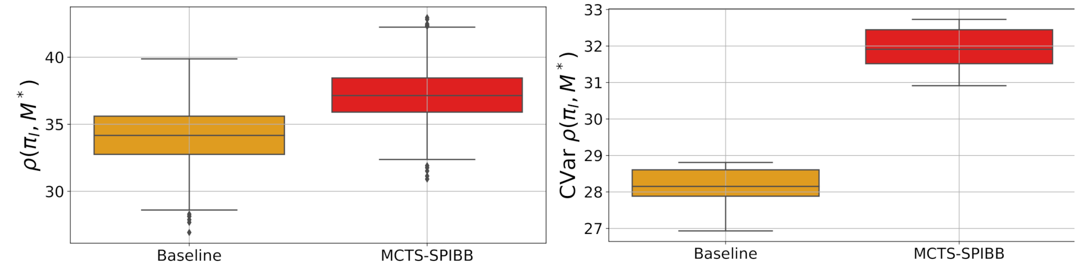

Figure 9. Results on scalability. a) Performance $\rho(\pi_I, M^*)$ of Baseline and MCTS-SPIBB policy and, b) 1%-CVar on WetChicken domain on a 70×70 grid.

Results. Figure 8 shows the results of the experiments performed on the benchmark proposed in (Scholl et al., 2022b). Tests are performed using the baseline policy, Basic RL, SPIBB, some SPIBB variants presented in (Scholl et al., 2022b), R-min, DUIPI, RAMDP, and MCTS-SPIBB. The x-axis displays different dataset dimensions $|\mathcal{D}|$ and the y-axis displays the performance of the improved policy (on top) and the 1%-CVaR of the performance of the improved policy (at the bottom). All tests are performed on a *WetChicken* environment with 25 states, as in (Scholl et al., 2022b). We used the software provided in (Scholl et al., 2022b) to perform the experiments and we added MCTS-SPIBB to the benchmark. The charts show that the improved policy generated by MCTS-SPIBB is safe (i.e., it never has performance lower than the baseline) on small datasets and has performance equivalent to SPIBB (hence comparable to the other state-of-the-art SPI algorithms) with larger datasets.

Figure 9 shows the performance of the improved policy computed by MCTS-SPIBB on a *WetChicken* environment based on a 70×70 grid, namely, with $|S| = 4900$. On this environment dimension, the state-of-the-art SPI algorithm fails because of the state space dimensionality (the algorithms could not run on our hardware for memory reasons). The policy generated by MCTS-SPIBB has instead higher performance than the baseline. The dataset used in these experiments has dimension $|\mathcal{D}| = 500000$ and the number of simulations used by MCTS-SPIBB is $m = 1000$. Comparisons with approximated methods, e.g., SPIBB-DQN (Laroche et al., 2019), are not shown because these methods do not provide theoretical guarantees (not even asymptotic) of optimality.