Scalable Safe Policy Improvement via Monte Carlo Tree Search

Alberto Castellini, Federico Bianchi, Edoardo Zorzi, Thiago D. Simao, Alessandro Farinelli, Matthijs T. J. Spaan

alberto.castellini@univr.it

International Conference on Machine Learning (ICML 2023)
26/07/2023
Honolulu, Hawaii, USA

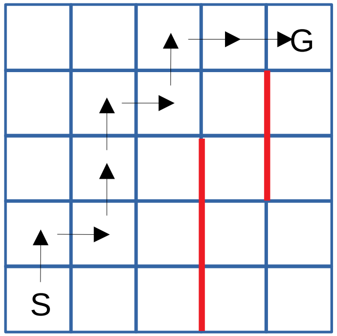
Overview

- **Problem definition:** Scalability in Safe Policy Improvement
- **Our contribution:** Monte Carlo Tree Search SPIBB (MCTS-SPIBB)
- **Results:** convergence, safety, scalability

Problem Definition: Scalability in Safe Policy Improvement

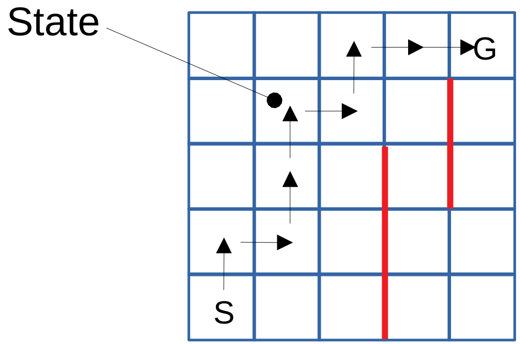
Environment

True MDP: M^{*}=<S, A, T*, R, γ>



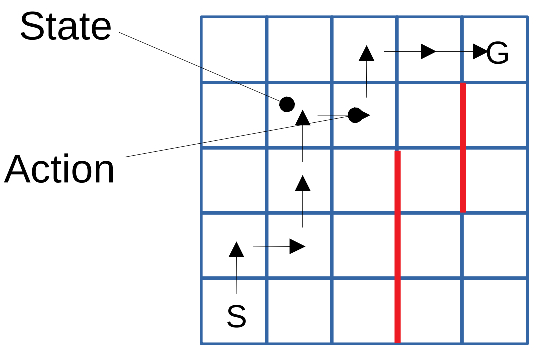
Environment

True MDP: M^*=<S, A, T*, R, γ>



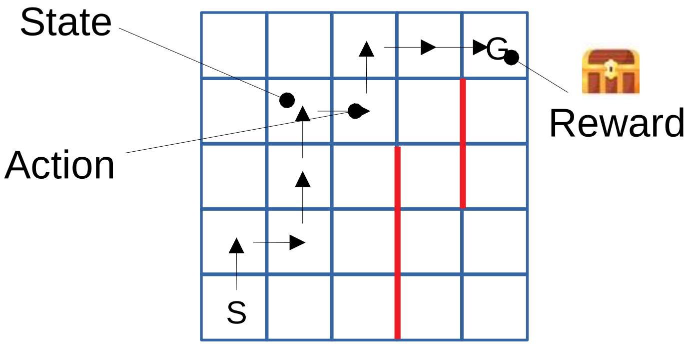
Environment

True MDP: M^{*}=<S, A, T*, R, γ>



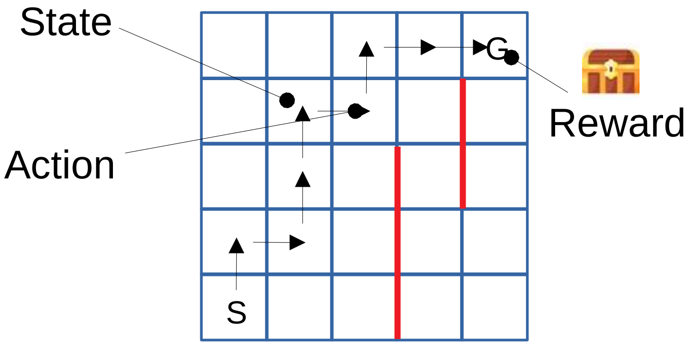
Environment

True MDP: M^*=<S, A, T*, R, γ>



Environment

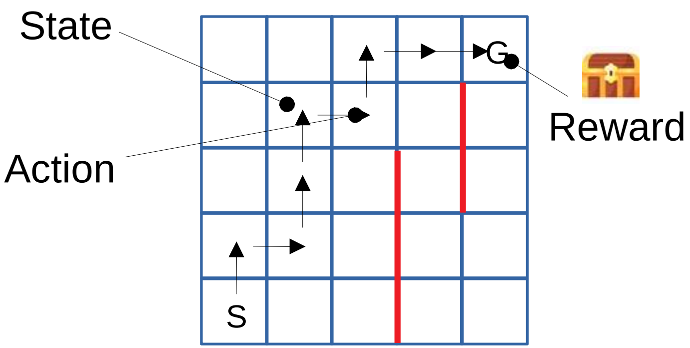
True MDP: $M^*=<S, A, T^*, R, \gamma>$



Transition model $T^*: S \times A \rightarrow \mathcal{P}(S)$

Environment

True MDP: M^{*}=<S, A, T*, R, γ>

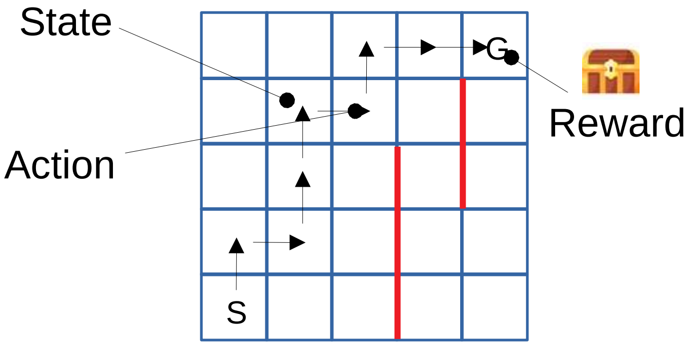


Transition model T^{*}: S x A → *P*(S)

Policy: π: S → *P*(A)

Environment

True MDP: M^*=<S, A, T*, R, γ>



Transition model T^*: S x A → *P*(S)

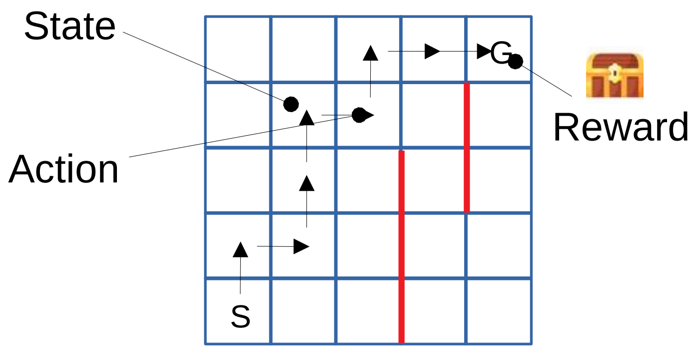
Policy: π: S → *P*(A)

Policy performance: expected return of the initial state $\rho(\pi, M) = V^{\pi}_{M}(s_0)$

Safe Policy Improvement

Environment

True MDP: M^*=<S, A, T*, R, γ>



Transition model T^*: S x A → *P*(S)

Policy: π: S → *P*(A)

Policy performance: expected return of the initial state $\rho(\pi, M) = V^{\pi}_{M}(s_0)$

Safe Policy Improvement (SPI)

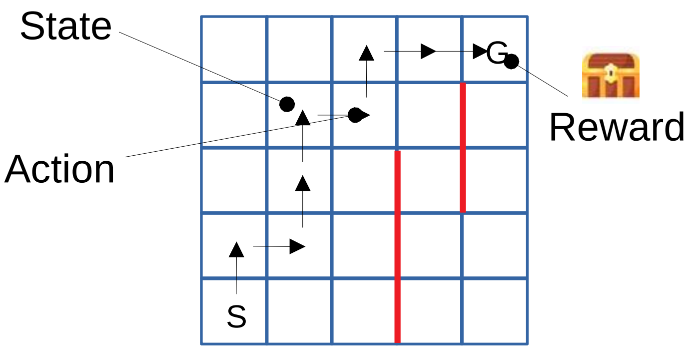
Baseline policy π_0

T^* unknown

Safe Policy Improvement

Environment

True MDP: $M^{*}=<S, A, T^{*}, R, \gamma>$



Transition model $T^{*}: S \times A \rightarrow P(S)$

Policy: $\pi: S \rightarrow P(A)$

Policy performance: expected return of the initial state $\rho(\pi, M) = V^{\pi}_{M}(s_0)$

Safe Policy Improvement (SPI)

Baseline policy π_0

T^{*} unknown

Goal

To generate a new policy π_I that **ouperforms** π_0 with an **admissible performance loss** $\xi \in R^{+}$ and **confidence level** δ, with $\delta \in [0,1]$

Safe Policy Improvement with Baseline Bootstrapping (SPIBB) [Laroche et al., 2019]

Baseline policy
π_0

↓

True Environment
<S, A, T*, R, γ>

↓

Dataset of trajecories (D)
s_0, a_0, s_0', r_0
$s_1, a_1, s_{1,}', r_1$
...

Safe Policy Improvement with Baseline Bootstrapping (SPIBB) [Laroche et al., 2019]

Baseline policy
π_0

True Environment
<S, A, T*, R, γ>

Dataset of trajecories (D)
s_0, a_0, s_0', r_0
$s_1, a_1, s_{1,}', r_1$
...

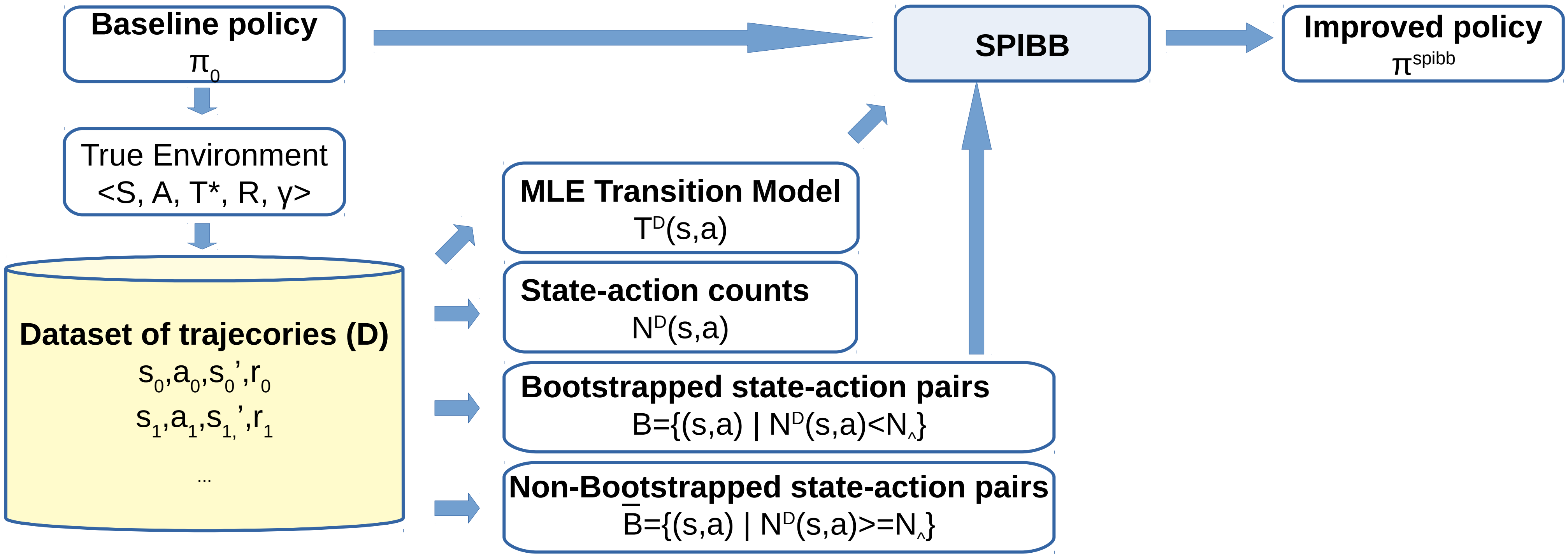
MLE Transition Model
$T^D(s,a)$

State-action counts
$N^D(s,a)$

Bootstrapped state-action pairs
$B=\{(s,a) \mid N^D(s,a)<N_\wedge\}$

Non-Bootstrapped state-action pairs
$\overline{B}=\{(s,a) \mid N^D(s,a)>=N_\wedge\}$

Safe Policy Improvement with Baseline Bootstrapping (SPIBB) [Laroche et al., 2019]



Safe Policy Improvement with Baseline Bootstrapping (SPIBB) [Laroche et al., 2019]

Baseline policy
π_0

True Environment
<S, A, T*, R, γ>

Dataset of trajecories (D)
s_0, a_0, s_0', r_0
s_1, a_1, s_1', r_1
...

MLE Transition Model
$T^D(s,a)$

State-action counts
$N^D(s,a)$

Bootstrapped state-action pairs
$B=\{(s,a) \mid N^D(s,a)<N_\wedge\}$

Non-Bootstrapped state-action pairs
$\bar{B}=\{(s,a) \mid N^D(s,a)>=N_\wedge\}$

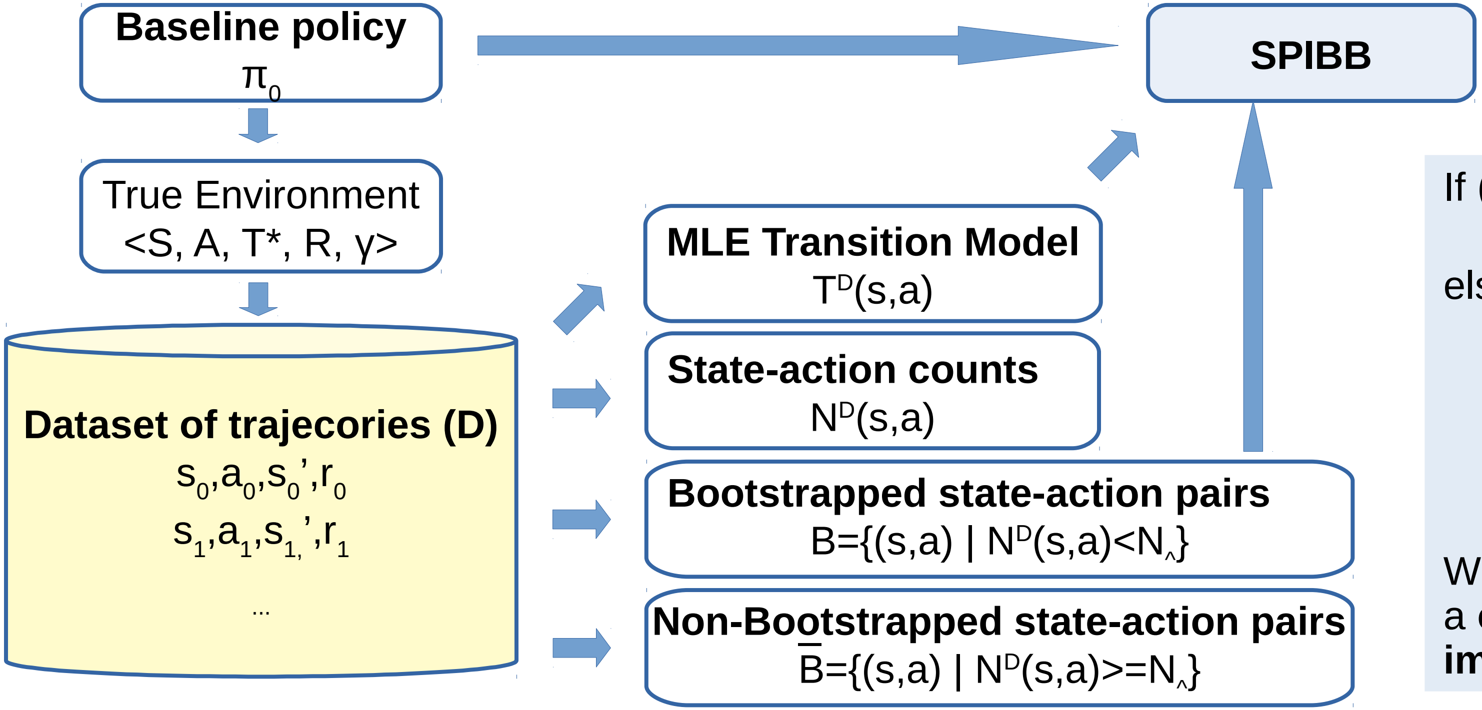
SPIBB

Improved policy
π^{spibb}

If $(s,a) \in B$ then
$\quad \pi^{spibb}(s,a)=\pi_0(s,a)$
else
$\quad$ If $(s,a)=\operatorname{argmax}_{s,a}(Q(s,a))$
$\quad\quad \pi^{spibb}(s,a)=1-\pi_0(s,a)$
$\quad$ Else
$\quad\quad \pi^{spibb}(s,a)=0$

Where Q(s,a) is computed using a **constrained** version of **policy improvement**

Safe Policy Improvement with Baseline Bootstrapping (SPIBB) [Laroche et al., 2019]



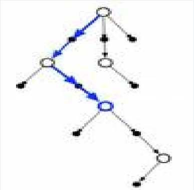
If $(s,a) \in B$ then
$\pi^{spibb}(s,a)=\pi_0(s,a)$
else
If $(s,a)=\text{argmax}_{s,a}(Q(s,a))$
$\pi^{spibb}(s,a)=1-\pi_0(s,a)$
Else
$\pi^{spibb}(s,a)=0$

Where Q(s,a) is computed using a **constrained** version of **policy improvement**

Problem: SPIBB complexity is O(|S|² * |A|), hence it cannot scale to large environments.
How can we make SPIBB scale?

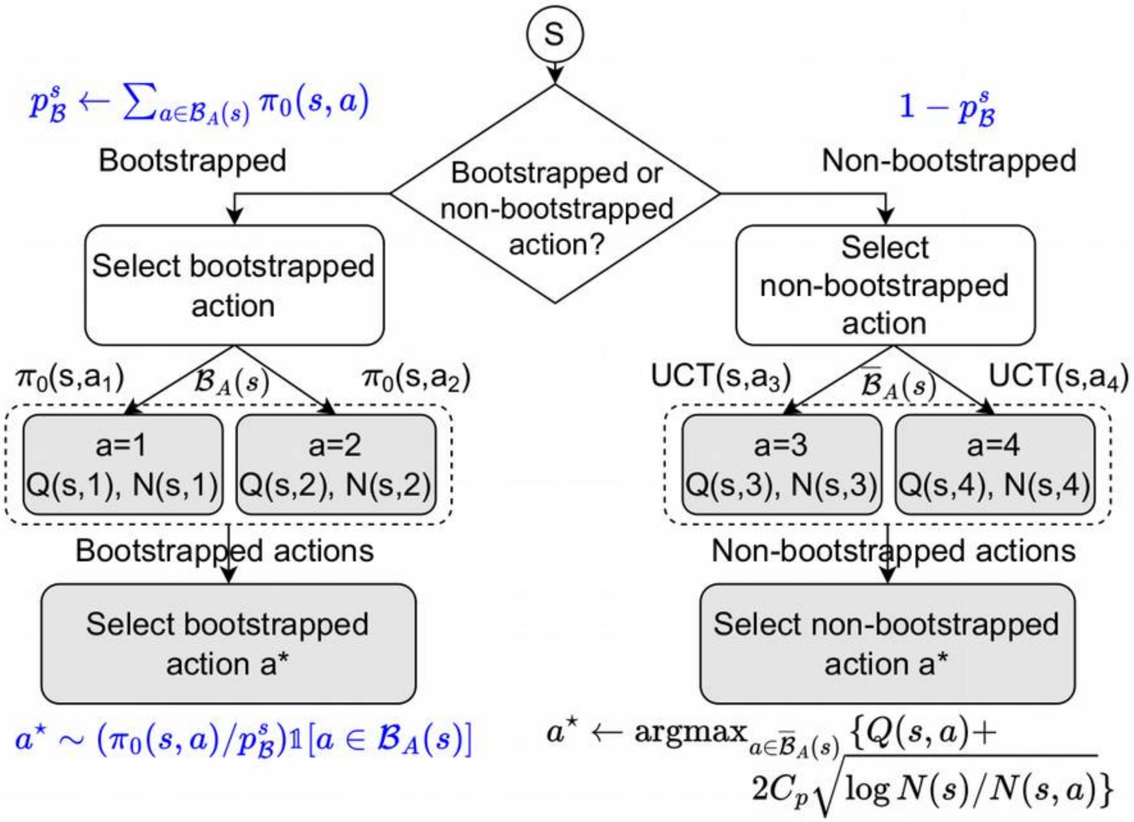
Monte Carlo Tree Search SPIBB (MCTS-SPIBB)

MCTS-SPIBB: Main Idea

SPIBB + MCTS  = MCTS-SPIBB

- **MCTS-SPIBB** uses **Monte Carlo Tree Search (MCTS)** to scale to large domains
- **Complexity**: depends on the number of **simulations** performed to generate the tree
- **Main contribution**: integration of the **SPIBB approach** in **UCT** by means of a suitable **action selection strategy**

MCTS-SPIBB: Action Selection Strategy



Main contribution: **This strategy is proved to make MCTS-SPIBB converge to SPIBB when the number of simulations tends to infinity**

MCTS-SPIBB: Algorithm

Algorithm 1 MCTS-SPIBB

Input: s: current state; π_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; π_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; π_0: baseline policy; $p^s_\mathcal{B}$: total probability of bootstrapped actions; $roll$: rollout flag

$\theta \sim \mathcal{U}([0,1])$ // Uniform sampling from $[0,1]$
if $\theta \leq p^s_\mathcal{B}$ **then**
  $p(\cdot) \leftarrow (0,\cdots,0)$ // Init. bootstrapped probabilities
  for $a \in \mathcal{B}_A(s)$ **do**
    $p(a) \leftarrow \pi_0(s,a)/p^s_\mathcal{B}$
  end for
  $a^\star \sim p(\cdot)$ // Sample bootstrapped action
else
  if $\neg roll$ **then**
    // Sample non-bootstrapped action using UCT
    $a^\star \leftarrow \text{argmax}_{a\in\overline{\mathcal{B}}_A(s)}\{Q(s,a) + 2C_p\sqrt{\frac{\log N(s)}{N(s,a)}}\}$
  else
    $a^\star \sim \pi_{rollout}(s,\cdot)$ // Sample uniformly
  end if
end if
return $a^\star$

Algorithm 4 Rollout

Input: s : state node; d: current depth; π_0: baseline policy; $\mathcal{B}_A(s)$: bootstrapped action set; $p^s_\mathcal{B}$: total probability of bootstrapped actions

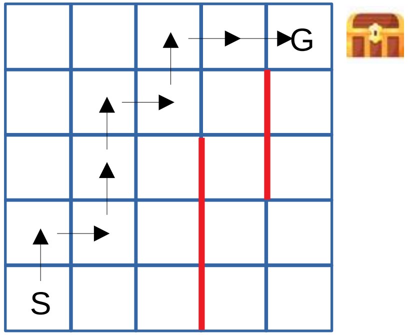
if $\gamma^d < \varepsilon$ **then**
  return 0
end if
$a^\star \leftarrow$ SelectAction($s, \mathcal{B}_A(s), \{\}, \pi_0, p^s_\mathcal{B}, True$)
$s' \sim T^\mathcal{D}(s,a^\star,\cdot)$; $r \leftarrow R(s,a^\star)$
return $r + \gamma \cdot$ Rollout($s', d+1, \pi_0, \mathcal{B}_A(s), p^s_\mathcal{B}$)

See the full MCTS-SPIBB algorithm in the paper

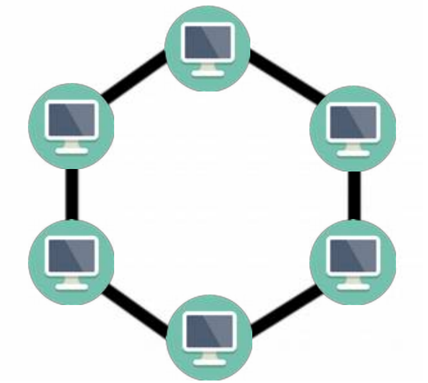
Results: convergence, safety, scalability

Domains

Gridworld

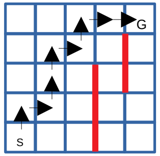


SysAdmin

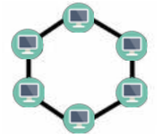


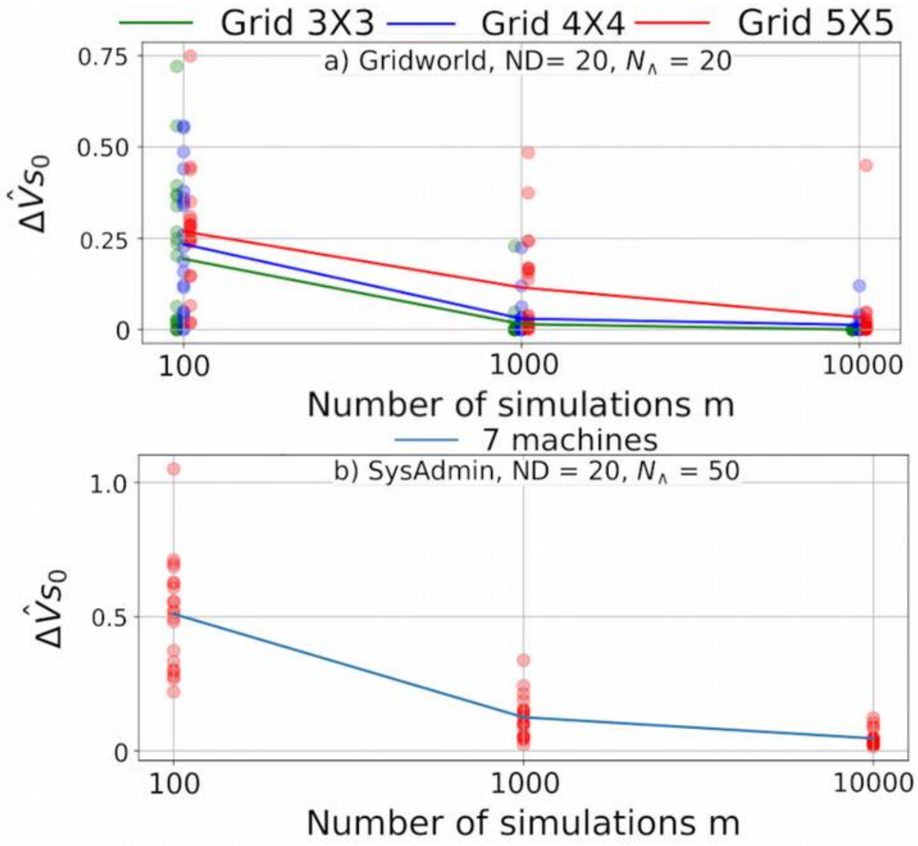
1. MCTS-SPIBB: convergence

SysAdmin



SysAdmin

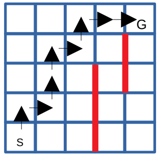


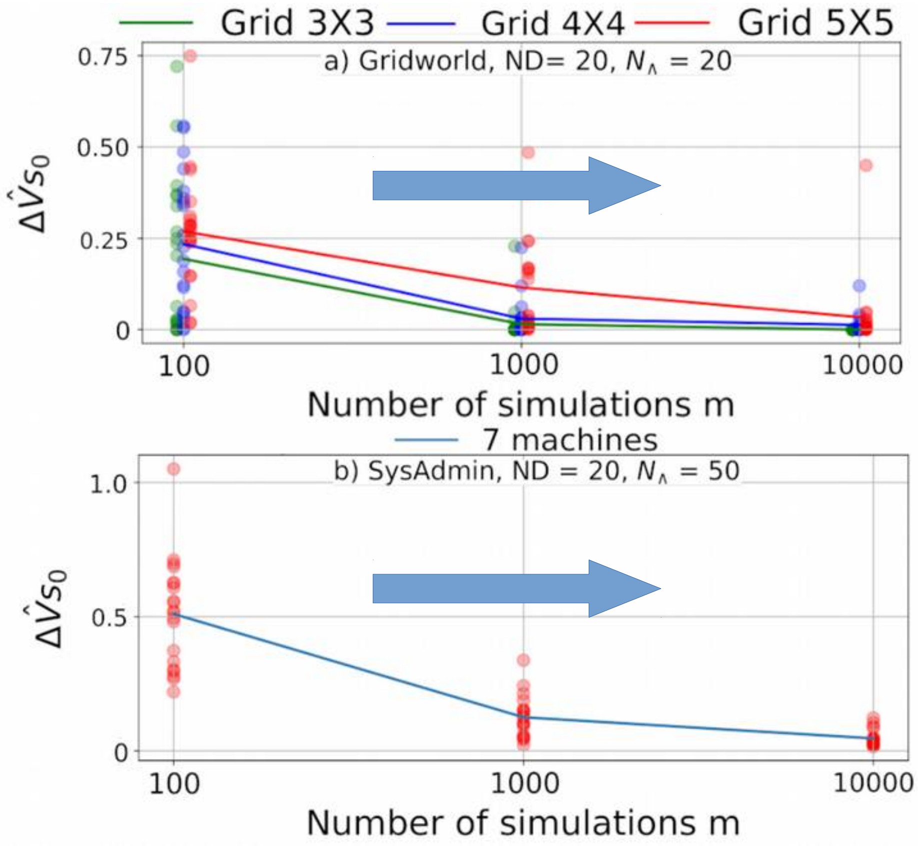


Theoretical analysis about MCTS-SPIBB convergence in the main paper (proofs in Appendix A)

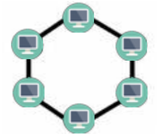
1. MCTS-SPIBB: convergence

SysAdmin





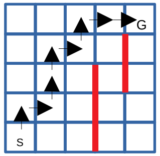
SysAdmin



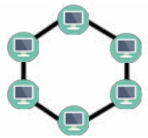
Theoretical analysis about MCTS-SPIBB convergence in the main paper (proofs in Appendix A)

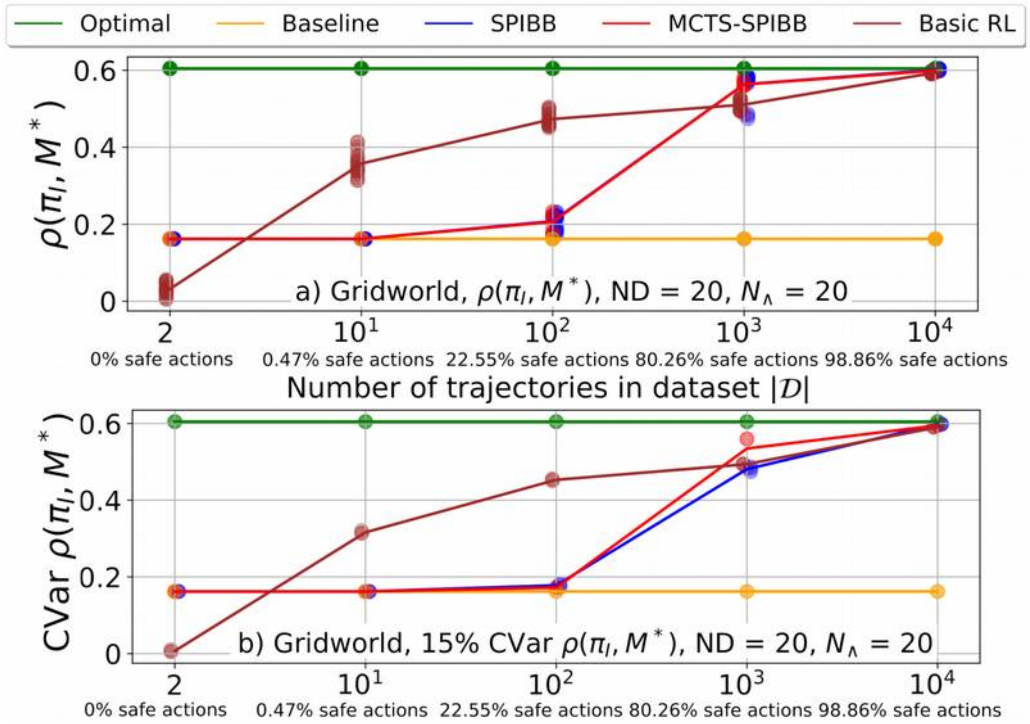
2. MCTS-SPIBB: Safety

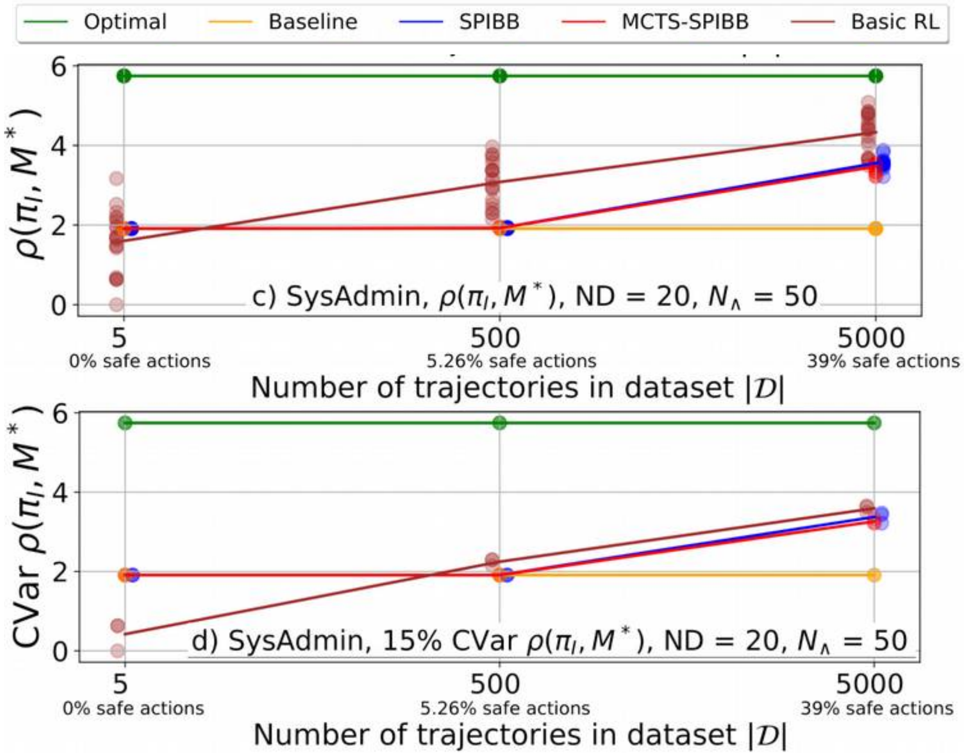
Gridworld



SysAdmin

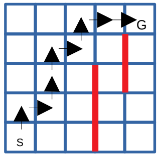




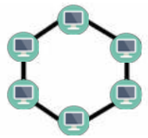


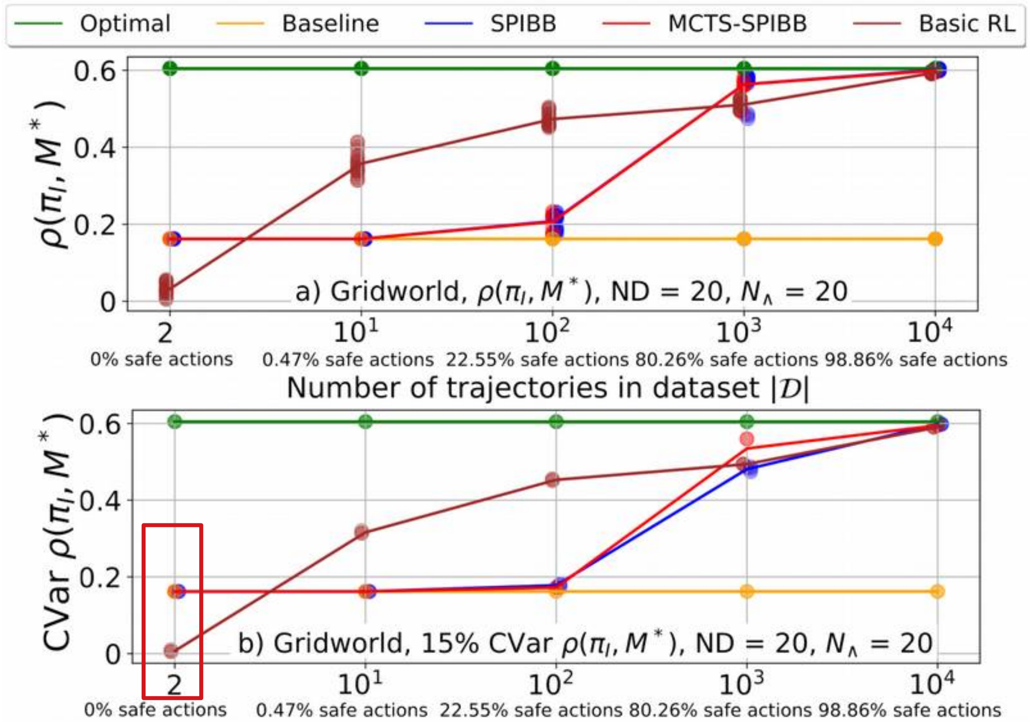
2. MCTS-SPIBB: Safety

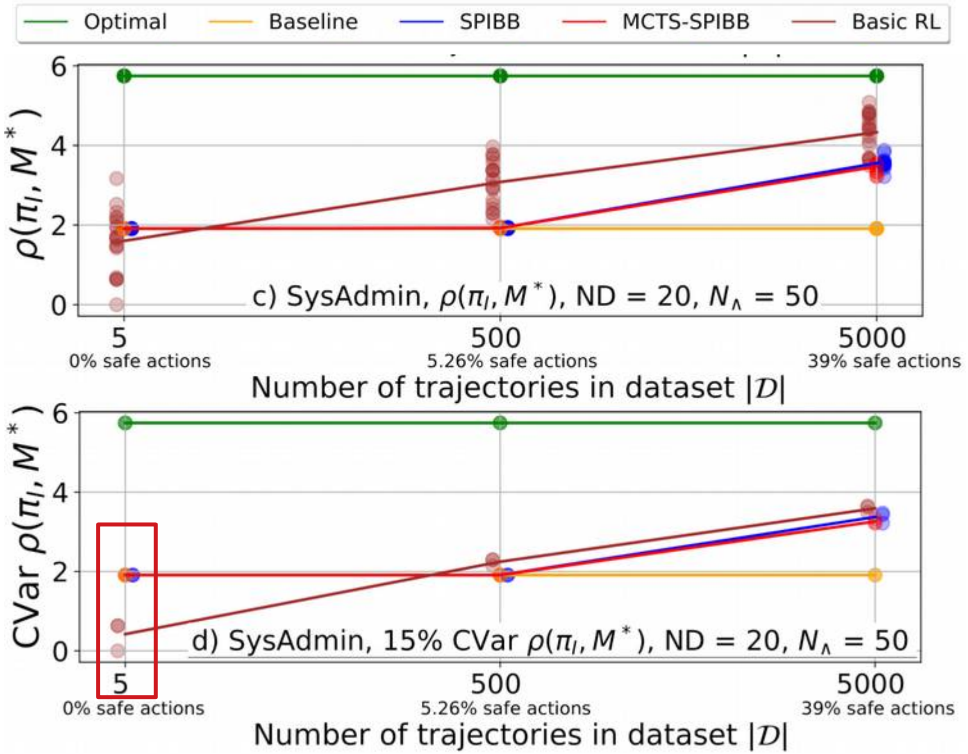
Gridworld



SysAdmin

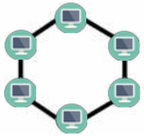


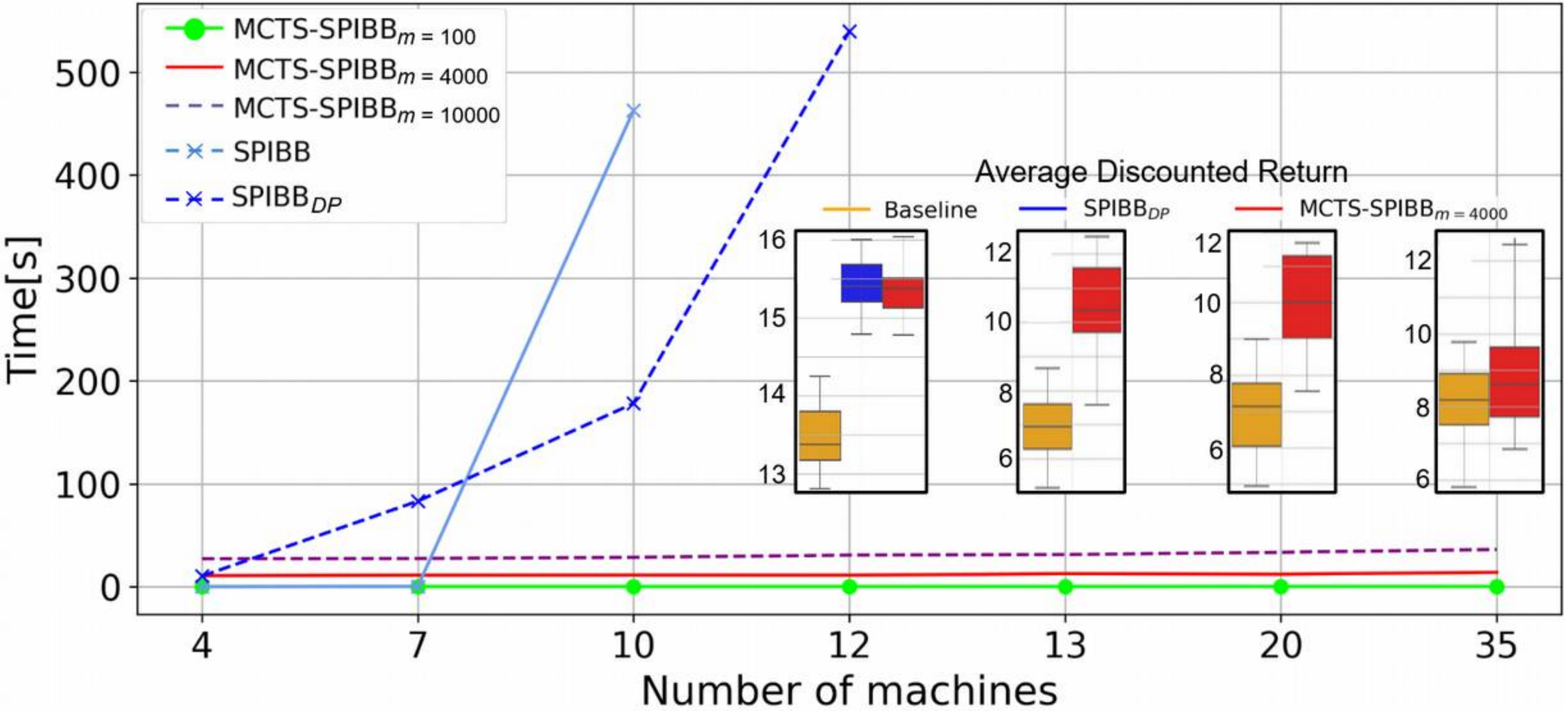




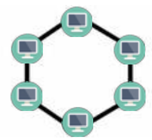
3. MCTS-SPIBB: Scalability

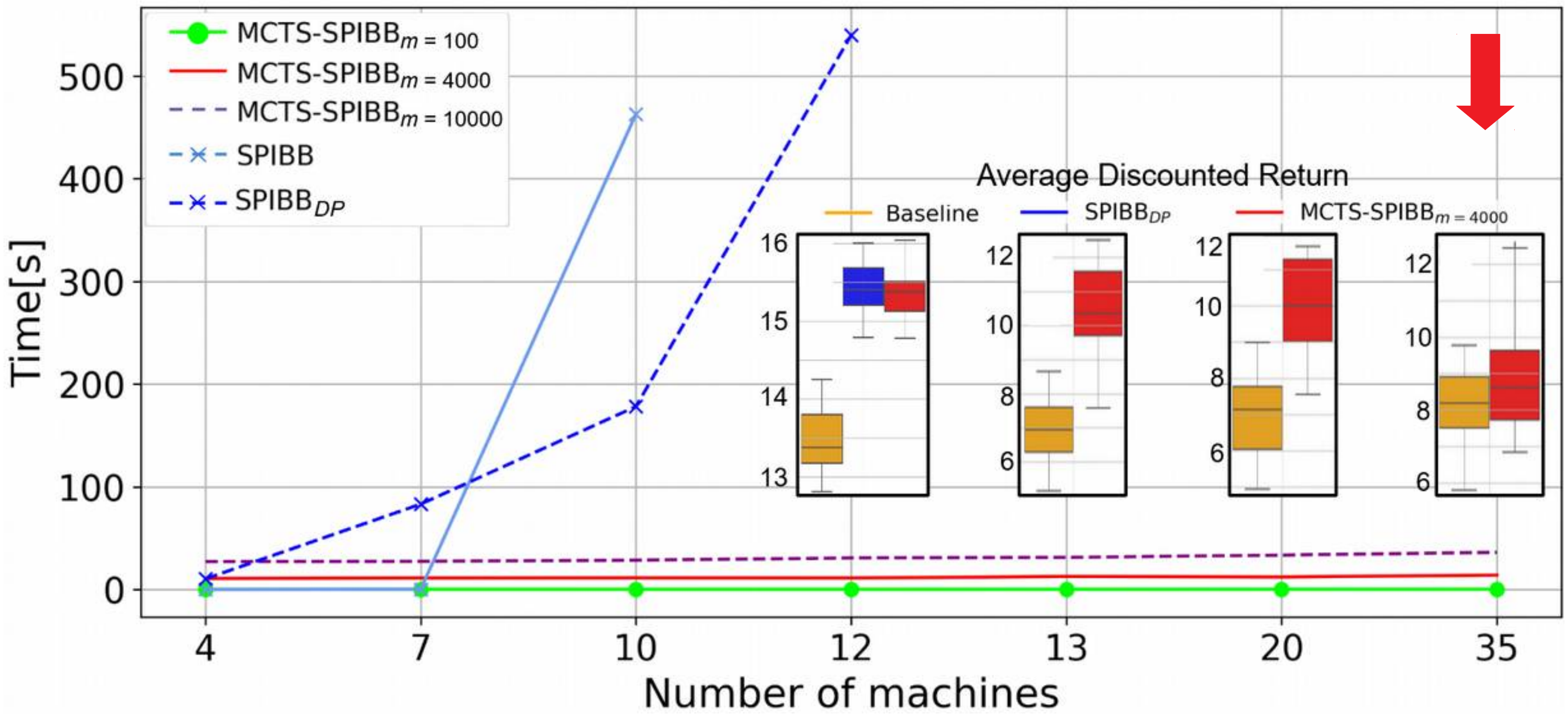
SysAdmin





SysAdmin





MCTS-SPIBB$_{m=100}$
MCTS-SPIBB$_{m=4000}$
MCTS-SPIBB$_{m=10000}$
SPIBB
SPIBB$_{DP}$
Time[s]
0
100
200
300
400
500
Number of machines
4
7
10
12
13
20
35
Average Discounted Return
Baseline
SPIBB$_{DP}$
MCTS-SPIBB$_{m=4000}$

Take Home Message

MCTS-SPIBB allows to **scale SPI** to **very large domains.**

This is an **important** result towards **applying SPI to real-world problems**.

Thank you!