Scalable Safe Policy Improvement for Factored Multi-Agent MDPs

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

Abstract

In this work, we focus on safe policy improvement in multi-agent domains where current state-of-the-art methods cannot be effectively applied because of large state and action spaces. We consider recent results using Monte Carlo Tree Search for Safe Policy Improvement with Baseline Bootstrapping and propose a novel algorithm that scales this approach to multi-agent domains, exploiting the factorization of the transition model and value function. Given a centralized behavior policy and a dataset of trajectories, our algorithm generates an improved policy by selecting joint actions using a novel extension of Max-Plus (or Variable Elimination) that constrains local actions to guarantee safety criteria. An empirical evaluation on multi-agent SysAdmin and multi-UAV Delivery shows that the approach scales to very large domains where state-of-the-art methods cannot work.

1. Introduction

Multi-Agent Reinforcement Learning (MARL) (Zhang et al., 2021; Chalkiadakis & Boutilier, 2003) has recently gained strong interest from the AI community due to the theoretical and application challenges it faces. Real-world applications of MARL are manifold. Some examples are fleets of mobile robots collecting and delivering goods in warehouses, or teams of drones used for environmental monitoring (Albrecht et al., 2023; Zuccotto et al., 2024). Significant progress has been made in virtual scenarios, such as competitive real-time strategy games (Vinyals et al., 2019), where agents can explore without safety concerns. However, in many real-world applications, collecting experience can be impractical due to the safety issues inherent in the learning process (Mazzi et al., 2023). A core foothold towards the deployment of both RL and MARL applications in real-world domains is the ability to reliably make good decisions with minimum interactions with the environment (Dulac-Arnold et al., 2021).

Offline RL takes a step in this direction by proposing algorithms capable of learning policies from batches of historical data collected by *behavior policies* (e.g., sub-optimal expert-designed policies) (Levine et al., 2020), but it faces reliability challenges due to distribution shifts (i.e., a policy different from the behavior policy induces a new distribution over the trajectories) which can result in high variability in the performance. In this context, *Safe Policy Improvement* (SPI) aims at improving the behavioral policy by exploiting information based on historical data and ensuring that the new policy outperforms the behavior policy (Laroche et al., 2019) with probabilistic guarantees.

Although some progress has been made in offline multi-agent settings (Ma & Wu, 2023), to the best of our knowledge, multi-agent methods do not have the reliability guarantees of SPI algorithms. Multi-agent problems are challenging due to the combinatorial number of joint actions, which grows exponentially with the number of agents, as they need to coordinate to achieve optimal behavior (Oliehoek & Amato, 2016). Furthermore, reliability becomes an issue in the offline multi-agent setting due to data availability, resulting from limited exploration of the behavior policy in large-scale problems. A fully observable multi-agent problem can be treated as a single-agent problem with a centralized controller responsible for coordinating the agents. However, this approach is severely limited due to the large joint action space, which makes policy optimization intractable. To tackle these problems, in this work *we propose a factorized SPI approach for large-scale multi-agent problems, enhancing both sample efficiency and scalability.*

Previous work has demonstrated that by exploiting the independencies between state variables represented by a factorized transition model, an offline RL algorithm may need orders of magnitude fewer samples to compute an improved policy (Simão & Spaan, 2019). Additionally, a

[1]Department of Computer Science, University of Verona, Verona, Italy [2]Department of Software Science, Eindhoven University of Technology, Eindhoven, Netherlands [3]Department of Intelligent Systems, Delft University of Technology, Delft, Netherlands. Correspondence to: Federico Bianchi <federico.bianchi@univr.it>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

recent work (Castellini et al., 2023) proposes the application of Monte Carlo Tree Search (MCTS) to Safe Policy Improvement with Baseline Bootstrapping (SPIBB) (Laroche et al., 2019), a state-of-the-art SPI method, showing asymptotic convergence as the number of simulations increases. MCTS-SPIBB (Castellini et al., 2023) enhances the time efficiency and scalability of SPIBB in single-agent scenarios with large state spaces but it struggles in multi-agent settings because the size of the joint action space becomes too large. The three main contributions of this work, summarized in the following, aim to overcome these limitations of MCTS-SPIBB and to propose a novel approach to perform SPI in multi-agent scenarios.

Our first contribution (regarding the scalability of multi-agent SPI) concerns the proposal of a scalable Factored-Value MCTS-based multi-agent SPI algorithm (FV-MCTS-SPIBB). It exploits the factorization of the value function and employs two novel action selection strategies, Constrained Max-Plus and Constrained Var-El. These strategies differ from the standard Max-Plus (Wainwright et al., 2004) and Var-El (Guestrin et al., 2003) that have already been employed for policy optimization (FV-MCTS) (Choudhury et al., 2021; Amato & Oliehoek, 2015) but never for SPI. The local computations, message-passing for Constrained Max-Plus and maximization for Constrained Var-El are carefully crafted to ensure that the SPIBB safety constraints on joint actions are always satisfied. The proposed method differs significantly from MCTS-SPIBB (Castellini et al., 2023), which is a single-agent method relying on a standard computation of the value function, that cannot, for this reason, reliable scale to multi-agent scenarios with large joint action spaces. Our method, instead, tractably selects cooperative actions by decomposing the global payoff function into a sum of local terms. Using the constrained action-selection strategies, FV-MCTS-SPIBB can select optimal joint actions without evaluating the entire (huge) joint action space.

Our second contribution (regarding multi-agent SPI sample efficiency) is an efficient local state-action counting based on the factorization of the transition model. This method computes state-action counts considering local actions instead of joint actions, thus extending the strategy proposed in (Simão & Spaan, 2019). In this way, the proposed approach obtains larger counts (exploiting the prior knowledge in the factorization of the transition model) which are then exploited by FV-MCTS-SPIBB to generate better policies. FV-MCTS-SPIBB can improve the baseline policy only in non-bootstrapped state-action pairs, namely, pairs having counts larger than a threshold.

Our third contribution is a theoretical analysis combined with an empirical evaluation of the proposed method. The theoretical analysis shows that our algorithm, under the assumptions of the factorization of the transition model and the value function, asymptotically converges to SPIBB (Laroche et al., 2019). This ensures both the optimality and safety of our approach. The empirical evaluation shows that our method can significantly improve the behavior policy in large-scale multi-agent domains for which no state-of-the-art SPI algorithms can compute improved policies with safety guarantees. We consider two large multi-agent domains, multi-agent SysAdmin (Guestrin et al., 2003) and multi-UAV Delivery (Choudhury et al., 2021). These environments have state space sizes up to 10^{41} and action space sizes up to 10^{16}, which are orders of magnitude larger than those used in the SPI literature[1].

2. Related Work

The main research fields related to our work are SPI and multi-agent planning/RL.

Safe Policy Improvement. Safe policy improvement is a sub-field of offline reinforcement learning that aims to improve a given policy π_0 while guaranteeing specific performance bounds compared to π_0 (Levine et al., 2020). Scholl et al. (2022) introduce a taxonomy of safe policy improvement methods, categorizing them based on how they utilize uncertainty in state-action pairs. A category of interest concerns constrained learnable policies, and in this context Delage & Mannor (2010) introduce a percentile criterion for SPI. Petrik et al. (2016) bound this criterion from below within the set of admissible policies. The percentile criterion from Delage & Mannor (2010) is also reformulated by Laroche et al. (2019) to allow deviations from the behavior policy in state-action pairs with sufficiently low uncertainty. This is achieved by bootstrapping the behavior policy in state-action pairs with high uncertainty. The approach has been specialized to factored domains, maintaining policy iteration as a policy optimization technique (Simão & Spaan, 2019). The algorithm proposed by Laroche et al. (2019) was also re-implemented using MCTS to improve scalability (Castellini et al., 2023). However, despite various extensions and alternative strategies, none of these approaches has been applied to large multi-agent systems, as we do in this paper because they cannot be used effectively with large action spaces.

Multi-agent Planning and RL. In the context of multi-agent planning and RL (Oliehoek & Amato, 2016; Kochenderfer et al., 2022; Albrecht et al., 2023), there exist offline and online methods. *Offline* methods involve computing policies for the entire state space (Oliehoek & Amato, 2016) with time and space complexity usually dependent on the

[1]Notice that the environments are much more complex than those used in the SPI literature. For instance, Scholl et al. (2022) and Laroche et al. (2019) use MDPs with 25 states and fewer than 10 actions.

size of the state and action spaces, therefore they are not suitable for large multi-agent scenarios. *Online* methods, such as MCTS-based methods (Browne et al., 2012), can scale because they only deal with states accessible from the current state, but they still encounter challenges when dealing with large action spaces, in particular in multi-agent domains (Zerbel & Yliniemi, 2019). Amato & Oliehoek (2015) use coordination graph and factorized global payoff function to define interactions among agents in partially observable environments and compute optimal centralized policies with Var-El (Guestrin et al., 2003) under specific assumptions. Choudhury et al. (2021) use MCTS with Max-Plus (Pearl, 1988) to generate optimal policies in multi-agent settings and extends the approach presented by Amato & Oliehoek (2015) to a scalable anytime method. However, none of these methods is designed for SPI. Instead, the approach proposed in this paper specifically addresses multi-agent SPI.

3. Background

This section describes state-of-the-art mathematical frameworks and algorithms used in the rest of the paper. Due to space limitations, we only define the main notation here. Markov Decision Processes, Multi-Agent Markov Decision Processes and MCTS are described in Section A of the supplementary material.

3.1. Factored Multi-Agent Markov Decision Processes

A Factored Multi-Agent Markov Decision Process (FMMDP) (Guestrin et al., 2003; Strehl et al., 2007) is a tuple $M = \langle S, \alpha, \{A_i\}_{i\in\alpha}, T, R, \gamma\rangle$, where S represents the set of all states; α denotes a finite set of n agents; $A = \times_i A_i$ represents a finite set of joint actions, with each A_i indicating the set of actions available to agent i; $T : S \times A_1 \times \cdots \times A_n \to \mathcal{P}(S)$ is a stochastic transition function that takes a joint state $\bar{s} \in S$ and a joint action $\bar{a} \in A$ as input and returns a probability distribution over the next states; $R : S \times A \to \bar{R} \in [-R_{min}, R_{max}]$[2] is a bounded stochastic reward function and $\gamma \in [0, 1]$ is the discount factor. To evaluate the overall performance of a policy π within the FMMDP M, we compute its expected return in the initial state $\bar{s}_0$ as $\rho(\pi, M) = V^{\pi}_{M}(\bar{s}_0)$, where $V^{\pi}_{M}(\bar{s})$ represents the state value of $\bar{s}$. $Q^{\pi}_{M}(\bar{s}, \bar{a})$ is the value of action $\bar{a}$ in state $\bar{s}$. Additionally, V_{max} represents the known upper bound of the absolute value of the return, with the inequality $V_{max} \leq {}^{R_{max}}/_{1-\gamma}$ providing a specific constraint on this upper bound. In FMMDP the state space is factored as $S = S_1 \times \cdots \times S_K$, such that $\bar{s}[S_k] \in Dom(S_k)$ indicates the value assigned to factor S_k in the joint state $\bar{s}$. The outcome of each factor S_k is independent of the remaining, so the probabilistic transition function can be written as $T(\bar{s}' \mid \bar{s}, \bar{a}) = \prod_{k=1}^{K} P_k(\bar{s}'[S_k] \mid \bar{s}, \bar{a})$, where P_k is the conditional probability distribution of the state variable S_k. A factored representation reduces sample complexity and generalizes better from deterministic behavior policies (Simão & Spaan, 2019). Given a set of dependency identifiers $\mathcal{J}$ to indicate common dependencies between different state-action pairs, we can define the dependency functions $D_k : S \times A \to \mathcal{J}$. Using this function, the probabilistic transition function can be compactly represented by $T(\bar{s}' \mid \bar{s}, \bar{a}) = \prod_{k=1}^{K} P_k(\bar{s}'[S_k] \mid D_k(\bar{s}, \bar{a}))$. Intuitively, given a factor S_k and two state-action pairs $\bar{s}, \bar{a}$ and $\bar{s}', \bar{a}'$, if $D_k(\bar{s}, \bar{a}) = D_k(\bar{s}', \bar{a}')$ then $P_k(\cdot \mid \bar{s}, \bar{a}) = P_k(\cdot \mid \bar{s}', \bar{a}')$.

Coordination graphs and value function factorization. Coordination graphs (CGs) (Guestrin et al., 2001; 2003) are a standard tool for representing agent coordination mechanisms. A CG is a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E}\rangle$ with a node $i \in \mathcal{V}$ for each agent and an edge $(i, j) \in \mathcal{E}$ for each pair of agents i and j with interdependent payoff. The global payoff can be factorized considering the coordination graph as $Q(\bar{a}) = \sum_{i\in\mathcal{V}} Q_i(a_i) + \sum_{(i,j)\in\mathcal{E}} Q_{ij}(a_i, a_j)$ (Choudhury et al., 2021). Here, Q_{ij} represents a local payoff function for agents i and j connected by an edge (i, j), while Q_i denotes the individual utility function for each agent (refer to Section B of the supplementary material for a detailed description).

3.2. FV-MCTS with Max-Plus and Var-El

To make MCTS scale to large (multi-agent) domains with an exponential number of actions, Choudhury et al. (2021) propose Factored-Value Monte Carlo Tree Search with Max-Plus (FV-MCTS-Max-Plus). This approach uses a factored representation of the payoff function induced by a coordination graph and introduces Max-Plus for action selection (see Algorithm 4 and 5 in the supplementary material). The joint action is computed through message passing for a maximum number of rounds, where iteratively each agent i sends a message to its neighbors $j \in \Gamma(i)$. At the end of the message passing phase, each agent i selects its optimal action by selecting the action a_i that maximizes the function $\{Q_i(a_i) + \sum_{j\in\Gamma(i)} \mu_{ji}(a_i)\}$. Max-Plus's complexity scales linearly with the number of edges of the CG (Pearl, 1988). An alternative version of the algorithm using Variable Elimination instead of Max-Plus (Amato & Oliehoek, 2015) is FV-MCTS-Var-El (see Section C in the supplementary material). Var-El guarantees convergence to optimal action selection for any type of CG but requires significantly more computation than Max-Plus. Max-Plus guarantees convergence only in acyclic graphs but empirically provides very good results also on cyclic structures (Choudhury et al., 2021).

[2] According to the definition of MMDP (Boutilier, 1996), the reward function can be defined globally or individually for each agent. We use the second possibility. Consequently, the update of Q-values is performed based on a vector of rewards.

3.3. Safe Policy Improvement

The SPIBB (Laroche et al., 2019) algorithm aims to compute a new policy π_I that outperforms the behavior policy π_0 with high probability. Specifically, the improvement is performed considering the following criterion

$$\max_{\pi_I \in \Pi} \rho(\pi_I, M^{\mathcal{D}}) \text{ s.t. } \rho(\pi_I, M) \geq \rho(\pi_0, M) - \zeta, \forall M \in \Xi,$$

namely, SPI ensures that π_I, learned on the Maximum Likelihood Estimation (MLE) MDP $M^{\mathcal{D}}$, outperforms π_0 with no more than a specified performance loss ζ and a confidence level of $\delta \in [0, 1]$ in the set of admissible MDPs Ξ. In practice, the behavior policy π_0 is executed in the true environment M^*, where a dataset is collected and then used to compute the estimated MDP $M^{\mathcal{D}}$. Two disjoint sets of state-action pairs are also computed: the *bootstrapped* set, which contains pairs that have not been observed enough times in the dataset, and its complement, the *non-bootstrapped* set. These sets are then used by SPIBB in a modified policy iteration (Sutton & Barto, 2018) procedure to compute an optimal policy that follows the behavior policy π_0 in the bootstrapped state-action pairs while improving π_0 in the non-bootstrapped pairs. Further details on SPIBB are provided in Section D of the supplementary material.

In the factored SPIBB setting (Simão & Spaan, 2019) global state-action counts (considering joint states) are replaced by more efficient local counts (considering factored states), and the set of bootstrapped state-action pairs is defined as:

$$\mathcal{B}_m = \left\{(\bar{s}, \bar{a}) \in S \times A \mid \exists S_k : n_{\mathcal{D}}(D_k(\bar{s}, \bar{a})) < m_k\right\} \quad (1)$$

with $n_{\mathcal{D}} : \mathcal{J} \to \mathbb{N}$ counting how many times a component has been observed in dataset $\mathcal{D}$ and m_k is the count threshold for factor S_k. Factored SPIBB shows that any *optimal policy* from the set of constrained policies:

$$\Pi_0 = \left\{\pi \mid \pi(\bar{s}, \bar{a}) = \pi_0(\bar{s}, \bar{a}), (\bar{s}, \bar{a}) \in \mathcal{B}_m\right\} \quad (2)$$

ensures SPI guarantees. Defining $\mathcal{B}(\bar{s}) = \{\bar{a} \mid (\bar{s}, \bar{a}) \in \mathcal{B}_m\}$ as the set of bootstrapped actions for each state $\bar{s}$, an *optimal policy* $\pi^\star$ from Π_0 assigns the same probabilities to bootstrapped actions as the behavior policy, i.e., $\pi^\star(\bar{s}, \bar{a}) = \pi_0(\bar{s}, \bar{a})$ for all $\bar{a} \in \mathcal{B}(\bar{s})$, and the remaining probability $p = 1 - \sum_{\bar{a} \in \mathcal{B}(\bar{s})} \pi_0(\bar{s}, \bar{a})$ to the best non-bootstrapped action, i.e., action $\bar{a}^\star$ with the highest Q-value s.t. $\bar{a}^\star \notin \mathcal{B}(\bar{s})$.

4. Method

We first provide an overview of our method for multi-agent SPI, named Factored-Value MCTS-SPIBB (FV-MCTS-SPIBB), and then describe the algorithm in detail. The safe policy improvement algorithm FV-MCTS-SPIBB proposed introduces two original elements. First, an efficient technique for computing state-action counts that splits the full dataset $\mathcal{D}$ into datasets, according to the factorization of the transition model and it computes local state-action counts for each factor (see Section 4.1). Second, a scalable technique for safely computing the improved policy based on the joint use of FV-MCTS and Constrained Max-Plus (or Constrained Var-El) to SPI (see Section 4.2).

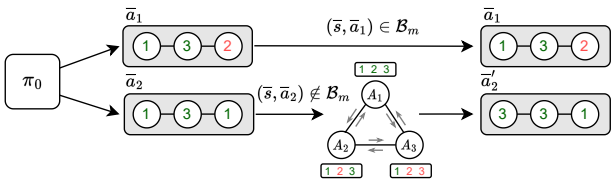

Figure 1. Action selection strategy (Algorithm 2). A synthetic example with three agents (circles), each one with three local actions. In this case, an agent corresponds to a factor. In red "bootstrapped" local actions (i.e., $n_{\mathcal{D}}(D_k(\bar{s}, \bar{a})) < m_k$), in green "non-bootstrapped" local actions. On top: a "bootstrapped" local action makes the joint action $\bar{a}_1$ bootstrapped, so it is returned as it is; at the bottom, only "non-bootstrapped" local actions have been selected so $\bar{a}_2$ is also non-bootstrapped: the agents communicate via message-passing to compute and return the best non-bootstrapped joint action.

4.1. Efficient local state-action counting

Let the FMMDP $M^* = \langle S, \alpha, \{A_i\}_{i \in \alpha}, T^*, R, \gamma \rangle$ be the true environment, and $\mathcal{D}$ a dataset collected using the behavior policy π_0 in M^*. Each sample stored in $\mathcal{D}$ is a quadruple $(\bar{s}, \bar{a}, \bar{s}', \bar{r})$. The dataset is used to compute the MLE FMMDP $M^{\mathcal{D}} = \langle S, \alpha, \{A_i\}_{i \in \alpha}, T^{\mathcal{D}}, R, \gamma \rangle$, where $T^{\mathcal{D}}$ is factorized according to dependency functions D_k. From $\mathcal{D}$, we compute the *component counters*, $n_{\mathcal{D}}(j) = \sum_{\bar{s}, \bar{a}, \bar{s}' \in \mathcal{D}} \sum_k \mathbb{1}(D_k(\bar{s}, \bar{a}) = j)$, and the *realization counters*, $n_{\mathcal{D}}(s'_k, j) = \sum_{\bar{s}, \bar{a}, \bar{s}' \in \mathcal{D}} \mathbb{1}(D_k(\bar{s}, \bar{a}) = j$ and $s'[S_k] = \bar{s}'_k)$, where $j \in \mathcal{J}$ and $s'_k \in Dom(S_k)$. Based on these counters, we can estimate each transition component of $T^{\mathcal{D}}$ as defined in Section A of the supplementary material, namely, $\hat{P}_k\big(s'[S_k] \mid j\big) = \frac{n_{\mathcal{D}}(s'_k, j)}{n_{\mathcal{D}}(j)}$.

We extend the approach proposed in (Simão & Spaan, 2019) by considering local actions a_i involved in the component D_k rather than using joint actions $\bar{a}$. This results in larger counts compared to standard counts using joint actions (see (Simão & Spaan, 2019)) with a related reduction of the set of bootstrapped joint state-action pairs. The efficient state-action counting exploits the factorization of the transition model and state-action counting at the agent level. This is fundamental for the algorithm because flat approaches with state-action counting at the joint state and joint action levels obtain very low counts, i.e., a very small number of state-action pairs would be put in the non-bootstrapped set using such a counting method. An in-depth explanation of the local state-action counting strategy with a dedicated

Algorithm 1 Factored-Value MCTS-SPIBB

Require: state $\bar{s}$; behavior policy π_0; MLE transition model $T^{\mathcal{D}}$, simulation *depth*; exploration constant c; coordination graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$;

1: $N_i(\bar{s}, a_i) = Q_i(\bar{s}, a_i) = 0 \; \forall i \in \mathcal{V}, \forall a_i \in A_i$
2: $N_{ij}(\bar{s}, a_i, a_j) = Q_{ij}(\bar{s}, a_i, a_j) = 0 \; \forall (i,j) \in \mathcal{E}, a_i \in A_i, a_j \in A_j$

3: **function** FV-MCTS-SPIBB($\bar{s}$, depth)
4: **while** time limit not reached **do**
5: SIMULATE($\bar{s}$, depth)
6: $\bar{a}^\star$ = ACTION-SELECTION(0) ▷ Algorithm 2
7: **return** $\bar{a}^\star$ ▷ Best joint action

8: **function** SIMULATE($\bar{s}$, depth)
9: **if** depth = 0 **then**
10: **return** $\bar{0}$
11: $\bar{a}$ = ACTION-SELECTION(c) ▷ Algorithm 2
12: $\bar{s}', \bar{r} \sim T^{\mathcal{D}}(\bar{s}, \bar{a}), R(\bar{s}, \bar{a})$
13: $\bar{r}' = \bar{r} + \gamma \cdot$ SIMULATE($\bar{s}'$, depth - 1)
14: UPDATE-STATS($\bar{s}$, $\bar{a}$, $\bar{r}'$)

Algorithm 2 Action Selection

Require: State $\bar{s}$; behavior policy π_0; non-bootstrapped action optimization type *Opt*

1: **function** ACTION-SELECTION(c)
2: $\bar{a} \sim \pi_0(\bar{s}, \cdot)$
3: **if** $(\bar{s}, \bar{a}) \in \mathcal{B}_m$ **then**
4: **return** $\bar{a}$
5: **else**
6: **if** *Opt=Max-Plus* **then**
7: $\bar{a}$ = MAX-PLUS(c) ▷ Algorithm 3
8: **else**
9: $\bar{a}$ = VAR-EL(c) ▷ Algorithm 8
10: **return** $\bar{a}$

Algorithm 3 Constrained Max-Plus

Require: Coordination Graph $\mathcal{G}=\langle \mathcal{V}, \mathcal{E} \rangle$; state node statistics N, N_i, N_{ij}, Q_i, Q_{ij}; max iterations M; node exploration constant c

1: **function** MAX-PLUS(c)
2: $\bar{\mathcal{B}}_m^{A_i}(\bar{s}) = \{a \in A_i \mid n_{\mathcal{D}}(D_k(\bar{s}, \bar{a})) \geq m_k, \forall k, \forall \bar{a} \in A[A_i = a]\}, \forall i$
3: $\mu_{ij}(a_j) = 0$ for $(i,j) \in \mathcal{E}, a_i \in \bar{\mathcal{B}}_m^{A_i}(\bar{s}), a_j \in \bar{\mathcal{B}}_m^{A_j}(\bar{s})$
4: **for** t = 1 to M **do**
5: **for** every agent i **do**
6: **for** all neighbors $j \in \Gamma(i)$ **do**
7: $\mu_{ij}(a_j) = \max_{a_i \in \bar{\mathcal{B}}_m^{A_i}(\bar{s})} \{Q_i(a_i) + Q_{ij}(a_i, a_j) + \sum_{\ell \in \Gamma(i) \setminus \{j\}} \mu_{\ell i}(a_i)\}$
8: $\mu_{ij}(a_j)$ -= $\frac{1}{|\bar{\mathcal{B}}_m^{A_j}(\bar{s})|} \sum_{a_j \in \bar{\mathcal{B}}_m^{A_j}(\bar{s})} \mu_{ij}(a_j)$
9: Send message $\mu_{ij}(a_j)$ to agent j
10: **if** $\mu_{ij}(a_j)$ close to previous msg. **then**
11: break
12: **for** every agent i **do**
13: $q_i(a_i) = Q_i(\bar{s}, a_i) + \sum_{j \in \Gamma(i)} \mu_{ji}(a_i)$
14: **if** $c \neq 0$ **then**
15: **for** all neighbors $j \in \Gamma(i)$ **do**
16: $q_i(a_i)$ += $c\sqrt{\frac{\log(N+1)}{N_i(\bar{s}, a_i)}}$
17: $\bar{a}_i = \text{argmax}_{a_i \in \bar{\mathcal{B}}_m^{A_i}} q_i(a_i)$
18: **return** $\bar{a}$ ▷ Constrained selection $\bar{a} \in \bar{\mathcal{B}}_m$

experiment on sample efficiency and more details about the dependency function D_k can be found in Sections I and K of the supplementary material.

4.2. Scalable multi-agent SPI: FV-MCTS-SPIBB

The proposed algorithm **FV-MCTS-SPIBB (Algorithm 1)** performs the SPI of a multi-agent behavior policy π_0 by simulating from the current state to generate the MC tree which estimates action values. What characterizes FV-MCTS-SPIBB (see blue lines in the algorithms) is *i)* the action selection strategy (lines 6, 11), which implements the multi-agent SPI approach, *ii)* the state transition (line 12), which utilizes the MLE transition model $T^{\mathcal{D}}$. Simulations are executed recursively by the SIMULATE function. The ROLLOUT (not shown for space limits) is performed in function SIMULATE and it chooses actions according to Algorithm 2. Node counts and Q-value statistics are updated in the function UPDATE-STATS as in (Choudhury et al., 2021) (see Section E of the supplementary material).

The new **action selection strategy (Algorithm 2)**, which represents the main methodological contributions of this work, is efficiently performed using π_0 in bootstrapped state-action pairs and Constrained Max-Plus or Constrained Var-El in non-bootstrapped pairs. Instead of always using coordination through message passing (as in FV-MCTS-Max-Plus, Section 3.2), the proposed strategy first selects a joint action $\bar{a}$ for the current joint state $\bar{s}$ using the behavior policy π_0 (line 2). Then, if $(\bar{s}, \bar{a})$ is bootstrapped, the joint action is returned (lines 3 and 4). Otherwise, i.e., when the pair is non-bootstrapped (lines 7 and 9), a round of *constrained* message passing (see Algorithms 3 for Max-Plus and Algorithm 8 for Var-El) is carried out by the agents to pick the best joint action $a^\star \notin \mathcal{B}_m$. This ensures that, as the number of simulations grows, our method picks actions as the Π_0-optimal policy, as discussed in Section 3.3. Figure 1 provides a graphical representation of the two cases. As

shown in that picture, message passing is specialized to the SPI case, namely, it is used to compute Q-values only of non-bootstrapped joint actions, and when used, it considers for each agent only the non-bootstrapped local actions of each agent. This mechanism is non-trivial and strategies different from the proposed one cannot guarantee the safe improvement. The proposed method deeply differs from standard MCTS-SPIBB (Castellini et al., 2023), which in turn relies on a standard computation of the value function, inapplicable to multi-agent scenarios with large joint action spaces, since it requires evaluating every single joint action. Furthermore, our algorithm never computes the complete set of non-bootstrapped joint actions (the computation would be intractable in large environments). In line 3 of Algorithm 2, the algorithm checks if a joint state-action pair is bootstrapped by evaluating if in the joint action, there exists a local bootstrapped action. If such a local bootstrapped action does not exist, then the joint action is considered non-bootstrapped. In the rest of the algorithm, we refer only to the set of non-bootstrapped local actions.

The **constrained Max-Plus strategy (Algorithm 3)** is a constrained version of Max-Plus with UCB-inspired exploration (Choudhury et al., 2021). *This method is specifically designed to address the SPI problem, hence it is completely original.* First, for each agent $i \in \alpha$, it defines the set of non-bootstrapped actions $\bar{\mathcal{B}}_m^{A_i}(\bar{s})$ for the current joint state $\bar{s}$ (line 2). This set contains all those local actions a for agent i such that all joint actions $\bar{a}$, whose i-th action A_i equal a (which is a set denoted as $A[A_i = a] = \{\bar{a} \in A \mid \bar{a}[A_i] = a\}$), satisfy the inequality $n_{\mathcal{D}}(D_k(\bar{s}, \bar{a})) \geq m_k$. Hence, if each agent picks actions from this set the resulting joint action will be non-bootstrapped. The algorithm, then, initializes messages μ_{ij} only for local actions a_i and a_j belonging to $\bar{\mathcal{B}}_m^{A_i}(\bar{s})$ (line 3). It performs message-passing until convergence (lines 4-11) by maximizing only on local non-bootstrapped actions (line 7). After normalizing the messages for each receiving agent $j \in \alpha$ (line 8), the Constrained Max-Plus computes the local Q-values $q_i(a_i)$ for each agent $i \in \alpha$, taking into account contributions from other agents (line 13). Then, it incorporates a UCB-inspired exploration term (line 16), selects the best local non-bootstrapped action a_i for each agent, and forms the joint action $\bar{a}$ (line 17), which is then returned. In Section E of the supplementary material, we also present a Constrained version of Var-El.

5. Theoretical Analysis

This analysis shows that FV-MCTS-SPIBB converges to SPIBB (Laroche et al., 2019) under the following assumptions: 1) the number of simulations performed by MCTS tends to infinity; 2) convergence of Max-Plus (or Var-El)[3]; 3) UCB can be factorized according to Amato & Oliehoek (2015); Choudhury et al. (2021). Under these assumptions, FV-MCTS-SPIBB asymptotically converges to a policy that is Π_0-optimal in the MLE MDP $M^{\mathcal{D}}$ and provides a policy which is a safe improvement of the behavior policy, Theorem 1 and Theorem 2 of (Laroche et al., 2019). The convergence builds upon the following results: *i)* convergence of MCTS-SPIBB (Castellini et al., 2023) to SPIBB, as the number of simulations grows; *ii)* convergence of Max-Plus to optimal action selection for acyclic CG (Wainwright et al., 2004) or convergence of Var-El to optimal action selection with any CG (Guestrin et al., 2003); *iii)* optimality of FV-MCTS using Max-Plus (Choudhury et al., 2021) or Var-El (Amato & Oliehoek, 2015) as action selection strategies. The optimality in the last point assumes that the UCB exploration strategy can be decomposed into components, which seems natural and has empirical confirmations but it is still an open problem, as explained by Amato & Oliehoek (2015); Choudhury et al. (2021)[4].

Theorem 5.1 (Safe Policy Improvement for FMMDPs)**.** *Let Π_0 be the set of policies under the constraint (2), meaning, to follow π_0 in every bootstrapped state-action pair $(s, a) \in \mathcal{B}_m$. Let UCB be component-wise in FV-MCTS-SPIBB, such that at each level of the MCTS the action selection converges to the optimum policy in Π_0. Then, under a suitable factorization of the Q-value function, the returned policy $\pi^{\odot} \in \operatorname{argmax}_{\pi \in \Pi_b} \rho(\pi, M^{\mathcal{D}})$ is at least a ζ-approximate safe policy improvement over π_0 with high probability $1 - \delta$, with $\zeta = \frac{4\epsilon V_{max}}{1-\gamma} - V(\pi^{\odot}, M^{\mathcal{D}}) + V(\pi_0, M^{\mathcal{D}})$.*

Let us consider FV-MCTS-SPIBB with Max-Plus (the analysis for the Var-El version is similar). Pearl (1988) and Wainwright et al. (2004) prove that Max-Plus converges to optimality with acyclic coordination graphs. In our case, the optimal selection of non-bootstrapped joint actions is guaranteed by two elements: first, we constrain each agent to send only messages related to its non-bootstrapped local actions (Algorithm 3, lines 7-9); second, the space of non-bootstrapped joint actions contains only actions $\bar{a}$ that are composed exclusively of non-bootstrapped local actions (see Figure 1 and the description of Algorithm 2 for details).

[3]Max-Plus guarantees convergence to optimality with acyclic CG. Empirically it provides approximately optimal results even on cyclic structures. Var-El ensures convergence for any type of CG, but the complexity of the algorithm is exponential in the treewidth, a parameter related to the non-acyclicity of the graph. Finding the treewidth of a graph is a difficult problem (NP), and it can be easily estimated with Depth-First Search (DFS). Therefore, it is possible to decide whether to use Max-Plus or Var-El by evaluating the treewidth of the graph.

[4]Empirical support for this decomposability is shown in all our empirical tests involving FV-MCTS-SPIBB-Max-Plus/Var-El (e.g., Fig. 2, 3, and 7), in which observed performance improvement and safety require the convergence of MCTS-based Q-value estimation.

This guarantees that the joint action $\bar{a}$ selected by Constrained Max-Plus (Algorithm 3) is indeed optimal in the space of non-bootstrapped joint actions. However, Q-values computed by Max-Plus in MCTS are estimates drawn from statistics Q_i and Q_{ij} (updated at each simulation according to the procedure in Algorithm 7 of the supplementary material) and these estimates are based on UCB exploration. The convergence of these estimates to real Q-values is guaranteed only under the assumption of component-wise UCB (Choudhury et al., 2021). As explained by Choudhury et al. (2021); Amato & Oliehoek (2015) this is still an open problem but empirical evidence indicates the effectiveness of this strategy. A similar analysis can be performed for FV-MCTS-SPIBB with Var-El, referring to the convergence results in Guestrin et al. (2003) (see Section E in the supplementary material). In this case, no restriction on acyclic graphs is imposed. Given the convergence of the action selection strategy, the possibility of introducing MCTS in SPIBB is theoretically proved in Castellini et al. (2023). The effectiveness of Max-Plus and Var-El in improving MCTS scalability within multi-agent contexts is described by Choudhury et al. (2021) and Amato & Oliehoek (2015), respectively.

Efficient local state-action counting. The proposed efficient counting method reduces the number of samples necessary to improve the behavior policy in multi-agent factored domains compared to the original SPIBB algorithm. The theoretical analysis from Simão & Spaan (2019) for Factored SPIBB (F-SPIBB) on the single agent can be specialized to multi-agent settings, where actions belong to the space $A = \times_i A_i$. Such specialization, referred to as MF-SPIBB hereafter, remains dependent on policy iteration. Considering the set of policies Π_0 as defined in our Equation (2) (which, in turn, refers to our definition of bootstrapped *joint* state-action pairs), the improved policy returned by MF-SPIBB is guaranteed to be at least a ζ-approximate safe policy improvement over the behavior policy π_0 with a high probability of $1 - \delta$ (Theorem 1) (Simão & Spaan, 2019).

However, scaling to large domains remains challenging due to the inherent algorithmic complexity of policy iteration. To solve these problems, FV-MCTS-SPIBB combines the efficient counting of MF-SPIBB, with the scalable multi-agent SPI which uses constrained Max-Plus (or Var-El). This integration improves sample efficiency in data utilization and scalability compared to policy iteration-based algorithms. Extending the Equation (8) of Proposition 1 of (Simão & Spaan, 2019) we show that the number of samples necessary to improve the behavior policy in multi-agent factored settings is reduced compared to SPIBB algorithm (Laroche et al., 2019). See SectionS E and I in the supplementary material for a detailed analysis of the number of samples and sample efficiency. Section G provides details about the relationship between the quality of the MDP and safety/convergence.

6. Experimental evaluation

We present the domains and the experimental setting for evaluating the scalability and safety of FV-MCTS-SPIBB.

6.1. Domains

Multi-agent SysAdmin is a standard MMDP benchmark (Guestrin et al., 2003). In a network of interconnected machines, each agent controls a single machine described by two variables. Initially, all machines are turned on. Agents can, at each time step, activate their machines, or take no action. The actions may result in status changes, and agents are rewarded for machines reaching the *(good, success)* state. A ring and a star network are considered (see Figure 5.a, b in Section F of the supplementary material).

Multi-UAV Delivery was proposed in (Choudhury et al., 2021). Drones start from random cells and aim to reach target regions. Each drone can take 10 actions, including movement, staying in place, and *boarding* in its assigned target region. Successful *boarding* rewards drones with $+1000$, and they receive intermediate rewards based on proximity. Drones incur penalties for movement away from targets, and penalties for collisions with other drones or simultaneous *boarding* in the same region.

We note that the environments are much more complex than those used in the SPI literature. For instance, (Scholl et al., 2022) and (Laroche et al., 2019) use MDPs with 25 states and fewer than 10 actions. In contrast, our settings span a wide range, reaching approximately 10^{30} to 10^{41} states, and 10^9 to 10^{16} actions at the higher end of the spectrum. Refer to Section F for a detailed description of the domains.

6.2. Algorithms

We compare our method with all the main state-of-the-art SPI algorithms, namely, **SPIBB** and **Lower-SPIBB** (Laroche et al., 2019); **DUIPI** (Schneegass et al., 2008); **Adv-Approx-Soft-SPIBB** and **Lower-Approx-Soft-SPIBB** (Scholl et al., 2022); **Approx-Soft-SPIBB** (Nadjahi et al., 2019) (all based on Policy Iteration); and **MCTS-SPIBB** (Castellini et al., 2023). We also use the **FV-MCTS-Max-Plus** (Choudhury et al., 2021) and **FV-MCTS-Var-El** (Amato & Oliehoek, 2015) algorithms with MLE transition model as unsafe baselines (i.e., the policy they produce can deteriorate if the MLE model is based on small datasets). We highlight in advance that no other SPI algorithm in the literature, besides MCTS-SPIBB, can scale to more than 3 agents in the two domains considered in this analysis: therefore they do not show in the figures. For more details see Section J in the supplementary material.[5]

[5]Code available at https://github.com/Isla-lab/fv-mcts-spibb

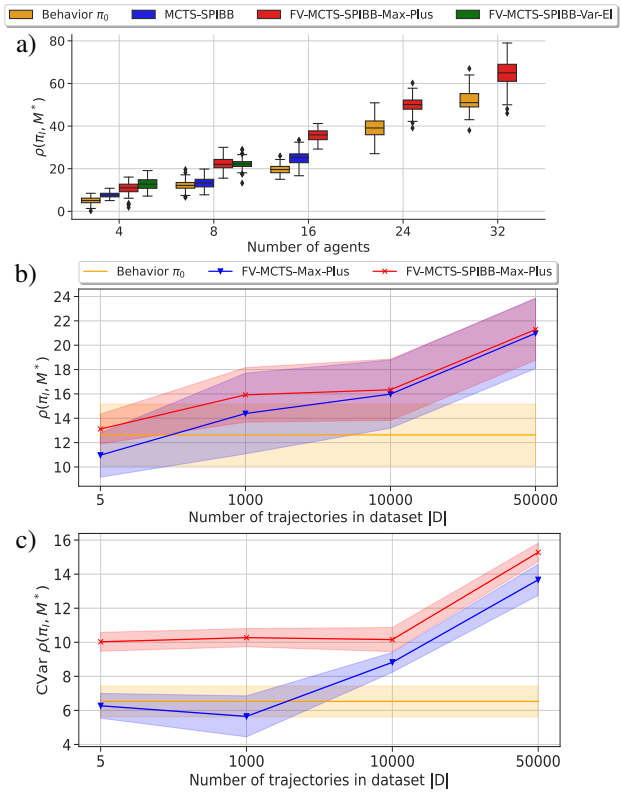

Figure 2. Multi-agent SysAdmin with ring topology: performance. a) Scalability of FV-MCTS-SPIBB (Max-Plus and Var-El): performance as the number of agents increases. b, c) Safety of FV-MCTS-SPIBB-Max-Plus ($\bar{\rho}(\pi, M^*)$ and CVaR) as the number of trajectories increases. Results obtained using Var-El instead of Max-Plus in the Supplementary Material.

6.3. Results

The scalability and safety results of FV-MCTS-SPIBB are shown in the following, considering the standard measure average return $\bar{\rho}(\pi, M^*)$. In SysAdmin it is computed averaging the returns of 100 runs of 20 steps each, while in Multi-UAV Delivery it is computed averaging the returns of 50 runs of 50 steps each. A Hyperparameter tuning on the sensitivity of MCTS parameters and additional results are available in Sections H and L of the supplementary material.

6.3.1. Multi-agent SysAdmin with ring topology

Scalability. Among the SPI methods tested on this domain, only MCTS-SPIBB and our approach worked (i.e., SPIBB and other SPI methods cannot scale to such a large action space). Figure 2.a shows box-plots of the average return $\bar{\rho}(\pi, M^*)$ (y-axis) as the number of agents varies from 4 to 32. For FV-MCTS-SPIBB-Max-Plus and FV-MCTS-SPIBB-Var-El, we use the following parameters: 100 simulations, an exploration constant empirically found to be best at $c = n$. (with n number of agents), MCTS tree depth of 20-steps, $\gamma = 0.9$, and 8 iterations of message passing in Max-Plus. With MCTS-SPIBB we used similar parameters but 1000 simulations since it does not exploit the model factorization and it requires more simulations.

With 4 agents all methods improved upon the behavior policy π_0 (orange box). FV-MCTS-SPIBB-Var-El (green box) slightly outperforms FV-MCTS-SPIBB-Max-Plus (red box) and both approaches outperform MCTS-SPIBB (blue box). With 8 agents FV-MCTS-SPIBB-Max-Plus and FV-MCTS-SPIBB-Var-El provide a (similar) significant improvement upon the behavior policy, but the gap between these methods and MCTS-SPIBB increases (i.e., FV-MCTS-SPIBB methods have performance around 22.0 and MCTS-SPIBB around 13.0, while the behavior policy reaches about 12.0). With 16 agents, FV-MCTS-SPIBB-Var-El cannot compute actions in a reasonable time, due to the exponential complexity of Var-El, namely, it is exponential on the induced width of the CG that depends on the elimination order (Dechter, 1999). MCTS-SPIBB achieved some improvements upon the baseline but performed less than FV-MCTS-SPIBB-Max-Plus. With 24 and 32 agents, only FV-MCTS-SPIBB-Max-Plus can generate a performance improvement compared to the behavior policy since MCTS-SPIBB breaks because of the too large number of actions available.

Safety. To empirically test whether the improvement generated by FV-MCTS-SPIBB-Max-Plus and FV-MCTS-SPIBB-Var-El is safe we evaluate the performance of these approaches on a domain with 8 machines (i.e., agents) using datasets of various sizes $|\mathcal{D}| = [5, 1000, 10000, 50000]$. For each dataset size, we perform 100 runs of 20 steps, then we compute the average return $\bar{\rho}(\pi, M^*)$ (where the average is performed across the 100 runs) and the 15% Conditional Value-at-risk (15%-CVaR), which is the mean performance over the 15% worst runs. Parameters are the same used in scalability tests. We explore the MLE transition model impact without safety considerations, examining FV-MCTS-Max-Plus and FV-MCTS-Var-El.

In Figures 2.b and 2.c, we observe that FV-MCTS-Max-Plus with the MLE model is not safe, as expected. When the number of trajectories in $\mathcal{D}$ is reduced to 5, the performance (i.e., both $\bar{\rho}(\pi, M^*)$ and 15%-CVaR, see the blue lines) becomes lower than that of the behavior policy (yellow lines). In comparison, the performance of FV-MCTS-SPIBB-Max-Plus (red lines) is always higher than the π_0. In particular, FV-MCTS-SPIBB-Max-Plus has $\bar{\rho}(\pi, M^*) = 13.1$ while FV-MCTS-Max-Plus $\bar{\rho}(\pi, M^*) = 10.9$, namely, $\approx 16\%$ less because the accuracy of the transition model $T^{\mathcal{D}}$ with $|\mathcal{D}|$ is very small. With $|\mathcal{D}| = 50000$, both methods improve their average performance to around 22.0. Similar results are observed also using Var-El for action selection instead of Max-plus (see Figures 9.a,b in the Supplementary Material).

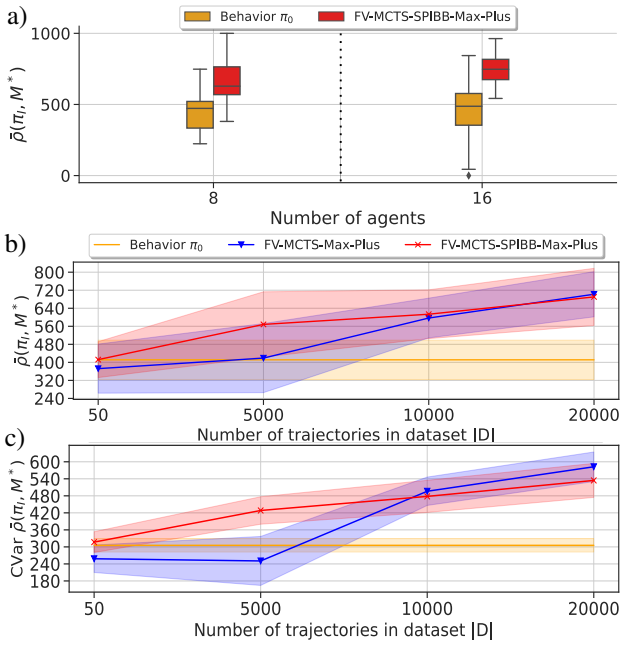

Figure 3. Multi-UAV Delivery: performance. a) Scalability of FV-MCTS-SPIBB-Max-Plus: performance as the number of agents increases. b, c) Safety of FV-MCTS-SPIBB-Max-Plus as the number of trajectories increases.

6.3.2. Multi-agent SysAdmin with star topology

Similar tests performed on multi-agent SysAdmin with star topology (see Figure 5.b) confirmed the results obtained with the ring topology. In Figure 10 of the supplementary material we show, for instance, the charts about the safety of FV-MCTS-SPIBB-Max-Plus and FV-MCTS-Max-Plus with MLE transition model on a domain with 16 machines.

6.3.3. Multi-UAV Delivery

This domain is particularly challenging because it has a dynamic coordination graph that can generate multiple cycles as agents move (Figure 5.c).

Scalability. In this domain, we can only assess the scalability of FV-MCTS-SPIBB-Max-Plus, since it is the only SPI method capable of handling such large domain sizes. We evaluate it with 8 and 16 agents. For FV-MCTS-SPIBB-Max-Plus we use 1000 simulations, exploration constant $c = n$ (i.e., number of agents), tree depth of 20 steps, $\gamma = 1.0$, and 8 iterations of message passing in Max-Plus. Figure 3.a shows that with 8 agents FV-MCTS-SPIBB-Max-Plus achieves a significant average improvement from 457.7 to 655.6. This performance advantage persists as the domain size increases to 16 agents, with an improvement from 454.2 to 748.1.

Safety. The safety of the improvement generated by FV-MCTS-SPIBB-Max-Plus is shown in a domain with 8 agents (Figures 3.b and 3.c). Again, FV-MCTS-Max-Plus with the MLE transition model (blue line), is used to show that with small datasets the inaccuracies of the MLE model lead to unsafe policies if SPI strategies are not used. We use dataset sizes $\mathcal{D} = [50, 5000, 10000, 20000]$, 100 simulations, an exploration constant $c = n + 2$, a tree depth of 10 steps, $\gamma = 1.0$, and 8 iterations of message passing. Figures 3.b show that with a small number of trajectories ($d = 50$), FV-MCTS-Max-Plus performs lower than π_0 (i.e., 372.1) while FV-MCTS-SPIBB-Max-Plus exhibits performance similar to π_0 (i.e., 412.1). With $d = 5000$, the average performance and 15%-CVaR of FV-MCTS-Max-Plus remains below π_0 (see Figure 3.c for the 15%-CVaR), whereas FV-MCTS-SPIBB-Max-Plus performs better than π_0. As the number of trajectories increases, the average performance of both methods tends to be the same value.

7. Conclusion

We introduced a first multi-agent SPI algorithm based on two main methodological contributions, an efficient method (based on transition model factorization) for counting state-action pairs in the available dataset, and an efficient online strategy (based on value function factorization and MCTS) for safely computing the improved policy. After formally defining the algorithm, we provide a theoretical analysis that identifies the conditions for which convergence and safety can be guaranteed. Moreover, we provide an empirical evaluation considering two large-scale multi-agent benchmark domains for which no state-of-the-art SPI algorithm can work. Future work could extend the proposed methodology to a competitive or mixed scenario and on its application in settings where data are collected sequentially.

Acknowledgements

This paper has been prepared as a part of a collaboration between the University of Verona and Leonardo Labs belonging to Leonardo SpA.

Impact Statement

We present a safe approach for scalable and reliable RL. Safety is an increasingly important topic in many application areas of modern machine learning, where algorithms are applied to real-world problems, and decision-makers need guarantees. These problems are usually computationally intensive, so many safe RL algorithms cannot be effectively applied. The net impact of our approach is to provide a safe scalable algorithm with guarantees that can be applied to realistic settings. The publication of our work can help bring attention to scalable safety approaches. We do not expect any negative influence on the field and society.

References

Albrecht, S. V., Christianos, F., and Schäfer, L. *Multi-Agent Reinforcement Learning: Foundations and Modern Approaches*. MIT Press, Cambridge, MA, USA, 2023.

Amato, C. and Oliehoek, F. A. Scalable planning and learning for multi-agent POMDPs. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI)*, pp. 1995–2002. AAAI Press, 2015.

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

Boutilier, C. Planning, learning and coordination in multi-agent decision processes. In *Proceedings of the 6th Conference on Theoretical Aspects of Rationality and Knowledge (TARK)*, pp. 195–210. Morgan Kaufmann Publishers Inc., 1996.

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., Bianchi, F., Zorzi, E., Simão, T. D., Farinelli, A., and Spaan, M. T. J. Scalable safe policy improvement via Monte Carlo Tree Search. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, pp. 3732–3756. PMLR, 2023.

Chalkiadakis, G. and Boutilier, C. Coordination in multi-agent reinforcement learning: a Bayesian approach. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, AAMAS 2003, pp. 709–716, New York, NY, USA, 2003. Association for Computing Machinery.

Choudhury, S., Gupta, J. K., Morales, P., and Kochenderfer, M. J. Scalable anytime planning for multi-agent MDPs. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, AAMAS 2021, pp. 341–349. IFAAMAS, 2021.

Dechter, R. Bucket elimination: A unifying framework for reasoning. *Artificial Intellingence*, 113(1–2):41–85, 1999.

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

Dulac-Arnold, G., Levine, N., Mankowitz, D. J., Li, J., Paduraru, C., Gowal, S., and Hester, T. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning*, 110(9):2419–2468, 2021.

Guestrin, C., Koller, D., and Parr, R. Multi-agent planning with factored MDPs. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 1523–1530. MIT Press, 2001.

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

Kochenderfer, M. J., Wheeler, T. A., and Wray, K. H. *Algorithms for Decision Making*. MIT press, Cambridge, MA, USA, 2022.

Kocsis, L. and Szepesvári, C. Bandit based Monte-Carlo planning. In *Proceedings of the 17th European Conference on Machine Learning (ECML)*, pp. 282–293. Springer-Verlag, 2006.

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. arXiv preprint arXiv:2005.01643, 2020.

Ma, J. and Wu, F. Learning to coordinate from offline datasets with uncoordinated behavior policies. In *Proceedings of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 1258–1266. ACM, 2023.

Mazzi, G., Castellini, A., and Farinelli, A. Risk-aware shielding of Partially Observable Monte Carlo Planning policies. *Artificial Intelligence*, 324:103987, 2023.

Nadjahi, K., Laroche, R., and Combes, R. T. d. Safe policy improvement with soft baseline bootstrapping, July 2019. arXiv:1907.05079 [cs, stat].

Oliehoek, F. A. and Amato, C. *A Concise Introduction to Decentralized POMDPs*. Springer Briefs in Intelligent Systems. Springer, Heidelberg, Germany, 2016.

Pearl, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, Calif., 1988.

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, USA, 2014.

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.

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

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

Strehl, A. L., Diuk, C., and Littman, M. L. Efficient structure learning in factored-state MDPs. In *Proceedings of the 22nd National Conference on Artificial Intelligence*, pp. 645–650. AAAI Press, 2007.

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

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., Oh, J., Horgan, D., Kroiss, M., Danihelka, I., Huang, A., Sifre, L., Cai, T., Agapiou, J. P., Jaderberg, M., Vezhnevets, A. S., Leblond, R., Pohlen, T., Dalibard, V., Budden, D., Sulsky, Y., Molloy, J., Paine, T. L., Gülçehre, Ç., Wang, Z., Pfaff, T., Wu, Y., Ring, R., Yogatama, D., Wünsch, D., McKinney, K., Smith, O., Schaul, T., Lillicrap, T. P., Kavukcuoglu, K., Hassabis, D., Apps, C., and Silver, D. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782): 350–354, 2019.

Wainwright, M., Jaakkola, T., and Willsky, A. Tree consistency and bounds on the performance of the Max-Product algorithm and its generalizations. *Statistics and Computing*, 14(2):143–166, 2004.

Zerbel, N. and Yliniemi, L. Multiagent Monte Carlo Tree Search. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pp. 2309–2311. IFAAMAS, 2019.

Zhang, K., Yang, Z., and Başar, T. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of reinforcement learning and control*, pp. 321–384, 2021.

Zuccotto, M., Castellini, A., Torre, D. L., Mola, L., and Farinelli, A. Reinforcement learning applications in environmental sustainability: a review. *Artificial Intelligence Review*, 57(88):1–68, 2024.

Supplementary Material

A. Additional Background

MDP. A *Markov Decision Process* (MDP) (Puterman, 2014) is formally represented as a tuple $M = \langle S, A, T, R, \gamma \rangle$. In this representation, S is a finite set of states; A is a finite set of actions; $T : S \times A \to \mathcal{P}(S)$ is a stochastic transition function, where $\mathcal{P}(X)$ represents the set of probability distributions over a finite set X; $R : S \times A \to [-R_{max}, R_{max}]$ is a bounded stochastic reward function and $\gamma \in [0, 1]$ is the discount factor. To evaluate the overall performance of a policy π within the MDP M, we compute its expected return in the initial state s_0 as $\rho(\pi, M) = V_M^\pi(s_0)$, where $V_M^\pi(s)$ represents the state value of s and $Q_M^\pi(s, a)$ the action value of (s, a). Additionally, V_{max} represents the known upper bound of the absolute value of the return, with the inequality $V_{max} \leq {}^{R_{max}}/_{1-\gamma}$ providing a specific constraint on this upper bound.

MMDP. In *Multi-agent* MDPs (MMDPs) (Boutilier, 1996) multiple agents cooperate to achieve a common goal. Formally, an MMDP is represented as a tuple $M = \langle S, \alpha, \{A_i\}_{i \in \alpha}, T, R, \gamma \rangle$, where S represents the set of all states; α denotes a finite set of n agents; $A = \times_i A_i$ represents a finite set of joint actions, with each A_i indicating the set of actions available to agent i (this set grows exponentially with the number of agents); $T : S \times A_1 \times \cdots \times A_n \to \mathcal{P}(S)$ is a stochastic transition function that takes a state $s \in S$ and a joint action $\bar{a} \in A$ as input and returns a probability distribution over the next states; R and γ have the same meaning they have in MDPs.

FMMDP. In *Factored* MMDP (FMMDP) (Guestrin et al., 2003; Strehl et al., 2007) the state space is factored as $S = S_1 \times \cdots \times S_K$, such that $\bar{s}[S_k] \in Dom(S_k)$ indicates the value assigned to factor S_k in the joint state $\bar{s}$. The outcome of each factor S_k is independent of the remaining, so the probabilistic transition function can be written as $T(\bar{s}' \mid \bar{s}, \bar{a}) = \prod_{k=1}^{K} P_k(\bar{s}'[S_k] \mid \bar{s}, \bar{a}))$, where P_k is the conditional probability distribution of the factor S_k. Furthermore, given a set of dependency identifiers $\mathcal{J}$ to indicate common dependencies between different state-action pairs, we can define the dependency functions $D_k : S \times A \to \mathcal{J}$. Using this function, the probabilistic transition function can be compactly represented by $T(\bar{s}' \mid \bar{s}, \bar{a}) = \prod_{k=1}^{K} P_k(\bar{s}'[S_k] \mid D_k(\bar{s}, \bar{a}))$. Intuitively, given a state-variable S_k and two state-action pairs $\bar{s}, \bar{a}$ and $\bar{s}', \bar{a}'$, if $D_k(\bar{s}, \bar{a}) = D_k(\bar{s}', \bar{a}')$ then $P_k(\cdot \mid \bar{s}, \bar{a}) = P_k(\cdot \mid \bar{s}', \bar{a}')$.

Monte Carlo Tree Search. Monte Carlo Tree Search (MCTS) is a technique used by agents to make decisions based on their current state (Browne et al., 2012). It creates a tree structure rooted in the current state to estimate the Q-values, which help the agent choose the best action after running a certain number of simulations. During each simulation, MCTS uses a strategy called Upper Confidence Bound applied to Trees (UCT) (Kocsis & Szepesvári, 2006) to decide on actions. UCT balances exploration and exploitation within the tree by updating two statistics for each node (state): the average discounted return achieved by selecting a specific action and the number of times that action has been chosen from that node. UCT extends the UCB1 strategy (Auer et al., 2002) to sequential decision-making, ensuring a trade-off between exploring new actions and exploiting known ones. It selects the action with the best upper confidence bound, which is calculated based on the average return and a constant parameter. After completing all the simulations, the action with the highest average return at the root node is executed in the real environment.

B. Details about coordination graphs

A CG is a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ that has a node $i \in \mathcal{V}$ for each agent and an edge $(i, j) \in \mathcal{E}$ for each pair of agents (i.e., nodes) i and j with interdependent payoffs (Guestrin et al., 2001; Choudhury et al., 2021). The coordination graph induces a factorization of the global payoff function

$$Q(\bar{a}) = \sum_{i \in \mathcal{V}} Q_i(a_i) + \sum_{(i,j) \in \mathcal{E}} Q_{ij}(a_i, a_j). \tag{3}$$

Here, each agent has an individual payoff function Q_i (highlighted in blue in Figure 4) and a local payoff function Q_{ij} (highlighted in red) that represents the interaction between agents i and j.

In this work, we present two domains with different types of CGs, including cyclic, acyclic, static, and dynamic graphs. Specifically, we evaluate the methods on a multi-agent SysAdmin with a ring network, as shown in Figure 1.a (in the paper). In this case, the CG is static because it does not change when agents transition states, and it is also cyclic. Additionally, we evaluate the methods in a multi-agent SysAdmin scenario with a star network, where the CG is always static but acyclic. The second domain is multi-UAV Delivery, as depicted in Figure 1.c (in the paper). In this case, it can be both acyclic and cyclic, which means that the graph can change with every state transition. In the multi-UAV Delivery domain, agents start

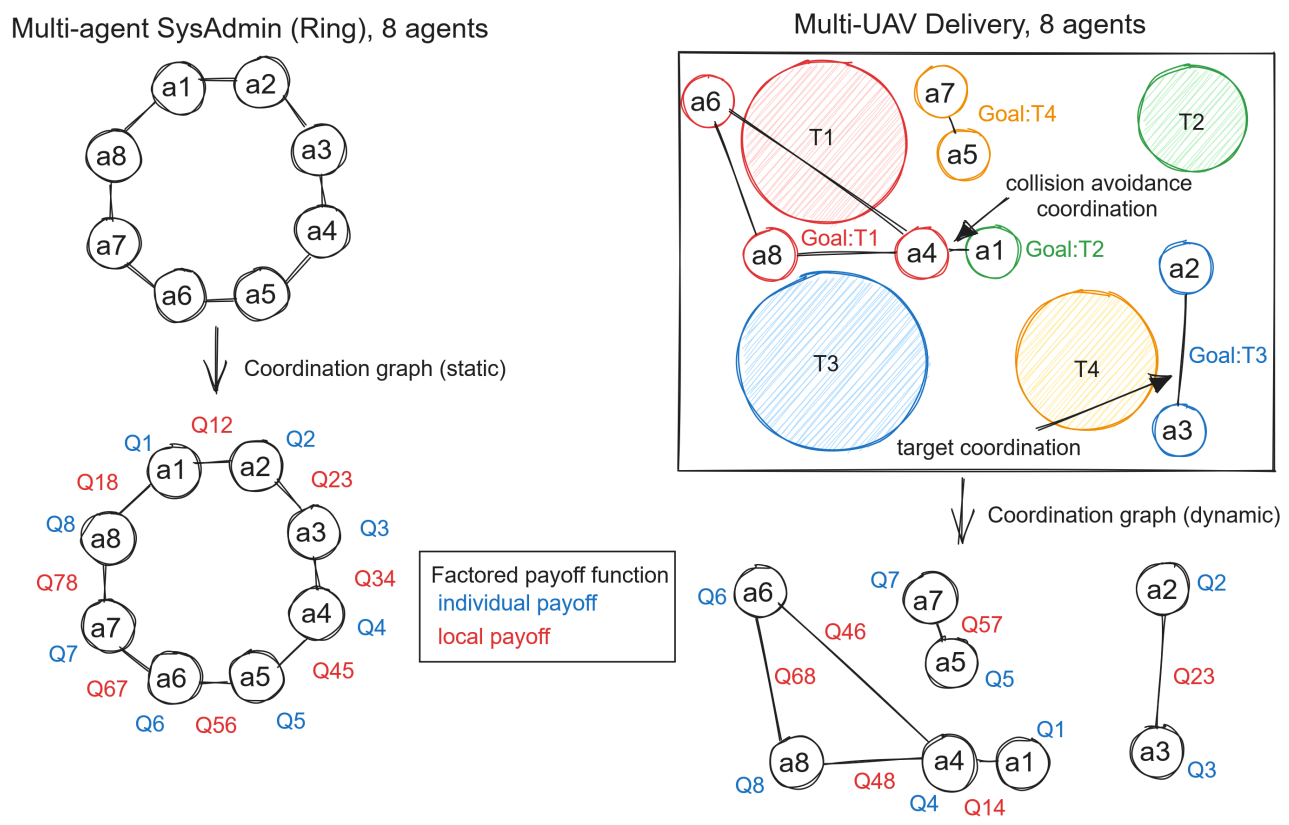

Figure 4. Example of coordination graph and factored payoff function on multi-agent SysAdmin with ring topology (left) and multi-UAV Delivery (right).

from random positions and are randomly assigned to target regions. To coordinate agents assigned to the same target (as shown in Figure 4, where nodes representing agents have the same target color), a CG is used to connect these agents with an edge. At each step, agents move and change their state, dynamically altering the CG by creating new connections. As seen in Figure 4, agents a_1 and a_4, assigned to different targets, are connected in the same CG because one of the objectives is to avoid agent collisions, requiring them to coordinate.

C. Details about Factored-Value MCTS with Max-Plus and Variable Elimination

Algorithm 4 represents both FV-MCTS-Max-Plus and FV-MCTS-Var-El. The first algorithm, proposed in (Choudhury et al., 2021), uses Max-plus with UCB-inspired exploration (see Algorithm 5) to select the best joint action, while the second algorithm, inspired by (Amato & Oliehoek, 2015), uses Variable Elimination with UCB-inspired exploration (see Algorithm 6) to select the best joint action. Since the two algorithms have several common elements, they have been merged into a unique algorithm. FV-MCTS-Max-Plus requires input parameter *action-selection=Max-Plus* while FV-MCTS-Var-El requires *action-selection=Var-El*. The algorithm initializes statistics (simulation counters N and estimated Q-values Q) for all nodes and edges in the coordination graph for the given state $\bar{s}$. The main function, FV-MCTS, runs multiple simulations starting from state $\bar{s}$. After completing these simulations, it calculates and returns a joint action $\bar{a}^\star$. Simulations are executed recursively by the SIMULATE function, similar to standard MCTS. Statistics are updated in the usual way by the function UPDATE-STATS, with the addition of considering edge statistics N_{ij} and Q_{ij}, as they are necessary for computing optimal actions. The ROLLOUT (not in the pseudo-code) is performed in function SIMULATE for states out of the MC tree it chooses actions according to uniformly random distribution.

FV-MCTS with Variable Elimination (FV-MCTS-Var-El). As previously defined, the global payoff can be factorized as $Q(\bar{a}) = \sum_{i \in \mathcal{V}} Q_i(a_i) + \sum_{(i,j) \in \mathcal{E}} Q_{ij}(a_i, a_j)$ (Choudhury et al., 2021). Alternatively, it can be factorized as, $Q(\bar{a}) =$

Algorithm 4 Factored-Value MCTS with Max-Plus and Variable Elimination (Amato & Oliehoek, 2015; Choudhury et al., 2021)

Require: state $\bar{s}$; MLE transition model $T^{\mathcal{D}}$; depth; exploration constant c; action-selection

$N_i(\bar{s}, a_i) = Q_i(\bar{s}, a_i) = 0 \; \forall i \in \mathcal{V}, \forall a_i \in A_i$ ▷ Initialize node statistics
$N_{ij}(\bar{s}, a_i, a_j) = Q_{ij}(\bar{s}, a_i, a_j) = 0 \; \forall (i,j) \in \mathcal{E}, a_i \in A_i, a_j \in A_j$ ▷ Initialize edge statistics
function FV-MCTS($\bar{s}$, depth)
  while time limit not reached **do**
    SIMULATE($\bar{s}$, depth)
  if action-selection = Max-Plus **then**
    $\bar{a}^\star$ = MAX-PLUS(0) ▷ No exploration here
  else
    $\bar{a}^\star$ = VAR-EL(0) ▷ No exploration here
  return $\bar{a}^\star$ ▷ Best joint action

function SIMULATE($\bar{s}$, depth)
  if depth = 0 **then**
    return $\bar{0}$
  if action-selection = Max-Plus **then**
    $\bar{a}$ = MAX-PLUS(c)
  else
    $\bar{a}$ = VAR-EL(c)
  $\bar{s}', \bar{r} \sim T(\bar{s}, \bar{a})^{\mathcal{D}}$, R($\bar{r}$, $\bar{a}$)
  $\bar{r}'$ = $\bar{r}$ + $\gamma\cdot$ SIMULATE($\bar{s}'$, depth - 1)
  UPDATE-STATS($\bar{s}$, $\bar{a}$, $\bar{r}'$)

function UPDATE-STATS($\bar{s}$, $\bar{a}$, $\bar{r}$)
  for every agent i **do**
    $N_i(\bar{s}, a_i)$ += 1
    $Q_i(\bar{s}, a_i)$ += $\frac{r_i - Q_i(\bar{s}, a_i)}{N_i(\bar{s}, a_i)}$
  for every edge $(i,j) \in \mathcal{G}(\bar{s})$ **do**
    $N_{ij}(\bar{s}, a_i, a_j)$ += 1
    $r_e = r_i + r_j$
    $Q_{ij}(\bar{s}, a_i, a_j)$ += $\frac{r_e - Q_{ij}(\bar{s}, a_i, a_j)}{N_{ij}(\bar{s}, a_i, a_j)}$

$\sum_i Q_i(a_i)$ (Amato & Oliehoek, 2015). We can exactly compute the best joint action $\text{argmax}_{\bar{a}} \, Q(\bar{a})$ using Var-El (Guestrin et al., 2003). Var-El eliminates variables while maximizing over them, collecting local payoffs dependent on those variables. After eliminations, it finds the optimal joint action by propagating agent actions in reverse, maximizing conditional functions. The algorithm has exponential complexity in the induced width of the CG, which depends on the elimination order (Dechter, 1999). In the context of online planning, Var-El is integrated into MCTS (Amato & Oliehoek, 2015). It maintains local statistics for each component, such as the local mean payoff $Q_i(\bar{s}, a_i)$ and the number of times component i selects action a_i, $N(\bar{s}, a_i)$. Local mean payoffs are updated incrementally as follows, $Q_i(\bar{s}, a_i) = Q_i(\bar{s}, a_i) + {}^{(r_i - Q_i(\bar{s}, a_i))}/_{N(\bar{s}, a_i)}$, where r_i is the reward. In constructing the Monte Carlo tree, joint actions are chosen using a version of UCB, ensuring exploration.

D. Safe Policy Improvement with Baseline Bootstrapping (SPIBB)

The *Safe Policy Improvement with Baseline Bootstrapping* (Laroche et al., 2019) algorithm reformulates the percentile criterion (Delage & Mannor, 2010; Petrik et al., 2016) to make the search for an efficient and provably safe policy tractable. The SPIBB algorithm searches for an efficient and provably safe policy tractable by considering the objective

$$\max_{\pi \in \Pi} \rho(\pi, M^{\mathcal{D}}) \text{ s.t. } \rho(\pi, M) \geq \rho(\pi_0, M) - \zeta, \forall M \in \Xi, \tag{4}$$

Algorithm 5 Max-Plus with UCB-inspired exploration (Choudhury et al., 2021)

Require: Coordination Graph $\mathcal{G}(\bar{s}) = \langle \mathcal{V}, \mathcal{E} \rangle$; state node statistics $N, N_i, N_{ij}, Q_i, Q_{ij}$; max iterations M; node exploration constant c

function MAX-PLUS(c)
  $\mu_{ij}(a_j) = \mu_{ji} = 0$ for $(i,j) \in \mathcal{E}, a_i \in A_i, a_j \in A_j$
  for t = 1 to M **do**
    for every agent i **do**
      for all neighbors $j \in \Gamma(i)$ **do**
        Compute $\mu_{ij}(a_j) = \max_{a_i} \left\{ Q_i(\bar{s}, a_i) + Q_{ij}(\bar{s}, a_i, a_j) + \sum_{\ell \in \Gamma(i) \setminus \{j\}} \mu_{\ell i}(a_i) \right\}$ ▷ $\Gamma(i)$ is the set of neighbors of i
        $\mu_{ij}(a_j)$ -= $\frac{1}{|\mathcal{A}_j|} \sum_{a_j \in \mathcal{A}_j} \mu_{ij}(a_j)$ ▷ Message normalization
        Send message $\mu_{ij}(a_j)$ to agent j
        if $\mu_{ij}(a_j)$ close to previous message **then**
          break
  for every agent i **do**
    $q_i(a_i)$ = $Q_i(\bar{s}, a_i) + \sum_{j \in \Gamma(i)} \mu_{ji}(a_i)$
    if $c \neq 0$ **then**
      for all neighbors $j \in \Gamma(i)$ **do**
        $q_i(a_i)$ += $c\sqrt{\frac{\log(N+1)}{N_i(\bar{s}, a_i)}}$
    a'_i = $\text{argmax}_{a_i}\, q_i(a_i)$
  return a'

Algorithm 6 Variable Elimination with UCB-inspired exploration (Amato & Oliehoek, 2015)

Require: Coordination Graph $\mathcal{G}(\bar{s}) = \langle \mathcal{V}, \mathcal{E} \rangle$; variable order $\mathcal{O}$; state node statistics $N, N_i, N_{ij}, Q_i, Q_{ij}$; node exploration constant c

function VAR-EL(c)
  $\mathcal{F} \leftarrow \emptyset$
  for each edge $(i,j) \in \mathcal{E}$ **do**
    Add to $\mathcal{F}$ the function $Q_{ij}(\bar{s}, a_i, a_j)$
  for variable name $v \in \mathcal{O}$ **do**
    $f_i \leftarrow \emptyset$
    for function $f \in \mathcal{F}$ **do**
      if f involves variable v **then**
        Add to f_i the function f and remove it from $\mathcal{F}$
    if $c \neq 0$ **then**
      $\phi_v \leftarrow \max_{a_v} \left(\sum_{f \in f_i} f + c\sqrt{\frac{\log(N+1)}{N_i(\bar{s}, a_v)}} \right)$
    else
      $\phi_v \leftarrow \max_{a_v} \sum_{f \in f_i} f$
    Add ϕ_v to $\mathcal{F}$
  Let f_{max} be the single function remaining in $\mathcal{F}$
  Solve f_{max} and let $\bar{a} = \text{argmax}\, f_{\text{max}}$
  return a'

where Ξ is the set of admissible MDPs: $\Xi(M^{\mathcal{D}}, e) = \{M = \langle S, A, T, R, \gamma \rangle \mid \forall (s,a) \in S \times A, \|T(\cdot|s,a) - T^{\mathcal{D}}(\cdot|s,a)\|_1 \leq e(s,a)\}$, and $e : S \times A \to \mathbb{R}$ is an arbitrary function representing the uncertainty over the estimated transition model $T^{\mathcal{D}}$, defined in such a way that $\Xi(M^{\mathcal{D}}, e)$ includes the true MDP with high probability. Then, a policy is considered safe if, given a confidence level δ and an approximation parameter ζ, with a high probability of $1 - \delta$, the improved policy is ζ-approximately as good as the behavior policy π_0 for all $M \in \Xi(M^{\mathcal{D}}, e)$. SPIBB distinguishes two distinct sub-sets, the

Algorithm 7 Factored-Value MCTS-SPIBB

Require: state $\bar{s}$; behavior policy π_0; bootstrapped joint state-action set $\mathcal{B}_m$; non-bootstrapped joint state-action set $\bar{\mathcal{B}}_m$; MLE transition model $T^{\mathcal{D}}$, simulation *depth*; exploration constant c; coordination graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$; non-bootstrapped action optimization type *Opt*

1: $N_i(\bar{s}, a_i) = Q_i(\bar{s}, a_i) = 0 \; \forall i \in \mathcal{V}, \forall a_i \in A_i$
2: $N_{ij}(\bar{s}, a_i, a_j) = Q_{ij}(\bar{s}, a_i, a_j) = 0 \; \forall (i,j) \in \mathcal{E}, a_i \in A_i, a_j \in A_j$

3: **function** FV-MCTS-SPIBB($\bar{s}$, depth)
4: **while** time limit not reached **do**
5: SIMULATE($\bar{s}$, depth)
6: $\bar{a}^\star$ = ACTION-SELECTION(0) ▷ Algorithm 2
7: **return** $\bar{a}^\star$ ▷ Best joint action

8: **function** SIMULATE($\bar{s}$, depth)
9: **if** depth = 0 **then**
10: **return** $\bar{0}$
11: $\bar{a}$ = ACTION-SELECTION(c) ▷ Algorithm 2
12: $\bar{s}', \bar{r} \sim T^{\mathcal{D}}(\bar{s}, \bar{a}), R(\bar{s}, \bar{a})$
13: $\bar{r}' = \bar{r} + \gamma \cdot$ SIMULATE($\bar{s}'$, depth - 1)
14: UPDATE-STATS($\bar{s}$, $\bar{a}$, $\bar{r}'$)

15: **function** UPDATE-STATS($\bar{s}$, $\bar{a}$, $\bar{r}$)
16: **for** every agent i **do**
17: $N_i(\bar{s}, a_i)$ += 1
18: $Q_i(\bar{s}, a_i)$ += $\frac{r_i - Q_i(\bar{s}, a_i)}{N_i(\bar{s}, a_i)}$
19: **for** every edge $(i,j) \in \mathcal{G}(\bar{s})$ **do**
20: $N_{ij}(\bar{s}, a_i, a_j)$ += 1
21: $r_e = r_i + r_j$
22: $Q_{ij}(\bar{s}, a_i, a_j)$ += $\frac{r_e - Q_{ij}(\bar{s}, a_i, a_j)}{N_{ij}(\bar{s}, a_i, a_j)}$

bootstrapped subset $\mathcal{B} = \{(s,a) \in S \times A \mid n_{\mathcal{D}}(s,a) < N_\wedge\}$, containing state-action pairs observed fewer than $N_\wedge$ times in $\mathcal{D}$, and the *non-bootstrapped set*, its complement. In this context, $n_{\mathcal{D}}(s,a)$ represents the state-action counter. Policies improved using SPIBB are essentially optimal policies that satisfy the *SPIBB constraint*. Namely, they are constrained to belong to the space of policies $\Pi_0 = \{\pi \in \Pi : \pi(s,a) = \pi_0(s,a) : \forall (s,a) \in \mathcal{B}\}$. This way, policy optimization is conducted, using policy iteration, in the MLE MDP $M^{\mathcal{D}}$ over the constrained policy space Π_0: $\operatorname{argmax}_{\pi \in \Pi_0} \rho(\pi, M^{\mathcal{D}})$.

E. FV-MCTS-SPIBB

Algorithm 7 provides a detailed description of the proposed FV-MCTS-SPIBB algorithm. We mark in blue the lines that differ from the standard algorithms. The algorithm initializes statistics (simulation counters N and estimated Q-values Q) for all nodes and edges in the coordination graph for the given state $\bar{s}$. The main function, FV-MCTS-SPIBB, runs multiple simulations starting from state $\bar{s}$ (line 5). After completing these simulations, it calculates and returns a joint action $\bar{a}^\star$ (lines 6 and 7). Simulations are executed recursively by the SIMULATE function (line 8-14), similar to standard MCTS. The structure of the algorithm is similar to that of (Choudhury et al., 2021). The main differences, highlighted in blue, are the action selection strategy (line 11), which implements the SPI approach, and the state transition (line 12), which utilizes the MLE transition model $T^{\mathcal{D}}$ rather than the true model. Statistics are updated in the usual way (line 14) by the function UPDATE-STATS (lines 15-22), with the addition of considering edge statistics N_{ij} and Q_{ij}, as they are necessary for computing optimal non-bootstrapped actions. The ROLLOUT (not in the pseudo-code for the space limits) is performed in function SIMULATE for states out of the MC tree: it chooses actions according to Algorithm 2 but using a uniformly random distribution for non-bootstrapped actions.

FV-MCTS-SPIBB with Constrained Variable Elimination. Algorithm 8 is a constrained version of Variable Elimination

Algorithm 8 Constrained Variable Elimination

Require: Coordination Graph $\mathcal{G}=\langle\mathcal{V},\mathcal{E}\rangle$; variable order $\mathcal{O}$; state node statistics N, N_i, N_{ij}, Q_i, Q_{ij}; node exploration constant c

1: **function** VAR-EL(c)
2: $\quad\bar{\mathcal{B}}_m^{A_i}(\bar{s}) = \{a \in A_i \mid n_{\mathcal{D}}(D_k(\bar{s},\bar{a})) \geq m_k, \forall k, \forall \bar{a} \in A[A_i = a]\}, \forall i$
3: $\quad\mathcal{F} \leftarrow \emptyset$
4: $\quad$**for** each edge $(i,j) \in \mathcal{E}$ **do**
5: $\quad\quad$Add to $\mathcal{F}$ the function $Q_{ij}(\bar{s}, a_i, a_j)$
6: $\quad$**for** variable name $v \in \mathcal{O}$ **do**
7: $\quad\quad f_i \leftarrow \emptyset$
8: $\quad\quad$**for** function $f \in \mathcal{F}$ **do**
9: $\quad\quad\quad$**if** f involves variable v **then**
10: $\quad\quad\quad\quad$Add to f_i the function f and remove it from $\mathcal{F}$
11: $\quad\quad$**if** $c \neq 0$ **then**
12: $\quad\quad\quad\phi_v \leftarrow \max_{a_v \in \bar{\mathcal{B}}_m^{A_v}} \left(\sum_{f \in f_i} f + c\sqrt{\frac{\log(N+1)}{N_i(\bar{s},a_v)}}\right)$
13: $\quad\quad$**else**
14: $\quad\quad\quad\phi_v \leftarrow \max_{a_v \in \bar{\mathcal{B}}_m^{A_v}} \sum_{f \in f_i} f$
15: $\quad\quad$Add ϕ_v to $\mathcal{F}$
16: $\quad$Let f_{max} be the single function remaining in $\mathcal{F}$
17: $\quad$Solve f_{max} and let $\bar{a} = \text{argmax}\, f_{\text{max}}$
18: $\quad$**return** $\bar{a}$ $\quad\triangleright$ Constrained selection $\bar{a} \in \bar{\mathcal{B}}_m$

with UCB-inspired exploration (see Algorithm 6). The algorithm starts by computing the set of non-bootstrapped actions $\bar{\mathcal{B}}_m^{A_i}(\bar{s})$ for each agent $i \in \alpha$ (line 2). Next, it generates a set $\mathcal{F}$ that contains Q-functions $Q_{ij}(\bar{s}, a_i, a_j)$ for each edge (i,j) in the coordination graph $\mathcal{G}$ (lines 4-5). The algorithm then proceeds in a given order, denoted by $\mathcal{O}$. It removes functions from $\mathcal{F}$ that involve a particular agent v and replaces them with a factor ϕ_v that considers the Q-functions of all edges connected to agent v (lines 6-15). In this process, when calculating the new agent Q-values ϕ_v, it maximizes over local non-bootstrapped actions in $\bar{\mathcal{B}}_m^{A_v}(\bar{s})$ instead of considering all possible actions, as is done in standard Variable Elimination. Additionally, the algorithm introduces a UCB-inspired exploration term to ϕ_v when the exploration constant c is non-null when the algorithm is used in MCTS simulations (line 12).

Additional theoretical analysis on Var-El. In (Guestrin et al., 2003) it is proven that Variable Elimination converges to optimality when applied to coordination graphs of any type (although it has higher complexity than Max-Plus). In our case, the constraints we introduced to consider only non-bootstrapped local actions, and the way we define the set of non-bootstrapped joint actions, guarantee that the joint action selected through Variable Elimination is indeed optimal in the space of non-bootstrapped joint actions. However, it is important to recognize that in this case, the algorithm relies on Q-value statistics, which needs an exploration mechanism. The UCB-inspired exploration we introduced in Algorithm 8 is, in turn, inspired by that used in (Amato & Oliehoek, 2015), and, although its validity has been show empirically, no theoretical proof is available yet, similarly as for Max-Plus with UCB-inspired exploration.

Additional theoretical analysis on efficient local state-action counting. The minimum number of samples needed for each component (see Equation (1)), to identify non-bootstrapped state-action pairs that lead to safety improvement, is $m_k = {}^{2K^2}/_{\epsilon^2} \cdot \log\left({}^{|\mathcal{Q}|2^{|\text{Dom}(S_k)|}}/_{\delta}\right)$. Extending the Equation (8) of Proposition 1 of (Simão & Spaan, 2019), we get that K is the number of factors, $\mathcal{Q}$ is the set of all transition components, and ϵ is an error term used to define the distance between the true and the MLE transition model such that $\mathbb{P}(\|T^*(\cdot|\bar{s},\bar{a}) - T^{\mathcal{D}}(\cdot|\bar{s},\bar{a})\|_1 \geq \epsilon) \leq \delta \ \forall(\bar{s},\bar{a}) \notin \bar{\mathcal{B}}_m$. This equation shows that given a specific error ϵ, the number of samples necessary to improve the behavior policy in multi-agent factored domains is reduced compared to SPIBB algorithm (Laroche et al., 2019), due to the term $|A||S|$ being replaced by $|\mathcal{Q}|$, and the term $|S|$ being replaced by $Dom(S_k)$. We notice that, even with the improved sample efficiency, MF-SPIBB still cannot scale to large domains due to the intrinsic algorithm complexity of policy iteration. The proposed FV-MCTS-SPIBB algorithm solves this problem by integrating the efficient counting of MF-SPIBB (which improves sample-efficiency in the

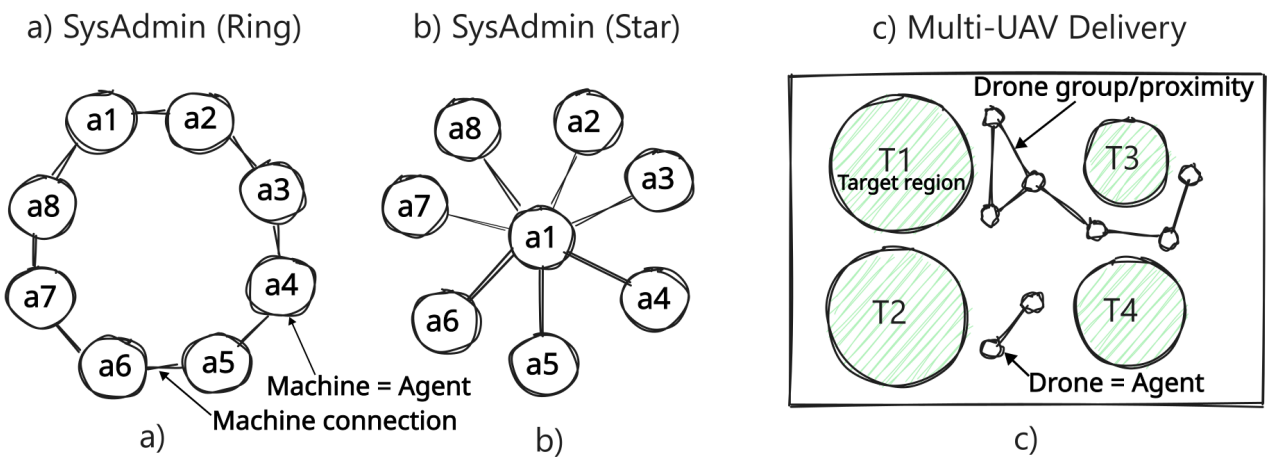

Figure 5. Domains. a,b) Multi-agent SysAdmin with ring/star topology (static and cyclic/acyclic coordination graph); c) multi-UAV Delivery (dynamic and cyclic coordination graph).

usage of dataset $\mathcal{D}$) with MCTS with Max-Plus/Var-El (which improves algorithm efficiency and scalability compared to policy iteration based algorithms).

F. Domains

Multi-agent SysAdmin is a standard MMDP benchmark (Guestrin et al., 2003). In a network of interconnected machines, each agent controls a single machine described by two variables: *status* (*good*, *faulty*, or *dead*) and *load* (*idle*, *loaded*, or *success*). Initially, all machines are *good* and *idle*. Agents can, at each time step, activate their machines, or take no action. The actions may result in status changes, and agents are rewarded for machines reaching the *(good, success)* state. The state and action spaces grow exponentially with the number of agents n, with state space size $|S| = 9^n$ and action space size $A = 2^n$. Two network topologies are considered: a ring network (Figure 5.a) and a star network (Figure 5.b).

Multi-UAV Delivery was proposed in (Choudhury et al., 2021). Drones start from random cells and aim to reach circular target regions (T1-T4 in Figure 5.c). Each drone can take 10 actions, including movement, staying in place, and *boarding* in its assigned target region. Successful *boarding* rewards drones with $+1000$, and they receive intermediate rewards based on proximity. Drones incur penalties for movement away from targets, and penalties for collisions with other drones or simultaneous *boarding* in the same region. We note that the environments are much more complex than those used in the SPI literature. For instance, (Scholl et al., 2022) and (Laroche et al., 2019) use MDPs with 25 states and fewer than 10 actions. In contrast, our settings span a wide range, reaching approximately 10^{30} to 10^{41} states, and 10^9 to 10^{16} actions at the higher end of the spectrum.

G. Details about the relationship between quality of the MDP and safety/convergence

The quality of MDP estimation is inherently evaluated by the algorithm. This quality depends on how many times the state-action pairs (i.e., a sub-part of the transition model) have been observed in the dataset of trajectories. If a state-action pair has not been observed enough times, then it is a bootstrapped pair (see line 3 of Algorithm 2), and the action is returned with the probability of the behavior policy, hence the performance cannot be less than that of the behavior policy. In this case, the policy is a safe improvement since it has the same performance as the behavior policy. If, instead, the state-action pair has been observed enough times in the set of trajectories, then it is a non-bootstrapped pair, hence it is selected only if it is optimal, namely, if the state-action Q-value is the maximum for that state. The optimality is computed by Constrained Max-Plus or Constrained Var-El considering only the safe transitions of the state-action pair, namely the transitions that have been observed enough time, for which the uncertainty is low. In this way, the probability of making an error in selecting the optimal action is small, and the optimal action can only improve the performance compared to the behavior policy, namely, the policy is again a safe improvement since it has a small probability of having lower performance than the behavior policy.

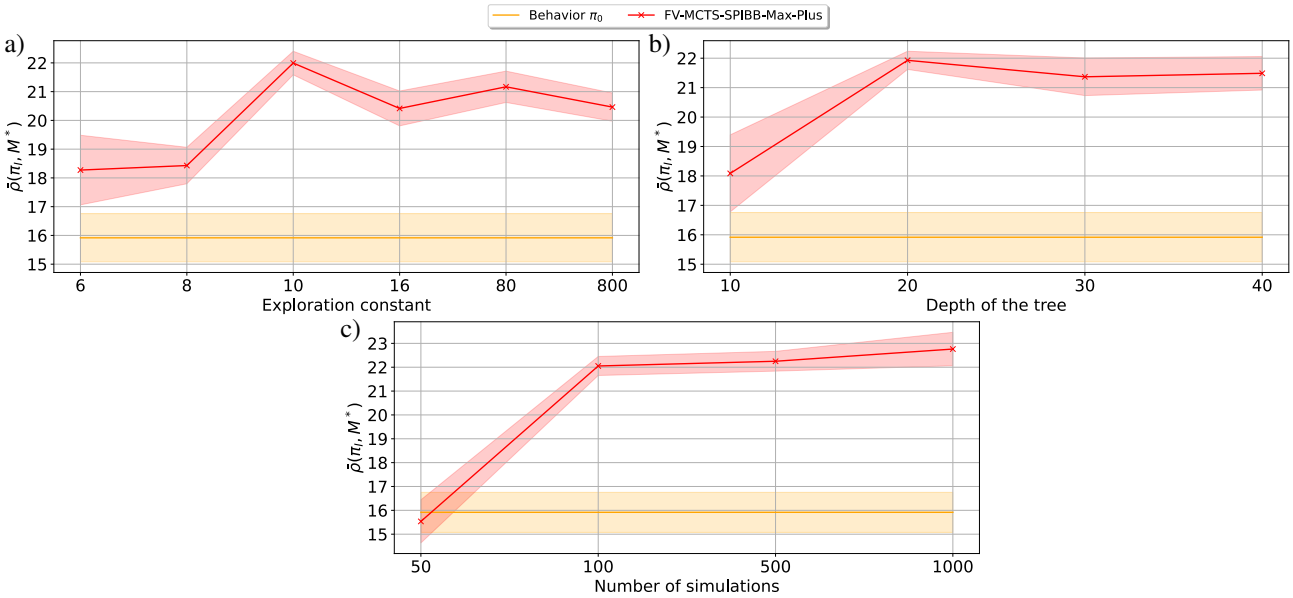

Figure 6. Multi-agent SysAdmin with ring topology: ablation study on the sensitivity of MCTS parameters, namely: a) on the UCT exploration constant; b) on tree depth of MCTS; c) on the number of simulations.

Some background on these concepts is provided in (Laroche et al., 2019). For the same reason, the quality of the transition model (or parts of it) does not even affect the convergence of the MCTS, since in case of low quality of the transition model MCTS converges to the baseline policy (without reaching any improvement), but it still converges.

H. Hyperparameters tuning - Sensitivity to MCTS parameters

We provide the results of the experiments executed to identify the best parameters for our FV-MCTS-SPIBB. In particular, we evaluated different values of the UCT exploration constant, the tree depth, and the number of simulations, on the multi-agent SysAdmin with 8 agents.

Exploration constant. We set the tree depth to 20, the number of simulations to 100, dataset size to 5000, and performed 5 experiments of 20 steps each, varying the exploration constant. Results are shown in Figure 6.a. The best exploration constant is equal to the number of agents + 2, which is the value that we also used in the paper.

Tree depth. We fixed the exploration constant to 10, the number of simulations to 100, dataset size to 5000, and performed 5 experiments of 20 steps each, varying the tree depth. Results are shown in Figure 6.b. The best tree depth is 20, which is the value that we also used in the paper.

Number of simulations. We fixed the exploration constant to 10, tree depth to 20, dataset size to 5000, and performed 5 experiments of 20 steps each, varying the number of simulations. Results are shown in Figure 6.c. Good performance is achieved from 100 simulations, which is the value we also used in the paper. Increasing the number of simulations to 1000 obtains a limited performance increase but also an increase in time required to perform the experiments.

I. Details about efficient state-action counting

FV-MCTS-SPIBB exploits the information contained in the dataset of trajectories to improve memory efficiency and achieve performance improvement in multi-agent scenarios. The algorithm extends (Simão & Spaan, 2019), where a method for improving the efficiency of state-action counting is presented. That technique, however, works on single-agent scenarios, hence it does not consider single-agent actions but only joint actions. Our efficient state-action counting achieves higher counts by exploiting the factorization of the transition model (see Section 4.2) and considering state-action counting at the agent level. This is fundamental for the algorithm because flat approaches with state-action counting at the joint state and joint action levels obtain very low counts, i.e., no state-action pairs would be put in the non-bootstrapped set using

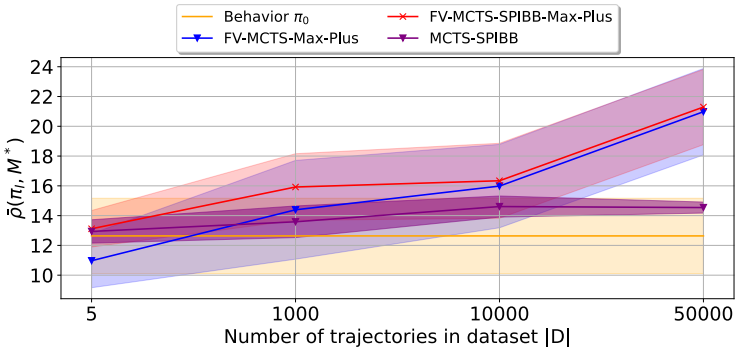

Figure 7. Multi-agent SysAdmin with ring topology: a) comparison on sample efficiency of FV-MCTS-SPIBB-Max-Plus (our method, red line), FV-MCTS-Max-Plus (Choudhury et al., 2021) (blue line) and MCTS-SPIBB (Castellini et al., 2023) (purple line) measured in terms of performance $\bar{\rho}(\pi, M^*)$ as the number of trajectories increases.

such a counting method. An evaluation of the impact that sample efficiency has on performance can be seen in Section 6. Figure 2 shows a comparison between FV-MCTS-SPIBB-Max-Plus (red boxplot), which exploits the factorization of the transition model and the value function, and MCTS-SPIBB (blue boxplot), which does not exploit these factorizations. If we focus, for instance, on the test with 8 agents, then we see that the performance of the improved policies generated by FV-MCTS-SPIBB-Max-Plus is higher (mean: 22.3, std: 2.8) than that of the improved policies generated by MCTS-SPIBB (mean: 13.4, std: 2.7), although we used only 100 simulations with FV-MCTS-SPIBB-Max-Plus and 1000 simulations with MCTS-SPIBB. The motivation for this difference is the higher sample efficiency of FV-MCTS-SPIBB-Max-Plus, which better exploits the information contained in the dataset of trajectories to achieve the performance improvement, as the dataset used by the two methods is the same, and MCTS-SPIBB had enough simulation to converge.

To further evaluate the sample efficiency, we also conducted another experiment comparing the performance of FV-MCTS-SPIBB-Max-Plus (our method), FV-MCTS-Max-Plus (Choudhury et al., 2021), MCTS-SPIBB (Castellini et al., 2023), and the behavior policy on a multi-agent SysAdmin system varying the dataset size. Experimental setting: number of agents = 8; number of simulations = 100; depth = 20; exploration constant = 10. Results are shown in Figure 7. The state-of-the-art method MCTS-SPIBB ensures safety and has better performance than the behavior policy, but it has lower performance than the proposed FV-MCTS-SPIBB-Max-Plus (mainly with few trajectories) since the latter has a higher sample efficiency. It is known from the literature (Oliehoek & Amato, 2016; Choudhury et al., 2021) that flat MCTS-based approaches in multi-agent scenarios require a larger number of simulations than factorized approaches.

J. Additional results on state-of-the-art baseline SPI methods

To provide a complete comparison between state-of-the-art SPI methods and the proposed FV-MCTS-SPIBB on multi-agent domains, we show the results of our test on SysAdmin with 3 agents, which is the largest number of agents that state-of-the-art methods can solve. We compare, in particular, SPIBB, MCTS-SPIBB, and FV-MCTS-SPIBB. We used the SPIBB code from (Laroche et al., 2019)[6], but it could not work even on small domains since it is based on matrix inversion. Hence we re-implemented it using dynamic programming to enable its execution on larger domains. It stopped working on instances of multi-agent SysAdmin with 4 agents. All other state-of-the-art algorithms mentioned in (Scholl et al., 2022) are mostly variants of SPIBB, with fewer safety guarantees (all algorithms with Soft in the acronym), hence they have scaling capabilities similar to SPIBB. Figure 8 shows that our FV-MCTS-SPIBB obtains similar performance to SPIBB and MCTS-SPIBB. However, in the main paper, we show that it can scale to a much larger number of agents because it exploits i) the factorization on the transition model to efficiently compute state-action counts; ii) the factorization of the value function (induced by the coordination graph) to select optimal joint actions without evaluating all of them; iii) the online MCTS strategy to compute the policy (i.e., action values) only for the states visited, instead of considering all possible joint state-action pairs.

[6]https://github.com/RomainLaroche/

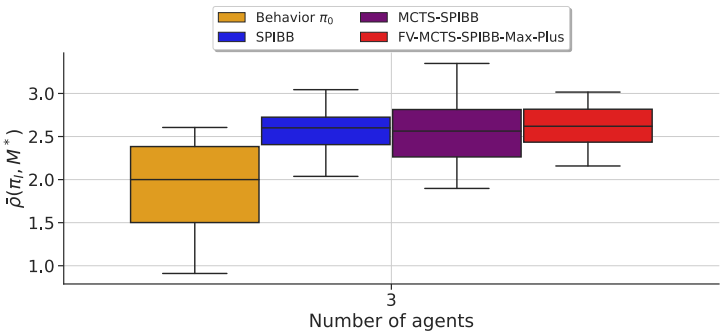

Figure 8. Multi-agent SysAdmin with ring topology: additional results of baseline SPI methods on multi-agent SysAdmin with 3 agents.

K. Details about Dependency identifiers

The dependency identifiers are domain-specific because they encode relations among state-action pairs. If the distribution over the future state $\bar{s}_{t+1}$ is identical conditioned on two different state-action pairs $(\bar{s}, \bar{a})$ and $(\bar{s}', \bar{a}')$, then they must be tagged with the same (arbitrary) identifier. For example, consider a simple two-machine domain (similar to Sysadmin) with a joint action space of dimension 2 (where each single action can be 0 or 1, which mean respectively *turn on the machine* and *turn it off*). Here, the status (encoded, for simplicity, also with 0 and 1) of the first machine depends only on the first action, and likewise, the status of the second machine depends only on the second one. So, $\bar{s} = 01$ and $\bar{a} = 11$ are one possible state and one possible action, respectively, in this domain. To encode the independence of the first machine's status on the second action we use dependency identifiers: in particular, we will tag with the arbitrary identifier j (for the first machine) both $(\bar{s}, \bar{a}) = (01, 11)$ and $(\bar{s}', \bar{a}') = (01, 10)$, that is, $D_1((01, 11)) = D_1((01, 10)) = j$ because the distribution over the first machine next statuses is the same whether the second action is 0 or 1 (again, it depends only on the first action).

L. Additional results on multi-agent SysAdmin with star and ring topology

In this section, we present some additional results. These are related to the FV-MCTS-SPIBB-Max-Plus method on the multi-agent SysAdmin domain with star topology with 16 agents, as FV-MCTS-SPIBB-Var-El fails to scale up to this dimension.

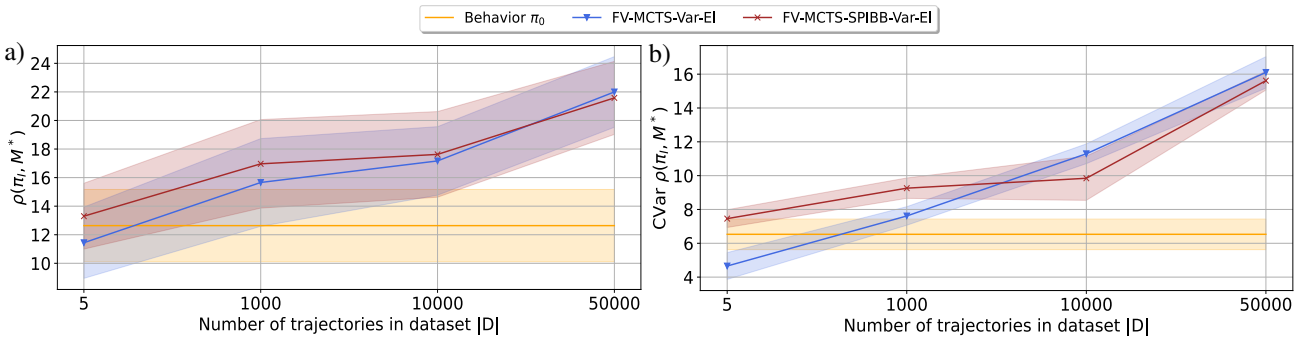

Figure 9. Multi-agent SysAdmin with ring topology: performance. a, b) Safety of FV-MCTS-SPIBB-Var-El ($\bar{\rho}(\pi, M^*)$ and CVaR) as the number of trajectories increases.

Multi-agent SysAdmin with ring topology (Var-El). For the multi-agent SysAdmin with ring topology, we consider the case with 8 agents and analyze the safety of the FV-MCTS-SPIBB-Var-El algorithm in comparison to FV-MCTS-Var-El. In this case, the coordination graph is cyclic (see Figure 5.a in the paper). The algorithm consistently ensures safe policy improvement across all trajectory sizes, whereas the unsafe version FV-MCTS-Var-El can be seen going under the behavior policy average performance with small dataset sizes ($d = 5$).

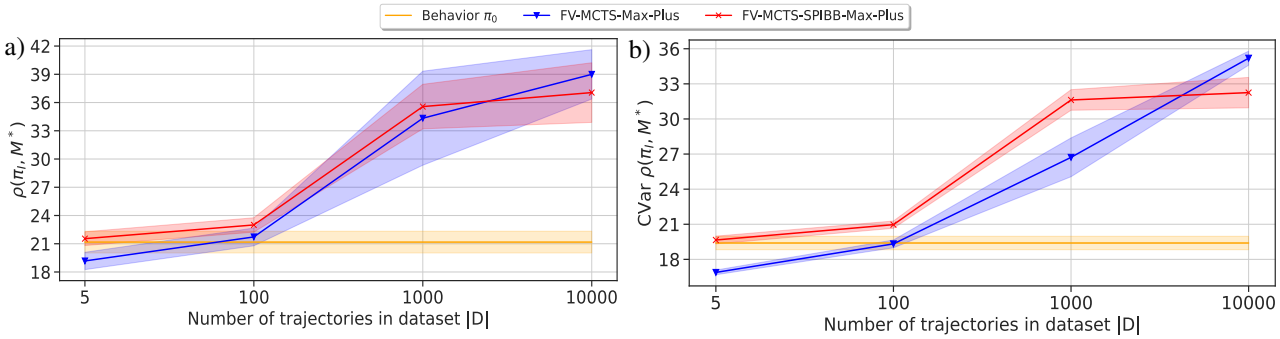

Figure 10. Multi-agent SysAdmin with star topology: performance. a, b) Safety of FV-MCTS-SPIBB-Max-Plus ($\bar{\rho}(\pi, M^*)$ and CVaR) as the number of trajectories increases.

Multi-agent SysAdmin with star topology. For the multi-agent SysAdmin with a star topology, instead, we use 16 agents to analyze the safety of the FV-MCTS-SPIBB-Max-Plus algorithm in comparison to FV-MCTS-Max-Plus. In this case, the coordination graph is acyclic (see Figure 5.b in the paper), because only one machine (the central one) is linked to every other machine, and there are no other links present. The algorithm consistently ensures safe policy improvement across all trajectory sizes $d \in [5, 100, 1000, 10000]$, for both the average performance $\bar{\rho}(\pi, M^*)$ (Figure 10.a) and the 15%-CVaR (Figure 10.b). In contrast, FV-MCTS-Max-Plus fails to outperform the behavior policy π_0 with the smallest dataset size.

Specifically, when $d = 5$, the average result for FV-MCTS-Max-Plus (19.2), over all 100 runs (Figure 10.a, blue line), is approximately 10% lower than π_0 (21.2), whereas FV-MCTS-SPIBB-Max-Plus (red line) still shows improvement (21.5), demonstrating safety with small dataset sizes. When the amount of data is increased ($d = 100$), both algorithms show improvement over the behavior policy, with FV-MCTS-SPIBB-Max-Plus outperforming its non-safe counterpart (23.0 vs. 21.7). The results become more comparable when the dataset size is increased to the maximum ($d = 10000$), where FV-MCTS-Max-Plus outperforms FV-MCTS-SPIBB-Max-Plus (38.9 vs. 37.1), albeit with higher overall variance. Even when considering the 15%-CVaR (Figure 10.b), the results with smaller datasets are worse for FV-MCTS-Max-Plus. While it achieves a performance of almost 36 at $d = 10000$, it barely outperforms the behavior π_0 at $d = 100$ (20.3 vs. 19.4), and it degrades at $d = 5$, where it achieves a performance of 17.8 compared to the behavior's performance of 19.4, resulting, approximately, in a 9% loss. In all cases, FV-MCTS-SPIBB-Max-Plus achieves a performance at least as good as the behavior policy, demonstrating safety.