Scalable Safe Policy Improvement for Single and Multi-Agent Systems

Federico Bianchi[1,*], Alberto Castellini[1] and Alessandro Farinelli[1]

[1]Department of Computer Science, University of Verona, Str. le Grazie, 15, 37134 Verona, Italy

Abstract

Safe Policy Improvement (SPI) is crucial in domains where reliable decision-making must be achieved with limited environmental interaction, given the high costs and risks involved. Although existing SPI algorithms ensure improved safety over baseline policies, they struggle to scale to large and complex problems. In this work, we discuss new approaches to enhance the scalability and safety of SPI for both single-agent and multi-agent systems. For single-agent scenarios, we introduce MCTS-SPIBB, which combines Monte Carlo Tree Search with Safe Policy Improvement with Baseline Bootstrapping, and SDP-SPIBB, a scalable dynamic programming approach that extends SPI to large domains while preserving safety guarantees. For multi-agent settings, we present Factored Value-MCTS-SPIBB, the first SPI method to address large-scale multi-agent problems effectively. Through theoretical and empirical evaluation, we show that our algorithms scale efficiently and maintain the safety properties of SPI, thus making SPI applicable to complex and large-scale scenarios.

Keywords

Reinforcement Learning, Safe Policy Improvement, Single-agent systems, Multi-agent systems

1. Introduction

Safety [1] is critical for deploying Reinforcement Learning (RL) algorithms in real-world scenarios [2], especially in domains like autonomous driving, healthcare, robotics and environmental monitoring [3, 4, 5], where reliable decision-making is essential, and data collection can be risky or expensive [6, 7, 8, 9]. Safe Policy Improvement [10] is a specialization of offline RL [11, 12] that assumes knowledge of a baseline policy and limited interaction with the environment from which a dataset is collected. It provides probabilistic guarantees that the new policy's performance will improve over the baseline, thereby addressing reliability issues inherent to offline RL, such as distributional shifts and extrapolation errors [13, 14], that arise when the policy encounters states and actions not well represented in the training data. SPI methods can be broadly categorized into two main groups based on how they manage uncertainty in the agent's states and actions: i) methods that handle uncertainty by reducing the estimated values of uncertain actions and ii) methods that handle uncertainty by restricting the space policies that can be learned. SPIBB [15] is a state-of-the-art method for SPI that constrains the space of learnable policies and extends the optimal policy iteration algorithm by bootstrapping from the baseline policy in states where the actions have high uncertainty, i.e., states and actions not sufficiently represented in the collected dataset. This strategy effectively limits the search for improved policies to a region where the model's estimates are sufficiently reliable. However, the computational complexity of SPIBB and other SPI methods restricts their applicability to real-world problems, largely due to the complexity of the underlying algorithms, such as policy iteration [10].

In this work, we discuss key contributions to improve SPI scalability in both single-agent and multi-agent systems. For single-agent systems, we propose Monte Carlo Tree Search SPIBB (MCTS-SPIBB), which integrates MCTS [16, 17, 18, 19] with SPIBB for scalable policy computation in large state spaces. Additionally, we propose SDP-SPIBB, which reduces SPIBB complexity by focusing policy updates only on relevant state-action subspaces, enabling it to scale to large domains. For multi-agent systems, we introduce Factored Value Monte Carlo Tree Search SPIBB (FV-MCTS-SPIBB), which leverages

11th Italian Workshop on Artificial Intelligence and Robotics (AIRO 2024)

*Corresponding author.

federico.bianchi@univr.it (F. Bianchi); alberto.castellini@univr.it (A. Castellini); alessandro.farinelli@univr.it (A. Farinelli)

action-value factorization to scale efficiently in state and action spaces with dimension which grows exponentially on the number of agents. In this context, we also propose two novel action-selection strategies, Constrained Max-Plus and Constrained Variable Elimination (Var-El), which guarantee safety criteria defined by SPIBB. Additionally, the factorization of the transition model allows the algorithm to trust a larger number of state-action pairs, improving the baseline policy more effectively. Empirical evaluations on large-scale problems highlight the effectiveness of these methods in improving policy performance while maintaining safety guarantees.

2. Preliminaries

In this section, we introduce the background for Safe Policy Improvement and Safe Policy Improvement with Baseline Bootstrapping.

2.1. Safe Policy Improvement

Let an unknown finite Markov Decision Process (MDP) $M^* = \langle S, A, T^*, R, \gamma \rangle$ represent a true environment where T^* is an unknown transition model and R a known reward function. $\Pi = \{S \to \Delta_A\}$ is the set of stochastic policies, where Δ_A denotes the set of probability distributions over the set of actions A. Given a policy subset $\Pi' \subseteq \Pi$, a policy Π' is Π'-optimal for an MDP M when it maximizes its performance on Π' [15]:

$$\rho(\pi', M) = \max_{\pi \in \Pi'} \rho(\pi, M). \tag{1}$$

SPI approaches focus on the Offline RL setting where the algorithm does its best at learning a policy from a fixed set of experiences. Given a dataset $\mathscr{D} = (s_j, a_j, r_j, s'_j)_{j=1}^n$ collected by a baseline policy π_0, let $N_{\mathscr{D}}(s, a)$ denote the number of visits to the state-action pair $(s, a) \in \mathscr{D}$. We construct the Maximum Likelihood Estimator (MLE) $M^{\mathscr{D}} = \langle S, A, T^{\mathscr{D}}, R, \gamma \rangle$ of M^* as follows:

$$T^{\mathscr{D}}(s' \mid s, a) = \frac{\sum_{(s_j=s, a_j=a, s'_j=s')} \mathbb{1}}{N_{\mathscr{D}}(s, a)} \tag{2}$$

This is common in real-world domains where the transition model must be estimated or inferred from small amounts of data. The *safety* of the improvement must be guaranteed, specifically, π_I must outperform π_0 with an admissible performance loss. A significant approach in this context is the *percentile criterion* [20], which aims to improve policy performance while maintaining a high probability of improvement over a baseline policy π_0. The percentile criterion is defined as follows:

$$\pi_C = \arg\max_{\pi \in \Pi} \mathbb{E}[\rho(\pi, M) \mid M \sim \mathbb{P}_{MDP}(\cdot \mid \mathscr{D})], \tag{3}$$

subject to the constraint:

$$\mathbb{P}(\rho(\pi_I, M) \geq \rho(\pi_0, M) - \zeta \mid M \sim \mathbb{P}_{MDP}(\cdot \mid \mathscr{D})) \geq 1 - \delta, \tag{4}$$

where $\mathbb{P}_{MDP}(\cdot|\mathscr{D})$ represents the posterior probability distribution of the MDP parameters given the dataset $\mathscr{D}$, ρ denotes the policy performance, ζ is an approximation parameter (or precision level), and $1 - \delta$ denotes a high confidence level.

2.2. Safe Policy Improvement with Baseline Bootstrapping

The SPIBB [15] algorithm reformulates the *percentile criterion* and aims to maximize the policy's performance in the estimated MDP $M^{\mathscr{D}}$, guaranteeing that the improved policy π_I is ζ-approximately at least as good as the baseline policy π_0:

$$\max_{\pi_I \in \Pi} \rho(\pi_I, M^{\mathscr{D}}), \text{ s.t. } \forall M \in \Xi,\ \rho(\pi_I, M) \geq \rho(\pi_0, M) - \zeta, \tag{5}$$

within the set of admissible MDPs Ξ:

$$\Xi(M^{\mathscr{D}}, e) = \{M \mid \forall (s,a) \in S \times A, \|T(s,a,\cdot) - T^{\mathscr{D}}(s,a,\cdot)\|_1 \leq e(s,a)\}, \tag{6}$$

where $e : S \times A \to \mathbb{R}$ is an error function depending on $\mathscr{D}$ and δ. Based on Theorem 8 from Petrik et al. [21], SPIBB guarantees that if all state-action pair counts $N_{\mathscr{D}}(s,a)$ meet the condition:

$$N_{\mathscr{D}}(s,a) \geq N_\wedge = \frac{8V_{max}^2}{\zeta^2(1-\gamma)^2} \log\left(\frac{2|S||A|2^{|S|}}{\delta}\right), \tag{7}$$

and $M^{\mathscr{D}}$ is the Maximum Likelihood Estimation MDP, then with high probability $1-\delta$, the optimal policy $\pi^* = \arg\max_{\pi \in \Pi} \rho(\pi, M^{\mathscr{D}})$ in $M^{\mathscr{D}}$ is ζ-approximately safe in the true environment M^*:

$$\rho(\pi^*, M^*) \geq \rho(\pi^*, M^{\mathscr{D}}) - \zeta \geq \rho(\pi_0, M^*) - \zeta. \tag{8}$$

These conditions ensure that performance estimates in $M^{\mathscr{D}}$ generalize safely to M^*. To implement this, SPIBB splits state-action pairs into two subsets, the bootstrapped subset $\mathscr{B} = \{(s,a) : N_{\mathscr{D}}(s,a) < N_\wedge\}$, which includes state-action pairs that occur fewer than $N_\wedge$ times in $\mathscr{D}$ and the non-bootstrapped set $\overline{\mathscr{B}} = \{(s,a) : N_{\mathscr{D}}(s,a) \geq N_\wedge\}$, which includes state-action pairs that occur at least $N_\wedge$ times in $\mathscr{D}$. Hereafter, we assume that the terms baseline policy and behavior policy can be used interchangeably.

3. Method

We developed three scalable SPI algorithms, two for single-agent domains [22] and one for multi-agent domains [23]. The main ideas and contributions are explained below.

3.1. Safe Policy Improvement for single-agent systems

MCTS-SPIBB. The first algorithm is a Monte Carlo Tree Search [16] extension of SPIBB [15]. As MCTS can approximate optimal policies generated by policy iteration, MCTS-SPIBB can approximate Π_0-optimal policies generated by SPIBB, starting from a baseline π_0. Since MCTS-SPIBB computes the policy online and locally it can scale to larger state problems than SPIBB.

The core idea of MCTS-SPIBB is to extend UCT [24] while considering the safety constraint on action selection. This presents several challenges, such as the fact that UCT selects actions based on Q-values while the safety constraint is on action selection probabilities, and this constraint's impact accumulates throughout the layers of the Monte Carlo tree. For a given state s in the tree, actions are divided into two categories: bootstrapped state-action pairs $(s,a) \in \mathscr{B}$ and non-bootstrapped pairs $(s,a) \in \overline{\mathscr{B}}$. When the simulation reaches state s a bootstrapped action is selected with a probability $p^s_{\mathscr{B}} = \sum_{a \in \mathscr{B}_A(s)} \pi_0(s,a)$, where $\mathscr{B}_A(s)$ represents the set of bootstrapped actions for state s, while a non-bootstrapped action is selected with probability $p^s_{\overline{\mathscr{B}}} = 1 - p^s_{\mathscr{B}}$. If a bootstrapped action is selected, it is chosen according to the probability distribution of the baseline policy $\pi_0(s,\cdot)$. If a non-bootstrapped action is chosen, it is selected using the UCT strategy, which considers current Q-value estimates and visit counts, ensuring that the optimal action is chosen given enough simulations. During the rollout phase, baseline probabilities are applied to bootstrapped actions, while non-bootstrapped actions are selected uniformly. At the end of the simulations, the estimated Q-values $Q(s,a)$ for the root state s are used to compute the probabilities for the improved policy $\pi^\circ(s,a)$ as follows: *i)* $\pi_0(s,a)$ if $a \in \mathscr{B}(s)$, *ii)* $1 - p^s_{\mathscr{B}}$ if $a = \arg\max_{a' \in \overline{\mathscr{B}}_A(s)} Q(s,a')$, and *iii)* 0 otherwise. The proposed action selection strategy integrates UCT and baseline probabilities in the MCTS allowing the generation of improved policies with probabilistic guarantees on the improvement.

The complexity of MCTS-SPIBB scales linearly with the number of Monte Carlo simulations m, namely, it is $O(m)$. This means that the computational effort required by MCTS-SPIBB is directly proportional to the number of simulations performed. Each simulation in MCTS-SPIBB is used to estimate the value of different actions by exploring potential future states. As the number of simulations

increases, the accuracy of the action value estimates improves, resulting in a corresponding linear increase in the computational complexity.

Scalable Dynamic Programming SPIBB (SDP-SPIBB). SPIBB uses policy iteration to generate the improved policy. This algorithm has a complexity $O(|S|^2|A|^2)$ due to the four nested loops over states, actions, next states, and next actions required to update the Q-function. However, in SPI, the value updates can be performed only on state-action pairs where MLE transition probabilities are non-zero, which correspond to state-action pairs observed a sufficient number of times in the dataset of trajectories. This observation allows us to reduce the complexity of SPIBB from $O(|S|^2|A|^2)$ to a term depending only on the dataset size $|\mathscr{D}|$, and in particular on the size of the non-bootstrapped state-action pairs, $\overline{\mathscr{B}}$. These pairs have been observed at least $N_\wedge$ times in the dataset. This complexity change provides a strong increase of scalability because the dataset size is usually much smaller than $O(|S|^2|A|^2)$ in large domains. However, since the complexity still depends on the dataset size, it can grow in applications where huge amounts of data are collected over time (e.g., streaming data).

3.2. Safe Policy Improvement for multi-agent systems

FV-MCTS-SPIBB. Let the Factored Multi-agent MDP (FMMDP) $M^* = \langle S, \alpha, \{A_i\}_{i\in\alpha}, T^*, R, \gamma\rangle$ [25] represent the true environment, where α is a set of agents and A_i is the set of actions of agent i. A central behavior policy π_0 is executed in this environment controlling all agents, and a dataset $\mathscr{D}$ of trajectories is collected. Each sample in $\mathscr{D}$ consists of a joint state, joint action, joint next state, and joint reward, denoted as $(\bar{s}, \bar{a}, \bar{r}, \bar{s}')$. This dataset is then used to compute the MLE FMMDP $M^{\mathscr{D}} = \langle S, \alpha, A_{i|i\in\alpha}, T^{\mathscr{D}}, R, \gamma\rangle$, where the transition model $T^{\mathscr{D}}$ is factorized according to dependency functions D_k. We define the set of bootstrapped joint state-action pairs as

$$\mathscr{B}_m = \left\{(\bar{s}, \bar{a}) \in S \times A \mid \exists S_k \,:\, n(D_k(\bar{s}, \bar{a})) < m_k\right\} \tag{9}$$

and the set of non-bootstrapped joint state-action pairs as

$$\bar{\mathscr{B}}_m = \left\{(\bar{s}, \bar{a}) \in S \times A \mid \forall S_k \,:\, n(D_k(\bar{s}, \bar{a})) \geq m_k\right\}. \tag{10}$$

FV-MCTS-SPIBB is based on the *percentile criterion*, adapted to FMMDPs, and achieves scalability in multi-agent systems by leveraging the factorization of the action-value function induced by a coordination graph (CG) $\mathscr{G} = \langle \mathscr{V}, \mathscr{E}\rangle$ [26, 27], where each agent is represented as a node $i \in \mathscr{V}$ and each pair of agents that needs to coordinate is represented by an edge $(i, j) \in \mathscr{E}$. The action-value function is decomposed as

$$Q(\bar{a}) = \sum_{i\in\mathscr{V}} Q_i(a_i) + \sum_{i,j\in\mathscr{E}} Q_{ij}(a_i, a_j), \tag{11}$$

providing an action-value function for each agent and each edge of the graph. This decomposition greatly reduces the number of joint actions to consider at each state in MCTS. To select among non-bootstrapped actions, FV-MCTS-SPIBB uses two novel action selection strategies: Constrained Max-Plus and Constrained Variable Elimination, both of which guarantee that the selected actions are optimal (despite a large number of possible joint actions) and that the resulting policy safely improves upon the behavior policy. FV-MCTS-SPIBB extends the Factored SPIBB approach from [28] by considering local actions a_i associated with the components D_k, rather than using joint actions $\bar{a}$.

This modification exploits the factorization of the transition model, allowing state-action counts at the agent level, which results in larger counts than other SPIBB variants that rely on joint actions. This requires transition function factorizability, which leads to greater potential for improving the behavior policy. Flat approaches in the SPI literature count state-action pairs at the joint state and joint action levels, producing a smaller set of non-bootstrapped joint state-action pairs $\bar{B}_m$ compared to our approach, which counts state-action pairs at local levels. Full details about the FV-MCTS-SPIBB algorithm can be found in [23]. FV-MCTS-SPIBB's complexity depends on the action selection strategy. In the case of Max-Plus, the complexity is linear in the size of the CG, and guarantees of convergence to optimality are provided for acyclic CGs. On cyclic CGs, these guarantees do not hold, but empirically,

Max-Plus provides approximately optimal results even on cyclic structures. The complexity of the Var-Elimination algorithm is exponential in the treewidth, a parameter related to the graph's cyclicity. The algorithm guarantees convergence for any type of CG, but finding the treewidth of a graph is a difficult (NP-hard) problem, although it can be easily estimated with Depth-First Search (DFS). Therefore, it is possible to decide whether to use Max-Plus or Var-Elimination by evaluating the treewidth of the graph.

4. Empirical analysis

In this section, we present results focusing on the scalability and safety of our proposed methods in the multi-agent SysAdmin domain [26, 27].

4.1. Benchmark

In the multi-agent SysAdmin domain, each agent controls a machine characterized by two state variables: a *status*, which can be *good*, *faulty*, or *dead*, and a *load*, which can be *idle*, *loaded*, or *success*, both initially set to *good* and *idle*. At each step, agents can activate their machines or do nothing, aiming to achieve *(good, success)* states for rewards. Although the coordination graph is static, complexity arises from reasoning about joint actions and their network-wide effects. Poor coordination can result in suboptimal outcomes, such as unnecessary simultaneous reboots. As the number of agents n increases, the size of the state and action spaces grows exponentially. Specifically, the complexity follows $|S| = 9^n$ and $|A| = 2^n$, where $|S|$ is the number of possible states and $|A|$ is the number of possible actions.

4.2. Experimental overview

Among the SPI methods tested, only MCTS-SPIBB, SDP-SPIBB, and FV-MCTS-SPIBB can handle the problem, since SPIBB [15] and other SPI approaches [10] cannot effectively scale to such large domains. Figure 1 provides box-plots of the average return $\bar{\rho}(\pi, M^*)$ (y-axis) as the number of agents increases from 4 to 32. For FV-MCTS-SPIBB-Max-Plus and FV-MCTS-SPIBB-Var-El, we use the following parameters: 100 simulations, an empirically determined exploration constant of $c = n$ (where n is the number of agents), an MCTS tree depth of 20 steps, $\gamma = 0.9$, and 8 iterations of message passing in Constrained Max-Plus. For MCTS-SPIBB, similar parameters are used, but 10000 simulations are required, as it does not leverage model factorization and needs more simulations.

4.3. Results on scalability

With 4 agents, all methods show improvements over the behavior policy π_0 (orange box). FV-MCTS-SPIBB-Var-El (green box) slightly outperforms FV-MCTS-SPIBB-Max-Plus (red box), with both methods outperforming MCTS-SPIBB (blue box) and SDP-SPIBB (steelblue). With 8 agents, FV-MCTS-SPIBB-Max-Plus and FV-MCTS-SPIBB-Var-El provide significant and similar improvements over the behavior policy, but the performance gap between these methods and MCTS-SPIBB widened (i.e., FV-MCTS-SPIBB methods achieves around 22.0, while MCTS-SPIBB and SDP-SPIBB score around 15.0, and the behavior policy reaches approximately 13.0). With 16 agents, FV-MCTS-SPIBB-Var-El cannot compute actions within a reasonable time due to the exponential complexity of Var-El, which is tied to the induced width of the CG and the elimination order. MCTS-SPIBB and SDP-SPIBB show some improvement over the baseline, but FV-MCTS-SPIBB-Max-Plus outperforms them. With 24 and 32 agents, only FV-MCTS-SPIBB-Max-Plus can improve performance compared to the behavior policy, as MCTS-SPIBB and SDP-SPIBB break due to the exponential number of available actions. This experiment shows that FV-MCTS-SPIBB-Max-Plus is the only approach able to scale to large multi-agent domains, in which the dimensions of the state and action spaces become huge because they grow exponentially on the number of agents (e.g., multi-agent SysAdmin with 32 agents has 10^{30} possible joint states and 10^9 possible joint actions).

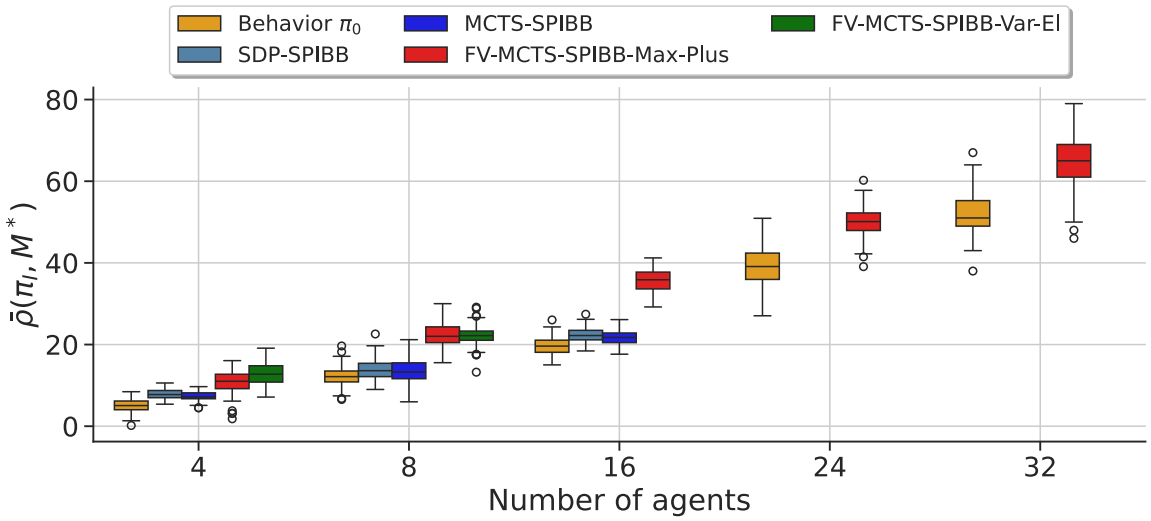

Figure 1: Multi-agent SysAdmin [26]: Scalability and safety of MCTS-SPIBB, SDP-SPIBB and FV-MCTS-SPIBB (with Max-Plus and Var-El) performance as the number of agents increases.

5. Conclusion

In this work, we discuss approaches for scalable, safe policy improvement in both single-agent and multi-agent systems. We introduced two novel Safe Policy Improvement methods for single-agent systems for large-scale problems. The first method, MCTS-SPIBB, is an MCTS-based approach and the second, SDP-SPIBB, is based on dynamic programming. For multi-agent systems, we introduced FV-MCTS-SPIBB, an extension of MCTS-SPIBB that scales assuming a transition model and Q-function factorization. Our empirical evaluation, conducted in a large-scale benchmark domain, shows that both SDP-SPIBB and MCTS-SPIBB can scale and achieve policy improvement in single-agent scenarios where other state-of-the-art SPI algorithms cannot work. FV-MCTS-SPIBB outperforms all other algorithms in multi-agent scenarios. By addressing the computational limitations of current SPI algorithms, particularly SPIBB methods, this work expands the range of problems that can be safely addressed with reinforcement learning, contributing to developing more reliable AI systems.

While these contributions represent significant advancements over the state-of-the-art SPI, they also raise important directions for future research. An open challenge in this context is to provide theoretical guarantees on policy improvement when the policy is approximated by a general function (e.g., a neural network). In particular, incorporating function approximation techniques, such as linear models or deep neural networks, into our SPI algorithms may enhance scalability by addressing bottlenecks related to the space complexity associated with large-scale problems. Furthermore, applying the proposed methodologies to real-world systems, such as autonomous vehicles or robotic platforms, would provide valuable insights into their practical utility and limitations.

Acknowledgments

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

References

[1] J. García, F. Fernández, A Comprehensive Survey on Safe Reinforcement Learning, JMLR 16 (2015) 1437–1480.

[2] R. S. Sutton, A. G. Barto, Reinforcement Learning: An Introduction, second ed., The MIT Press, 2018.

[3] J. Cacace, R. Caccavale, A. Finzi, R. Grieco, Combining human guidance and structured task execution during physical human–robot collaboration, Journal of Intelligent Manufacturing 34 (2022) 3053–3067.

[4] R. De Benedictis, G. Beraldo, G. Cortellessa, F. Fracasso, A. Cesta, A transformer-based approach for choosing actions in social robotics, in: Social Robotics, Springer, 2022, pp. 198–207.

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

[6] D. Meli, A. Castellini, A. Farinelli, Learning logic specifications for policy guidance in POMDPs: an inductive logic programming approach, Journal of Artificial Intelligence Research (JAIR) 79 (2024) 725–776.

[7] R. Cipollone, G. De Giacomo, M. Favorito, L. Iocchi, F. Patrizi, Exploiting multiple abstractions in episodic RL via reward shaping, Proceedings AAAI Conference on Artificial Intelligence 37 (2023) 7227–7234.

[8] G. Mazzi, A. Castellini, A. Farinelli, Risk-aware shielding of Partially Observable Monte Carlo Planning policies, Artificial Intelligence 324 (2023) 103987.

[9] G. Mazzi, A. Castellini, A. Farinelli, Active generation of logical rules for pomcp shielding, in: Proceedings AAMAS 2022, IFAAMAS, 2022, pp. 1696–1698.

[10] P. Scholl, F. Dietrich, C. Otte, S. Udluft, Safe policy improvement approaches and their limitations, in: Agents and Artificial Intelligence, Springer International Publishing, Cham, 2022, pp. 74–98.

[11] S. Levine, A. Kumar, G. Tucker, J. Fu, Offline reinforcement learning: tutorial, review, and perspectives on open problems, arXiv preprint arXiv:2005.01643, 2020.

[12] R. F. Prudencio, M. R. Maximo, E. L. Colombini, A survey on offline reinforcement learning: Taxonomy, review, and open problems, IEEE Trans. Neural Networks and Learning Systems (2023).

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

[14] A. Kumar, J. Fu, M. Soh, G. Tucker, S. Levine, Stabilizing off-policy Q-learning via bootstrapping error reduction, in: Proceedings of the 32th Conference on Neural Information Processing Systems (NeurIPS), Curran Ass. Inc., 2019, pp. 11761–11771.

[15] R. Laroche, P. Trichelair, R. Tachet Des Combes, Safe policy improvement with baseline bootstrapping, in: Proceedings 36th International Conference on Machine Learning (ICML), PMLR, 2019, pp. 3652–3661.

[16] C. Browne, E. J. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. P. Liebana, S. Samothrakis, S. Colton, A survey of monte carlo tree search methods, IEEE Transactions on Computational Intelligence and AI in Games 4 (2012) 1–43.

[17] A. Castellini, G. Chalkiadakis, A. Farinelli, Influence of State-Variable Constraints on Partially Observable Monte Carlo Planning, in: Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI, International Joint Conferences on Artificial Intelligence Organization, 2019, pp. 5540–5546. doi:10.24963/ijcai.2019/769.

[18] M. Zuccotto, M. Piccinelli, A. Castellini, E. Marchesini, A. Farinelli, Learning state-variable relationships in pomcp: A framework for mobile robots, Frontiers in Robotics and AI 9 (2022) 1–18.

[19] M. Zuccotto, E. Fusa, A. Castellini, A. Farinelli, Online model adaptation in monte carlo tree search planning, Optimization and Engineering (2024).

[20] E. Delage, S. Mannor, Percentile optimization for Markov Decision Processes with parameter uncertainty, Operations Research 58 (2010) 203–213.

[21] M. Petrik, M. Ghavamzadeh, Y. Chow, Safe policy improvement by minimizing robust baseline regret, in: Proceedings of the 30th International Conference on Neural Information Processing Systems (NIPS 2016), Curran Associates Inc., Red Hook, NY, USA, 2016, p. 2306–2314.

[22] A. Castellini, F. Bianchi, E. Zorzi, T. D. Simão, A. Farinelli, M. T. J. Spaan, Scalable safe policy improvement via Monte Carlo tree search, in: Proceedings of the 40th International Conference on Machine Learning (ICML 2023), PMLR, 2023, pp. 3732–3756.

[23] F. Bianchi, E. Zorzi, A. Castellini, T. D. Simão, M. T. J. Spaan, A. Farinelli, Scalable safe policy improvement for factored multi-agent MDPs, in: Proceedings of the 41st International Conference on Machine Learning (ICML 2024), PMLR, 2024, pp. 3952–3973.

[24] L. Kocsis, C. Szepesvári, Bandit based monte-carlo planning, in: Proceedings of the 17th European Conference on Machine Learning (ECML 2006), Springer-Verlag, 2006, p. 282–293.

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

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

[27] C. Guestrin, D. Koller, R. Parr, S. Venkataraman, Efficient solution algorithms for factored MDPs, Journal of Artificial Intelligence Research 19 (2003) 399–468.

[28] T. D. Simão, M. T. J. Spaan, Safe policy improvement with baseline bootstrapping in factored environments, in: Proceedings AAAI Conference on Artificial Intelligence, AAAI Press, 2019, pp. 4967–4974.