Structure Learning for Safe Policy Improvement

Simão, Thiago D.; Spaan, Matthijs T.J.

DOI
10.24963/ijcai.2019/479

Publication date
2019

Document Version
Final published version

Published in
Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence

Structure Learning for Safe Policy Improvement

Thiago D. Simão and **Matthijs T. J. Spaan**
Delft University of Technology, The Netherlands
{t.diassimao, m.t.j.spaan}@tudelft.nl

Abstract

We investigate how Safe Policy Improvement (SPI) algorithms can exploit the structure of factored Markov decision processes when such structure is unknown a priori. To facilitate the application of reinforcement learning in the real world, SPI provides probabilistic guarantees that policy changes in a running process will improve the performance of this process. However, current SPI algorithms have requirements that might be impractical, such as: (i) availability of a large amount of historical data, or (ii) prior knowledge of the underlying structure. To overcome these limitations we enhance a Factored SPI (FSPI) algorithm with different structure learning methods. The resulting algorithms need fewer samples to improve the policy and require weaker prior knowledge assumptions. In well-factorized domains, the proposed algorithms improve performance significantly compared to a flat SPI algorithm, demonstrating a sample complexity closer to an FSPI algorithm that knows the structure. This indicates that the combination of FSPI and structure learning algorithms is a promising solution to real-world problems involving many variables.

1 Introduction

Reinforcement Learning (RL) algorithms aim to make smart decisions during interactions with an unknown environment. *Safe* RL mitigates undesirable effects experienced during the learning process of such algorithms [García and Fernández, 2015]. For instance, in applications where a policy π_b, called the behavior policy, is already in use, deploying a new policy π computed by an RL algorithm might decrease the performance of the system. In these situations, it is important to provide guarantees that π is better than π_b, otherwise one would prefer to keep executing π_b to avoid such risks. To do so, a safe RL algorithm can use data collected while executing π_b to compute π and estimate π's performance.

In particular, Safe Policy Improvement (SPI) is the problem of computing a policy that is better than π_b with high confidence [Thomas *et al.*, 2015]. In this setting, an RL algorithm can either return an improved policy, which will replace π_b, or return π_b, indicating that it did not find a better policy and that π_b should continue to be used. An SPI algorithm is considered *safe* if it has a high probability of returning a policy at least as good as π_b.

Until recently, SPI algorithms used flat representations [Thomas *et al.*, 2015; Petrik *et al.*, 2016; Laroche *et al.*, 2019]. This limited their application since, to be reliable, they would need a large number of samples before returning an improved policy instead of π_b. In previous work [Simão and Spaan, 2019], we showed how SPI algorithms that use a factored representation require fewer samples to return an improved policy by exploiting the independence between the variables of the problem to generalize past experiences.

Nevertheless, this factored approach assumes that the local structure is known a priori, which can be quite impractical in many applications. For instance, in applications with many variables, it is difficult for an expert to provide the exact structure of the problem. Therefore, it is necessary to investigate how to relax this assumption, without resorting to flat representations to maintain a good sample complexity. We aim to develop SPI algorithms that are sample efficient even when the structure of the problem is unknown.

The problem of learning the structure has already been investigated from different perspectives. For example, algorithms have been proposed to improve exploration efficiency [Strehl *et al.*, 2007; Diuk *et al.*, 2009] and to increase the accuracy of off-policy evaluation methods [Hallak *et al.*, 2015]. These methods, however, do not consider the safety of the learning agent. In fact, typical approaches consider "optimism in the face of uncertainty", which is highly undesirable in a safe RL setting.

To address safety in environments with an unknown structure, we propose an SPI framework that can use different structure learning algorithms to estimate the dynamics of the environment. We provide two algorithms that instantiate the new SPI framework using different structure learning algorithms and prove that they have safety guarantees. Our experiments compare these algorithms to an algorithm that has access to the structure of the problem [Simão and Spaan, 2019] and to an algorithm that ignores the underlying structure [Laroche *et al.*, 2019]. They show that, depending on the structure learning algorithm and how well the problem can be factorized, this framework can yield algorithms with performance competitive to an algorithm that knows the structure.

2 Background

In this section we review basic concepts related to Reinforcement Learning and how to derive sample efficient algorithms given different levels of prior knowledge about the structure of the problem.

2.1 MDPs and Factored MDPs

The Markov Decision Process (MDP) [Puterman, 1994] is a common framework to model the interaction between a decision maker (the agent) and its environment (the world). Formally, an MDP M is represented by a set of states of the world S, a set of actions of the agent A, a transition function $P(s'|s,a)$ that represents the probability of moving to $s' \in S$ after executing $a \in A$ in $s \in S$, a reward function $R : S \times A \to \mathbb{R}$ that indicates the reward obtained after executing $a \in A$ in $s \in S$, and the discount factor $\gamma \in [0,1)$ that captures the agent's preference for immediate rewards over future rewards. The solution of an MDP is a policy $\pi : S \times A \to [0,1]$ that represents the behavior of the agent. The optimal policy maximizes the sum of the expected discounted reward $V(\pi, M) = E[\sum_{t=0}^{\infty} \gamma^t R_t \mid \pi, S_0 = s_0]$, where R_t is a random variable indicating the reward the agent receives at time step t.

A sequence of interactions between the agent and the environment is represented by an episode $h = [s_t, a_t, r_t, s_{t+1}, \cdots]$ where $a_t \sim \pi(s_t)$ is the action applied in the state s_t and $r_t = R(s_t, a_t)$ is the reward obtained and $s_{t+1} \sim P(\cdot \mid s_t, a_t)$. We denote a set of episodes by $\mathcal{D} = \{h_i \mid i \in [1, |D|]\}$.

A Factored MDP (FMDP) [Boutilier *et al.*, 1995] is a compact formalization of MDPs for problems where the state space S is represented by a set of *state variables* $X = \{X_1, \ldots, X_{|X|}\}$, where each variable might assume a value x_i from its domain $dom(X_i)$. Two assumptions are commonly made that allow us to compactly represent the transition function using Dynamic Bayesian Networks (DBNs). First, the outcome of each variable is independent of the outcome of the other variables:

$$P(s' \mid s, a) = \prod_{i=1}^{|X|} P(s'[X_i] \mid s, a),$$

where $s[\cdot]$ denotes the value of a variable X_i (or a set of variables $\Delta \subseteq X$) on the $s \in S$ and $P(x_i \mid s, a)$ is the probability of observing $x_i \in dom(X_i)$ conditioned on (s, a).

Second, the dynamics of a variable $X_i \in X$ for a given action $a \in A$ depend only on its parents, a subset of the variables $\text{Pa}_a(X_i) \subseteq X$. In this way, the probability distribution of each variable X_i can be conditioned only on $\text{Pa}_a(X_i)$: $P(x_i \mid s, a) = P(x_i \mid s[\text{Pa}_a(X_i)], a)$, which yields a compact representation of the transition function:

$$P(s' \mid s, a) = \prod_{i=1}^{|X|} P(s'[X_i] \mid s[\text{Pa}_a(X_i)], a). \quad (1)$$

2.2 Model-based Exploration

In this section we review sample-efficient RL algorithms, focusing on how they exploit the structure of the environment.

The Knows What It Knows (KWIK) framework was developed to help RL agents stay aware of under-explored parts of the environment [Li *et al.*, 2011]. A KWIK learner must only return ϵ-accurate predictions with high probability $1 - \delta$, otherwise it can return "I do not know" ($\perp$). However, a KWIK learner can only return $\perp$ a limited number of times. This way, a KWIK learner can be used by model-based RL algorithms to learn the transition function of a flat MDP or the components of an FMDP.

Sophisticated exploration strategies can reduce the number of interactions necessary to find an optimal policy. R-max is a family of model-based algorithms that quickly build an accurate estimate of the transition function, by incentivizing the agent to visit under-explored parts of the environment. These algorithms use a KWIK-like method to keep track of the set of state-action pairs that are still considered unknown.

The original R-max was proposed for flat representations [Brafman and Tennenholtz, 2002], in this case the set of known state-action pairs is defined as follows:

$$K_m = \{(s,a) \in S \times A \mid n(s,a) \geq m\}, \quad (2)$$

where m is a minimum number of observations to consider the state-action pair (s,a) as known, and $n(s,a)$ is the number of times (s,a) was observed. Note that the R-max algorithm implicitly uses a KWIK algorithm to estimate the transition function when m is defined according to the required precision ϵ and the expected level of confidence $1 - \delta$.

Extensions of this algorithm have been proposed for FMDPs with different assumptions regarding the prior knowledge about the structure of the DBN [Guestrin *et al.*, 2002; Strehl, 2007; Diuk *et al.*, 2009; Chakraborty and Stone, 2011]. In general, these extensions use a subalgorithm $\mathcal{A}_{X_i,a}$ for each variable-action pair $(X_i, a) \in X \times A$ to estimate the distribution of the components of the FMDP. If the subalgorithm is KWIK admissible, given a state s, this subalgorithm must return the probability distribution of X_i if it has a low parametric uncertainty over this distribution, otherwise it must return $\perp$. The main algorithm only consider a state-action pair $(s,a) \in S \times A$ as known, if all the subalgorithms related to the action a consider (s,a) as known:

$$K = \{(s,a) \in S \times A \mid \mathcal{A}_{X_i,a}(s) \neq \perp, \forall X_i \in X\}. \quad (3)$$

For problems where the structure of the problem is known a priori the subalgorithm $\mathcal{A}_{X_i,a}$ only needs to keep track of $\hat{P}(X_i = x_i \mid s[\text{Pa}_a(X_i)], a)$[1] using a maximum likelihood estimate [Guestrin *et al.*, 2002]. Each subalgorithm also has a parameter m_i, that indicates the minimum number of observations to consider the distribution of X_i known. Therefore, given a state s, the subalgorithm defines whether a component is known or not as follows:

$$\mathcal{A}^m_{X_i,a}(s) = \begin{cases} \perp & \text{if } n(s[\text{Pa}_a(X_i)]) < m, \\ \hat{P}(\cdot \mid s[\text{Pa}_a(X_i)]), & \text{otherwise,} \end{cases}$$

where $n(s[\text{Pa}_a(X_i)])$ is the number of times a was executed in states where the parents of X_i were in the same configuration as in s. In the next section, we present algorithms that can be used when the structure of the problem is unknown.

[1] Hereinafter, we omit the action a and the variable X_i whenever they are in the scope of a subalgorithm $\mathcal{A}_{X_i,a}$.

2.3 Structure Learning

When the structure is unknown the subalgorithm $\mathcal{A}_{X_i,a}$ needs to initially search for the set of parents Pa_{a,X_i}. This problem has been extensively studied [Degris *et al.*, 2006; Strehl *et al.*, 2007; Diuk *et al.*, 2009; Chakraborty and Stone, 2011]. Here we present two structure learning algorithms with KWIK guarantees.

Given a maximum in-degree d, which bounds the size of the set of parents $\text{Pa}_a(X_i)$, these algorithms need to choose a candidate from $Comb(X, d)$ that contains the true set of parents, where $Comb(X, d)$ is the collection of subsets of X of size d.

The **Structure Learning (SL) algorithm** [Strehl *et al.*, 2007] relies on the fact that the probability distribution of the variable X_i is independent of non-parents given the true parents. To choose the set of parents the SL algorithm estimates the distribution for each pair of candidates $(\Delta_i, \Delta_j) \in Comb(X, d) \times Comb(X, d)$. Given a state s, this algorithm chooses Δ_i as parents if two conditions are met: (i) it has collected enough samples of the realizations of Δ_i and all the other candidates were in the same configuration as in state s:

$$n(s[\Delta_i \cup \Delta_j]) \geq m, \forall \Delta_j \in Comb(X, d) \setminus \{\Delta_i\}, \quad (4)$$

and (ii) the distribution estimate of all pairs of candidates that include Δ_i are similar, that is, the distribution divergence between different pairs that include Δ_i is smaller than a given ϵ_1:

$$\|\hat{P}(\cdot \mid \Delta_i \cup \Delta_j) - \hat{P}(\cdot \mid \Delta_i \cup \Delta_k)\|_1 \leq \epsilon_1, \forall j, k \neq i. \quad (5)$$

In this way, given a state s, the SL algorithm returns the probability distribution of X_i if it finds a subset of variables Δ_i that satisfies both conditions:

$$\mathcal{A}^m_{X_i,a}(s) = \begin{cases} \perp & \text{if } \nexists \Delta_i \in Comb(X, d) \\ & \text{s.t. } \Delta_i \models (4) \text{ and } \Delta_i \models (5), \\ \hat{P}(\cdot \mid s[\Delta_i \cup \Delta_j]) & \text{where } j \neq i, \text{otherwise.} \end{cases}$$

The ***k*-meteorologists algorithm** [Diuk *et al.*, 2009] makes the same assumptions as the SL algorithm and initializes the set of tracked candidates R with $Comb(X, d)$. However, it relies on a different fact to find the best candidate: the squared error of the distribution function computed according to a candidate containing the true parents is smaller than the one computed without the true parents. Given a transition sample $(s, a, s') \in S \times A \times S$, the squared error of the set of parents $\Delta \subseteq X$ is given by $e = (1 - \hat{P}(s'[X_i] \mid s[\Delta]))^2$. Therefore, this algorithm keeps track of the accumulated squared error for each pair of candidate parents and their number of mismatches $c_{i,j}$. After two candidates have disagreed enough times ($c_{i,j} \geq c$), the k-meteorologists algorithm discards the one with the largest accumulated error. When asked for the distribution of variable X_i for a given state-action pair, the k-meteorologists algorithm returns the average of of the probability distribution of the remaining candidates R, if they are confident in the distribution ($n(s[\Delta_i]) \geq m$) and if they agree on such:

$$\forall \Delta_i, \Delta_j \in R \times R : \|\hat{P}(\cdot \mid s[\Delta_i]) - \hat{P}(\cdot \mid s[\Delta_j])\|_1 < \epsilon_1;$$

otherwise, it returns $\perp$.

Algorithm 1 Policy-based SPIBB (Π_b-SPIBB)

Input: Previous experiences $\mathcal{D}$
Input: Parameters ϵ, δ
Input: Behavior policy π_b
Output: Safe Policy
1: Estimate $\hat{P}(\cdot \mid s, a), \forall (s, a) \in S \times A$
2: $m = \frac{2}{\epsilon^2} \log \frac{|S||A|2^{|S|}}{\delta}$
3: Compute $\mathfrak{B} = \overline{K_m}$ ▷ (2)
4: Compute Π_b ▷ (7)
5: **return** $\arg\max_{\pi \in \Pi_b} V(\pi, \hat{M})$

3 Safe Policy Improvement

Safe RL aims to develop reliable agents to be deployed in real-world applications. In particular, when a behavior policy π_b is already in execution, one must provide guarantees that a new policy π, computed by an RL algorithm, outperforms π_b. We consider a batch RL setting, where given π_b and a set of past interactions $\mathcal{D}$ collected by executing π_b, the RL algorithm needs to compute a new policy to be deployed. Note that in this setting, the agent does not interact with the environment during the learning process. The main concern is to avoid returning policies that have a poor performance.

We call an RL algorithm *safe* if, given $\mathcal{D}$ and π_b, it has high probability $1 - \delta$ of returning a new policy π that is better than π_b:

$$\Pr_{M \sim \mathbb{P}(\cdot \mid \mathcal{D})} (V(\pi, M) \geq V(\pi_b, M) - \zeta) \geq 1 - \delta,$$

where ζ is an admissible error and $\mathbb{P}(\cdot \mid \mathcal{D})$ is the posterior distribution over all possible MDPs given the batch of previous experiences. Unfortunately, finding a policy π that maximizes the expected value $V(\pi, M)$ under the constraints above is intractable if one does not make any assumptions about $\mathbb{P}(\cdot \mid \mathcal{D})$ [Delage and Mannor, 2010].

3.1 The SPIBB Framework

To develop tractable SPI algorithms, Laroche *et al.* [2019] propose the SPI by Baseline Bootstrapping (SPIBB) criterion, where the goal is to compute a policy that is better than the behavior policy on the MDPs whose dynamics are close to the estimated MDP $\Xi(\hat{M})$, that is:

$$\max_{\pi \in \Pi} V(\pi, \hat{M}) \text{ s.t.}$$
$$\forall M' \in \Xi(\hat{M}) : V(\pi, M') \geq V(\pi_b, M') - \zeta. \quad (6)$$

We refer to Laroche *et al.* [2019] and Petrik *et al.* [2016] for more details on how $\Xi(\hat{M})$ is computed such that it contains the true MDP with high probability.

To solve (6), Laroche *et al.* [2019] propose to bootstrap the behavior policy in parts of the environment with high parametric uncertainty. These parts of the environment are represented by a set of state-action pairs $\mathfrak{B} \subseteq S \times A$. The Policy Based SPIBB (Π_b-SPIBB) algorithm solves the estimated MDP while constraining the policies to use the same probability as the behavior policy in every bootstrapped state-action

Algorithm 2 Factored Π_b-SPIBB

Input: Previous experiences $\mathcal{D}$
Input: Parameters ϵ, δ
Input: Behavior policy π_b
Input: DBN Structure Pa
Output: Safe Policy
1: Estimate $\hat{P}(\cdot \mid \text{Pa}_a(X_i), a), \forall (X_i, a) \in X \times A$
2: Compute $\hat{P}(. \mid s, a), \forall (s, a) \in S \times A$ ▷ (1)
3: Initialize $\vec{m}$ with $m_i = \frac{2|X|^2}{\epsilon^2} \log \frac{|\mathcal{Q}|2^{|dom(X_i)|}}{\delta}$
▷ $\mathcal{Q}$ is the number of parameters of the FMDP
4: $\mathfrak{B} = \overline{K_{\vec{m}}}$ ▷ (3)
5: Compute Π_b ▷ (7)
6: **return** $\arg\max_{\pi \in \Pi_b} V(\pi, \hat{M})$

pair $(s, a) \in \mathfrak{B}$. Formally, the constrained set of policies is defined as:

$$\Pi_b = \{\pi \mid \pi(s, a) = \pi_b(s, a) : \forall \pi \in \Pi, \forall (s, a) \in \mathfrak{B}\}, \quad (7)$$

and the safe policy is:

$$\pi = \arg\max_{\pi \in \Pi_b} V(\pi, \hat{M}).$$

Algorithm 1 presents an overview of this method. Note that computing a new policy in this framework is relatively simple and can be done using slightly adapted value iteration or policy iteration algorithms.

3.2 The Factored SPIBB Framework

In previous work [Simão and Spaan, 2019], we show that the Π_b-SPIBB framework can be extended to factored environments. Algorithm 2 shows an overview of the Factored Π_b-SPIBB algorithm. Note that this algorithm takes as input the structure of the DBN, represented by the set of parents of each variable-action pair. The main enhancement of this algorithm is a reduction in the number of samples required to stop bootstrapping a state-action pair.

An important contribution of this method is to overcome the limitation of the SPIBB framework where the probability of taken an action a in state s would always be 0, if that probability was 0 in the behavior policy. In Factored SPIBB, the agent can generalize past experiences to predict the outcome of an action that was never taken.

The Factored Π_b-SPIBB algorithm assumes that the structure of the FMDP is known a priori, a strong assumption that frequently is not satisfied. In the next section we present a method to overcome this limitation.

4 Structure Learning for Safe Policy Improvement

In this section, we propose a more general version of the Factored SPIBB framework for problems where the structure of the problem is unknown.

Structure Learning Π_b-SPIBB keeps track of the distribution of each transition component using a separate subalgorithm $\mathcal{A}_{X_i,a}, \forall (X_i, a) \in X \times A$. The subalgorithms used by this framework can be borrowed from the factored RL literature (Section 2.3). Algorithm 3 presents an overview of the

Algorithm 3 Structure Learning Π_b-SPIBB

Input: Previous experiences $\mathcal{D}$
Input: Parameters ϵ, δ
Input: Behavior policy π_b
Input: Subalgorithms $\mathcal{A}$
Output: Safe Policy
1: **for all** $(X_i, a) \in X \times A$ **do**
2: Initialize $\mathcal{A}_{X_i,a}$ with $\frac{\epsilon}{|X|}$ and $\frac{\delta}{|X||A|}$
3: Present $\{(s, a', s') \in \mathcal{D} \mid a' = a\}$ to $\mathcal{A}_{X_i,a}$
4: **end for**
5: $\mathfrak{B} = \emptyset$
6: **for all** $(s, a) \in S \times A$ **do**
7: **if** $\exists X_i \in X : \mathcal{A}_{X_i,a}(s) = \perp$ **then**
8: $\mathfrak{B} = \mathfrak{B} \cup \{(s, a)\}$
9: $\hat{P}(s' \mid s, a) = 0, \forall s' \in S$
10: **else**
11: $P_i = \mathcal{A}_{X_i,a}(s), \forall X_i \in X$
12: $\hat{P}(s' \mid s, a) = \prod_{i=1}^{|X|} P_i(s'[X_i]), \forall s' \in S$ ▷ (1)
13: **end if**
14: **end for**
15: Compute Π_b according to $\mathfrak{B}$ ▷ (7)
16: **return** $\arg\max_{\pi \in \Pi_b} V(\pi, \hat{M})$

new framework. Different from the original SPIBB algorithm (Algorithm 1), this framework takes as input a class of subalgorithms $\mathcal{A}$ that must be chosen according to the prior knowledge available. Note that if the given subalgorithm knows the underlying structure, this algorithm would be equal to the Factored Π_b-SPIBB algorithm.

First, Algorithm 3 instantiates one subalgorithm for each variable-action pair and presents the relevant transitions from the batch of previous experiences $\mathcal{D}$ to it (lines 1-4). Next, it estimates the transition function and, at the same time, it builds the set of bootstrapped state-action pairs (lines 5-14). Line 8, shows how the set of bootstrapped state-action pairs $\mathfrak{B}$ is constructed. For a given state-action pair (s, a), if one of the subalgorithms related to a returns $\perp$ given s (line 7), then the pair (s, a) is added to $\mathfrak{B}$. Note that for these pairs, at least one of the subalgorithms does not return a distribution, therefore we assume that these state-action pairs are absorbing. Finally, the safe policy is computed in the same way as by the original SPIBB algorithm.

For the theoretical analysis of the proposed algorithm, we first show that the transition function of non-bootstrapped state-action pairs is precise with high probability.

Proposition 1. *When the Structure Learning Π_b-SPIBB algorithm is equipped with a subalgorithm $\mathcal{A}$ that is KWIK-admissible, all non-bootstrapped state-action pairs have a transition error smaller than ϵ with high probability $1 - \delta$:*

$$\Pr(\forall (s, a) \notin \mathfrak{B} : \|P(\cdot \mid s, a) - \hat{P}(\cdot \mid s, a)\|_1 \leq \epsilon) \geq 1 - \delta.$$

Proof. According to Strehl [2007, Corollary 1], if all $|X|$ relevant subalgorithms return a distribution with error smaller than $\frac{\epsilon}{|X|}$ then the error of the estimated transition function is smaller than ϵ. Using a union bound, we conclude that the probability that there is a factor with error larger than $\frac{\epsilon}{|X|}$ is

smaller than $\sum_1^{|X||A|} \frac{\delta}{|X||A|} = \delta$. Given that the subalgorithm used is KWIK admissible, the above assumptions hold, which concludes our proof[2]. □

We can use Proposition 1 to show that the proposed algorithm is safe.

Theorem 1. *(Safe Policy Improvement of the Structure Learning Π_b-SPIBB Algorithm). Let Π_b be the set of policies under the constraint of following π_b in every bootstrapped state-action pair $(s, a) \in \mathfrak{B}$. Then, the policy π_{pol} computed by the Structure Learning Π_b-SPIBB algorithm, is at least a ζ-approximate safe policy improvement over π_b with high probability $1 - \delta$, with*

$$\zeta = \frac{4\epsilon V_{\max}}{(1-\gamma)} - V(\pi_{pol}, \hat{M}) + V(\pi_b, \hat{M}).$$

Proof. We use Proposition 1 (above) to replace Proposition 1 in the proof of Theorem 2 by Laroche *et al.* [2019]. □

In conclusion, the algorithm Structure Learning Π_b-SPIBB is safe when equipped with a KWIK admissible algorithm.

5 Empirical Analysis

We evaluate the Structure Learning Π_b-SPIBB framework combined with the two structure learning algorithms presented before (SL and k-meteorologists) in three domains. The two baselines for comparison are the Factored Π_b-SPIBB algorithm [Simão and Spaan, 2019], that knows the structure of the problem, and the (flat) Π_b-SPIBB algorithm [Laroche *et al.*, 2019], that does not consider this structure. We also consider an algorithm without safety guarantees, which is referred to as Factored Basic RL. This algorithm has access to the structure of the problem and computes a greedy policy according to an estimate of the factored transition function of the problem.

All algorithms use a flat estimate of the transition function and a flat Value Iteration algorithm with a discount factor of 0.99. We assume that the reward function is known in all algorithms. The problems used are: (i) the Taxi domain with a horizon of 200 steps [Dietterich, 1998], (ii) the SysAdmin domain with 9 machines in a bidirectional ring topology and a horizon of 40 steps [Guestrin *et al.*, 2003], and (iii) the Stock-Trading domain with 3 sectors and 2 stocks per sector with a horizon of 40 steps [Strehl *et al.*, 2007]. We follow a setup similar to the experiments by Laroche *et al.* [2019], where the behavior policy π_b is defined as a softmax over the optimal value function. The softmax temperature is set to 2 for the Taxi and Stock-Trading domains and to 3 for the SysAdmin domain.

Each experiment is executed in three steps varying the number of episodes in the batch $\mathcal{D}$ of previous experiences:

(i) create $\mathcal{D}$ by executing the behavior policy,

(ii) present $\mathcal{D}$ and π_b (in the case of the safe algorithms) to each algorithm and compute a new policy, and

(iii) estimate the performance of each new policy by averaging the discounted returns of 1000 trials.

		Taxi	SysAdmin	Stock-Trading
Π_b-SPIBB	m	10.00	100.00	10.00
Factored Π_b-SPIBB	m	20.00	10.00	20.00
Π_b-SPIBB SL	m	20.00	10.00	20.00
	ϵ_1	0.01	0.20	0.30
Π_b-SPIBB	m	10.00	10.00	20.00
k-meteorologists	ϵ_1	0.01	0.00	0.01
	c	2000.00	2000.00	300.00

Table 1: Parameters used by each algorithm

Table 1 reports the parameters used by each algorithm. These values were chosen in order to reduce the number of samples required to improve the policy, while keeping a safe behavior. Figures 1 and 2 present the results. In every plot the x-axis shows the number of trials in the batch collected with the behavior policy. We highlight that the experiments with different batch sizes are independent of each other, that is, the trajectories collected where $|\mathcal{D}| = x$ are not related to the trajectories collected where $|\mathcal{D}| < x$. In both figures, each column shows the results obtained in different domains: Taxi (left), SysAdmin (middle) and Stock-Trading (right).

5.1 Searching for the Best Candidate Structure

Figure 1 shows how the estimated structure improves as the structure learning algorithms receive more data. For the SL algorithm it shows the number of parents missing in the selected candidate: $\sum_{X_i,a \in X \times A} |\text{Pa}_a(X_i) \setminus \Delta_{X_i,a}|$, where $\Delta_{X_i,a}$ is the candidate chosen by subalgorithm $\mathcal{A}_{X_i,a}$. For the k-meteorologists algorithm it shows the average number of missing parents between all remaining candidates:

$$\sum_{X_i,a \in X \times A} \frac{\sum_{\Delta \in R_{X_i,a}} |\text{Pa}_a(X_i) \setminus \Delta|}{|R_{X_i,a}|},$$

where $R_{X_i,a}$ is the set of remaining candidates of $\mathcal{A}_{X_i,a}$.

In the Taxi domain (Figure 1 left) the overall number of missing parents is small, which is expected since this domain has only four variables. We note that the k-meteorologists algorithm needs significantly more samples to discard the candidates that do not contain the true parents. That occurs because the number of mismatches (c) must be large to avoid erroneously discarding a candidate.

In the SysAdmin domain (Figure 1 middle) and in the Stock-Trading domain (Figure 1 right), the structure learning algorithms exhibit a similar behavior. Both algorithms select candidates missing many parents for small batches. However, the k-meteorologists algorithm has a sudden drop in the number of missing parents. This is explained by the fact that all the subalgorithms crossed the minimum number of mismatches when the batch is larger than a certain threshold.

We also note that the SL algorithm does not display a monotonic improvement in these domains. This is because

[2]This proof follows the same principle used by Li *et al.* [2011, p. 413, Problem 9] to show that the output combination algorithm is KWIK admissible. However, to prove that Factored MDPs with unknown structure are KWIK-learnable, Li *et al.* [2011] use a different algorithm that includes the action as one of the input variables.

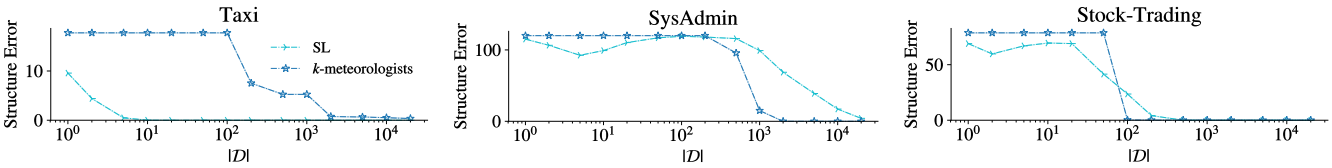

Figure 1: Structure learning error of the SL and k-meteorologists algorithms

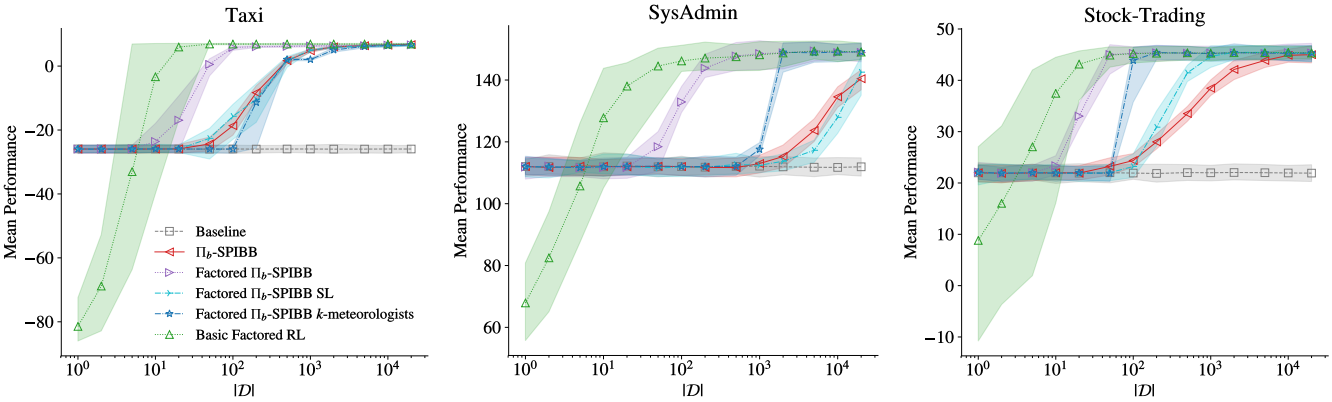

Figure 2: Average performance of the computed policy over 50 repetitions, along with the 1% quantile and 99% quantile (shaded area)

to compute the distribution divergence of two sets of candidates (5), we only consider configurations that have been observed at least once. Without this measure, the SL algorithm is not able to learn the structure of the Taxi domain, where some configurations are never observed. Therefore, in the SysAdmin and Stock-Trading domains, when $|\mathcal{D}|$ is small, the SL algorithm ignores some of the configurations and can improve the estimated structure. However, when $|\mathcal{D}|$ contains more configurations it becomes more conservative again, returning $\perp$ until it gets enough data for all configurations.

5.2 Policy Improvement

Next, we evaluate the performance of the Factored Π_b-SPIBB equipped with structure learning algorithms (Figure 2). We present the average performance of 50 repetitions and to measure the risk of the algorithms the 1%-quantile and 99%-quantile (shaded area).

First we observe that in the Taxi domain (Figure 2 left), the structure learning approaches are slightly more conservative than the flat Π_b-SPIBB algorithm. This is not surprising, since, as pointed out by Strehl *et al.* [2007], a DBN is not the ideal structure to capture the independence between the variables of this domain. Comparing the methods using structure learning algorithms, we see that the Factored Π_b-SPIBB SL algorithm requires less samples to exceed the performance of the behavior policy than the Factored Π_b-SPIBB k-meteorologists algorithm. This is because, as mentioned before, the k-meteorologists algorithm requires a large number of mismatches to discard a candidate, while the SL algorithm can choose the best candidate with fewer samples.

The advantages of using a structure learning algorithm are more prominent in well-factorized environments, where the maximum in-degree d is much smaller than the number of state variables. The experiments with the SysAdmin and Stock-Trading domains illustrate this fact (Figure 2 middle and right). We note that in the Stock-Trading domain, the Factored Π_b-SPIBB algorithm needs around 50 trajectories to find the optimal policy while the Factored Π_b-SPIBB k-meteorologists algorithm needs 200 trajectories and the flat Π_b-SPIBB algorithm needs 10000 trajectories. Their relative performance is similar in the SysAdmin domain. In summary, using the k-meteorologists algorithm, Structure Learning Π_b-SPIBB demonstrates a behavior closer to the Factored Π_b-SPIBB algorithm and manages to find an improved policy with an order of magnitude fewer samples than the flat Π_b-SPIBB algorithm.

6 Conclusions

We presented a Safe Policy Improvement framework for factored environments with unknown structure. Relaxing the assumption that the underlying structure is known a priori makes this method applicable in a wider range of problems. Furthermore, when equipped with an efficient structure learning method, this framework can still exploit the factored structure of the environment and typically requires fewer samples than a flat algorithm to improve the behavior policy.

Studying how to exploit other types of structure such as decision trees and linear dynamics in this setting is a promising line of future work.

Acknowledgments

This research received funding from the Netherlands Organisation for Scientific Research (NWO).

References

[Boutilier *et al.*, 1995] Craig Boutilier, Richard Dearden, and Moisés Goldszmidt. Exploiting Structure in Policy Construction. In *Proc. Int. Joint Conf. on Artificial Intelligence*, pages 1104–1113. Morgan Kaufmann, 1995.

[Brafman and Tennenholtz, 2002] Ronen I. Brafman and Moshe Tennenholtz. R-MAX - A General Polynomial Time Algorithm for Near-Optimal Reinforcement Learning. *Journal of Machine Learning Research*, 3:213–231, 2002.

[Chakraborty and Stone, 2011] Doran Chakraborty and Peter Stone. Structure Learning in Ergodic Factored MDPs without Knowledge of the Transition Function's In-Degree. In *Proc. of International Conference on Machine Learning*, pages 737–744. Omnipress, 2011.

[Degris *et al.*, 2006] Thomas Degris, Olivier Sigaud, and Pierre-Henri Wuillemin. Learning the Structure of Factored Markov Decision Processes in Reinforcement Learning Problems. In *Proc. of International Conference on Machine Learning*, volume 148 of *ACM International Conference Proceeding Series*, pages 257–264. ACM, 2006.

[Delage and Mannor, 2010] Erick Delage and Shie Mannor. Percentile Optimization for Markov Decision Processes with Parameter Uncertainty. *Operations Research*, 58(1):203–213, 2010.

[Dietterich, 1998] Thomas G Dietterich. The MAXQ Method for Hierarchical Reinforcement Learning. In *Proc. of International Conference on Machine Learning*, volume 98, pages 118–126, 1998.

[Diuk *et al.*, 2009] Carlos Diuk, Lihong Li, and Bethany R. Leffler. The Adaptive k-meteorologists Problem and Its Application to Structure Learning and Feature Selection in Reinforcement Learning. In *Proc. of International Conference on Machine Learning*, pages 249–256. ACM, 2009.

[García and Fernández, 2015] Javier García and Fernando Fernández. A Comprehensive Survey on Safe Reinforcement Learning. *Journal of Machine Learning Research*, 16:1437–1480, 2015.

[Guestrin *et al.*, 2002] Carlos Guestrin, Relu Patrascu, and Dale Schuurmans. Algorithm-Directed Exploration for Model-Based Reinforcement Learning in Factored MDPs. In *Proc. of International Conference on Machine Learning*, pages 235–242. Morgan Kaufmann, 2002.

[Guestrin *et al.*, 2003] Carlos Guestrin, Daphne Koller, Ronald Parr, and Shobha Venkataraman. Efficient Solution Algorithms for Factored MDPs. *Journal of Artificial Intelligence Research*, 19:399–468, 2003.

[Hallak *et al.*, 2015] Assaf Hallak, François Schnitzler, Timothy Arthur Mann, and Shie Mannor. Off-policy Model-based Learning under Unknown Factored Dynamics. In *Proc. of International Conference on Machine Learning*, pages 711–719. JMLR.org, 2015.

[Laroche *et al.*, 2019] Romain Laroche, Paul Trichelair, and Rémi Tachet des Combes. Safe Policy Improvement with Baseline Bootstrapping. In *Proc. of International Conference on Machine Learning*, 2019.

[Li *et al.*, 2011] Lihong Li, Michael L. Littman, Thomas J. Walsh, and Alexander L. Strehl. Knows What It Knows: A Framework For Self-Aware Learning. *Machine Learning*, 82(3):399–443, 2011.

[Petrik *et al.*, 2016] Marek Petrik, Mohammad Ghavamzadeh, and Yinlam Chow. Safe Policy Improvement by Minimizing Robust Baseline Regret. In *Advances in Neural Information Processing Systems 29*, pages 2298–2306. Curran Associates, Inc., 2016.

[Puterman, 1994] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1994.

[Simão and Spaan, 2019] Thiago D. Simão and Matthijs T. J. Spaan. Safe Policy Improvement with Baseline Bootstrapping in Factored Environments. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*, 2019.

[Strehl *et al.*, 2007] Alexander L. Strehl, Carlos Diuk, and Michael L. Littman. Efficient Structure Learning in Factored-State MDPs. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*, pages 645–650. AAAI Press, 2007.

[Strehl, 2007] Alexander L Strehl. Model-Based Reinforcement Learning in Factored-State MDPs. In *2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, pages 103–110. IEEE, 2007.

[Thomas *et al.*, 2015] Philip S. Thomas, Georgios Theocharous, and Mohammad Ghavamzadeh. High Confidence Policy Improvement. In *Proc. of International Conference on Machine Learning*, pages 2380–2388. JMLR.org, 2015.