In *the proc of 28th International Conference on Machine Learning (minor typos corrected)*,
Bellevue, Washingtion, USA, 2011.

Structure Learning in Ergodic Factored MDPs without Knowledge of the Transition Function's In-Degree

Doran Chakraborty CHAKRADO@CS.UTEXAS.EDU
Peter Stone PSTONE@CS.UTEXAS.EDU
Department of Computer Science, University of Texas, 1 University Station C0500, Austin, Texas 78712, USA

Abstract

This paper introduces *L*earn *S*tructure and *E*xploit RMax (LSE-RMax), a novel model based structure learning algorithm for ergodic factored-state MDPs. Given a planning horizon that satisfies a condition, LSE-RMax provably guarantees a return very close to the optimal return, with a high certainty, without requiring any prior knowledge of the in-degree of the transition function as input. LSE-RMax is fully implemented with a thorough analysis of its sample complexity. We also present empirical results demonstrating its effectiveness compared to prior approaches to the problem.

1. Introduction

Representing a Markov Decision Process (MDP) (Sutton & Barto, 1998) with a large state space is challenging due to the curse of dimensionality. One popular way of doing so is factoring the state space into discrete factors (a.k.a features), and using formalisms such as the Dynamic Bayesian Network (DBN) (Guestrin et al., 2002) to succinctly represent the state transition dynamics. A DBN representation for the factored transition function can capture the fact that the transition dynamics of a factor is often dependent on only a subset of other factors. Such an MDP is called a factored-state MDP (FMDP).

From a high level, this paper addresses the problem of first learning the factored transition function, and second planning for near optimal behavior, in a FMDP, through efficient exploration and exploitation. In reinforcement learning (RL) parlance, the first problem is often called the Structure Learning Problem (Strehl et al., 2007; Diuk et al., 2009).

Some of the earlier work in RL for FMDPs, are extensions of provably sample efficient RL algorithms for general MDPs. Algorithms such as Factored-E^3 (Kearns & Koller, 1999) and Factored-RMax (Guestrin et al., 2002) achieve near optimal behavior in a polynomial number of samples, but require prior knowledge of the structure of the transition function, in the form of complete DBN structures with unknown conditional probability tables. More recently, there have been approaches proposed that do not assume prior knowledge of the structure of the transition function, and constructs it from experience in the environment (Degris et al., 2006). However, these results are mostly empirical successes in particular domains, with no formal analysis of sample complexity.

To the best of our knowledge, SLF-RMax (Strehl et al., 2007) is the first algorithm to solve the Structure Learning Problem with a formal guarantee on sample complexity. More recently, (Diuk et al., 2009) proposed Met-RMax, which improves upon SLF-RMax with a better sample complexity guarantee. However, akin to SLF-RMax, Met-RMax requires as input the in-degree (K) of the transition function's DBN structure. In this work, we are interested in scenarios where prior knowledge of K is unavailable. To that end, we propose the first Structure Learning algorithm for ergodic FMDPs, called Learn Structure and Exploit with RMax (LSE-RMax), that given a planning horizon that satisfies a condition, provably guarantees a return close to the optimal return, without requiring knowledge of K as input. We explicitly lay down the condition that the planning horizon needs to satisfy for the above claim to hold. Our analysis focuses on time averaged return.

We propose two variants of LSE-RMax, for two versions of the problem.
1. The first deals with a specific case where factors can be arranged in a sequence with the relevant fac-

Appearing in *Proceedings of the 28th International Conference on Machine Learning*, Bellevue, WA, USA, 2011.

tors (which determine the transition function) preceding the irrelevant ones in the sequence. For example, the probability of a machine running on a particular time step in a network may depend on the states of its K closest neighbors from the past step. Though this fact is common knowledge, K may be unknown. The sequence of factors is then an incremental ordering of the states of all neighbors based on adjacency, with the immediate neighbors heading the sequence. We call problems of such a nature Sequential Structure Learning problems. We believe many real world problems can be mapped to problems of such nature, and hence can leverage from our solution.
2. The second deals with the general case where such an ordering is unknown. Problems of such nature constitute the more general Structure Learning problem.

LSE-RMax is fully implemented, and we present a thorough analysis of its sample complexity. We also present empirical results from two domains demonstrating LSE-RMax's superiority over Met-RMax.

The remainder of the paper is organized as follows. Section 2 presents the background necessary to understand our work, Sections 3– 5 present our algorithms, Section 6 presents empirical results on benchmark domains from literature, and Section 7 concludes.

2. Background and Concepts

This section introduces the background, and concepts necessary to understand our work.

A finite *MDP*, is given by a tuple $\{S, A, P, R\}$ where S is the finite set of states, A is the finite set of actions available to a learner, $P : S \times A \times S' \mapsto [0, 1]$ is the transition function, and $R : S \times A \mapsto [0 : 1]$ is the bounded reward function. A *factored-state MDP* (FMDP) is a finite MDP where each state consists of n discrete factors. Formally S consists of a tuple $<X_1, \ldots, X_i, \ldots X_n>$ where each X_i assumes a value in the discrete range $\{0, \ldots, d\}$. Similarly to prior work on structure learning in FMDPs (Degris et al., 2006; Diuk et al., 2009), we assume i) that the reward function is known, and ii) that the transition function satisfies the independence criterion: $P(s_t, a_t, s_{t+1}) = \prod_{i=1}^{n} P_{i,a_t}(s_t, s_{t+1}(i))$ where $P_{i,a_t}(s_t, s_{t+1}(i))$ represents the probability of transitioning to $s_{t+1}(i)$ when action a_t is taken in state s_t. $s_{t+1}(i)$ denotes the value of factor X_i in state s_{t+1}.

Each $P_{i,a}$ has its own DBN structure, and may depend only on certain factors of the state. For ease of presentation, we assume that the *in-degree* (number of parent factors in the DBN representation) for every $P_{i,a}$ is K. If such is not the case, then K can be seen as the maximum of all such in-degrees for all $P_{i,a}$'s. Let $Par_{i,a}$ be the parent factors for $P_{i,a}$.

A *policy* π is a strategy for choosing actions from a state in a FMDP. A FMDP combined with a policy π yields a Markov process on the states. We say π is *ergodic* if the induced Markov process is ergodic (all states are reachable from any state). To simplify the analysis, and keeping in line with prior work that claim formal learning time guarantees for average reward RL (Kearns & Singh, 2002; Brafman & Tennenholtz, 2003), we focus on FMDPs for which every policy is ergodic, the so called *ergodic* FMDPs. In an ergodic FMDP, the stationary distribution of any policy is independent of the start state.

Let p be a T step path in the FMDP starting from state s. Also let $\mathbf{Pr}^\pi[p]$ and $U(p)$ be the probability of that path while executing π, and the sum of rewards accrued along that path respectively. Then the *T-step expected time averaged return* from state s while executing π is given by $U_T^\pi(s) = \frac{1}{T}\sum_p \mathbf{Pr}^\pi[p]U(p)$. The *asymptotic average return* from s while executing π is given by $U^\pi(s) = lim_{T\to\infty} U_T^\pi(s)$. For ergodic FMDPs, $U^\pi(s)$ is independent of s, and can be denoted as just U^π (Kearns & Singh, 2002). The optimal policy, π^*, is the policy that maximizes U^π, yielding return U^*. Henceforth, we refer to the time averaged return of a policy as simply its *return*.

Since this research seeks guarantees on the return of a learning algorithm after a finite number of steps, we need to take into account some notion of the mixing time of the policies in the FMDP. For an $0 < \epsilon < 1$, the *ϵ-return mixing time* of π is the smallest T s.t. $\forall T' \geq T$ and $\forall s$, $|U_{T'}^\pi(s) - U^\pi| \leq \epsilon$. In general to compete with π^* and achieve a return close to U^*, it is essential to have knowledge of the ϵ-return mixing time $T(\epsilon)$ of π^* (Kearns & Singh, 2002). Then given a $T(\epsilon)$ as input, we would like our learning algorithm to provably achieve a return sufficiently close to U^*, with a high certainty, in a reasonable sample complexity.

Trivially, the problem can be solved by assuming that every $P_{i,a}$ is dependent on all n factors, and running Factored-RMax (Guestrin et al., 2002). The downside of this approach is that it requires an order of $\approx \tilde{O}(d^n)$ data samples. Needless to say, the above approach does not scale for large values of n. Our objective is to solve the problem in a sample complexity which is polynomial in d^K. Without assuming prior knowledge of K, we propose LSE-RMax that achieves it, but only for certain restrictive choices of $T(\epsilon)$. To formally specify this restriction on the choice of the $T(\epsilon)$, we introduce a bit more notation.

First, a *model* $\hat{P}$ of the transition function P is comprised of sub-models for each individual $P_{i,a}$, denoted by $\hat{P}_{i,a}$. Each $\hat{P}_{i,a}$ is defined over a set of probable parent factors $\hat{Par}_{i,a}$, and specifies some distribution over $\{0,\ldots,d\}$, for each possible instantiation of $\hat{Par}_{i,a}$. Note, a model can be any function of factor instantiations to probability distributions over $\{0,\ldots,d\}$. Second, when we say *planning* over T steps based on a model $\hat{P}$, we mean executing a T-step policy in the true FMDP, which is optimal w.r.t. the FMDP induced by $\hat{P}$. The T-step policy is computed by running a planning algorithm, such as Value Iteration (Sutton & Barto, 1998) in that fictitious FMDP.

Then, LSE-RMax can provably guarantee a return sufficiently close to U^*, with a high certainty, in the desired sample complexity on the order of $\approx \tilde{O}(d^{2K})$, if the $T(\epsilon)$ provided as input satisfies the condition:

Condition 1. *Planning over $T(\epsilon)$ steps based on a transition function $\hat{P}$ which comprises an insufficient set of parent factors for some $P_{i,a}$ (i.e., $Par_{i,a} \not\subseteq \hat{Par}_{i,a}$), from any start state, always results in an expected return $< U^* - 3\epsilon$ in the true FMDP.*

Another way of interpreting Condition 1 is if we achieve a $T(\epsilon)$ step expected return $\geq U^* - 3\epsilon$ in the true FMDP from any start state, then we must be planning based on a $\hat{P}$ that is based on the correct set of parent factors for every $P_{i,a}$. To facilitate our theoretical claims, we assume that such a $T(\epsilon)$ exists for our FMDP, and we are aware of it. Note that the ϵ in Condition 1 is the corresponding ϵ value for which $T(\epsilon)$ is the ϵ-return mixing time.

In practice when we implement RMax type of algorithms, we pick an (m, T) pair. m is the number of visits required to each unknown parameter, and is a fixed value in most implementations. T is the planning horizon (same as $T(\epsilon)$). In the context of LSE-RMax, our goal is then to guess a big enough T such that achieving a T-step near optimal return from any start state requires knowing all of the parent factors. We believe for most of the FMDPs that we may encounter in practice, it may not be hard to guess a big enough T that satisfies (or closely satisfies) our objective. Even in cases where our chosen T fails to strictly satisfy Condition 1, we may still get good results as long as it is big enough to ensure that achieving a T-step near optimal return from any start state requires knowing most of the crucial parent factors. We present further empirical evidence of this in Section 6.

Our assumption of knowing a $T(\epsilon)$ that satisfies Condition 1 can be seen as an alternative to knowing K. However if the choice is between guessing a big enough $T(\epsilon)$ that satisfies Condition 1, or a big K that serves as a conservative upper bound if the in-degree, we are better off guessing the former because the sample complexity grows exponentially in K ($\approx \tilde{O}(d^K)$), while only polynomially in $T(\epsilon)/\epsilon$. Thus theoretically we have more latitude to conservatively guess a big $T(\epsilon)$ (hence a small ϵ), than a big K.

Since the structure of each $P_{i,a}$ is unknown a priori, the crux of our problem is figuring out how to bias exploration so that the structure of each $P_{i,a}$ can be learned quickly. To solve this general *Structure Learning problem* (SLP), we begin by solving a simpler learning problem which makes a more restrictive assumption about the structure of the factored transition function. Specifically, we assume that for each $P_{i,a}$, we have an ordering of the parent factors, and the first K factors in the ordering determines $P_{i,a}$, K being unknown. We call a problem of this nature the *Sequential Structure Learning Problem* (SSLP). A solution to SSLP holds the key to solving SLP. We begin by introducing our solution to SSLP.

3. Sequential Structure Learning Problem

For ease of presentation, in this section and Section 4, we denote $P_{i,a}$, s_t and $s_{t+1}(i)$ as F, x_t and y_t respectively. x_t can be seen as the input to F, and $y_t \sim F(x_t)$ as the observed output. We denote the i'th factor value of x as $x[i]$. Then for SSLP, the factor values $\{x[0],\ldots,x[K]\}$ completely determine y, with K being unknown. The learning problem is then to learn a good approximation of F, from the observed (x_t, y_t) pairs. We call our learning algorithm for SSLP the *Sequential Structure Learning Algorithm* (SSLA).

3.1. Sequential Structure Learning Algorithm

We begin by introducing the concept of a model w.r.t. SSLA. SSLA maintains a model for every K, ranging from 0 to n, and chooses the best amongst them on every time step. A model of *size* k only consults the first k factor value of the input x. Internally, for every possible value $\mathbf{b_k}$ of the first k factors, it maintains a value $M_k(\mathbf{b_k})$ which is the maximum likelihood distribution of the observed y's given that the first k factor value of the input x is $\mathbf{b_k}$. Whenever a new input that starts with $\mathbf{b_k}$ is observed, we say a *visit* to $\mathbf{b_k}$ has occurred. Each maximum likelihood distribution is based on all the visits to $\mathbf{b_k}$ until time $t-1$.

Each model of size k makes a prediction $\hat{F}_k(x)$ for an input x. The prediction is the empirical distribution captured for the first k factor value of x, if the distribution has been computed from a sufficient number of samples. Otherwise, $\perp$ is returned ("I don't know").

Thus, if the first k factor value of x is $\mathbf{b_k}$, then

$$\hat{F}_k(x) = \begin{cases} M_k(\mathbf{b_k}) & \text{if } c(\mathbf{b_k}) \geq m_k \\ \perp & \text{otherwise} \end{cases} \quad (1)$$

where $c(\mathbf{b_k})$ is the number of times $\mathbf{b_k}$ has been visited, and m_k is a system defined parameter unique to each k. We discuss later how m_k is chosen for each k. The role of $\perp$ will be clear when we specify our main algorithm in Section 5.

Alg. 1 gives an outline of SSLA. On each step it takes as input the current input instance x_t. It calls Find-Best-Model-Ssla to get the current best estimate model of F, denoted as $\hat{F}$. It then makes a prediction $\hat{F}(x_t)$. The environment subsequently reveals y_t. It then updates all its internal data structures.

The main objective of SSLA is to learn to predict $\hat{F}(x_t)$ s.t. it is a close approximation of $F(x_t)$. In that regard, we say $\hat{F}(x)$ is an ϵ-approx of $F(x)$ when $\hat{F}(x) \neq \perp$, and $||F(x) - \hat{F}(x)||_\infty \leq \epsilon$. In order to achieve its objective, SSLA relies on Find-Best-Model-Ssla (Alg. 2) to keep returning a near accurate model. Before we dive into the technical details of Alg. 2, we introduce the concept of Δ_k.

Δ_k is the maximum difference in prediction between consecutive models of size k and $k+1$. For any $\mathbf{b_k}$, we define a set $Aug(\mathbf{b_k})$ as the set of all $k+1$ factor values which have $\mathbf{b_k}$ as their first k factor value. Then,

$$\Delta_k = max_{\mathbf{b_k}, \mathbf{b_{k+1}} \in Aug(\mathbf{b_k}))} ||M_k(\mathbf{b_k}) - M_{k+1}(\mathbf{b_{k+1}})||_\infty$$

s.t. $c(\mathbf{b_{k+1}}) \geq m_{k+1}$. We will choose m_k's such that $c(\mathbf{b_{k+1}}) \geq m_{k+1}$ will always imply $c(\mathbf{b_k}) \geq m_k$. If there exists no such $\mathbf{b_{k+1}}$, then by default $\Delta_k = -1$.

Intuitively, all models of size $\geq K$ can learn F accurately (as they consist of all of the relevant factors), with the bigger models requiring more samples to do so. On the other hand, models of size $< K$ cannot fully represent F without losing information. From a high level, Alg. 2 chooses $\hat{F}$ by comparing models of increasing size to determine at which point there is no larger model that is more predictive of F. The outline of Alg. 2 is then as follows.

1. On every time step, $\forall 0 \leq k < n$, compute Δ_k and σ_k (line 1). We skip the details on how the σ_k's are computed until Section 3.2. Intuitively, the σ_k's are generated in a fashion such that if $k \geq K$, then w.h.p. [1] they assume a value greater than their corresponding Δ_k. Alg. 2 then searches for that smallest value of k such that all the subsequent Δ_k's are less than their corresponding σ_k's, and concludes that this k is the true value of K. More formally the σ_k's are computed

[1] with a high probability

Algorithm 1: SSLA

input : x_t
$\hat{F} \leftarrow$ Find-Best-Model-Ssla()
predict $\hat{F}(x_t)$
Observe y_t, update all models and c values

Algorithm 2: Find-Best-Model-Ssla

$\hat{F} \leftarrow F_n$
1 for all $0 \leq k < n$, compute Δ_k and σ_k
2 **for** $0 \leq k < n$ **do**
3     $flag \leftarrow$ true
4   **for** $k \leq k' < n$ **do**
5   **if** $\Delta_{k'} \geq \sigma_{k'}$ **then**
6             $flag \leftarrow$ false
7   break
8   **if** $flag$ **then**
9         $\hat{F} \leftarrow F_k$
10   break
11
12 return $\hat{F}$

such that the following condition is satisfied:

$$\forall K \leq k < n\text{: } \mathbf{Pr}[\Delta_k < \sigma_k] \geq 1 - \rho \quad (2)$$

where ρ is a very small probability, and a constant. In other words, even without the knowledge of K, we compute the σ_k's such that the difference between two consecutive models of size $\geq K$ is less than σ_k, w.h.p of at least $1 - \rho$. Note, although we compute a σ_k for every $0 \leq k < n$, Eqn. 2 only holds for $K \leq k < n$;
2. Then, iterate over values of k from 0 to $n-1$ and choose the minimum k s.t. $\forall k \leq k' < n$, the condition $\Delta_{k'} < \sigma_{k'}$ is satisfied (lines 2 - 12). Return F_k as $\hat{F}$;

We now state the sufficient condition on the exploration required that would ensure that $\hat{F}(x_t)$ is an ϵ-approx of $F(x_t)$, w.h.p.

Lemma 3.1. *Given $0 < \epsilon, \delta < 1$, once a $\mathbf{b_K}$ is visited $O(\frac{K^2}{\epsilon^2} log(\frac{nd^K}{\delta}))$ times, then $\forall x$ whose first K factor value is $\mathbf{b_K}$, $\hat{F}(x)$ is an ϵ-approx of $F(x)$, w.h.p of at least $1 - \delta$.*

Proof Sketch: Let at any t, the probability with which Alg. 2 selects a model of size $< K$ be p. If all sub-optimal models of size $< K$ are rejected, then it selects $\hat{F}_K$ with probability at least $1-(n-K)\rho$ (from Eqn. 2, and using Union bound). Therefore, the probability with which it selects a model $\leq K$ is at least $p + (1-p)(1-(n-K)\rho) \geq 1 - n\rho$. Models with size $> K$ are selected with a low probability of at most $n\rho$, and hence are ignored. This is exactly in line with our goal: find the shortest most descriptive model.

If Alg. 2 selects $\hat{F}_K$ as $\hat{F}$, then we have the best model. If it selects a shorter model $\hat{F}_k$ as $\hat{F}$, then with a probability of failure at most $n\rho$, we have a model which approximates $\hat{F}_K(b_K)$ with an error of at most $\sum_{k\leq k'<K}\Delta_{k'} \leq \sum_{k\leq k'<K}\sigma_{k'}$, for a $\mathbf{b_K}$. This follows directly from the definition of Δ_k, and line 5 of the Alg. 2. The latter ensures that the following is true: $\Delta_k < \sigma_k, \Delta_{k+1} < \sigma_{k+1}, \ldots, \Delta_{K-1} < \sigma_{K-1}$. Furthermore, from Hoeffding's bound, it follows that if a $\mathbf{b_K}$ is visited $O(1/\psi^2 log(1/\rho))$ times, then $M_K(\mathbf{b_K})$ is a ψ-approx of $F(\mathbf{b_K})$, with probability of failure at most ρ. Revisit Eqn. 1 for a recap on how $\hat{F}_K$ relates to M_K.

Combining the above two observed facts and applying Union bound, it follows that once a $\mathbf{b_K}$ is visited $m_K \geq O(1/\psi^2 log(1/\rho)))$ times, then w.h.p of at least $1-(n+1)\rho$, for a x_t whose first K factor value is $\mathbf{b_K}$, $\hat{F}(x_t)$ is an $(\sum_{k\leq k'<K}\sigma_{k'}+\psi)$-approx of $F(x_t)$. Thus all we need to do is choose m_K such that $\sum_{k\leq k'<K}\sigma_{k'}+\psi$ is bounded by ϵ. It can be shown that if $m_K = O(K^2/\epsilon^2 log(d^K/\rho))$, then the above holds for all possible $\mathbf{b_K}$'s. Then by choosing $\rho = \delta/(n+1)$, we complete the proof. □

Henceforth, we denote the visits term in Lemma 3.1 for any arbitrary k as $V(k)$, i.e. $V(k) = O(\frac{k^2}{\epsilon^2} log(\frac{nd^k}{\delta}))$. We set m_k s.t. a model of size k stops predicting $\perp$ for a $\mathbf{b_k}$ once it is visited $V(k)$ times, i.e., $m_k = V(k)$. We defer a discussion on how we bias our action selection mechanism to ensure that Lemma 3.1 is satisfied for each $P_{i,a}$ and all its K parent factor values until Section 5, where we specify our full-fledged algorithm for performing structure learning in FMDPs.

3.2. Computation of σ_k

We now specify how the σ_k's are computed. For each k, the goal is to select a value for σ_k s.t. Eqn. 2 is satisfied. If $\Delta_k = -1$, then we choose $\sigma_k = 1$, and the condition is trivially satisfied. Hence the rest of the derivation will focus on the case when $\Delta_k \neq -1$

In the computation of Δ_k, SSLA chooses a specific $\mathbf{b_k}$, a $\mathbf{b_{k+1}} \in Aug(\mathbf{b_k})$ and a factor value $j \in \{0,\ldots,d\}$ for which the models M_k and M_{k+1} differ maximally on that particular time step. Let $M_k(\mathbf{b_k}, j)$ be the probability value assigned to j by $M_k(\mathbf{b_k})$. Without loss of generality, assume $M_k(\mathbf{b_k}, j) \geq M_{k+1}(\mathbf{b_{k+1}}, j)$. Then $\Delta_k < \sigma_k$ implies satisfying $M_k(\mathbf{b_k}, j) - M_{k+1}(\mathbf{b_{k+1}}, j) < \sigma_k$. For the range $K \leq k < n$, we can rewrite it as,

$$\begin{aligned}(M_k(\mathbf{b_k}, j) - E(M_k(\mathbf{b_k}, j))) +\\ (E(M_{k+1}(\mathbf{b_{k+1}}, j)) - M_{k+1}(\mathbf{b_{k+1}}, j)) < \sigma_k\end{aligned} \quad (3)$$

This step follows from the fact that the first K factors completely determine F: $E(M_k(\mathbf{b_k}, j)) = E(M_{k+1}(\mathbf{b_{k+1}}, j))$. One way to satisfy Inequality 3 is by ensuring that $M_k(\mathbf{b_k}, j) - E(M_k(\mathbf{b_k}, j)) < \sigma 1$, and $E(M_{k+1}(\mathbf{b_{k+1}}, j)) - M_{k+1}(\mathbf{b_{k+1}}, j)) < \sigma 2$, and subsequently setting $\sigma_k = \sigma 1 + \sigma 2$.

Observing that $\mathbf{b_k}$, and $\mathbf{b_{k+1}}$ are random variables for which M_k and M_{k+1} differ maximally, it follows that $\mathbf{Pr}(M_k(\mathbf{b_k}, j) - E(M_k(\mathbf{b_k}, j)) \geq \sigma 1) \leq \frac{\rho}{2}$, and $\mathbf{Pr}(E(M_{k+1}(\mathbf{b_{k+1}}, j)) - M_{k+1}(\mathbf{b_{k+1}}, j) \geq \sigma 2) \leq \frac{\rho}{2}$, is always satisfied if we choose $\sigma 1 = \sqrt{1/2c(\mathbf{b_k})log(2d^k/\rho)}$, and $\sigma 2 = \sqrt{1/2c(\mathbf{b_{k+1}})log(2d^{k+1}/\rho)}$. Thus, by setting $\sigma_k = \sigma 1 + \sigma 2$, we have $\mathbf{Pr}(\Delta_k < \sigma_k) > 1-\rho$.

This concludes our specification of SSLP. Next, we build on SSLA to propose our algorithm for SLP.

4. Structure Learning Problem

Unlike SSLP, now we do not assume any prior information about the ordering of factors. Clearly, a very straightforward extension of SSLA can solve the problem: arrange all factors in some order, and execute SSLA. The downside of this approach is that we now require sufficient visits to all of the K' factor values, where K' is the smallest value $\in [0, n]$ that spans over all parent factors. In many domains it is possible to guess a decent ordering of factors, and for such cases the above approach provides an efficient solution. However, in the worst case the last factor in the sequence may be a parent factor, and in that case SSLA can only solve the problem by requiring sufficient visits to all possible n factor values. Our goal in this section is to do better sample complexity wise. To this end we introduce our *Structure Learning Algorithm* (SLA).

Structure Learning Algorithm

SLA shares the same algorithmic structure as SSLA (Alg. 1) except it makes a call to Find-Best-Model-Sla. In spirit similar to SSLA, the main objective of SLA is to eventually start predicting $\hat{F}(x_t)$ which is an ϵ-approx of $F(x_t)$ $\forall x_t$, w.h.p., by observing as few online samples as possible. In contrast to SSLP, since an ordering of the factors is unavailable a priori, Find-Best-Model-Sla maintains a model for every possible combination of factors. From a high level, Find-Best-Model-Sla chooses $\hat{F}$ by searching for the shortest model s.t. no bigger model consisting of a superset of its factors is more predictive.

Let $\mathcal{P}$ be the power set of the factor set $\{X_1, \ldots, X_n\}$, and $\mathcal{P}_k \subseteq \mathcal{P}$ be all the subets of $\mathcal{P}$ with cardinality k. The steps employed by Find-Best-Model-Sla (Alg. 3) are then as follows. Iterate over $0 \leq k < n$ and for every k generate $\mathcal{P}_k$ (Step 2). Iterate over every element $\wp_k$ from this set, i.e., $\wp_k$ is a set of k

factors (Step 3). Let the model comprising the factors in $\wp_k$ be $\hat{F}_{\wp_k}$. Now generate all possible sequences of size $k+1$ to n, where the first k factors in the sequence are the factors from $\wp_k$ (Steps 5-9). Then run FIND-BEST-MODEL-SSLA on each of these sequences (Step 10). If the $\hat{F}$ returned by FIND-BEST-MODEL-SSLA for all of these sequences is $\hat{F}_{\wp_k}$, then we are sure w.h.p, that from the samples we have seen so far, $\hat{F}_{\wp_k}$ is the best model (Steps 11-17). If not, keep repeating the process until a particular $\hat{F}_{\wp_k}$ satisfies the condition. In the worst case, return the model comprising all factors (Step 18).

We claim that if all combinations of $2K$ factor values in a x_t are visited $nV(2K)$ times, then the $\hat{F}(x_t)$ predicted by SLA is an ϵ-approx of $F(x_t)$, w.h.p of at least $1-\delta$. Trivially, if $2K \geq n$, then all n factor values need to be visited $nV(n)$ times. Hence the rest of the analysis focuses on the scenario when $2K < n$.

To understand the claim, consider the following example. Let the factor set be $\{X_1, X_2, X_3, X_4, X_5\}$, and let each X_i assume binary values. Also let $x_t = \{X_1 = 0, X_2 = 1, X_3 = 0, X_4 = 1, X_5 = 1\}$, and $K = 2$. Then all possible combinations of 4 factor values in x_t are $\{X_1 = 0, X_2 = 1, X_3 = 0, X_4 = 1\}$, $\{X_1 = 0, X_2 = 1, X_3 = 0, X_5 = 1\}$, $\{X_2 = 1, X_3 = 0, X_4 = 1, X_5 = 1\}$, and so on (in total 5). Then we require all of these factor values to be visited $5V(4)$ times. The formal proof of the claim is similar to that for SSLP. We instead give an intuitive justification for it by way of the above example.

Assume that F depends on parent factors $\{X_1, X_2\}$. Let $\hat{F}_{X1,X2}$ be the model based on them. First, we show that if the choice boils down to selecting whether $\hat{F}_{X1X2}$ is the best model or not, Alg. 3 always selects $\hat{F}_{X1X2}$ with some probability. Alg. 3 then checks for all possible sequences that start with $\{X_1, X_2\}$. Step 10 returns $\hat{F}_{X1X2}$, w.h.p of at least $1-\delta$ for each such sequence (following reasoning similar to SSLP, i.e., by our choice of σ's). Since there can be at most $\sum_{k=1}^{n-2}\binom{n}{k} < 2^n$ such sequences, step 10 returns $\hat{F}_{X1X2}$, with a probability of at least $1-2^n\delta$ (by Union bound). Thus a model of size > 2 is only selected with a probability of at most $2^n\delta$.

If Alg. 3 selects $\hat{F}_{X1X2}$, then we have the best model. Suppose it selects a model of size ≤ 2 other than $\hat{F}_{X1X2}$. Let it be $\hat{F}_{X3X4}$. This means that at some point Alg. 3 must have executed SSLA on the sequence $X_3X_4X_1X_2$ (or $X_3X_4X_2X_1$), and Step 10 returned $\hat{F}_{X_3X_4}$. However if the factor value $\{X_1 = 0, X_2 = 1, X_3 = 0, X_4 = 1\}$ is visited $V(4)$ times, then from Lemma 3.1, we know that $\hat{F}_{X3X4}(x_t)$ is an ϵ-approx of $F(x_t)$, w.h.p of at least $1-\delta$.

Algorithm 3: FIND-BEST-MODEL-SLA

1 **for** $0 \leq k < n$ **do**
2   Generate $\mathcal{P}_k$
3   **for** *every* $\wp_k \in \mathcal{P}_k$ **do**
4     $flag \leftarrow true$
5   **for** $1 \leq k' <= n-k$ **do**
6   Generate $\mathcal{P}_{k'}$
7   **for** *every* $\wp_{k'} \in \mathcal{P}_{k'}$ **do**
8   **if** $\wp_{k'} \cap \wp_k \neq \phi$ **then**
          continue
9   *seq* $\leftarrow$ a sequence starting with $\wp_k$ followed by $\wp_{k'}$
10         $\hat{F} \leftarrow$ execute FIND-BEST-MODEL-SSLA on *seq*
11   **if** $\hat{F} \neq \hat{F}_{\wp_k}$ **then**
12           $flag \leftarrow false$
13   break
14   **if** $!flag$ **then**
15   break
16   **if** $flag$ **then**
17   return $\hat{F}_{\wp_k}$
18 return model comprising of all factors

So we are sure that the model returned by Alg. 3 has to be of size ≤ 2, w.h.p of at least $1-2^n\delta$, but unsure about the exact model. Hence, to be on the safe side we require all combinations of 4 ($2K = 4$ here) factor values in a x_t to be visited $V(4)$ times, s.t. $\hat{F}(x_t)$ is an ϵ-approx of $F(x_t)$, w.h.p of at least $1-2^n\delta$. Then reassigning $\delta \leftarrow \delta/2^n$ brings us to our main theoretical result concerning SLA.

Lemma 4.1. *Given $0 < \epsilon, \delta < 1$, if all the combinations of $2K$ factor values in a x_t are visited $O(\frac{(2K)^2}{\epsilon^2} log(\frac{nd^{2K}2^n}{\delta})) < nV(2K)$ times, then $\hat{F}(x_t)$ is an ϵ-approx of $F(x_t)$, w.h.p of at least $1-\delta$.*

For SLA, we set m_k s.t. a model of size k stops predicting $\perp$ for a k factor value once that is visited $nV(k)$ times, i.e., $m_k = nV(k)$. Again we defer a discussion on how we bias our action selection mechanism to ensure that Lemma 4.1 is satisfied for each $P_{i,a}$, and the $2K$ factor values associated with it until Section 5, where we introduce the full blown LSE-RMax.

5. LSE-RMax

We now introduce Learn Structure and Exploit with RMax (LSE-RMax), which is modeled closely on RMax (Brafman & Tennenholtz, 2003). To learn each factored transition function $P_{i,a}$ individually, LSE-RMax uses a separate instance of SLA (or SSLA depending on the type of the problem), denoted as $\mathcal{A}_{i,a}$. For both SSLA and SLA, we set $\epsilon \leftarrow \frac{\epsilon}{nT(\epsilon)}$ and $\delta \leftarrow \frac{\delta}{n|A|}$. Our analysis focuses solely on LSE-RMax with SLA. LSE-RMax with SSLA works analogously.

LSE-RMax runs in phases, where a phase lasts for at most $T(\epsilon)$ time steps. The phases are of two types: regular and exploratory. On each *regular* phase LSE-RMax computes a $T(\epsilon)$ step policy, and executes it. The Bellman equations (Sutton & Barto, 1998) behind the policy computation are as follows,

$$Q_t(s,a) = \begin{cases} T(\epsilon) \text{ if } \exists i, \text{ s.t, } \hat{P}_{i,a}(s) = \perp, \text{ o.w.,} \\ R(s,a) + \sum_{s'} \prod_i \hat{P}_{i,a}(s, s'(i)) max_{a'} Q_{t-1}(s', a') \end{cases}$$

The $\hat{P}_{i,a}$'s are predicted by their corresponding $\mathcal{A}_{i,a}$'s. For a state action pair (s,a), if any $\hat{P}_{i,a}(s) = \perp$, it means that LSE-RMax has no way of estimating $Q_t(s,a)$. To facilitate exploration to these state action pairs, LSE-RMax gives them an imaginary bonus of $T(\epsilon)$, the maximum total reward achievable in $T(\epsilon)$ steps. For all other pairs, LSE-RMax performs the conventional Bellman backup.

If we just keep running regular phases, then there is a possibility that LSE-RMax may converge to exploiting a sub-optimal model of the transition function because of insufficient exploration. In order to avoid that, LSE-RMax deviates for a single phase, after every ϕ regular phases. We call this phase an *exploratory* phase. The objective of this phase is to facilitate the exploration needed to satisfy the condition in Lemma 4.1 for each individual $P_{i,a}$. In this phase, LSE-RMax chooses a k randomly from 0 to n. The $T(\epsilon)$ step policy computation, which is similar to the one for a regular phase apart from one key difference, is as follows: For all (s,a) pairs that contain a k factor value $\mathbf{b_k}$ in s for which action a has not been taken $m_k = nV(k)$ times, give them the imaginary bonus. For every other (s,a) pair, use the $\hat{P}_{i,a}(s)$'s, predicted by the corresponding $\mathcal{A}_{i,a}$'s, to perform the Bellman back up.

The analysis that leads to our main theoretical result for LSE-RMax is similar to that of RMax, and we skip it for space constraints. Instead we highlight the key parts where it differs from a standard RMax analysis.

From the Implicit Explore and Exploit Lemma of RMax (Lemma 6 of (Brafman & Tennenholtz, 2003)), the policy followed by RMax on every $T(\epsilon)$ step phase either attains an expected return $\geq U^* - 3\epsilon$ in the true FMDP, or explores with probability $\geq \epsilon$. Now in expectation, LSE-RMax chooses $2K$ once in every $n+1$ such exploratory phases. Since there are only finite number of explorations to perform, then at some point LSE-RMax must execute such a phase where it chooses $2K$ as the random value from $[0, n]$, and achieves an expected return $\geq U^* - 3\epsilon$. This follows from the reasoning that if RMax is exploring with probability $< \epsilon$, then it must be confining itself to "known" state action pairs (state action pairs for which it believes it has a near accurate model). Now, from Lemma 4.1, it is true that the predictions made by each $P_{i,a}$ for these "known" state action pairs are near accurate with an error of at most $\frac{\epsilon}{nT(\epsilon)}$. Then through the standard analysis of the Implicit Explore and Exploit Lemma, the expected return for this phase is $\geq U^* - 3\epsilon$.

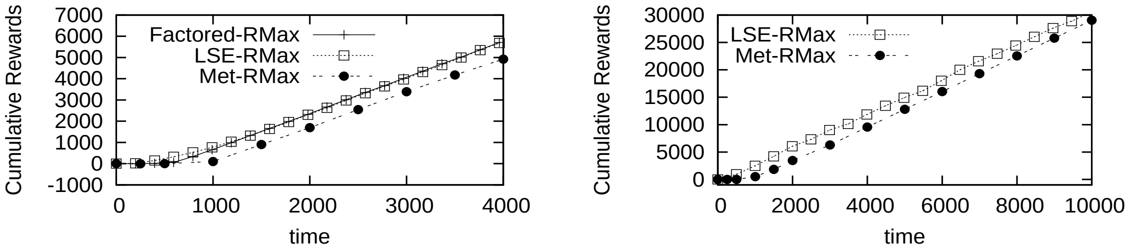
Then from Condition 1, and the fact noted earlier that each $\hat{P}_{i,a}$ is of size at most K w.h.p of at least $1 - \frac{\delta}{n|A|}$, it follows that the transition function of this approximate MDP must be based on the correct parent factors for all $P_{i,a}$'s, w.h.p of at least $1-\delta$. Hence, LSE-RMax has identified the correct parent factors for each $P_{i,a}$. From then onwards, in every regular phase LSE-RMax uses this model of the transition function, and eventually achieves an expected return $\geq U^* - 3\epsilon$ over all the regular phases. However, the return from an exploratory phase may yet still be arbitrary. By appropriately choosing $\phi = \lceil \frac{1-4\epsilon}{\epsilon} \rceil$, we then keep the expected return over every $\phi + 1$ phases $\geq U^* - 4\epsilon$. Analyzing in terms of overall return then brings us to our main theorem regarding LSE-RMax.

Theorem 5.1. *For a $T(\epsilon)$ that satisfies Condition 1, and $0 < \delta < 1$, w.h.p of at least $1 - 3\delta$, LSE-RMax achieves a return $\geq U^* - 6\epsilon - \delta$, in time steps: 1) $O(\frac{n^4 K^3 |A| d^K T(\epsilon)^3}{\epsilon^7} log(\frac{n^2|A|d^K}{\delta}) log^2(\frac{1}{\delta}))$ for SSLP; 2) $O(\binom{n}{2K} \frac{n^5 K^2 |A| d^{2K} T(\epsilon)^3}{\epsilon^7} log(\frac{n^2|A|d^{2K}}{\delta}) log^2(\frac{1}{\delta}))$ for SLP;*

The additional loss of δ is because, with probability at most δ, on each $T(\epsilon)$ step iteration, LSE-RMax can still plan based on a model that is sub-optimal, and hence achieve a return of 0 over that iteration.

6. Results

In this section we test LSE-RMax's empirical performance in two popular domains compared to its closest competitor Met-RMax (implemented as closely as possible to match their descriptions in the literature), and Factored-RMax which is aware of the DBN structures a priori. Theoretically in context of LSE-RMax, we need to set the value of m_k to $nV(k)$ for SLP, or to $V(k)$ for SSLP. Note that these estimates for m_k proven theoretically are extremely conservative, and for most practical scenarios, much lower values may suffice. In our experiments we set a fixed m for each k by doing a parameter search over a coarse grid, and choosing the best value. Also we ran LSE-RMax without an exploratory phase, i.e., $\phi = \infty$. The explorations from the regular phases were enough to determine the structure of the transition function. For, Met-RMax we set K to the in-degree (the tightest K). The m values chosen for the benchmarks are from the results reported in (Diuk et al., 2009). For all the al-



(a) Stock Trading (b) System Administrator

Figure 1. Results in two domains

gorithms, we use $T = 20$. Additionally for LSE-RMax we use $\rho = 0.99$, needed for the computation of the σ_k's (Section 3.2). All of our results have been averaged over 50 runs, and are statistically significant.

Stock Trading (Strehl et al., 2007): In this case LSE-RMax uses SLA to learn each factored transition function, with $m = 20$. LSE-RMax has a significantly higher cumulative reward than Met-RMax (Fig. 1(a)). In fact its performance is as good as Factored-RMax. The reason behind being LSE-RMax figures out the correct structure for almost 85 % of the factors within 1000 time steps, and hence behaves as Factored-RMax from there on. In fact it consistently follows the optimal policy from the 1000 time step in all runs. Prior to that LSE-RMax still accrues a high cumulative reward because it plans based on decent sub-optimal models.

System Administrator (Guestrin et al., 2003): Here we assume that it is known beforehand that the state of a machine depends on its own state, the action taken, and the states of its K closest neighbors from the past step, K being unknown. Thus in this case LSE-RMax uses SSLA, with $m = 30$. Our results are for the 8 machines in a ring topology, and assuming that every machine has the same transition structure. Again the cumulative reward accrued by LSE-RMax is significantly better than that of Met-RMax (Fig. 1(b)), and almost equivalent to that of Factored-RMax (plot omitted for clarity). In this case LSE-RMax figures out the correct parent factors in about 400 time steps.

7. Conclusion and Future Work

This paper introduces LSE-RMax, an algorithm for structure learning in FMDPs that given a big enough planning horizon that satisfies a certain condition, provably guarantees a return close to the optimal return, without requiring any prior knowledge of the in-degree of the transition function. We present two versions of LSE-RMax, tailored for two cases. In both cases, we argue that LSE-RMax has a competitive sample complexity. Our ongoing research agenda includes empirical analysis of LSE-RMax in the various factored domains prevalent in the RL literature.

Acknowledgments: This work has taken place in the Learning Agents Research Group (LARG) at UT Austin. LARG research is supported in part by NSF (IIS-0917122), ONR (N00014-09-1-0658), and the FHWA (DTFH61-07-H-00030).

References

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

Degris, Thomas, Sigaud, Olivier, and Wuillemin, Pierre-Henri. Learning the structure of Factored Markov Decision Processes in reinforcement learning problems. In *Proceedings of the 23rd International Conference on Machine learning' 06*, pp. 257–264.

Diuk, Carlos, Li, Lihong, and Leffler, Bethany R. The adaptive k-meteorologists problem and its application to structure learning and feature selection in reinforcement learning. In *Proceedings of the 26th International Conference on Machine Learning'09*, pp. 249–256.

Guestrin, Carlos, Patrascu, Relu, and Schuurmans, Dale. Algorithm-directed exploration for model-based reinforcement learning in Factored MDPs. In *Proceedings of the 19th International Conference on Machine Learning'02*, pp. 235–242.

Guestrin, Carlos, Koller, Daphne, Parr, Ronald, and Venkataraman, Shobha. Efficient solution algorithms for Factored MDPs. *Journal of the Artificial Intelligence Research*, pp. 399–468, 2003.

Kearns, Michael and Singh, Satinder. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, pp. 209–232, 2002.

Kearns, Michael J. and Koller, Daphne. Efficient reinforcement learning in Factored MDPs. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence' 99*, pp. 740–747.

Strehl, Alexander L., Diuk, Carlos, and Littman, Michael L. Efficient structure learning in Factored-State MDPs. In *Proceedings of the 22nd National Conference on Artificial intelligence*, pp. 645–650.

Sutton, Richard S. and Barto, Andrew G. *Reinforcement Learning: An Introduction*. MIT Press, 1998.