Efficient Reinforcement Learning in Factored MDPs

Michael Kearns
AT&T Labs
mkearns@research.att.com

Daphne Koller
Stanford University
koller@cs.stanford.edu

Abstract

We present a provably efficient and near-optimal algorithm for reinforcement learning in Markov decision processes (MDPs) whose transition model can be *factored* as a dynamic Bayesian network (DBN). Our algorithm generalizes the recent E^3 algorithm of Kearns and Singh, and assumes that we are given both an algorithm for approximate planning and the graphical structure (but not the parameters) of the DBN. Unlike the original E^3 algorithm, our new algorithm exploits the DBN structure to achieve a running time that scales polynomially in the number of *parameters* of the DBN, which may be exponentially smaller than the number of global states.

1 Introduction

Kearns and Singh [Kearns and Singh, 1998] recently presented a new algorithm for reinforcement learning in Markov decision processes (MDPs). Their E^3 algorithm (for *Explicit Exploit or Explore*) achieves near-optimal performance in a running time and a number of actions which are polynomial in the number of states and a parameter T (which is the time horizon in the case of discounted return, and the mixing time of the optimal policy in the case of infinite-horizon average return). The E^3 algorithm makes no assumptions on the structure of the unknown MDP, and the resulting polynomial dependence on the number of states makes E^3 inapplicable to the case of very large MDPs. In particular, it cannot be applied to MDPs in which the transition probabilities are represented in the factored form of a *dynamic Bayesian network* (DBN). MDPs with very large state spaces, and such *DBN-MDPs* in particular, are becoming increasingly important as reinforcement learning methods are attempted on problems of growing difficulty and complexity [Boutilier *et al.*, 1999].

In this paper, we extend the E^3 algorithm to the case of DBN-MDPs. The E^3 algorithm relies on the ability to find optimal strategies in MDPs. While this problem is intractable in large MDPs, significant progress has been made recently on *approximate* solution algorithms for DBN-MDPs [Boutilier *et al.*, 1999]. Our new DBN-E^3 algorithm therefore assumes the existence of a procedure for finding approximately optimal policies in any *given* DBN-MDP. Our algorithm also assumes that the qualitative structure of the transition model is known, i.e., the underlying graphical structure of the DBN. This assumption is often reasonable, as the qualitative properties of a domain are often understood. Using the planning procedure as a subroutine, DBN-E^3 explores the space, learning the parameters it considers relevant. It achieves near-optimal performance in a running time and a number of actions that are polynomial in T and the *number of parameters in the DBN-MDP*, which in general is exponentially smaller than the number of global states. We further show that, under very reasonable assumptions, the mixing time T of a policy in a DBN-MDP is polynomial in the number of parameters of the DBN-MDP, thereby eliminating the other potential source of exponential dependence in our algorithm.

2 Preliminaries

We begin by introducing some of the basic concepts of MDPs and factored MDPs. A *Markov Decision Process (MDP)* is defined as a tuple (S, A, R, T) where: S is a set of states; A is a set of actions; R is a *reward function* $R : S \mapsto [0, R_{max}]$, such that $R(s)$ represents the reward obtained by the agent in state s; T is a *transition model* $T : S \times A \mapsto \Delta_S$, such that $T(s' \mid s, a)$ represents the probability of landing in state s' if the agent takes action a in state s.

Most simply, MDPs are described explicitly, by writing down a set of transition matrices and reward vectors — one for each action a. However, this approach is impractical for describing complex processes. Here, the set of states is typically described via a set of random variables $\mathbf{X} = \{X_1, \ldots, X_n\}$, where each X_i takes on values in some finite domain D_i. A state in this context is an assignment of a value $x_i \in D_i$ to each variable X_i; we use $\mathbf{x}$ to denote such a state. In such an MDP, the set of states S is exponentially large in the number of variables. Thus, it is impractical to represent the transition model explicitly using transition matrices.

The framework of *dynamic Bayesian networks (DBNs)* allows us to describe a certain important class of such MDPs in a compact way. Processes whose state is described via a set of variables typically exhibit a weak form of decoupling — not all of the variables at time t directly influence the transition of a variable X_i from time t to time $t + 1$. For example, in a simple robotics domain, the location of the robot at time $t + 1$ may depend on its position, velocity, and orientation at time t, but not on what it is carrying.

Let $a \in A$ be an action. We first want to specify the transition model $T(\mathbf{x}' \mid \mathbf{x}, a)$. Let X_i denote the variable X_i at the current time and X_i' denote the variable at the next time step. The transition model for action a will consist of two parts — an underlying *transition graph* associated with a, and parameters associated with that graph. The transition graph is a 2-layer directed acyclic graph whose nodes are $\{X_1, \ldots, X_n, X_1', \ldots, X_n'\}$. All edges in this graph are directed from nodes in $\{X_1, \ldots, X_n\}$ to nodes in $\{X_1', \ldots, X_n'\}$, and we denote the parents of X_i' in the graph by $\mathrm{Pa}_a(X_i')$. Intuitively, the transition graph for a specifies the *qualitative* nature of probabilistic dependencies in a single time step — namely, the new setting of X_i depends only on the current setting of the variables in $\mathrm{Pa}_a(X_i')$. To make this dependence *quantitative*, each node X_i' is associated with a *conditional probability table (CPT)* $P_a(X_i' \mid \mathrm{Pa}_a(X_i'))$. The transition probability $P(\mathbf{x}' \mid \mathbf{x}, a)$ is then defined to be $\prod_i P_a(x_i' \mid \mathbf{u}_i)$, where $\mathbf{u}_i$ is the value in $\mathbf{x}$ of the variables in $\mathrm{Pa}_a(X_i')$.

We also need to provide a compact representation of the reward function. As in the transition model, explicitly specifying a reward for each of the exponentially many states is impractical. Again, we use the idea of factoring the representation of the reward function into a set of *localized* reward functions, each of which only depends on a small set of variables. In our robot example, our reward might be composed of several subrewards: for example, one associated with location (for getting too close to a wall), one associated with the printer status (for letting paper run out), and so on. More precisely, let $\mathcal{R}$ be a set of functions $R_1, \ldots, R_k$; each function R_i is associated with a cluster of variables $\mathbf{C}_i \subset \{X_1, \ldots, X_n\}$, such that R_i is a function from $\mathrm{Val}(\mathbf{C}_i)$ to $I\!R$. Abusing notation, we will use $R_i(\mathbf{x})$ to denote the value that R_i takes for the part of the state vector corresponding to $\mathbf{C}_i$. The reward function associated with the DBN-MDP at a state $\mathbf{x}$ is then defined to be $R(\mathbf{x}) = \sum_{i=1}^{k} R_i(\mathbf{x}) \in [0, R_{max}]$.

The following definitions for finite-length paths in MDPs will be of repeated technical use in the analysis. Let M be a Markov decision process, and let π be a policy in M. A *T-path in M* is a sequence p of $T+1$ states (that is, T transitions) of M: $p = \mathbf{x}_1, \ldots, \mathbf{x}_T, \mathbf{x}_{T+1}$. The probability that p is traversed in M upon starting in state $\mathbf{x}_1$ and executing policy π is denoted $P_M^\pi[p] = \Pi_{k=1}^{T} P(\mathbf{x}_{k+1} \mid \mathbf{x}_k, \pi(\mathbf{x}_k))$.

There are three standard notions of the expected *return* enjoyed by a policy in an MDP: the asymptotic discounted return, the asymptotic average return, and the finite-time average return. Like E^3, our new generalization will apply to all three cases, and to convey the main ideas it suffices for the most part to concentrate on the finite-time average return. This is because our finite-time average return result can be applied to the asymptotic returns through either the *horizon time* $1/(1-\gamma)$ for the discounted case or the *mixing time* of the optimal policy in the average case.

Let M be a Markov decision process, let π be a policy in M, and let p be a T-path in M. The (expected) *average return along* p in M is $U_M(p) = (1/T)(R(\mathbf{x}_1) + \cdots + R(\mathbf{x}_{T+1}))$. The *$T$-step* average return from state $\mathbf{x}$ is $U_M^\pi(\mathbf{x}, T) = \sum_p P_M^\pi[p] U_M(p)$ where the sum is over all T-paths p in M that start at $\mathbf{x}$. Furthermore, we define the *optimal T-step average return* from $\mathbf{x}$ in M by $U_M^*(\mathbf{x}, T) = \max_\pi\{U_M^\pi(\mathbf{x}, T)\}$.

An important problem in MDPs is *planning*: finding the policy π^* that achieves optimal return in a given MDP. In our case, we are interested in achieving the optimal T-step average return. The complexity of all exact MDP planning algorithms depends polynomially on the number of states; this property renders all of these algorithms impractical for DBN-MDPs, where the number of states grows exponentially in the size of the representation. However, there has been recent progress on algorithms for approximately solving MDPs with large state spaces, and particularly on ones represented in a factored way as an MDP [Boutilier *et al.*, 1999]. The focus of our work is on the reinforcement learning task, so we simply assume that we have access to a "black box" that performs approximate planning for a DBN-MDP.

Definition 2.1: A *μ-approximation T-step planning algorithm* for a DBN-MDP is one that, given a DBN-MDP M, produces a (compactly represented) policy π such that $U_M^\pi(\mathbf{x}, T) \geq (1-\mu) U_M^*(\mathbf{x}, T)$. ∎

We will charge our learning algorithm a single step of comuptation for each call to the assumed approximate planning algorithm.

Our goal is to perform model-based reinforcement learning. Thus, we wish to learn an approximate model from experience, and then exploit it (or explore it) by planning given the approximate model. In this paper, we focus on the problem of learning the model *parameters* (the CPTs), assuming that the model *structure* (the transition graphs) is given to us. It is therefore useful to consider the set of parameters that we wish to estimate. As we assumed that the rewards are deterministic, we can focus on the probabilistic parameters. (Our results easily extend to the case of stochastic rewards.) We define a *transition component* of the DBN-MDP to be a distribution $P_a(X_i' \mid \mathbf{u})$ for some action a and some particular instantiation $\mathbf{u}$ to the parents $\mathrm{Pa}_a(X_i')$ in the transition model. Note that the number of transition components is at most $\sum_{a,i} |\mathrm{Val}(\mathrm{Pa}_a(X_i'))|$, but may be much lower when a variable's behavior is identical for several actions.

3 Overview of the Original E^3

Since our algorithm for learning in DBN-MDPs will be a direct generalization of the E^3 algorithm of Kearns and Singh — hereafter abbreviated KS — we begin with an overview of that algorithm and its analysis. It is important to bear in mind that the original algorithm is designed only for the case where the total number of states N is small, and the algorithm runs in time polynomial in N.

E^3 is what is commonly referred to as an *indirect* or *model-based* algorithm: rather than maintaining only a current policy or value function, the algorithm maintains a model for the transition probabilities and the rewards for some *subset* of the states of the unknown MDP M. Although the algorithm maintains a partial model of M, it may choose to *never* build a *complete* model of M, if doing so is not necessary to achieve high return.

The algorithm starts off by doing *balanced wandering*. By this we mean that the algorithm, upon arriving in a state, takes the action it has tried the fewest times from that state (breaking ties randomly). At each state it visits, the algorithm maintains the obvious statistics: the reward received at that state, and for each action, the empirical distribution of next states reached (that is, the estimated transition probabilities).

A crucial notion is that of a *known state* — a state that the algorithm has visited "so many" times that the transition probabilities for that state are "very close" to their true values in M. This definition is carefully balanced so that "so many" times is still polynomially bounded, yet "very close" suffices to meet the simulation requirements below. An important observation is that we cannot do balanced wandering indefinitely before at least one state becomes known: by the Pigeonhole Principle, we will soon start to accumulate accurate statistics at some state.

The most important construction of the analysis is the *known-state MDP*. If $\mathcal{S}$ is the set of currently known states, the known-state MDP is simply an MDP $M_{\mathcal{S}}$ that is naturally *induced* on $\mathcal{S}$ by the full MDP M. Briefly, all transitions in M between states in $\mathcal{S}$ are preserved in $M_{\mathcal{S}}$, while all other transitions in M are "redirected" in $M_{\mathcal{S}}$ to lead to a single new, absorbing state that intuitively represents all of the unknown and unvisited states. Although E^3 does not have direct access to $M_{\mathcal{S}}$, by virtue of the definition of the known states, it does have a good *approximation* $\hat{M}_{\mathcal{S}}$. The KS analysis hinges on two central technical lemmas. The first is called the Simulation Lemma, and it establishes that $\hat{M}_{\mathcal{S}}$ has good *simulation accuracy*: that is, the expected T-step return of any policy in $\hat{M}_{\mathcal{S}}$ is close to its expected T-step return in $M_{\mathcal{S}}$. Thus, at any time, $\hat{M}_{\mathcal{S}}$ is a useful *partial* model of M, for that part of M that the algorithm "knows" very well.

The second central technical lemma is the "Exploit or Explore" Lemma. It states that either the optimal (T-step) policy in M achieves its high return by staying (with high probability) in the set $\mathcal{S}$ of currently known states — which, most importantly, the algorithm can detect and replicate by finding a high-return *exploitation* policy in the partial model $\hat{M}_{\mathcal{S}}$ — or the optimal policy has significant probability of *leaving* $\mathcal{S}$ within T steps — which again the algorithm can detect and replicate by finding an *exploration* policy that quickly reaches the additional absorbing state of the partial model $\hat{M}_{\mathcal{S}}$. Thus, by performing two off-line computations on $\hat{M}_{\mathcal{S}}$, the algorithm is guaranteed to find either a way to get near-optimal return for the next T steps, or a way to improve the statistics at an unknown or unvisited state within the next T steps. KS show that this algorithm ensures near-optimal return in time polynomial in N.

Our goal is to derive a generalization of E^3 for DBN-MDPs, and to prove for it a result analogous to that of KS — but with a polynomial dependence not on the number of states N, but on the number of CPT parameters ℓ in the DBN model. Our analysis closely mirrors the original, but requires a significant generalization of the Simulation Lemma that exploits the structure of a DBN-MDP, a modified construction of $\hat{M}_{\mathcal{S}}$ that can be represented as a DBN-MDP, and a number of alterations of the details.

4 The DBN-E^3 Algorithm

Like the original E^3 algorithm, DBN-E^3 will build a model of the unknown DNB-MDP on the basis of its experience, but now the model will be represented in a compact, factorized form. More precisely, suppose that our algorithm is in state $\mathbf{x}$, executes action a, and arrives in state $\mathbf{x}'$. This experience will be used to update all the appropriate CPT entries of our model — namely, all the estimates $\hat{P}_a(x'_i \mid \mathbf{u}_i)$ are updated in the obvious way, where as usual $\mathbf{u}_i$ is the setting of $\mathrm{Pa}_a(X'_i)$ in $\mathbf{x}$. We will also maintain counts $C_a(x'_i, \mathbf{u}_i)$ of the number of times $\hat{P}_a(x'_i \mid \mathbf{u}_i)$ has been updated.

Recall that a crucial element of the original E^3 analysis was the notion of a *known state*. In the original analysis, it was observed that if N is the total number of states, then after $O(N)$ experiences some state must become known by the Pigeonhole Principle. We cannot hope to use the same logic here, as we are now in a DBN-MDP with an exponentially large number of states. Rather, we must "pigeonhole" not on the number of states, but on the number of parameters required to specify the DBN-MDP. Towards this goal, we will say that the *CPT entry* $\hat{P}_a(x'_i \mid \mathbf{u}_i)$ is *known* if it has been visited "enough" times to ensure that, with high probability

$$|\hat{P}_a(x'_i \mid \mathbf{u}_i) - \hat{P}_a(x'_i \mid \mathbf{u}_i)| \leq \alpha.$$

We now would like to establish that if, for an appropriate choice of α, *all* CPT entries are known, then our approximate DBN-MDP can be used to accurately estimate the expected return of any policy in the true DBN-MDP. This is the desired generalization of the original Simulation Lemma. As in the original analysis, we will eventually apply it to a generalization of $M_{\mathcal{S}}$, in which we deliberately restrict attention to only the known CPT entries.

4.1 The DBN-MDP Simulation Lemma

Let M and $\hat{M}$ be two DBN-MDPs over the same state space with the same transition graphs for every action a, and with the same reward functions. Then we say that $\hat{M}$ is an α-*approximation* of M if for every action a and node X'_i in the transition graphs, for every setting $\mathbf{u}$ of $\mathrm{Pa}_a(X'_i)$, and for every possible value x'_i of X'_i, $|P_a(X'_i = x'_i \mid \mathbf{u}) - \hat{P}_a(X'_i = x'_i \mid \mathbf{u})| \leq \alpha$, where $P_a(\cdot \mid \cdot)$ and $\hat{P}_a(\cdot \mid \cdot)$ are the CPTs of M and $\hat{M}$, respectively.

Lemma 4.1: *Let M be any DBN-MDP over n state variables, and let $\hat{M}$ be an α-approximation of M, where $\alpha = O((\epsilon/(T^2 n R_{max}))^2)$. Then for any policy π, and for any state $\mathbf{x}$, $|U^\pi_M(\mathbf{x}, T) - U^\pi_{\hat{M}}(\mathbf{x}, T)| \leq \epsilon$.*

Proof: (Sketch) Let us fix a policy π and state $\mathbf{x}$. Recall that for any next state $\mathbf{x}'$ and any action a, the transition probability factorizes via the CPTs as $P_a(\mathbf{x}' \mid \mathbf{x}) = \prod_i P_a(x'_i \mid \mathbf{u}_i)$. where $\mathbf{u}_i$ is the setting of $\mathrm{Pa}_a(X'_i)$ in $\mathbf{x}$. Let us say that $P(\mathbf{x}' \mid \mathbf{x}, a)$ *contains a β-small factor* if any of its CPT factors $P_a(x'_i \mid \mathbf{u}_i)$ is smaller than β. Note that a transition probability may actually be quite small itself (exponentially small in n) without necessarily containing a β-small factor.

Consider a random trajectory of T steps in M from state $\mathbf{x}$ following policy π. Let v be the maximum number of values of any variable X_i. We first prove that the probability that such a trajectory will cross at least one transition $P(\mathbf{x}' \mid \mathbf{x}, a)$ that contains a β-small factor is at most $Tnv\beta$. Essentially, the probability that at any step, a particular β-small transition will be taken by a particular variable X_i is at most β. A simple union argument over variables X_i, their values, and time steps gives the desired bound. Therefore, the total contribution to the difference $|U^\pi_M(\mathbf{x},T) - U^\pi_{\hat{M}}(\mathbf{x},T)|$ by these trajectories can be shown to be at most $T^2 R_{max} vn(\alpha+\beta)$. We will thus ignore such trajectories for now.

The key advantage of eliminating β-small factors is that we can convert additive approximation guarantees into multiplicative ones. Let p be any path of length T. If all the relevant CPT factors are greater than β, and we let $\Delta = \alpha/\beta$, it can be shown that

$$(1-\Delta)^{Tn} P^\pi_M[p] \leq \hat{P}^\pi_M[p] \leq (1+\Delta)^{Tn} P^\pi_M[p].$$

In other words, ignoring β-small CPT factors, the distributions on paths induced by π in M and $\hat{M}$ are quite similar. From this it follows that, for the upper bound (the lower bound argument is entirely symmetric)

$$U^\pi_{\hat{M}}(\mathbf{x},T) \leq (1+\Delta)^{Tn} U^\pi_M(\mathbf{x},T) + T^2 R_{max} nv(\alpha+2\beta).$$

For the choices $\beta = \sqrt{\alpha}$, $\alpha = O((\epsilon/(T^2 nv R_{max}))^2)$ the lemma is obtained. ∎

Returning to the main development, we can now give a precise definition of a known CPT entry. It is a simple application of Chernoff bounds to show that provided the count $C_a(x'_i, \mathbf{u}_i)$ exceeds $O(1/\alpha^2 \log(1/\delta))$, $\hat{P}_a(x'_i \mid \mathbf{u}_i)$ has additive error at most α with probability at least $1-\delta$. We thus say that this CPT entry is known if its count exceeds the given bound for the choice $\alpha = O((\epsilon/(T^2 nv R_{max}))^2)$ specified by the DBN-MDP Simulation Lemma. The DBN-MDP Simulation Lemma shows that if *all* CPT entries are known, then our approximate model $\hat{M}$ can be used to find a near-optimal policy in the true DBN-MDP M.

Note that we can *identify* which CPT entries are known via the counts $C_a(x'_i, \mathbf{u}_i)$. Thus, if we are at a state $\mathbf{x}$ for which at least one of the associated CPT entries $\hat{P}_a(x'_i \mid \mathbf{u}_i)$ is unknown, by taking action a we then obtain an experience that will increase the corresponding count $C_a(x'_i, \mathbf{u}_i)$. Thus, in analogy with the original E^3, as long as we are encountering unknown CPT entries, we can continue taking actions that increase the quality of our model — but now rather than increasing counts on a per-state basis, the DBN-MDP Simulation Lemma shows why it suffices to increase the counts on a per-CPT entry basis, which is crucial for obtaining the running time we desire. We can thus show that if we encounter unknown CPT entries for a number of steps that is polynomial in the total number ℓ of *CPT entries* and $1/\epsilon$, there can no longer be any unknown CPT entries, and we know the true DBN-MDP well enough to solve for a near-optimal policy.

However, similar to the original algorithm, the real difficulty arises when we are in a state with no unknown CPT entries, yet there do remain unknown CPT entries elsewhere. Then we have no guarantee that we can improve our model at the next step. In the original algorithm, this was solved by defining the known-state MDP $M_{\mathcal{S}}$, and proving the aforementioned "Exploit or Explore" Lemma. Duplicating this step for DBN-MDPs will require another new idea.

4.2 The DBN-E^3 "Exploit or Explore" Lemma

In our context, when we construct a known-state MDP, we must satisfy the additional requirement that the known-state MDP preserve the DBN structure of the original problem, so that if we have a planning algorithm for DBN-MDPs that exploits the structure, we can apply it to the known-state MDP. Therefore, we cannot just introduce a new "sink state" to represent that part of M that is unknown to us; we must also show how this "sink state" can be represented as a setting of the state variables of a DBN-MDP.

We present a new construction, which extends the idea of "known states" to the idea of "known transitions". We say that a transition component $P_a(X'_i \mid \mathbf{u})$ is *known* if all of its CPT entries are known. The basic idea is that, while it is impossible to check locally whether a state is known, it is easy to check locally whether a transition component is known.

Let $\mathcal{T}$ be the set of known transition components. We define the known-transition DBN-MDP $M_\mathcal{T}$ as follows. The model behaves identically to M as long as only known transitions are taken. As soon as an unknown transition is taken for some variable X'_i, the variable X'_i takes on a new *wandering* value w, which we introduce into the model. The transition model is defined so that, once a variable takes on the value w, its value never changes. The reward function is defined so that, once at least one variable takes on the wandering value, the total reward is nonpositive. These two properties give us the same overall behavior that KS got by making a sink state for the set of unknown states.

Definition 4.2: Let M be a DBN-MDP and let $\mathcal{T}$ be any subset of the transition components in the model. The *induced DBN-MDP on* $\mathcal{T}$, denoted $M_\mathcal{T}$, is defined as follows:

- $M_\mathcal{T}$ has the same set of state variables as M; however, in $M_\mathcal{T}$, each variable X_i has an additional possible value w. We use $\mathrm{Val}^M(X_i)$ to denote the set of original values of X_i, and $\mathrm{Val}^{M_\mathcal{T}}(X_i)$ to denote the expanded set.
- $M_\mathcal{T}$ has the same transition graphs as M. For each a, i, and $\mathbf{u} \in \mathrm{Val}^M(\mathrm{Pa}_a(X'_i))$, we have that $P^{M_\mathcal{T}}_a(X'_i \mid \mathbf{u}) = P^M_a(X'_i \mid \mathbf{u})$ if the corresponding transition component is in $\mathcal{T}$; in all other cases, $P^{M_\mathcal{T}}_a(w \mid \mathbf{u}) = 1$, and $P^{M_\mathcal{T}}_a(x_i \mid \mathbf{u}) = 0$ for all $x_i \in \mathrm{Val}^M(X_i)$.
- $M_\mathcal{T}$ as the same set $\mathcal{R}$ as M. For each $i = 1, \ldots, k$ and $\mathbf{c} \in \mathrm{Val}^M(\mathbf{C}_i)$, we have that $R^{M_\mathcal{T}}_i(\mathbf{c}) = R^M_i(\mathbf{c})$. For other vectors $\mathbf{c}$, we have that $R^{M_\mathcal{T}}_i(\mathbf{c}) = -R_{max}$. ∎

With this definition, we can prove the analogue to the "Exploit or Explore" Lemma.

Lemma 4.3: *Let M be any DBN-MDP, let $\mathcal{T}$ be any subset of the transition components of M, and let $M_\mathcal{T}$ be the induced MDP on M. For any $\mathbf{x} \in S$, any T, and any $1 > \tau > 0$, either there exists a policy π in $M_\mathcal{T}$ such that $U^\pi_{M_\mathcal{T}}(\mathbf{x},T) \geq U^*_M(\mathbf{x},T) - \tau$, or there exists a policy π in $M_\mathcal{T}$ such that the probability that a walk of T steps following π will take at least one transition not in $\mathcal{T}$ exceeds $\tau/((k+1)TR_{max})$.*

4.3 Putting It All Together

We now have all the pieces to finish the description and analysis of the DBN-E^3 algorithm. The algorithm initially executes balanced wandering for some period of time. After some number of steps, by the Pigeonhole Principle one or more transition components become known. When the algorithm reaches a known state $\mathbf{x}$ — one where all the transition components are known — it can no longer perform balanced wandering. At that point, the algorithm performs approximate off-line policy computations for two different DBN-MDPs. The first corresponds to attempted exploitation, and the second to attempted exploration.

Let $\mathcal{T}$ be the set of known transitions at this step. In the attempted exploitation computation, the DBN-E^3 algorithm would like to find the optimal policy on the induced DBN-MDP $M_{\mathcal{T}}$. Clearly, this DBN-MDP is not known to the algorithm. Thus, we use its approximation $\hat{M}_{\mathcal{T}}$, where the true transition probabilities are replaced with their current approximation in the model. The definition of $M_{\mathcal{T}}$ uses only the CPT entries of known transition components. The Simulation Lemma now tells us that, for an appropriate choice of α — a choice that will result in a definition of known transition that requires the corresponding count to be only polynomial in $1/\epsilon$, n, v, and T — the return of any policy π in $\hat{M}_{\mathcal{T}}$ is within ϵ of its return in $M_{\mathcal{T}}$. We will specify a choice for ϵ later (which in turn sets the choice of α and the definition of known state).

Let us now consider the two cases in the "Exploit or Explore" Lemma. In the exploitation case, there exists a policy π in $M_{\mathcal{T}}$ such that $U^{\pi}_{M_{\mathcal{T}}}(\mathbf{x}, T) \geq U^{*}_{M}(\mathbf{x}, T) - \tau$. (Again, we will discuss the choice of τ below.) From the Simulation Lemma, we have that $U^{\pi}_{\hat{M}_{\mathcal{T}}}(\mathbf{x}, T) \geq U^{*}_{M}(\mathbf{x}, T) - (\tau + \epsilon)$. Our approximate planning algorithm returns a policy π' whose value in $\hat{M}_{\mathcal{T}}$ is guaranteed to be a multiplicative factor of at most $1 - \mu$ away from the optimal policy in $\hat{M}_{\mathcal{T}}$. Thus, we are guaranteed that $U^{\pi'}_{\hat{M}_{\mathcal{T}}}(\mathbf{x}, T) \geq (1 - \mu)(U^{*}_{M}(\mathbf{x}, T) - (\tau + \epsilon))$. Therefore, in the exploitation case, our approximate planner is guaranteed to return a policy whose value is close to the optimal value.

In the exploration case, there exists a policy π in $M_{\mathcal{T}}$ (and therefore in $\hat{M}_{\mathcal{T}}$) that is guaranteed to take an unknown transition within T steps with some minimum probability. Our goal now is to use our approximate planner to find such a policy. In order to do that, we need use a slightly different construction $M'_{\mathcal{T}}$ ($\hat{M}'_{\mathcal{T}}$). The transition structure of $M'_{\mathcal{T}}$ is identical to that of $M_{\mathcal{T}}$. However, the rewards are now different. Here, for each $i = 1, \ldots, k$ and $\mathbf{c} \in \mathrm{Val}^{M}(\mathbf{C}_i)$, we have that $R_i^{M'_{\mathcal{T}}}(\mathbf{c}) = 0$; for other vectors $\mathbf{c}$, we have that $R_i^{M_{\mathcal{T}}}(\mathbf{c}) = 1$. Now let π' be the policy returned by our approximate planner on the DBN-MDP $\hat{M}'_{\mathcal{T}}$. It can be shown that the probability that a T-step walk following π' will take at least one unknown transition is at least $(1 - \mu)(\tau/((k + 1)TR_{max}) - \epsilon)/kT$.

To summarize: our approximate planner either finds an exploitation policy π in $\hat{M}_{\mathcal{T}}$ that enjoys actual return $U^{\pi}_{M}(\mathbf{x}, T) \geq (1 - \mu)(U^{*}_{M}(\mathbf{x}, T) - (\tau + \epsilon))$ from our current state $\mathbf{x}$, or it finds an exploitation policy in $\hat{M}'_{\mathcal{T}}$ that has probability at least $p = (1 - \mu)(\tau/((k + 1)TR_{max}) - \epsilon)/kT$ of improving our statistics at an unknown transition in the next T steps. Appropriate choices for ϵ and τ yield our main theorem, which we are now finally ready to describe.

Recall that for expository purposes we have concentrated on the case of T-step average return. However, as for the original E^3, our main result can be stated in terms of the asymptotic discounted and average return cases. We omit the details of this translation, but it is a simple matter of arguing that it suffices to set T to be either $(1/(1-\gamma))\log(1/\epsilon)$ (discounted) or the mixing time of the optimal policy (average).

Theorem 4.4: *(Main Theorem) Let M be a DBN-MDP with ℓ total entries in the CPTs.*

- *(Undiscounted case) Let T be the mixing time of the policy achieving the optimal average asymptotic return U^* in M. There exists an algorithm DBN-E^3 that, given access to a μ-approximation planning algorithm for DBN-MDPs, and given inputs $\epsilon, \delta, \ell, T$ and U^*, takes a number of actions and computation time bounded by a polynomial in $1/(1-\mu)$, $1/\epsilon$, $1/\delta$, ℓ, T, and R_{max}, and with probability at least $1 - \delta$, achieves total actual return exceeding $U^* - \epsilon$.*
- *(Discounted case) Let V^* denote the value function for the policy with the optimal expected discounted return in M. There exists an algorithm DBN-E^3 that, given access to a μ-approximation planning algorithm for DBN-MDPs, and given inputs ϵ, δ, ℓ and V^*, takes a number of actions and computation time bounded by a polynomial in $1/(1 - \mu), 1/\epsilon, 1/\delta, \ell$, the horizon time $T = 1/(1-\gamma)$, and R_{max}, and with probability at least $1 - \delta$, will halt in a state $\mathbf{x}$, and output a policy $\hat{\pi}$, such that $V^{\hat{\pi}}_{M}(\mathbf{x}) \geq V^*(\mathbf{x}) - \epsilon$.*

As with the original E^3, we can eliminate knowledge of the optimal returns in both cases via search techniques, and for the average case can give an "anytime" algorithm that "competes" against policies with longer and longer mixing times the longer it is run (details omitted).

5 Conclusions

Structured probabilistic models, and particularly Bayesian networks, have revolutionized the field of reasoning under uncertainty by allowing compact representations of complex domains. Their success is built on the fact that this structure can be exploited effectively by inference and learning algorithms. This success leads one to hope that similar structure can be exploited in the context of planning and reinforcement learning under uncertainty. This paper, together with the recent work on representing and reasoning with factored MDPs [Boutilier *et al.*, 1999], demonstrate that substantial computational gains can indeed be obtained from these compact, structured representations.

This paper leaves many interesting problems unaddressed. Of these, the most intriguing one is to allow the algorithm to learn the model structure as well as the parameters. The recent body of work on learning Bayesian networks from data [Heckerman, 1995] lays much of the foundation, but the integration of these ideas with the problems of exploration/exploitation is far from trivial.

References

[Boutilier *et al.*, 1999] C. Boutilier, T. Dean, and S. Hanks. Decision theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 1999. To appear.

[Heckerman, 1995] D. Heckerman. A tutorial on learning with Bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research, 1995.

[Kearns and Singh, 1998] M. Kearns and S.P. Singh. Near-optimal performance for reinforcement learning in polynomial time. In *Proc. ICML*, pages 260–268, 1998.