Algorithm-Directed Exploration for Model-Based Reinforcement Learning in Factored MDPs

Carlos Guestrin GUESTRIN@CS.STANFORD.EDU
Department of Computer Science, Stanford University, Stanford, CA, USA

Relu Patrascu RPATRASC@CS.UWATERLOO.CA
Dale Schuurmans DALE@CS.UWATERLOO.CA
Department of Computer Science, University of Waterloo, Waterloo, ON, Canada

Abstract

One of the central challenges in reinforcement learning is to balance the exploration/exploitation tradeoff while scaling up to large problems. Although model-based reinforcement learning has been less prominent than value-based methods in addressing these challenges, recent progress has generated renewed interest in pursuing model-based approaches: Theoretical work on the exploration/exploitation tradeoff has yielded provably sound model-based algorithms such as E^3 and R_{max}, while work on factored MDP representations has yielded model-based algorithms that can scale up to large problems. Recently the benefits of both achievements have been combined in the *Factored* E^3 algorithm of Kearns and Koller. In this paper, we address a significant shortcoming of *Factored* E^3: namely that it requires an oracle planner that cannot be feasibly implemented. We propose an alternative approach that uses a practical approximate planner, *approximate linear programming*, that maintains desirable properties. Further, we develop an exploration strategy that is targeted toward improving the performance of the linear programming algorithm, rather than an oracle planner. This leads to a simple exploration strategy that visits states relevant to tightening the LP solution, and achieves sample efficiency logarithmic in the size of the problem description. Our experimental results show that the targeted approach performs better than using approximate planning for implementing either *Factored* E^3 or *Factored* R_{max}.

1. Introduction

Research on reinforcement learning has consistently focused on developing algorithms that effectively manage the exploration/exploitation tradeoff [1, 11] and attempting to scale up to large real-world problems [8, 7]. Most research in these directions has focused on the direct value-approximation approach [4, 5, 25] rather than the indirect model-based approach, because the model-based approach has not been amenable to scaling up to large problems [24]. However, two recent developments in model-based reinforcement learning and Markov decision process (MDP) planning have created new opportunities for model-based techniques.

The E^3 algorithm of Kearn and Singh [15] is the first known technique for managing the exploration/exploitation tradeoff that is guaranteed to yield approximately optimal policies with near-minimal exploration.[1] This work has been recently simplified and generalized as R_{max} [6].

A parallel development has been significant progress in developing compact representations for large MDPs. These representations scale *logarithmically* (not linearly) in the size of the state space; that is, they are polynomial in the size of the state *description*. A polynomial size representation is often thought to be essential to scale up model-based MDP planning to realistic problems [2]. Of these approaches, the *factored* MDP representation developed by Koller and Parr has proven to be particularly convenient [17, 18, 12]. Combined with linear value function approximators it allows practically efficient planning algorithms based on linear programming to be easily implemented [13, 22].

Both lines of research—exploration/exploitation and compact representations—have recently been brought together in the *Factored* E^3 algorithm of Kearns and Koller [14]. This algorithm combines the theoretical exploration/exploitation guarantees of E^3 with the ability to scale up to large state spaces afforded by factored MDP representations. However, there is a shortcoming with the result: it relies on an oracle planning algorithm that must guarantee approximately optimal plans for the exploration and exploitation steps. Unfortunately, it is well known that even approximate planning is an inherently hard compuational problem in factored MDPs [19, 20], and it is thus unlikely that the *Factored* E^3 algorithm can ever be feasibly implemented as is.

The first main contribution of this paper is to investigate a practical approach to implementing factored versions of E^3 and R_{max} by using approximate planning algorithms while preserving what one can of the theoretical guarantees. Specifically, we consider approximate linear programming as a practical planning algorithm, since it can be efficiently applied to large problems (in factored form) without difficulty [13, 22]. The second contribution is to take this development one step further and demonstrate that the exploration/exploitation strategy itself can be tailored to the specific approximate planning algorithm being used. That is,

[1] That is, E^3's exploration requirements are near-minimal in terms of how they *scale* in problem size and approximation accuracy, but not not in terms of constant factors.

rather than target the exploration toward every unknown region of the domain, or toward the unknown regions relevant to an oracle planner, one can more effectively direct exploration toward only the domain uncertainties that affect the results of the approximate planner in a direct way. In fact, below we show that linear programming offers a particularly elegant avenue to identifying relevant uncertainties in the domain and therefore directing exploration effectively. We have implemented practical factored versions of all these algorithms—E^3, R_{max}, and our new relevance directed approach—and compared them to benchmark strategies on simple domains. Our experimental results confirm that the directed exploration scheme yields benefits over the other approaches.

2. Markov decision processes

We first consider the problem of calculating optimal behavior in a known stochastic domain, which can be formalized as planning in a *Markov Decision Process (MDP)*. An MDP is defined as a 4-tuple $(\mathbf{X}, \mathcal{A}, R, P)$ where $\mathbf{X}$ is a finite set of states, and we let $N = |\mathbf{X}|$; $\mathcal{A}$ is a set of actions; R is a *reward function* $R : \mathbf{X} \times \mathcal{A} \mapsto [0, R_{max}]$ such that $R(\mathbf{x}, a)$ represents the reward obtained in state $\mathbf{x}$ after taking action a; and P is a *Markovian transition model* such that $P(\mathbf{x}' \mid \mathbf{x}, a)$ represents the probability of going from state $\mathbf{x}$ to state $\mathbf{x}'$ after taking action a.

A stationary policy π for an MDP is a mapping $\pi : \mathbf{X} \mapsto A$, where $\pi(\mathbf{x})$ is the action the agent takes in state $\mathbf{x}$. We assume that the MDP has an infinite horizon and the agent is interested in maximizing the average reward received per time step:

$$\rho^{\pi} = \lim_{T \to \infty} \frac{1}{T} E\left[\sum_{t=1}^{T} R\left(\mathbf{x}^{(t)}, \pi(\mathbf{x}^{(t)})\right)\right],$$

where $\mathbf{x}^{(t)}$ is the state of the system at time t, and the expectation is taken over possible sequences of states. We assume that for any stationary policy π, the resulting Markov chain $P(\mathbf{x}' \mid \mathbf{x}, \pi(\mathbf{x}))$ is ergodic and hence the average reward ρ^{π} is unique, independent of the starting state [21].

The optimal policy π^* is characterized by a set of fixed-point equations

$$\rho^* + \mathcal{V}^*(\mathbf{x}) = \max_a \left[R(\mathbf{x}, a) + \gamma \sum_{\mathbf{x}'} P(\mathbf{x}' \mid \mathbf{x}, a)\mathcal{V}^*(\mathbf{x}')\right],$$

for all $\mathbf{x}$. Here, ρ^* is the maximum average reward. The function $\mathcal{V}(\mathbf{x})$ can be interpreted as the advantage of starting in state $\mathbf{x}$ rather than a random state chosen according to the stationary distribution. For any value function $\mathcal{V}$, we can define the policy obtained by acting greedily relative to $\mathcal{V}$ by

$$\mathit{Greedy}(\mathcal{V})(\mathbf{x}) = \arg\max_a R(\mathbf{x}, a) + \gamma \sum_{\mathbf{x}'} P(\mathbf{x}' \mid \mathbf{x}, a)\mathcal{V}(\mathbf{x}').$$

Finally, the greedy policy relative to the optimal value function $\mathcal{V}^*$ is the optimal policy $\pi^* = \mathit{Greedy}(\mathcal{V}^*)$.

There are several algorithms for computing the optimal policy in average reward MDPs [21]. One particularly convenient approach is linear programming which we exploit below. To formulate an explicit version of the required linear program, enumerate the states in $\mathbf{X}$ as $\mathbf{x}_1, \ldots, \mathbf{x}_N$. Then the variables for the linear program are given by $V_1, \ldots, V_N$, where V_i represents $\mathcal{V}(\mathbf{x}_i)$. The linear program is then:

$$\begin{array}{ll} \text{For variables:} & \rho, V_1, \ldots, V_N\ ; \\ \text{Minimize:} & \rho\ ; \\ \text{Subject to:} & \rho + V_i \geq R(\mathbf{x}_i, a) + \sum_k P(\mathbf{x}'_k \mid \mathbf{x}_i, a)V_k, \\ & \text{for all } \mathbf{x}_i \in \mathbf{X}, a \in \mathcal{A}. \end{array} \tag{1}$$

Here the policy is implicitly represented in the slack variables of the linear program. That is, for each $\mathbf{x}_i$ there will be at least one a for which the corresponding state value constraint is tight, and this will in fact be an optimal action for $\mathbf{x}_i$.

Overall, the problem of computing optimal policies for given MDPs raises two important caveats: First, the linear program in (1) contains a variable V_i for each complete state $\mathbf{x}_i$. However, the state space is usually very large in most settings (exponential in the number of state variables). Second, the problem assumes full knowledge of the model parameters; specifically, the transition probabilities and the reward function. These limitations are major hindrances to the application of MDPs to the control of real stochastic systems. We consider each of these shortcomings in turn.

First, to address the first issue of a large state space, we use the common approach of considering value function approximations that are compactly represented as a linear combination of *basis functions* $H = \{h_1, \ldots, h_k\}$. A *linear value function* over H is a function $\mathcal{V}$ that can be written as $\mathcal{V}(\mathbf{x}) = \sum_{j=1}^{k} w_j h_j(\mathbf{x}_j)$ for coefficients $\mathbf{w} = (w_1, \ldots, w_k)'$. Here $\mathbf{x}_j$ is the subset of the state variables that h_j depends on. Given a linear value function representation the linear programming approach can be adapted to produce an approximation to the optimal value function:

$$\begin{array}{ll} \text{For variables:} & \rho, w_1, \ldots, w_k\ ; \\ \text{Minimize:} & \rho\ ; \\ \text{Subject to:} & \rho + \sum_{j=1}^{k} w_j h_j(\mathbf{x}_i) \ \geq \\ & R(\mathbf{x}_i, a) + \sum_{\mathbf{x}'_l} P(\mathbf{x}'_l \mid \mathbf{x}_i, a) \sum_{j=1}^{k} w_j h_j(\mathbf{x}'_l); \\ & \text{for all } \mathbf{x}_i \in \mathbf{X}, a \in \mathcal{A}. \end{array} \tag{2}$$

This linear program is guaranteed to be feasible for any set of basis functions. (An analogous linear program was proposed for the discounted infinite horizon case in [23].) In general, there is no guarantee of the quality of the approximation $\sum_{j=1}^{k} w_j h_j$ obtained by such approach, but recent work [9] on the discounted reward case provides some analysis of the error relative to that of the best possible approximation in the subspace. This transformation has the effect of reducing the number of free variables in the linear program to $k + 1$, but the number of constraints remains $|\mathbf{X}| \times |\mathcal{A}|$. Fortunately, using the algorithms of [12, 13, 22] one can exploit the structure of a *factored MDP* (see Section 3) to obtain a compact representation and efficient solution to this linear program.

The second limitation of the explicit MDP planning approach, unknown model parameters, has been the focus of extensive work in the field of *reinforcement learning (RL)* [24]. Two recent algorithms, E^3 (*Explicit Exploit and Explore*) [15] and R_{max} [6], can obtain provably near-optimal performance in time polynomial in the number of states, among other quantities. Unfortunately, the number of states is usually too large to be handled explicitly (i.e. exponential) and hence these methods are not usually practical. Here again one could exploit structure in a factored MDP to make these algorithms feasible. Kearns and Koller [14] proposed *Factored* E^3, a version the E^3 algorithm for factored MDPs. In Section 5, we present this algorithm and, in Section 6, we propose *Factored* R_{max}, a similar extension for the R_{max} algorithm.

3. Factored MDPs

Factored MDPs allow one to exploit problem structure to represent exponentially large MDPs compactly. The idea of representing a large MDP using a factored model was first proposed by Boutilier et al. [3]. Our presentation of factored MDPs follows that of Koller and Parr [18]. In a factored MDP, the set of states is 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 $\mathrm{Dom}(X_i)$. A state $\mathbf{x}$ defines a value $x_i \in \mathrm{Dom}(X_i)$ for each variable X_i. We define a state transition model τ using a *dynamic Bayesian network (DBN)* [10]. Let X_i denote a variable at the current time and let X'_i denote the same variable at the successive step. The *transition graph* of a DBN is a two-layer directed acyclic graph G_τ whose nodes are $\{X_1, \ldots, X_n, X'_1, \ldots, X'_n\}$. We denote the parents of X'_i in the graph by $\mathit{Parents}_\tau(X'_i)$. For simplicity of exposition, we assume that $\mathit{Parents}_\tau(X'_i) \subseteq \mathbf{X}$; i.e., all arcs in the DBN are between variables in consecutive time slices. Each node X'_i is associated with a *conditional probability distribution (CPD)* $P_\tau(X'_i \mid \mathit{Parents}_\tau(X'_i))$. The transition probability $P_\tau(\mathbf{x}' \mid \mathbf{x})$ is then defined to be $\prod_i P_\tau(x'_i \mid \mathbf{u}_i)$, where $\mathbf{u}_i$ is the value in $\mathbf{x}$ of the variables in $\mathit{Parents}_\tau(X'_i)$. We can define the transition dynamics of an MDP by defining a separate DBN model $\tau_a = \langle G_a, P_a \rangle$ for each action a. [2]

Finally, we need to provide a compact representation of the reward function. We assume that the reward function is factored additively into a set of localized reward functions, each of which only depends on a small set of variables. We can formalize this concept of localized functions:

Definition 3.1 *A function f is* restricted *to a scope* $\mathrm{Scope}[f] = \mathbf{C} \subseteq \mathbf{X}$ *if* $f : \mathrm{Dom}(\mathbf{C}) \mapsto \mathbb{R}$. *Furthermore, if f is restricted to $\mathbf{Y}$ and $\mathbf{Y} \subset \mathbf{Z}$, we will use $f(\mathbf{z})$ as shorthand for $f(\mathbf{y})$ where $\mathbf{y}$ is the part of the instantiation $\mathbf{z}$ that corresponds to variables in $\mathbf{Y}$.* ∎

We can now characterize the concept of local rewards. Let $R_1, \ldots, R_r$ be a set of functions, where each R_i is restricted to variable cluster $\mathbf{W}_i \subset \{X_1, \ldots, X_n\}$. The reward function for state $\mathbf{x}$ is defined to be $\sum_{i=1}^r R_i(\mathbf{W}_i) \in \mathbb{R}$. As with the transition model, we assume there is a (potentially) different decomposition R_i^a for each action a.

[2] In multiagent systems, the action space is exponentially large. The results presented in this paper can be extended to such systems by applying the planning algorithm of Guestrin et al. [13].

Factorization allows very large MDPs to be represented compactly. However, we must still address the problem of planning in these MDPs. One might be tempted to believe that factored transition dynamics and rewards would result in a factored value function, which might allow a planner to consider compact value function and policy descriptions. Unfortunately, even in factored MDPs, the value function rarely maintains any internal structure [17].

Koller and Parr [17] suggest that there are many domains where the value function might be "close" to structured; that is, well-approximated by a linear combination of functions where each only refers to a small number of state variables. More precisely:

Definition 3.2 *A* factored (linear) value function *is a linear function over the basis $h_1, \ldots, h_k$, where each h_i is restricted to some subset of variables $\mathbf{C}_i$.* ∎

Value functions of this type have a long history in the area of multi-attribute utility theory [16].

4. Solving factored MDPs with linear programming

Factored value functions provide the key to performing efficient computations over the exponential-sized state spaces encountered in factored MDPs [17, 18, 12, 13, 22]. The key insight is that restricted domain functions (including our basis functions) allow for certain operations to be implemented very efficiently. These basic operations can be combined to create various practical algorithms. In this section, we briefly review how the linear programming approach outlined in (2) can be solved efficiently when the MDP is represented by a factored model [13, 22].

The first key operation is the computation of the backprojection g_i of a basis function h_i through the DBN τ_a of action a:

$$g_i^a(\mathbf{x}) = \sum_{\mathbf{x}'} P(\mathbf{x}' \mid \mathbf{x}, a) h_i(\mathbf{x}')$$

The expectation is a summation over an exponential number of future states. As shown in [17], this can be simplified substantially. Recall our assumption that the scope of each h_i is only a small subset of variables $\mathbf{C}_i$. Then, the scope of g_i is $\Gamma_a(\mathbf{C}_i) = \cup_{X'_j \in \mathbf{C}'_i} \mathit{Parents}_{\tau_a}(X'_j)$. Specifically, $g_i^a(\mathbf{x}) = \sum_{\mathbf{c}'_i} P(\mathbf{c}'_i \mid \mathbf{z}_i) h_i(\mathbf{c}'_i)$, where $\mathbf{z}_i$ is a value of $\Gamma_a(\mathbf{C}_i)$. Note that the cost of the computation depends linearly on $|\mathrm{Dom}(\Gamma_a(\mathbf{C}_i))|$, which depends on $\mathbf{C}_i$ (the scope of h_i) and on the complexity of the process dynamics.

Once the backprojections for the basis functions are precomputed, we can write the constraints for (2) as:

$$\rho + \sum_{j=1}^{k} w_j \left[h_j(\mathbf{x}) - g_j(\mathbf{x}) \right] \;\; \geq \;\; \sum_{j=1}^{r} R_j^a(\mathbf{x}) \, , \qquad (3)$$

for all $\mathbf{x}, a$. Although, we still have exponentially many constraints, the functions h_j, g_j and R_j^a all have restricted scope and thus depend only on small sets of variables. As demonstrated by Guestrin et al. [12, 13], these constraints can be represented compactly by an exponentially smaller set of constraints and LP variables using an algorithm resembling variable elimination. Similarly, Schuurmans and Patrascu [22] showed that these constraint sets can be built incrementally using an efficient constraint generation scheme.

Unfortunately, as discussed in Section 2, it is not currently known how to obtain strong formal guarantees of the quality of the policies obtained by approximate linear programming. Nonetheless, as demonstrated in [13, 22] these linear programs are very efficient and generate very reasonable policies. Currently, this approach seems to be the most appropriate for obtaining linear approximations for factored MDPs.

5. Factored E^3

Thus far, we have described a representation and an algorithm for planning in MDPs with exponentially large state spaces. Although, we have addressed our first concern about the standard MDP approach from Section 2, i.e., scaling up, we have not addressed the second concern: unknown model parameters. Clearly, estimating the unknown parameters in a DBN model of a factored MDP is easily achieved by a straightforward maximum likelihood or maximum a posteriori approach. However, the difficult issue is how to explore the environment efficiently to acquire the observations necessary to form sufficient estimates of the domain quantities.

Kearns and Koller [14] proposed the *Factored* E^3 algorithm to tackle this issue. *Factored* E^3 is a version of Kearns and Singh's E^3 algorithm [15] for factored MDPs. They simplify the problem by considering only the case where the DBN structure is known, and only the parameters of the MDP have to be estimated. We maintain this assumption in this paper.

The intent behind E^3 is to strike an effective balance between exploring uncertain regions of the state space to gather information, and exploiting knowledge of well-explored regions of the state space to optimize reward. Like E^3, *Factored* E^3 distinguishes "known" from "uncertain" states and employs two significant planning steps: calculating a policy that exploits the known part of the state space to maximize average reward (planning to exploit); and calculating a policy that leaves the known space to enter uncertain regions with high probability (planning to explore). If these planning algorithms are able to achieve provably near-optimal policies for their respective problems, *Factored* E^3 is guaranteed to approximate the optimal policy in reasonable time. Unfortunately, given a DBN representation of an MDP, it is an inherently hard problem to produce an approximately optimal policy [19, 20].

To implement a feasible version of *Factored* E^3 we employ an approximate planning strategy based on the linear programming approach outlined above. This linear program only gives an upper bound on the actual quality of the policies it produces. Thus a guaranteed decision between exploration and exploitation of the form required by [14, 15] cannot be achieved. Nevertheless, a reasonable decision can be made by comparing the objectives of the two linear programs and favoring exploration, unless its objective is too small.

6. Factored R_{max}

Recently, Brafman and Tennenholtz [6] proposed the R_{max} algorithm. As with E^3, this algorithm is guaranteed to yield a near-optimal policy for an MDP in polynomial time. However, the R_{max} algorithm is much simpler than E^3. Rather than explicitly deciding whether to exploit or explore, R_{max} makes the decision implicitly. The agent is continually exploiting *or* exploring and, in polynomial time, it is guaranteed to follow a near-optimal policy.

Following the generalization of E^3 to *Factored* E^3, one can also implement an approximate factored form of R_{max} using the linear programming approach. The weak upper bound that the linear program gives on the policy value ρ^π prevents the implicit explore or exploit theorem from being provable the same way as in [6]. However, *Factored* R_{max} can be directly implemented by using the linear program simply to provide an approximation to the required planning algorithm.

7. Algorithm-directed factored RL

Unfortunately, *Factored* E^3 and *Factored* R_{max} suffer from three strong limitations: First, the concept of *known-state* is quite limiting. A state is not considered during planning until it is known. On the other hand, the decision of when a state becomes known depends on a loose bound on the error of the probability estimate at that state. Thus, many visits are required before a state becomes relevant to the plan.

Second, both *Factored* E^3 and *Factored* R_{max} depend on the mixing time T of the asymptotically optimal policy. This quantity is central for deciding the number steps that the current (exploration) policy should be followed. Although, both E^3 and R_{max} can avoid knowing the horizon time by considering putative horizons $T = 1, 2, 3, \ldots$, thereby maintaining polynomial running time, this issue becomes much more difficult in the factored case. In particular, *Factored* E^3 requires the numeric value of the bound on the sub-optimality of the current policy for deciding when a particular T should be abandoned and the next should be tried. As discussed in Section 2, it is unlikely that such bounds would be available in practice, thus, this decision becomes particularly hard.

Third and most important, the theoretical guarantees of *Factored* E^3 and *Factored* R_{max} both depend on the existence of a planning algorithm for factored MDPs that can generate a compactly represented μ-optimal policy. Currently, to the best of our knowledge, there is no known algorithm that can achieve such a bound efficiently for general factored MDPs. Furthermore, we conjecture that, for sufficiently small μ, such an algorithm would require an exponential amount of space to represent the policy. Unfortunately, without a planning algorithm of this type, neither *Factored* E^3 nor *Factored* R_{max} maintains their strong theoretical guarantees. In addition, the stopping criterion for *Factored* E^3 depends fundamentally on the existence of such approximation algorithm. Thus, it becomes difficult to apply "Exploit or Explore" Lemma of [14].

In this section, we propose a more "pragmatic" approach to reinforcement learning for factored MDPs. Rather than learning the full factored model by assuming a strong approximation algorithm, we will adopt the linear programming approach for factored MDPs (Section 4) as our planning algorithm, and only learn the critical parts of the model needed for this algorithm. Thus, we call our approach *algorithm-directed factored reinforcement learning*.

7.1 Optimality criterion

In our approach, rather than attempting to guarantee a near-optimal approximation to the optimal value function, we will assume that the user is trying to obtain a linear approximation using the following linear programming approach:

$$
\begin{array}{ll}
\text{Variables:} & \rho, w_1, \ldots, w_k \;; \\
\text{Minimize:} & \rho \;; \\
\text{Subject to:} & \rho + \sum_{j=1}^{k} w_j h_j(\mathbf{x}_i) \geq \\
& \quad R(\mathbf{x}_i, a) + \sum_{\mathbf{x}'_l} P(\mathbf{x}'_l \mid \mathbf{x}_i, a) \sum_{j=1}^{k} w_j h_j(\mathbf{x}'_l); \\
& \qquad \forall \mathbf{x}_i \in \mathbf{X}, a \in \mathcal{A}; \\
& w_j \geq 0;\;\; j = 1, \ldots, k.
\end{array}
\tag{4}
$$

Furthermore, we will assume that basis functions are non-negative and bounded: $H_{max} \geq h_j(\mathbf{x}) \geq 0, \forall j, \mathbf{x}$. Without loss of generality and for simplicity of presentation, we will assume $H_{max} = 1$. Similarly, choosing positive weights does not necessarily make this LP different from the one in (2). In particular, if our basis function choice is appropriate (covering subset of variables), the solution of the two LPs are guaranteed to be equivalent.

Given a set of basis functions $h_1, \ldots, h_k$ and the true transition probabilities $P(\mathbf{x} \mid \mathbf{x}', a)$ and reward functions $R_i^a(\mathbf{x})$, one could apply the LP-based approximation algorithm from (4). This procedure would yield an approximate value function and an estimate of the average reward, given by the LP objective, which we denote by the *exact-model average reward estimate* ρ^*_{lp}. Note that this is not necessarily the average reward of the policy obtained by the LP approach, but it is the best LP objective achievable with the chosen set of basis functions given that the MDP model is known.

Unfortunately, in the context of reinforcement learning, the transition probabilities $P(\mathbf{x} \mid \mathbf{x}', a)$ and reward functions $R_i^a(\mathbf{x})$ are not known. For some estimate $\widehat{P}(\mathbf{x} \mid \mathbf{x}', a)$ and $\widehat{R}_i^a(\mathbf{x})$ to the model parameters, we define the *algorithm-directed error* as:

$$\left|\rho^*_{lp} - \widehat{\rho}\right| ;$$

where $\widehat{\rho}$ is the estimate of the average reward obtained by applying the LP-based approximation algorithm with the model parameters $\widehat{P}(\mathbf{x} \mid \mathbf{x}', a)$ and $\widehat{R}_i^a(\mathbf{x})$. In other words, it is the objective value of the linear program when the approximate model parameters are used in place of the true ones.

We can now define the optimality measure of our approach:

Definition 7.1 *For some* $\varepsilon > 0$*, define an* ε-optimal algorithm-directed approximation *as a model* $\widehat{P}(\mathbf{x} \mid \mathbf{x}', a)$ *and* $\widehat{R}_i^a(\mathbf{x})$ *which yields* $\left|\rho^*_{lp} - \widehat{\rho}\right| < \varepsilon$. ∎

In other words, given that approximate linear programming is to be used as the planning algorithm, we wish to estimate the parameters of the factored MDP so that the LP objective obtained with the approximate model is close to the one achieved with the true model.

7.2 Upper and lower bound LPs

In the first step of our algorithm, we propose an approach for generating upper and lower bounds on the exact-model average reward estimate ρ^*_{lp}; specifically $\overline{\rho}$ and $\underline{\rho}$ such that:

$$\overline{\rho} \geq \rho^*_{lp} \geq \underline{\rho} \;.$$

For this purpose, we will first need upper and lower bounds on the reward functions:

$$\overline{R}_i^a(\mathbf{x}) \geq \quad R_i^a(\mathbf{x}) \quad \geq \underline{R}_i^a(\mathbf{x}), \;\; \forall i, \mathbf{x}, a.$$

To maintain these bounds, we can start with the trivial bounds $[0, R_{max}]$ and update the values as the rewards are observed for various states.

Now consider the other part of the constraint, the backprojection of each basis h_i:

$$g_i^a(\mathbf{x}) \quad = \quad \sum_{\mathbf{x}'} P(\mathbf{x}' \mid \mathbf{x}, a) h_i(\mathbf{x}').$$

This operation is simply an expectation over a small set of state variables $\mathbf{x}'$ that yields a backprojected basis function whose scope is restricted to $\mathbf{B}_i^a = \Gamma_a(\mathbf{C}_i)$. We would like to maintain upper and lower bounds on the backprojections:

$$\overline{g}_i^a(\mathbf{b}_i^a) \geq g_i^a(\mathbf{b}_i^a) \geq \underline{g}_i^a(\mathbf{b}_i^a) \;\; ;$$

where $\mathbf{b}_i^a$ is an assignment to $\mathbf{B}_i^a$. We can estimate these upper and lower bounds by doing a simple empirical mean computation. Consider a transition visited along the exploration path $\langle \mathbf{x}, a, \mathbf{x}' \rangle$. If the starting state $\mathbf{x}$ for this transition is consistent with $\mathbf{b}_i^a$, then it can be used as a sample for the estimation of the backprojection g_i^a. More formally, let $\langle \mathbf{x}, a, \mathbf{x}' \rangle_{\mathbf{b}_i^a}$ be the set of $n_{\mathbf{b}_i^a} > 0$ such transitions.[3] Also, let the *mean backprojection* be given by:

$$\widehat{g}_i^a(\mathbf{b}_i^a) = \frac{1}{n_{\mathbf{b}_i^a}} \sum_{\langle \mathbf{x}, a, \mathbf{x}' \rangle_{\mathbf{b}_i^a}} h_i(\mathbf{x}') \;\; .$$

We can now define upper and lower bounds which will hold with high probability by:

$$\overline{g}_i^a(\mathbf{b}_i^a) = \widehat{g}_i^a(\mathbf{b}_i^a) + \varepsilon(n_{\mathbf{b}_i^a}); \;\; \text{and} \;\; \underline{g}_i^a(\mathbf{b}_i^a) = \widehat{g}_i^a(\mathbf{b}_i^a) - \varepsilon(n_{\mathbf{b}_i^a}) \, ; \tag{5}$$

where the error parameter $\varepsilon(n_{\mathbf{b}_i^a})$ is given by:

$$\varepsilon(n_{\mathbf{b}_i^a}) = \sqrt{\frac{1}{n_{\mathbf{b}_i^a}} \ln \frac{2k\,|\mathcal{A}|\,\mathbf{B}_{max}}{\gamma}} \, , \tag{6}$$

for any $\gamma > 0$, k basis functions and maximum backprojection scope size $\mathbf{B}_{max} = \max_{a,i} |\mathbf{B}_i^a|$. It is important to note

[3] When $n_{\mathbf{b}_i^a} = 0$ trivial upper and lower bounds should be used.

here that the sample complexity depends only *logarithmically* on the number of basis functions, number of actions and the size of the domain of backprojections; that is, it is only logarithmic in the size of the "problem description".

We can now obtain the desired upper and lower bounds:

Proposition 7.2 *Let ρ^*_{lp} be the average reward estimate obtained by solving the LP in (4) using the true model parameters $P(\mathbf{x} \mid \mathbf{x}', a)$ and $R^a_i(\mathbf{x})$. Furthermore, for any $\gamma > 0$, let $\overline{\rho}$ be the average reward estimate by solving the LP in (4) using the estimated upper bound backprojections $\overline{g}^a_i$ and rewards $\overline{R}^a_i(\mathbf{x})$; and let $\underline{\rho}$ be the estimate obtained using $\underline{g}^a_i$ and $\underline{R}^a_i(\mathbf{x})$. Then $\overline{\rho} \geq \rho^*_{lp} \geq \underline{\rho}$ with probability at least $1-\gamma$.*

Proof: First, we can prove using an application of Hoeffding's inequality that $\overline{g}^a_i(\mathbf{b}^a_i)$ and $\underline{g}^a_i(\mathbf{b}^a_i)$ as defined in Eq. (5) will be upper and lower bounds, respectively, on the true backprojection value $g^a_i(\mathbf{b}^a_i)$ for some assignment $\mathbf{b}^a_i$ with probability at least $1 - \frac{\gamma}{k|\mathcal{A}|\mathbf{B}_{max}}$. Thus, the upper and lower bounds will hold for all assignments with probability at least $1-\gamma$ by the union bound, since there are $k\,|\mathcal{A}|$ different backprojections with at most $\mathbf{B}_{max}$ assignments each.

The remainder of the proof assumes that the upper and lower bounds hold, thus arguments are true with high probability. Consider the lower bound case. Using $\underline{g}^a_i(\mathbf{x})$ in place of $g^a_i(\mathbf{x})$ makes the right side of the constraints in the LP in (4) smaller, and therefore the solution to the LP with the true model is still feasible. Thus, using the same set of basis weights as the ones in the solution to the true model, we can decrease the objective function, obtaining a smaller estimate of the average reward.

On the other hand, consider the upper bound case, where $\overline{g}^a_i(\mathbf{x})$ replaces $g^a_i(\mathbf{x})$. Here, the right side of the constraints in the LP in (4) becomes larger. Thus, the solution to the LP with the true model may no longer be feasible. Which means that the upper bound objective can be no smaller than ρ^*_{lp}.

Similar reasoning proves the influence of the upper and lower bounds on the reward. ∎

7.3 LP algorithm-directed exploration

By solving the upper and lower bound LPs in Section 7.2, we obtain a well-defined stopping criterion: For any $\varepsilon > 0$, if $\overline{\rho} - \underline{\rho} < \varepsilon$, we must have obtained an ε-optimal algorithm-directed approximation.

In addition to the stopping criterion, these upper and lower bound LPs give us information about which states need to be explored to make the bounds tighter. Note that at the solution point of the LP in (4), for a problem with k basis functions, only $k+1$ constraints will be active. These constraints correspond to $k+1$ critical state-action $(\mathbf{x}, a)$ pairs for which the inequality constraints become equality constraints. Now, consider the upper and lower bound LPs. Let us define $\overline{\textit{active}}$ and $\underline{\textit{active}}$ as the set of active state-action pairs in the upper and lower bound LPs, respectively. Intuitively, these are the critical states that need to be explored, as we now illustrate.

Consider an MDP model where some states-action pairs happen to have precise backprojection and reward values, but other state-action pairs are imprecise. More formally let:

$$\begin{aligned} \textit{Precise} &= \left\{ (\mathbf{x}, a) \;\middle|\; \begin{array}{l} \overline{g}^a_i(\mathbf{x}) = \underline{g}^a_i(\mathbf{x}) \;\forall\, i; \\ \text{and } \overline{R}^a_i(\mathbf{x}) = \underline{R}^a_i(\mathbf{x}),\ \forall i \end{array} \right\}; \\ \textit{Imprecise} &= \{(\mathbf{x}, a) \mid (\mathbf{x}, a) \notin \textit{Precise}\}. \end{aligned}$$

It is easy to show that if all of the *active* state-action pairs happen to be precise, then we have found an optimal algorithm-directed approximation:

Proposition 7.3 *Let $\overline{\textit{active}}$ be the set of active state-action pairs in the upper bound LP; and let Precise be the set of states with precise backprojections and rewards. If $\overline{\textit{active}} \subseteq$ Precise. Then $\overline{\rho} = \rho^*_{lp}$ with probability at least $1-\gamma$.*

Proof: First note that with probability at least $1-\gamma$, for $(\mathbf{x}, a) \in \textit{Precise}$, we have that:

$$\overline{g}^a_i(\mathbf{x}) = g^a_i(\mathbf{x}) = \underline{g}^a_i(\mathbf{x});\ \ \forall i.$$

Thus, the constraints corresponding to state-action pairs in *Precise* are equivalent in the upper bound LP, the lower bound LP, and the LP with the true transition probabilities.

In linear programs, one can remove all constraints other than the active constraints and the objective will remain unchanged. Furthermore, one can add new linear constraints to the linear program and the objective cannot decrease.

Now, consider the upper bound LP with active constraints given by state-action pairs in $\overline{\textit{active}}$. Let $\neg\overline{\textit{active}}$ denote the state-action pairs for the other constraints. Create a transformed LP by substituting the constraints for state-action pairs in $\neg\overline{\textit{active}}$ by their corresponding constraints in the LP with the true model. By the properties of linear programs, this substitution cannot decrease the objective. Thus, the new objective will be at least $\overline{\rho}$.

On the other hand, with probability at least $1-\gamma$, the constraints in the transformed LP will be the same as the ones in the LP with the true model, as all the states in $\overline{\textit{active}}$ are precise, by the assumption of the theorem. Thus, the objective of the transformed LP will be ρ^*_{lp}, which means that $\overline{\rho} \leq \rho^*_{lp}$.

Recall from Proposition 7.2 that $\overline{\rho} \geq \rho^*_{lp}$. Thus, we conclude that $\overline{\rho} = \rho^*_{lp}$. ∎

Proposition 7.3 suggests that one should direct exploration toward active LP states, because precise estimates on these states are sufficient to obtain an optimal approximation. Of course, the bounds will never be precise in reality, and one needs to use LP sensitivity analysis to determine error bars for $g^a_i(\mathbf{x})$ and $R^a_i(\mathbf{x})$ which ensure $\overline{\rho} - \underline{\rho} < \varepsilon$ given only nearly-precise estimates. Nevertheless, the intuition behind Proposition 7.3 provides sufficient motivation for our algorithm.

7.4 Algorithm

We can now describe our algorithm-directed reinforcement learning approach for factored MDPs. As in *Factored* E^3, we assume that the structure of the factored MDP is given, but the parameters are unknown.

During execution, we maintain upper and lower bounds on the backprojections g_i^a and on the reward function $R_i^a(\mathbf{x})$. As we visit state-action pairs, we update these functions.

After solving the upper and lower bound LPs, we can verify whether the stopping criterion has been reached. If not, we collect the active state-action pairs in the upper and lower bound LPs:

$$\mathit{ToVisit} = \overline{\mathit{active}} \bigcup \underline{\mathit{active}}.$$

As outlined in Section 7.3, visiting states in *ToVisit* is of fundamental importance in improving our LP approximation.

We can now define a factored MDP where the agent will receive rewards for exploring the states in *ToVisit*. In this MDP, the reward function will be decomposed according to the structure of the DBN. Specifically, for each $(\mathbf{x}^{(j)}, a^{(j)}) \in$ *ToVisit*, for each state variable X_i, we define a local reward function $R_{i,j}^{a^{(j)}}(\mathbf{x})$ with scope restricted to $\mathbf{W}_{i,j} = \mathit{Parents}_{\tau_{a^{(j)}}}(X_i')$. The reward function $R_{i,j}^{a^{(j)}}(\mathbf{x})$ takes value 1 if the state $\mathbf{x}$ is consistent with the assignment to the variables of $\mathbf{W}_{i,j}$ in $\mathbf{x}^{(j)}$ and 0 otherwise. In other words, the agent receives a reward for visiting the state that corresponds to the active constraints in the upper and lower bounds LPs. Note that, if a particular backprojection estimate $\widehat{g}_i^a(\mathbf{b}_i^a)$ is sufficiently precise, we should not necessarily add a reward for the state variables in $\mathbf{b}_i^a$ to avoid unnecessary exploration.

This construction yields a new, factored, exploration MDP. A policy for this MDP will guide the agent toward relevant unknown states. To obtain an exploration policy, we can again attempt to apply the LP approach in (4). However, the transition probabilities for this exploration MDP are not known precisely (as for the original MDP) so rather than applying the LP in (4) directly, we should apply the same upper and lower bound LPs strategy as in the original problem. If these have sufficiently tight upper and lower bounds, then we have a satisfactory exploration strategy. Otherwise, we can add the critical constraints of these new LPs to the *ToVisit* set and repeat the process. Note that after every recursion, states are added to the *ToVisit* set, but no states are deleted. Thus, if the set is the same after two successive recursions, then we know we cannot yet obtain a satisfactory exploration policy and have to explore by other means. This auxiliary exploration can be performed either with our current best approximation to the exploration policy or with a standard exploration strategy, such as balanced wandering.

8. Experimental results

To assess the quality of our algorithm we compare it with *Factored* E^3 and *Factored* R_{max} on three problems. Two of the problems consider a simple computer network administrator domain that has been investigated previously [12, 22]. The third problem we consider is a resource allocation problem.[4]

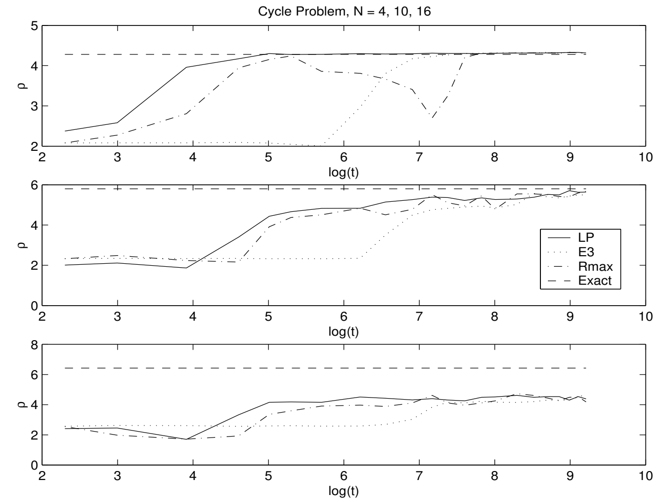

Figure 1. Policy quality on "cycle" network problem, N computers.

Figs. 1–3 report our results. For the network domains, each exploration strategy was run 20 times. The graphs plot the average reward value ρ_π obtained by the policies produced, π, as a function of exploration time t. Specifically, at given times, the current policy was extracted and separately evaluated to ascertain its asymptotic rate of reward ρ. (These evaluations were conducted by large auxiliary Monte Carlo runs.) Plotted also is the average reward obtained by an ideal LP planner that uses the exact MDP model. Figs. 1–2 show that the algorithm-directed technique exhibits superior behavior for smaller network problems. This is partially due to its switching from a "wander" mode to an "explore" as its statistics allow it to do so earlier than E^3 or R_{max}. However, the advantage becomes less predominant on larger problems ($N = 16$). Fig. 3 shows preliminary results on the resource problem (but using fewer runs). This problem has a large action space (corresponding to ways of allocating resources to problems) which causes the methods to take longer to converge. The results show good behavior for the algorithm-directed method. The error bars (not shown) are still loose, and more runs need to be completed. Nevertheless, the trends appear to be positive.

9. Conclusions

In this paper, we presented a new exploration/exploitation strategy for model-based reinforcement learning in factored

[4]The network administrator problem has binary state variables which correspond to the status of computers on a network. Machines crash with an increased probability if an ancestor is down. Actions are to reboot at most one machine per time step. Reward 1 is obtained for each machine that is up, plus an extra reward for a designated server machine. The transition probabilities for a machine depend on its status and the status of its direct ancestors in the directed network. In the resource allocation problem, there are tasks which become active spontaneously and have to be serviced by choosing resources from an available pool. Any resource can be applied to any task, and applying additional resources increases the probability of completing the task. However, once resources are depleted they are only regenerated at a slow random rate.

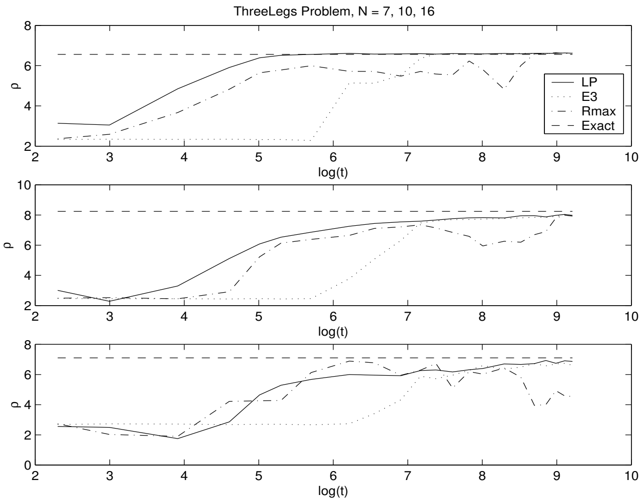

Figure 2. Policy quality on "3legs" network problem, N computers.

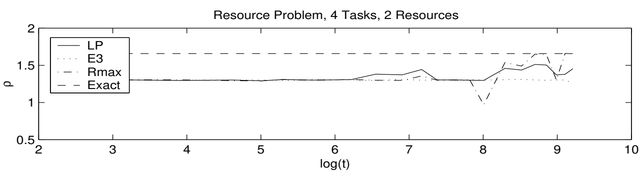

Figure 3. Resource allocation problem; 4 tasks, 2 resources.

MDPs. This strategy addresses strong practical limitations of the *Factored* E^3 algorithm [14]: First, rather than assuming the existence of an oracle that can perform near-optimal planning for factored MDPs, we tailor our exploration towards the particular planning algorithm used, approximate linear programming, which recently has been shown to scale to very large problems [13, 22]. By not targeting exploration toward every unknown region of the domain, nor toward unknown regions relevant to an oracle planner, our algorithm can more effectively direct exploration only toward the domain uncertainties that affect the results of the approximate planner directly. Furthermore, our strategy makes better use of the data collected. Rather than labeling states as *known* or *unknown* and only using transition probabilities for known states, as in *Factored* E^3, we incorporate the confidence intervals of the estimates of the model parameters onto the linear programming approach. These estimates have good sample efficiency, i.e., the number of samples needed for a given confidence level depends *logarithmically* in the number of actions, basis functions and the domain size of the backprojected basis functions. In contrast to *Factored* E^3, which needs to use hard thresholds to decide when a state becomes known, we can dynamically decide which confidence intervals need to be tightened. Our experimental results on benchmark problems demonstrate the effectiveness of this algorithm-directed exploration strategy. We believe that this new practical approach to model-based reinforcement learning will further the applicability of factored MDP models and algorithms to large-scale real-world problems.

Acknowledgments We are grateful to D. Koller, R. Parr and anonymous reviewers for many useful suggestions. This work was supported by the MURI program N00014-00-1-0637, NSERC, MITACS and CITO. C. Guestrin was supported by a Siebel Scholarship.

References

[1] D. Berry and B. Fristedt. *Bandit Problems: Sequential Allocation of Experiments*. Chapman and Hall, London, 1985.

[2] C. Boutilier, T. Dean, and S. Hanks. Decision theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 11:1 – 94, 1999.

[3] C. Boutilier, R. Dearden, and M. Goldszmidt. Exploiting structure in policy construction. In *IJCAI*, pages 1104–1111, 1995.

[4] J. Boyan. Least-squares temporal difference learning. 2000.

[5] S. Bradtke and A. Barto. Linear least-squares algorithms for temporal difference learning. *Mach. Learn.*, 2(1):33–58, 1996.

[6] R. Brafman and M. Tennenholtz. R-MAX - A general polynomial time algorithm for near-optimal reinforcement learning. In *Proc. IJCAI*, pages 953–958, 2001.

[7] T. Brown. Switch packet arbitration via queue-learning. In *Proc. NIPS-14*, 2001.

[8] R. Crites and A Barto. Improving elevator performance using reinforcement learning. In *NIPS-9*, pages 1017–1023, 1996.

[9] D.P. de Farias and B. Van Roy. The linear programming approach to approximate dynamic programming. 2001.

[10] Thomas Dean and Keiji Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5(3):142–150, 1989.

[11] R. Dearden, N. Friedman, and S. Russell. Bayesian Q-learning. In *Proc. AAAI*, pages 761–768, 1998.

[12] C. Guestrin, D. Koller, and R. Parr. Max-norm projections for factored MDPs. In *Proc. IJCAI*, pages 673–682, 2001.

[13] C. Guestrin, D. Koller, and R. Parr. Multiagent planning with factored MDPs. In *Proc. NIPS-14*, 2001.

[14] M. Kearns and D. Koller. Efficient reinforcement learning in factored MDPs. In *Proc. IJCAI*, 1999.

[15] M. Kearns and S. Singh. Near-optimal reinforcement learning in polynomial time. In *Proc. ICML*, 1998.

[16] R. Keeney and H. Raiffa. *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*. Wiley, New York, 1976.

[17] D. Koller and R. Parr. Computing factored value functions for policies in structured MDPs. In *Proc. IJCAI*, 1999.

[18] D. Koller and R. Parr. Policy iteration for factored MDPs. In *Proc. UAI*, 2000.

[19] C. Lusena, J. Goldsmith, and M. Mundhenk. Nonapproximability results for partially observable Markov decision processes. *JAIR*, 14:83–103, 2001.

[20] M. Mundhenk, J. Goldsmith, C. Lusena, and E. Allender. Complexity of finite-horizon Markov decision processes. *JACM*, 47(4):681–720, 2000.

[21] M. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, New York, 1994.

[22] D. Schuurmans and R. Patrascu. Direct value-approximation for factored MDPs. In *Proc. NIPS-14*, 2001.

[23] P. Schweitzer and A. Seidmann. Generalized polynomial approximations in Markovian decision processes. *J. Math. Analysis and Appl.*, 110:568 – 582, 1985.

[24] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[25] B. Van Roy. *Learning and Value Function Approximation in Complex Decision Processes*. PhD thesis, M.I.T, 1998.