Solving Structured Continuous-Time Markov Decision Processes

Kin Fai Kan and Christian R. Shelton
Department of Computer Science and Engineering
University of California, Riverside, CA 92521, USA
{kkan,cshelton}@cs.ucr.edu

Abstract

We present an approach to solving structured continuous-time Markov decision processes. We approximate the the optimal value function by a compact linear form, resulting in a linear program. The main difficulty arises from the number of constraints that grow exponentially with the number of variables in the system. We exploit the representation of continuous-time Bayesian networks (CTBNs) to describe the Markov process. We show that by exploiting the structure of the CTBN, we can reduce the growth in the number of constraints to be polynomial. We provide theoretic bounds on the quality of the approximation and experimental results on problems of different sizes, demonstrating the scalability and fidelity of our approach.

1 Introduction

Many real-world systems involve in continuous time. Networks (both traditional computer networks and sensor networks) are asynchronous with no natural clock to regulate their behavior. These and other systems, such as mobile robots (Ng, Pfeffer, & Dearden 2005), biological processes (El-Hay *et al.* 2006), computer users (Nodelman & Horvitz 2003), and transportation (Gopalratnam, Kautz, & Weld 2005), have all been modeled using continuous-time Bayesian networks (CTBNs).

A CTBN (Nodelman, Shelton, & Koller 2002) allows for the structured (and therefore compact) representation of the dynamics of a multi-variate stochastic system. Unlike a dynamic Bayesian network (DBN) (Dean & Kanazawa 1989) which represents the values of the system at discrete time points, a CTBN represents the distribution over complete trajectories of the system. This means that it makes assumptions about the independencies of the immediate and local interactions of the variables, not the independences about samples taken at particular time intervals. Thus the structure of the network is not dependent on the time-slice width.

Algorithms for learning (Nodelman, Shelton, & Koller 2005) and inference (Nodelman, Koller, & Shelton 2005) in CTBNs have been previously established. Yet, currently there are no methods for control or planning in such systems that exploit the structure to reduce computation time and memory. In this work we combine classic work in the field of continuous-time Markov decision processes (MDPs) with work from discrete-time structured MDPs (Guestrin, Koller, & Parr 2001; Schuurmans & Patrascu 2002; de Farias & Van Roy 2003) to produce results in continuous-time structured MDP planning.

We start by setting out the problem for flat (unstructured) continuous-time MDPs and providing performance bounds. We then describe, briefly, the CTBN model and how it helps us reduce the exponentially-large number of constraints. We conclude with experimental results.

2 Continuous-Time MDPs

A Markov Decision Process (MDP) consists of four components: a state space $\mathcal{X}$, an action space $\mathcal{A}$, a transition model, and a reward model. A policy is a mapping from $\mathcal{X}$ to $\mathcal{A}$. We assume that the MDP has an infinite horizon and future rewards are discounted exponentially with a discount factor, β. The goal is to find a policy that maximize the total discounted reward in every state.

For a continuous-time MDP, the transition model is a set of intensity matrices, one for each action a:

$$Q_a = \begin{bmatrix} -q_a(x_1) & q_a(x_1,x_2) & \cdots & q_a(x_1,x_N) \\ q_a(x_2,x_1) & -q_a(x_2) & \cdots & q_a(x_2,x_N) \\ \vdots & \vdots & \ddots & \vdots \\ q_a(x_N,x_1) & q_a(x_N,x_2) & \cdots & -q_a(x_N) \end{bmatrix}$$

where $q_a(x_i,x_j)$ is the intensity of transitioning from state x_i to state x_j by performing action a and $q_a(x_i) = \sum_{j\neq i} q_a(x_i,x_j)$.

Given Q_a, we can describe the transient behavior of $X(t)$ as follows. If $X(0)=x$ then the system stays in state x for an amount of time exponentially distributed with parameter $q_a(x)$.[1] Upon transitioning, X shifts to state x' with probability $\theta_a(x,x') = q_a(x,x')/q_a(x)$.

The reward model is a function $r(x,a)$ that represents the reward rate while taking action a in state x. There may also be a lump sum reward for transitioning from one state to another. For this paper, we ignore the lump sum (assume it is equal to zero). The rewards are discounted exponentially in time at a rate of β: a reward of 1 at time t is equivalent to a reward of $e^{\beta\delta}$ at time $t+\delta$.

[1] The mean of the exponential distribution is the reciprocal of its parameter.

The value function $V^{\pi}(x)$ is defined as the expected total discounted return of executing π from state x.

$$V^{\pi}(x) = \frac{r(x,\pi(x))}{\beta + q_{\pi(x)}(x)} + \sum_{x' \neq x} \frac{q_{\pi(x)}(x,x')}{\beta + q_{\pi(x)}(x)} V^{\pi}(x') \quad (1)$$

Equation (1) is the Bellman equation for a continuous-time MDP. The optimal value function V^* is the best possible expected total discounted rewards that can be attained by any policy. It satisfies a slightly different version of Bellman equation:

$$V^*(x) = \max_a \frac{r(x,a)}{\beta + q_a(x)} + \sum_{x' \neq x} \frac{q_a(x,x')}{\beta + q_a(x)} V^*(x') \; . \quad (2)$$

Given V^*, we can obtain the optimal policy easily. For any state x, we choose the action a that maximizes the right hand side of Equation (2).

2.1 Linear Programs for Continuous-Time MDPs

We can solve a continuous-time MDP by computing the optimal value function V^* and extracting the optimal policy from V^* afterward. It is well known that the problem of computing V^* can be formulated as a linear program:

$$\begin{aligned} \text{min. } & \sum_x V(x) \\ \text{s.t. } & V(x) \geq \frac{r(x,a)}{\beta + q_a(x)} + \sum_{x' \neq x} \frac{q_a(x,x')}{\beta + q_a(x)} V(x') \quad \forall x, a \, . \end{aligned}$$

We call this linear program the *exact LP*. In many domains, the state space is quite large, and we need to approximate the value function $V(x)$ with a more compact one. A simple scheme is to approximate $V(x)$ by a linear combination of k fixed basis functions:

$$V(x) = \sum_{i=1}^{k} w_i h_i(x) \quad (3)$$

where h_i and w_i are the ith basis function and its weight. Substituting Equation (3) into the exact LP, we obtain the *approximate LP*:

$$\begin{aligned} \text{min. } \sum_{i=1}^{k} w_i \alpha_i \quad \text{s.t. } & \sum_{i=1}^{k} w_i h_i(x) \geq \frac{r(x,a)}{\beta + q_a(x)} \qquad \forall x, a \\ & + \sum_{i=1}^{k} \sum_{x' \neq x} \frac{w_i q_a(x,x')}{\beta + q_a(x)} h_i(x') \end{aligned}$$

where $\alpha_i = \sum_x h_i(x)$.

The idea of using linear value function was proposed initially by Bellman & Dreyfus (1959) and has been further explored by Tsitsiklis & Van Roy (1996) and Koller & Parr (1999). de Farias & Van Roy (2003) have developed error bounds that provide performance guarantees for approximate LP with linear value functions in the discrete-time setting. Following their ideas, it is straightforward to prove analogous error bounds in the continuous time setting.

Theorem 1. *Let $e = [1, \ldots, 1]$ be in the span of the basis functions and N be the size of the state space. Then if $\tilde{w}$ is an optimal solution to the approximate LP,*

$$\|V^* - H\tilde{w}\|_{1,1/N} \leq 2\left(1 + \frac{q_{max}}{\beta}\right) \min_w \|V^* - Hw\|_\infty$$

where Hw represents the function $\sum_{i=1}^{k} w_i h_i(\cdot)$, $\|\cdot\|_{1,1/N}$ is the $\mathcal{L}_1$-norm weighted by $1/N$, and $q_{max} = \max_{a,x} q_a(x)$.

Theorem 2. *Let $\tilde{w}$ be a solution of the approximate LP. Then, for any $v \in \Re^k$ such that $(Hv)(x) > 0$ and $\max_{a \in \mathcal{A}} \sum_{i=1}^{k} v_i \sum_{x' \neq x} \frac{q_a(x,x')}{\beta + q_a x} \leq (Hv)(x)$ for all $x \in \mathcal{X}$,*

$$\|V^* - H\tilde{w}\|_1 \leq \frac{2eHv}{1 - \gamma_v} \min_w \|V^* - Hw\|_{\infty,1/Hv}$$

where $\gamma_v = \max_x \frac{\max_{a \in \mathcal{A}} \sum_{i=1}^{k} v_i \sum_{x' \neq x} \frac{q_a(x,x')}{\beta + q_a x}}{(Hv)(x)}$, $e = [1, \ldots, 1]$, and $\| \cdot \|_{\infty,1/Hv}$ is the $\mathcal{L}_\infty$-norm weighted by $1/Hv$.

Theorems 1 and 2 bound the difference between the objectives of the exact LP and the approximate LP. Theorem 1 suggests that the performance of the approximate LP depends on the ratio of the maximum state transition rate and the reward discount rate. The bound in Theorem 1 may be too loose for large MDPs because of the $\mathcal{L}_\infty$-norm. Theorem 2 remedies the situation by providing a refined bound which is expressed in terms of a weighted $\mathcal{L}_\infty$-norm.

3 Solving Structured Continuous-Time MDPs

Although the approximate LP reduces the number of optimization variables from the size of the state space $|\mathcal{X}|$ to a constant (the number of basis functions), the number of constraints remain the same ($|\mathcal{X}| \cdot |\mathcal{A}|$). To scale up to large state spaces, it is necessary to exploit the structure of the continuous-time MDP. As in Guestrin, Koller, & Parr (2001) and Schuurmans & Patrascu (2002), we make three standard assumptions: (1) factored continuous-time dynamics, (2) a factored reward function, and (3) the value basis functions each depend on only a small subset of the state variables. These three assumptions allow us to rewrite the approximate LP in a compact form. The resulting LP is called the *factored LP*.

3.1 Structured Continuous-Time MDPs

Factored MDPs (Dean & Kanazawa 1989) are one approach to represent large, structured MDPs compactly in the discrete-time setting. In the same spirit, we introduce structured continuous-time MDPs. In this framework, 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 $Dom(X_i)$. A state $\mathbf{x}$ defines a value $x_i \in Dom(X_i)$ for each variable X_i. We define for each action a, a separate state transition model using a continuous-time Bayesian network . Furthermore, 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 briefly review the key features of CTBNs (Nodelman, Shelton, & Koller 2002). Continuous-time Bayesian networks describe the joint dynamics of the state variables in a structured homogeneous Markov processes. A CTBN is a directed (possibly cyclic) dependency graph $\mathcal{G}$ over a set of n variables. Let $\mathbf{U}_X$ be the parents of X in $\mathcal{G}$. Each variable X is a Markov process and is associated with a conditional intensity matrix, $Q^{X|\mathbf{U}_X}$: a set of homogeneous intensity matrices $Q^{X|\mathbf{u}}$, one for each instantiation of values $\mathbf{u}$ to $\mathbf{U}_X$. Given $\mathbf{U}_X = \mathbf{u}$, the transition dynamics of X are determined by $Q^{X|\mathbf{u}}$ and are independent of all other variables. A basic assumption of CTBNs is that, as time is continuous, two distinct variables cannot transition at the exact same instant. Thus, in the joint intensity matrix, all intensities that correspond to two simultaneous changes are zero. Given a CTBN, we can describe the transient behavior of $\mathbf{X}(t)$ as follows. If $\mathbf{X}(0) = \mathbf{x}$, then the system stays in state $\mathbf{x}$ for an amount of time exponentially distributed with parameter $q(\mathbf{x})$:

$$\begin{aligned} q(\mathbf{x}) &= \sum_{\mathbf{x}'} q(\mathbf{x}, \mathbf{x}') \\ &= \sum_{i=1}^{n} \sum_{x_i' \neq x_i} q^{X_i|\mathbf{u}_i}(x_i, x_i') \end{aligned} \tag{4}$$

where x_i' is any value in $Dom(X_i)$, x_i and $\mathbf{u}_i$ are the values assigned to X_i and $\mathbf{U}_i$ in $\mathbf{x}$, and $q^{X_i|\mathbf{u}_i}(\cdot,\cdot)$ are intensity values in $Q^{X_i|\mathbf{u}_i}$. Upon transitioning, $\mathbf{X}$ changes X_j from x_j to some value $x_j' \neq x_j$ with probability $q^{X_j|\mathbf{u}_j}(x_j, x_j')/q(\mathbf{x})$.

3.2 Factored LP

We want to construct a compact LP that avoids an explicit enumeration of the constraints for the exponentially many states. To begin, for every constraint in the approximate LP, we multiply both sides of the inequality by $\beta + q_a(\mathbf{x})$ and move everything on the left hand side to the right hand side. We obtain the following inequality.

$$\begin{aligned} 0 \geq r(\mathbf{x}, a) + \sum_{i=0}^{k} w_i \Big[&- \beta h_i(\mathbf{x}) - q_a(\mathbf{x}) h_i(\mathbf{x}) \\ &+ \sum_{\mathbf{x}' \neq \mathbf{x}} q_a(\mathbf{x}, \mathbf{x}') h_i(\mathbf{x}') \Big] \quad \forall \mathbf{x}, a \ . \end{aligned} \tag{5}$$

The key is to show that the right hand side of Equation (5) has a factored representation. Recall that in CTBNs, only one single variable can change at any single instant and, given the values of the parent variables in CTBN, the rate that a variable changes is independent of all the other variables. We can rewrite Equation (5) in terms of the conditional intensities. We use $\mathbf{x}_{[j \leftarrow z]}$ to denote the state assignment that is the same as $\mathbf{x}$ except that X_j equals z. sc_i is the set of variables in the scope of $h(i)$. Rewriting the left hand side (except for $r(\mathbf{x}, a)$) we get

$$\begin{aligned} &\sum_{i=0}^{k} w_i \Big[- \beta h_i(\mathbf{x}) - \sum_{X_j \in \text{sc}_i} \sum_{x_j' \neq x_j} q_a^{X_j|\mathbf{u}_j}(x_j, x_j') h_i(\mathbf{x}) \\ &\qquad - \sum_{X_j \notin \text{sc}_i} \sum_{x_j' \neq x_j} q_a^{X_j|\mathbf{u}_j}(x_j, x_j') h_i(\mathbf{x}) \\ &\qquad + \sum_{X_j \in \text{sc}_i} \sum_{x_j' \neq x_j} q_a^{X_j|\mathbf{u}_j}(x_j, x_j') h_i(\mathbf{x}_{[j \leftarrow x_j']}) \\ &\qquad + \sum_{X_j \notin \text{sc}_i} \sum_{x_j' \neq x_j} q_a^{X_j|\mathbf{u}_j}(x_j, x_j') h_i(\mathbf{x}) \Big] \\ &= \sum_{i=0}^{k} w_i \Big[- \beta h_i(\mathbf{x}) - \sum_{X_j \in \text{sc}_i} \sum_{x_j' \neq x_j} q_a^{X_j|\mathbf{u}_j}(x_j, x_j') h_i(\mathbf{x}) \\ &\qquad + \sum_{X_j \in \text{sc}_i} \sum_{x_j' \neq x_j} q_a^{X_j|\mathbf{u}_j}(x_j, x_j') h_i(\mathbf{x}_{[j \leftarrow x_j']}) \Big] \ . \end{aligned}$$

Using $\max$ to combine the linear constraints for the same action a, we are left with $|A|$ non-linear constraints:

$$\begin{aligned} 0 \geq \max_{\mathbf{x}} \Big\{ r(\mathbf{x}, a) + \sum_{i=0}^{k} w_i \Big[&- \beta h_i(\mathbf{x}) \\ - \sum_{X_j \in \text{sc}_i} \sum_{x_j' \neq x_j} &q_a^{X_j|\mathbf{u}_j}(x_j, x_j') h_i(\mathbf{x}) \\ + \sum_{X_j \in \text{sc}_i} \sum_{x_j' \neq x_j} &q_a^{X_j|\mathbf{u}_j}(x_j, x_j') h_i(\mathbf{x}_{[j \leftarrow x_j']}) \Big] \Big\} \quad \forall a \ . \end{aligned} \tag{6}$$

Note that in Equation (6), for each i, the terms within the square brackets depend only on the variables in the scope of h_i and their parents. We assume that the optimal value function can be approximated well by a linear combination of some fixed basis functions, each of which depends on a small number of variables. The reward function $r(\mathbf{x}, a)$ is, by assumption, factored. Thus, we can use the technique proposed in Guestrin, Koller, & Parr (2001) to rewrite the $|\mathcal{A}|$ non-linear constraints into a small number of linear constraints. For example, if each reward function or value basis function depends on one variable, the number of linear constraints generated is exponential in the degree of the variables in the CTBNs rather than the complete set of variables. The technique is basically the same as variable elimination in Bayesian Networks. Instead of computing $\max$ by enumerating all possible assignment of $\mathbf{X}$, it maximizes one variable at a time. We illustrate the main idea using a simple example.

Example 1. *Consider the following constraint*

$$0 \geq \max_{x_1, x_2, x_3, x_4} f_1(x_1, x_2) + f_2(x_2, x_3) + f_3(x_3, x_4) + f_4(x_1, x_4).$$

We first maximize the left hand side w.r.t. variable x_4. We group terms that depend on x_4 together.

$$0 \geq \max_{x_1, x_2, x_3} f_1(x_1, x_2) + f_2(x_2, x_3) + \max_{x_4} f_3(x_3, x_4) + f_4(x_1, x_4)$$

Then, we introduce a new term $e_1(x_1, x_3)$ to represent the result of the maximization over x_4 and rewrite the above constraint as a non-linear constraint plus a linear constraint.

$$0 \geq \max_{x_1,x_2,x_3} f_1(x_1, x_2) + f_2(x_2, x_3) + e_1(x_1, x_3)$$
$$e_1(x_1, x_3) \geq f_3(x_3, x_4) + f_4(x_1, x_4), \quad \forall x_1, x_3, x_4 \ .$$

By repeating the above procedure for x_1, x_2, and x_3, we transform the single non-linear constraint into a number of linear constraints.

The factored linear value function not only allows us to rewrite the constraints in a compact form, but also makes computing α_i, which appears in the objective of the approximate LP, efficient. Recall that $\alpha_i = \sum_{\mathbf{x}} h_i(\mathbf{x})$. Since h_i is factored, we just need to sum over the domain of h_i instead of the whole state space. Also, it is easy to extract the policy from the factored linear value function. The major step is to compute the term $\sum_{\mathbf{x}' \neq \mathbf{x}} \frac{q_a(\mathbf{x},\mathbf{x}')}{\beta + q_a(\mathbf{x})} h_i(\mathbf{x}')$ in Equation (2). However, since it is assumed in CTBNs that all intensities corresponding to simultaneous changes are zero, the time complexity of the summation is linear in the number of variables in the scope of h_i.

3.3 Uniformization

There is an alternative approach to obtain the same factored LP. Instead of tackling the continuous-time MDP directly, we can convert the continuous-time MDP into a discrete-time MDP using the uniformization technique (Puterman 1994) and use the approximate LP for the discrete-time MDP. We choose a constant κ satisfying

$$\sup_{x \in \mathcal{X}} q(x) \leq \kappa < \infty$$

and define the reward function ($\tilde{r}$) and the transition model ($\tilde{T}$) as follows

$$\begin{aligned} \tilde{r}(x,a) &= r(s,a)\frac{\beta + q_a(x)}{\beta + \kappa}; \\ \tilde{T}_a(x,x') &= \begin{cases} \kappa - q_a(x) & x' = x, \\ q_a(x,x') & x' \neq x. \end{cases} \end{aligned}$$

The uniformization may be viewed as an equivalent process, in which the system state is observed at random times which are exponentially distributed with parameter κ. The reward function is adjusted to ensure that the transformed system has the same expected total discounted rewards. We write down the Bellman equation for this transformed model. After some algebraic manipulation, we obtain the same result as Equation 2.

We do wish to point out that this conversion method is *not* the same as constructing the discrete-time MDP corresponding to any fixed sampling rate. Uniformization constructs a discrete-time MDP in which each time step corresponds to a single transition in the continuous-time domain. Thus, steps in the new discrete Markov decision process do not correspond to fixed amounts of time in the original process.

By contrast, if we were to construct a DBN that corresponded to sampling the CTBN at a fixed rate, each DBN variable at time $t+1$ would have as parents all variables at time t that were its ancestors in the CTBN. Furthermore, barring cycles, it would also have the same variables at time $t+1$ as parents as well. Thus, even simple sparse CTBNs (such as the ring network in Figure 1) can lead to fully connected (with both inter- and intra-time-slice arcs) DBNs, if we sample them at a uniform rate.

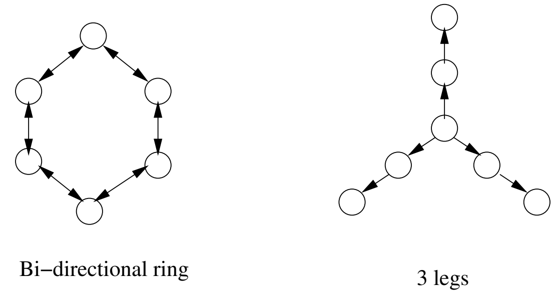


Figure 1: Network topologies

4 Experimental results

We tested the factored LP approach on the continuous time version of the SysAdmin problem from (Guestrin, Koller, & Parr 2001). A system administrator has to maintain a directed network of computers. The network topologies we considered are shown in Figure 1.

The state of the computer network is represented by a number of indicator variables, one for each computer indicating whether it is working or faulty. Computers fail randomly. The system administrator can choose to reboot one of the computers or do nothing. We assume that the amount of time a computer is working until it fails follows an exponential distribution with an intensity of 0.5. A faulty computer causes neighboring computers to fail more quickly. The intensity with which a computer fails is defined as $0.5 + 0.5 \times$ number of faulty neighbors. If the system administrator reboots a faulty computer, the amount of time before the faulty computer returns to be working follows an exponential distribution with parameter 2. A computer receives a reward of 1 per unit time when working. In ring networks, one computer is designated to receive a reward of 2 per unit time to introduce asymmetry. Faulty computers get contribute zero reward. The rewards are discounted exponentially in time at a rate of $\beta = 0.1$. The value basis functions we used are independent indicators for each computer, with value 1 if it is working and 0 otherwise, and a constant basis whose value is 1 for all states.

We implemented a program to generate the factored LP in Matlab and used the GNU Linear Programming Kit (GLPK) to solve the resulting linear programs. GLPK is intended for solving large-scale linear programming, mixed linear programming, other related problems. It is a simplex-based solver and it is able to handle problems with up to 100,000 constraints. GLPK is about 10-100 times slower than CPLEX 8.0 dual simplex on very large-scale problems. The experiments were performed on a PC with a 3GHz Pentium 4 processor.

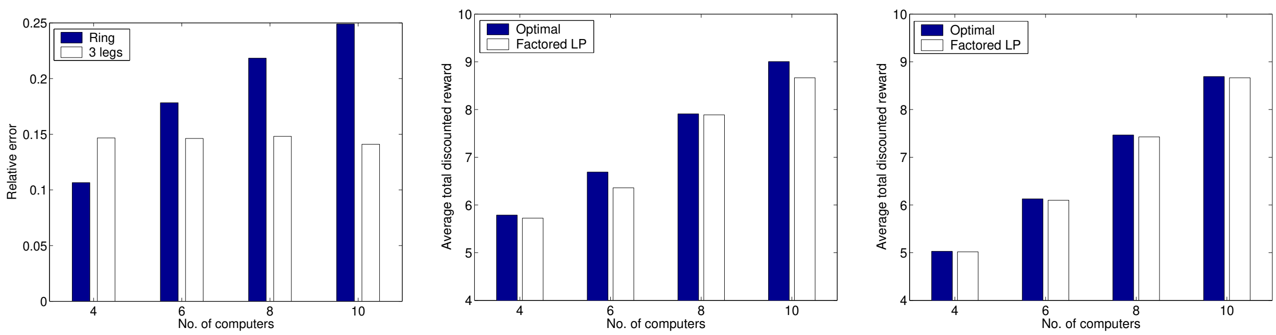
No. of computers	4	6	8	10
Time (ExactLP)	0.05	0.26	4.69	123.13
Time (FactoredLP)	0.12	0.34	1.00	2.40
No. of variables	249	685	1313	2133
No. of constraints	324	1014	2024	3354

(a) Ring network

No. of computers	4	6	8	10
Time (ExactLP)	0.08	0.54	5.31	148.47
Time (FactoredLP)	0.06	0.14	0.20	0.21
No. of variables	119	251	431	659
No. of constraints	154	328	566	868

(b) 3-Leg network

Table 1: Running time in seconds and the size of the factored LP (small networks)



(a) (b) (c)

Figure 2: Results for small networks: (a) Approximation quality, (b) Ring network performance, and (c) 3-leg network performance.

We evaluated the efficiency and the approximation quality of the factored LP approach on problems of various sizes. For small problems, we solved the exact LP. We compare the time taken to solve the exact LP and the factored LP. Table 1 shows the time taken to solve the LPs and the numbers of optimization variables and constraints generated by the factored LP. As the number of computers increases, the size of the factored LP increases and the running time of the factored LP increases. However, the running time of the factored LP increases much more slowly than that of the exact LP. Each computer in a ring network has two parents while each computer in a 3-leg network has only one parent (except the root computer in the center which has none). Not surprisingly, the factored LP generated more variables and constraints for a ring network than a 3-leg network of the same size.

We evaluated the error in our approximate value function relative to the true optimal value function: $\frac{\|Hw-V^*\|_\infty}{\|V^*\|_\infty}$. We also compared the quality of the policy extracted from our approximate value function with the true optimal policy using Monte Carlo simulations. In each simulation, we used 1000 trials and each trial lasted for 100 time units. Figure 2(a) shows the relative error of the factored LP approach. Figure 2(b–c) shows the rewards obtained by the optimal policy and the factored LP policy in the simulation. We observe that for the ring networks, the relative error of the factored LP increases as the number of computers increases. For the 3-leg networks, the relative error of the factored LP remains roughly the same. However, the policy extracted from the factored LP consistently performed almost as well as the optimal policy did in our simulation.

For large problems, we can no longer compute the true

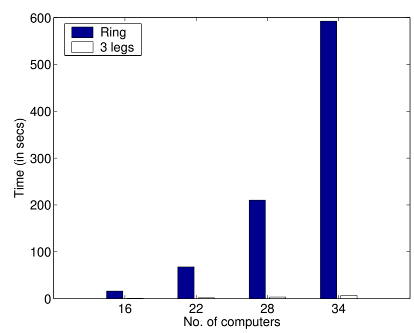


Figure 3: Running time of the factored LP (large networks)

optimal value function. Figure 3 shows how the time taken to solve the factored LP scales with the number of computers in the large ring network (the running time for the 3-leg network is negligible). Table 2 gives the numbers of variables and constraints generated. For comparison, the exact LP for a 34-node network has approximately 16 billion variables and 500 billion constraints.

We use an upper bound on the Bellman error to evaluate the approximation quality of the factored LP. In the discrete-time setting, the Bellman operator $\mathcal{T}^*$ is $\mathcal{T}^* = \max_a [R(x,a) + \gamma \sum_{x'} P(x'|x,a)V(x')]$, where $R(s,a)$ is the reward function, γ is the discount factor, and $P(x'|x,a)$ is the transition probability. The Bellman error, defined as $BellmanErr(V) = \|V - \mathcal{T}^*V\|_\infty$, can be used to provide a bound: $\|V^* - Hw\|_\infty \leq \frac{BellmanErr(Hw)}{1-\gamma}$ (Williams & Baird 1993). The Bellman error can in turn be bounded by $\min_a \|V - \mathcal{T}^a V\|_\infty$ where $\mathcal{T}^a V$ is just like $\mathcal{T}^*$ but with

No. of computers	16	22	28	34
R: No. of variables	5745	11062	18153	26914
R: No. of constraints	9264	18054	29724	44274
L: No. of variables	1631	3035	4871	7139
L: No. of constraints	2158	4024	6466	9484

Table 2: The size of the factored LP (large networks). "R:" indicates results for the Ring network while "L:" indicates results for the 3-leg network.

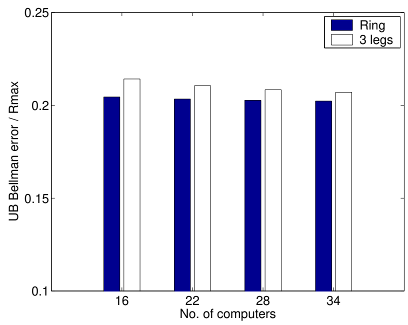


Figure 4: Approximation quality of the factored LP (large networks)

$\max_a$ replaced by a fixed action a (Schuurmans & Patrascu 2002). To apply this bound in our case, we convert the continuous time MDP into a discrete time MDP using the uniformization technique. Figure 4 shows the upper bound on the Bellman error (denoted by UB Bellman error) divided by the maximum total reward (denoted by R_{max}) for increasing number of computers. The ratio UB Bellman error$/R_{max}$ seems to decrease very slightly (but consistently) with the size of the problem.

5 Conclusions and Extensions

In this paper, we present an approach to solving structured continuous-time MDPs. Our method is based on solving a single linear program that approximates the best linear fit to the optimal value function. We exploit the representation of continuous-time Bayesian networks to describe the Markov process. We show that by exploiting the structure of the CTBN, we can rewrite the linear program in a compact form and hence make it easy to solve. We tested our method on a simplified version of a network maintenance task. Our results demonstrate that our method can produce high quality approximate value functions (and policies) and it scales effectively to large problems.

We would like to extend this work in a number of ways in future research. We think the linear program can be extended and improved in two ways. First, adding "lump sum" rewards should be possible by modifying the linear program. However, we still need additional work to insure that the resulting LP can be factored. Second, our constraint generation is not as fast as it could be. Techniques from Schuurmans & Patrascu (2002) should also apply here to speed up generation.

Much as Guestrin, Koller, & Parr (2001) can perform policy iteration in DBNs, we think the techniques in this paper can also be used to perform policy iteration in CTBNs. The basic methods for constraint representation remain the same. The method is more involved and it would be interesting to explore whether it results in better policies.

Finally, we would like to explore the possibility of planning in partially-observable continuous-time models. Younes & Simmons (2004) and Younes (2005) explore using phase-type distributions and algebraic decision diagrams (ADD) to model semi-Markov decision processes (a simple kind of partial observability). Their state space is not factored like that of a CTBN (the transition model uses an ADD) and they do not approximate the value function. However, it is worth noting that the phase-type distributions used by Younes & Simmons are similar to those that result from partially observed CTBNs (Nodelman, Shelton, & Koller 2005). A combination of the two methods might be able to solve large partially-observable continuous-time MDPs.

Acknowledgments

This work was funded by the DAF Air Force Office of Scientific Research (Young Investigator Award #FA9550-07-1-0076) and Intel Corporation.

References

Bellman, R., and Dreyfus, S. 1959. Functional approximation and dynamic programming. *Mathematical Tables and Other Aids to Computation* 13(68):247–251.

de Farias, D. P., and Van Roy, B. 2003. The linear programming approach to approximate dynamic programming. *Operations Research* 51(6):850–865.

Dean, T., and Kanazawa, K. 1989. A model for reasoning about persistence and causation. *Computational Intelligence* 5(3):142–150.

El-Hay, T.; Friedman, N.; Koller, D.; and Kupferman, R. 2006. Continuous time Markov networks. In *UAI*, 155–164.

Gopalratnam, K.; Kautz, H.; and Weld, D. S. 2005. Extending continuous time bayesian networks. In *AAAI*, 981–986.

Guestrin, C.; Koller, D.; and Parr, R. 2001. Max-norm projections of factored MDPs. In *IJCAI*, volume 1, 673–680.

Koller, D., and Parr, R. 1999. Computing factored value functions for policies in structured MDPs. In *IJCAI*, 1332–1339.

Ng, B.; Pfeffer, A.; and Dearden, R. 2005. Continuous time particle filtering. In *IJCAI*, 1360–1365.

Nodelman, U., and Horvitz, E. 2003. Continuous time Bayesian networks for inferring users' presence and activities with extensions for modeling and evaluation. Technical Report MSR-TR-2003-97, Microsoft Research.

Nodelman, U.; Koller, D.; and Shelton, C. R. 2005. Expectation propagation for continuous time Bayesian networks. In *UAI*, 431–440.

Nodelman, U.; Shelton, C. R.; and Koller, D. 2002. Continuous time Bayesian networks. In *UAI*, 378–387.

Nodelman, U.; Shelton, C. R.; and Koller, D. 2005. Expectation maximization and complex duration distributions for continuous time Bayesian networks. In *UAI*, 421–430.

Puterman, M. L. 1994. *Markov Decision Processes*. Wiley-Interscience.

Schuurmans, D., and Patrascu, R. 2002. Direct value-approximation for factored MDPs. In Dietterich, T. G.; Becker, S.; and Ghahramani, Z., eds., *NIPS*, 1579–1586.

Tsitsiklis, J. N., and Van Roy, B. 1996. Feature-based methods for large scale dynamic programming. *Machine Learning* 22(1–3):59–94.

Williams, R., and Baird, L. 1993. Tight performance bounds on greedy policies based on imperfect value functions. Technical report, Northeastern University.

Younes, H. L. S., and Simmons, R. G. 2004. Solving generalized semi-Markov decision processes using continuous phase-type distributions. In *AAAI*, 742–747.

Younes, H. L. S. 2005. Planning and execution with phase transitions. In *AAAI*, 1030–1035.