Planning and Execution with Phase Transitions

Håkan L. S. Younes
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213, USA
lorens@cs.cmu.edu

Abstract

We consider a special type of continuous-time Markov decision processes (MDPs) that arise when phase-type distributions are used to model the timing of non-Markovian events and actions. We focus, primarily, on the execution of phase-dependent policies. Phases are introduced into a model to represent relevant execution history, but there is no physical manifestation of phases in the real world. We treat phases as partially observable state features and show how a belief distribution over phase configurations can be derived from observable state features through the use of transient analysis for Markov chains. This results in an efficient method for phase tracking during execution that can be combined with the Q_{MDP} value method for POMDPs to make action choices. We also discuss, briefly, how the structure of MDPs with phase transitions can be exploited in structured value iteration with symbolic representation of vectors and matrices.

Introduction

The continuous-time Markov decision process (MDP), with exponentially distributed holding times in states, is an attractive model for decision theoretic control of asynchronous systems. Continuous-time MDPs can, in fact, be solved using the exact same techniques as discrete-time MDPs (Howard 1960; Lippman 1975; Puterman 1994), so recent progress in the AI literature on solving discrete-time MDPs (e.g. Boutilier, Dearden, & Goldszmidt 2000; Guestrin *et al.* 2003) applies to continuous-time models as well.

Many phenomena in nature are, however, best modeled with non-exponential distributions, for example, the lifetime of a product (Nelson 1985) or a computer process (Leland & Ott 1986). *Phase-type distributions* (Neuts 1981) can approximate any positive non-exponential distribution with a Markov chain. This means, in particular, that a decision process with non-exponential holding-time distributions, such as a semi-Markov decision process (Howard 1971) or *generalized* semi-Markov decision process (Younes & Simmons 2004), can be approximated by an MDP. This MDP can then be solved using standard techniques. A state of the approximating MDP includes phase information that models the history dependence of non-exponential distributions, so a policy for the MDP may be phase-dependent. Phases are not observable during execution of the policy, however, so to execute phase-dependent policies, we need a way to derive phase configurations from observable features of the world.

Younes & Simmons (2004) simulate phase transitions during execution to obtain phase configurations that can be used to guide action selection. This paper provides a more robust method for phase tracking based on *transient analysis* for Markov chains (Stewart 1994). We treat phases as partially observable state variables and provide a method for computing a belief distribution over phase configurations for any given situation during execution. Given an observation of the world, which includes the physical state and the time that any event has remained enabled without triggering, we use the Q_{MDP} value method for POMDPs (Littman, Cassandra, & Kaelbling 1995) to rank action choices. This method is well suited for MDPs with phase transitions because there is typically no gain in taking actions solely for the purpose of obtaining information about phase configurations. We demonstrate that this new phase-tracking technique clearly outperforms the simulation-based method suggested by Younes & Simmons (2004). We are able to earn near optimal reward with relatively few phases.

In addition to a new phase-tracking technique, we show how the specific structure of MDPs with phase transitions can be exploited by exact solution methods. In particular, we show that the planning time and memory requirements of structured value iteration with ADDs (Hoey *et al.* 1999) can be reduced significantly by taking into account the fact that some state variables represent phases.

We start by providing background on phase-type distributions and generalized semi-Markov decision processes (GSMDPs). This is meant to give the reader a better idea of how continuous-time MDPs with phase transitions arise.

Phase-Type Distributions

The memoryless property of the exponential distribution is instrumental in making optimal planning with continuous-time MDPs tractable. Continuous *phase-type distributions* (Neuts 1981) generalize the exponential distribution to permit history dependence in the form of *phases*, while maintaining analytical tractability.

In general, a phase-type distribution with n phases represents time from entry until absorption in a Markov chain with n transient states and a single absorbing state. An n-

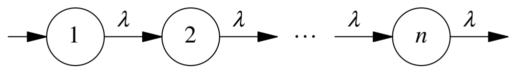

Figure 1: Erlang distribution.

phase continuous phase-type distribution is specified using $n^2 + 2n$ parameters (i and j range between 1 and n):

- λ_i, representing the exit rate for phase i.
- p_{ij}, representing the probability that phase i is followed by phase j; $q_i = 1 - \sum_{j=1}^{n} p_{ij}$ is the probability that absorption occurs immediately following phase i.
- π_i, representing the probability that the initial phase is i.

Let $\mathbf{Q} = [q_{ij}]$, with $q_{ii} = -\lambda_i(1 - p_{ii})$ and $q_{ij} = \lambda_i p_{ij}$ $(i \neq j)$, and let $\vec{\pi} = [\pi_i]$. Then the cumulative distribution function for a phase-type distribution is given by $F(t) = 1 - \vec{\pi} e^{\mathbf{Q}t} \vec{e}$, where $\vec{e}$ is a unit column vector of size n. The matrix $\mathbf{Q}$ is called the *generator* for a Markov chain.

The number of parameters needed to specify a phase-type distribution can be prohibitive. We restrict our attention to the *Erlang* distribution (Erlang 1917) in this paper, which has a single parameter λ for any n. An Erlang distribution can be thought of as a chain of n phases where the time spent in each phase, before a transition to the next phase occurs, is exponentially distributed with rate λ (Figure 1).

A phase-type distribution PH can be used to approximate a general positive distribution G, for example a Weibull distribution. The most straightforward approach is the *method of moments*, which matches the first k moments of G and PH. Closed-form solutions exist for matching up to three moments of any positive distribution, so the method of moments is fast. Younes & Simmons (2004) uses this method to approximate GSMDPs with continuous-time MDPs.

In this paper, we instead use an Erlang distribution with n phases that matches the mean μ of G ($\lambda = n/\mu$). This gives us freedom to select the number of phases. Furthermore, the distributions we use have a low coefficient of variation (the standard deviation divided by the mean). It is known that with n phases, the coefficient of variation is at least $n^{-0.5}$ for a continuous phase-type distribution, with $n^{-0.5}$ achieved exactly by an n-phase Erlang distribution (Aldous & Shepp 1987). Matching two moments of a distribution with low coefficient of variation requires a large number of phases.

If a good fit for the distribution function is desired (and the coefficient of variation is not less than $n^{-0.5}$), then more sophisticated, but also more expensive, methods are available (e.g. Asmussen, Nerman, & Olsson 1996).

Generalized Semi-Markov Decision Processes

Younes & Simmons (2004) introduced the *generalized semi-Markov decision process* (GSMDP) as a decision theoretic extension of the GSMP formalism for discrete event systems. A time-homogeneous GSMP (Glynn 1989) consists of a set of states S and a set of events E. We assume that these sets are finite in this paper. At any time, the process occupies some state $s \in S$ in which a subset $E(s)$ of the events are enabled. Associated with each event $e \in E$ is a positive distribution G_e and a next-state distribution $p_e(\cdot; s)$. The distribution G_e governs the time from when e becomes enabled until it triggers, provided e remains continuously enabled during that time period. The enabled events in a state race to trigger first. When e triggers in s, $p_e(s'; s)$ is the probability that the next state is s'.

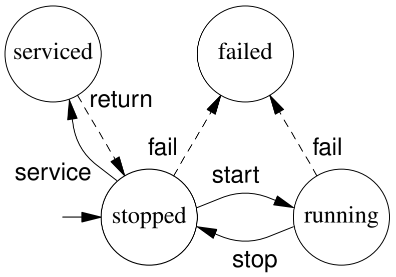

Figure 2: GSMP with fail event enabled across transitions.

To see that the GSMP truly is a generalization of the semi-Markov process, consider the model in Figure 2 with five events: start, stop, service, return, and fail. The fail event is enabled in multiple states. If the trigger-time distribution for fail is not memoryless (e.g. a Weibull distribution representing an increasing failure rate), then the time to failure in both the “stopped” and the “running” state may depend on the entire execution history of the process. In particular, the fail event remains enabled if a start or stop event occurs, but is disabled by a service event. The execution history of a GSMP can be captured by adding a real-valued clock τ_e, for each event e, to the description of states, with τ_e recording the time that e has remained continuously enabled without triggering. Thus, ordinarily, a *general state space* is required to model a finite-state GSMP as a semi-Markov process.

A decision dimension is introduced by distinguishing a set $A \subset E$ of actions and adding a reward structure. We assume a traditional reward structure for continuous-time decision processes with a lump-sum reward $k_e(s, s')$ associated with the transition from s to s' caused by the triggering of e, and a continuous reward rate $c_B(s)$ associated with the set of actions $B \subset A$ being enabled in s (cf. Howard 1971). Like Younes & Simmons (2004), we consider infinite-horizon discounted reward with discount rate α. This means that reward earned at time t counts as $e^{-\alpha t}$ or, alternatively, that there is a termination event with exponential trigger-time distribution having rate α (Howard 1960). As an example of a GSMDP, consider the model in Figure 2 with actions represented by solid arrows. A negative reward rate in the “serviced” state can be used to represent the cost of service.

If all trigger-time distributions are memoryless, then a GSMDP is simply a continuous-time MDP with a factored transition model. A GSMDP with state space S and event set E can be approximated by a continuous-time MDP with state space S' and event set E' by using phase-type distributions. Each non-exponential trigger-time distribution in the GSMDP is approximated by a phase-type distribution, as described by Younes & Simmons (2004). The new state space S' is still finite and includes phase information that can be seen as a discretization into *random-length intervals* of the clocks τ_e. The event set E' includes regular state transitions from the original model, but also phase transitions.

It is well-known that the continuous-time MDP with discounting is computationally equivalent to its discrete-time counterpart (Howard 1960; Lippman 1975). In fact, the standard solution method is *uniformization* (Puterman 1994), which can be interpreted as transforming a continuous-time MDP into an equivalent discrete-time MDP.

A continuous-time MDP can be solved directly as well. In state s, with actions B chosen to be enabled, the events $E_B(s) = E(s) \setminus (A \setminus B)$ are enabled. Let λ_e denote the rate of the exponential trigger-time distribution for event e. Treating α as the rate of a termination event, the exit rate for state s with actions B enabled is $\lambda_B^\alpha(s) = \alpha + \sum_{e \in E_B(s)} \lambda_e$. Since all trigger-time distributions are exponential, the probability that event e triggers first is $\lambda_e / \lambda_B^\alpha(s)$ and the probability that termination occurs before any events can trigger is $\alpha / \lambda_B^\alpha(s)$. With $\bar{r}_B(s) = c_B(s) + \sum_{e \in E_B(s)} \lambda_e \sum_{s' \in S} p_e(s'; s) k_e(s, s')$, we can express the Q-value of any pair $\langle s, B \rangle$ as follows:

$$Q(s, B) = \frac{\bar{r}_B(s)}{\lambda_B^\alpha(s)} + \sum_{e \in E_B(s)} \frac{\lambda_e}{\lambda_B^\alpha(s)} \sum_{s' \in S} p_e(s'; s) \max_{C \subset A} Q(s', C)$$
$$= R_B^\alpha(s) + \sum_{s' \in S} P_B^\alpha(s'; s) \max_{C \subset A} Q(s', C)$$

Here, $R_B^\alpha(s)$ is the expected reward in s with actions B enabled, and $P_B^\alpha(s'; s) = \sum_{e \in E_s(B)} p_e(s'; s) \lambda_e / \lambda_B^\alpha(s)$ is the probability that s' follows s. Note that $\sum_{s' \in S} P_B^\alpha(s'; s) < 1$ due to discounting.

Factored State Spaces and State Filtering

So far, we have assumed no specific structure for the state space S. Commonly, MDPs are specified using a *factored state space* with a set of state variables (Boutilier, Dearden, & Goldszmidt 2000). This section considers the exact solution of continuous-time MDPs with factored state spaces, and discusses a technique for exploiting the structure of such processes when some state variables represent *phases*. We assume a factored transition model, and associate an enabling condition ϕ_e (a logic formula over the state variables) with each event e such that $E(s) = \{e \mid s \models \phi_e\}$.

The Q-value recurrence in the previous section can be expressed in matrix-vector form. Let $\vec{R}_B^\alpha = [R_B^\alpha(i)]$, $\mathbf{P}_B^\alpha = [P_B^\alpha(j; i)]$, and $\vec{V}^* = [\max_{C \subset A} Q(i, C)]$. We then have

$$\vec{Q}(B) = \vec{R}_B^\alpha + \mathbf{P}_B^\alpha \cdot \vec{V}^* \ .$$

Given a factored representation of the state space S, defined by a set of state variables SV, it is possible to represent the vectors and matrices as ADDs (Clarke *et al.* 1993; Bahar *et al.* 1993) defined over variables $SV_\mathrm{b} \cup SV'_\mathrm{b} = \{v_1, \ldots, v_b, v'_1, \ldots, v'_b\}$, where SV_b is a binary encoding of the state variables SV using b bits and SV'_b are next-state versions of SV_b. The Q-value for any pair $\langle s, B \rangle$ can then be computed using structured value iteration with ADDs (Hoey *et al.* 1999).

ADDs generally permit a succinct representation of reward vectors and transition probability matrices, even for very large state spaces. This does not guarantee, however, that the Q-value vector for an action set also has a compact ADD representation. In fact, it is often the case that the ADD representation of a Q-value vector grows with each iteration, which adversely affects solution time and memory requirements. Moreover, factored representations can introduce *spurious states* (assignments to state variables that do not correspond to states in S). It is needless to compute Q-values for spurious states, but it may be difficult, in general, to determine if a specific variable assignment is feasible.

When some of the state variables represent phases, however, certain spurious states become obvious immediately. Let φ_e be a state variable representing the phase of the trigger-time distribution for event e. If e is disabled in state s, then the phase of e is inconsequential. By convention, let $\varphi_e = 1$ in this case. Any assignment of the state variables such that ϕ_e is false and $\varphi_e > 1$ is hence invalid. Furthermore, if at most one action can be enabled at any time (single agent system), then any assignment with $\varphi_a > 1$ and $\varphi_{a'} > 1$, for $a \neq a'$, is also invalid. Finally, a binary encoding of multi-valued state variables introduces spurious states if the number of values for all variables is not a power of 2, and this is true not only for phase variables.

Let $\vec{f} = [f(v_1, \ldots, v_b)]$, with $f(v_1, \ldots, v_b)$ equal to 0 if $v_1, \ldots, v_b$ is known to be invalid, and equal to 1 otherwise. The Q-value recurrence can now be expressed as

$$\vec{Q}(B) = \vec{f} \circ (\vec{R}_B^\alpha + \mathbf{P}_B^\alpha \cdot \vec{V}^*) \ .$$

The vector $\vec{f}$ acts as a *filter* that forces the Q-value to be zero for known invalid variable assignments (the operator $\circ$ represents element-wise multiplication). We can apply the filter in every iteration, or we can apply the filter once to $\vec{R}_B^\alpha$ and $\mathbf{P}_B^\alpha$ before the first iteration. In the evaluation section, we show that state filtering can reduce planning time and memory requirements notably.

Execution of Phase-Dependent Policies

When solving a continuous-time MDP with some events representing phase transitions, we get a phase-dependent policy. The phases are fictitious, however, and do not correspond to physical (observable) features in the real world. By solving the problem as if phases were observable, we temporarily ignore the observation model of the actual process. This section describes how to maintain a belief distribution over possible phase configurations during policy execution. This belief distribution, computed using regular *transient analysis* for Markov chains, is used together with the Q_MDP value method for POMDPs (Littman, Cassandra, & Kaelbling 1995) to evaluate action choices.

Phase Tracking through Transient Analysis

To take advantage of the phase-dependent aspects of a policy during execution, we need to infer phase configurations from observable features of the environment. We assume that the physical state of the process is fully observable and that we know when an event triggers. This means that, at any point in time, we have complete knowledge of the set of enabled events and, consequently, we know for how long any particular event has been enabled. An observation can thus be seen as consisting of the physical state s and a vector $\vec{\tau} = [\tau_i]$, where τ_i is the time that the ith non-Markovian event has been continuously enabled without triggering ($\tau_i = 0$ for disabled events). The goal is to compute a belief distribution $p(\vec{\varphi}; s, \vec{\tau})$ over phase configurations.

We note that the phase of an event is independent of any information concerning other events. This means, in particular, that $p(\vec{\varphi}; s, \vec{\tau}) = \prod_i p_i(\varphi_i; s, \tau_i)$, so we can perform belief tracking for each individual phase separately. If event e_i is disabled in s, then $p_i(\varphi_i; s, \tau_i)$ is 1 for $\varphi_i = 1$ and 0 for $\varphi_i > 1$. Otherwise, if e_i is enabled in s, then we compute $p_i(\varphi_i; s, \tau_i)$ using transient analysis, as described next.

Each phase φ_i represents the state in a transient Markov chain with generator matrix $\mathbf{Q}_i$ and initial-state distribution $\vec{\pi}_i$. The probability distribution over states at time t for this Markov chain is given by $\vec{\pi}_i e^{\mathbf{Q}_i t}$ (Stewart 1994, p. 407). Hence, we have $\vec{p}_i(s, \tau_i) = [p_i(j; s, \tau_i)] = \vec{\pi}_i e^{\mathbf{Q}_i \tau_i}$. This probability distribution can be computed numerically through the use of *uniformization*, as first described by Jensen (1953), which transforms a continuous-time Markov chain with generator matrix $\mathbf{Q}$ into a discrete-time Markov chain with transition probability matrix $\mathbf{P} = \mathbf{I} + \mathbf{Q}/q$, with $q \geq \max_i q_{ii}$. Through Taylor expansion, we get

$$\vec{\pi} e^{\mathbf{Q}t} = \vec{\pi} \sum_{k=0}^{\infty} e^{-q \cdot t} \frac{(q \cdot t)^k}{k!} \mathbf{P}^k \quad .$$

In practice, this infinite summation is truncated by using the techniques of Fox & Glynn (1988) so that the truncation error is bounded by an a priori error tolerance ϵ. The Fox–Glynn method requires $q \cdot t + \sqrt{2q \cdot t} \cdot o(\sqrt{\log 1/\epsilon}) + 3/2$ matrix-vector multiplications. The dependence on ϵ is generally negligible, so the time complexity is $O(q \cdot t \cdot M)$, where M is the time complexity for a single matrix-vector multiplication. For an n-phase Erlang distribution, M is $O(n)$.

The belief distribution $p(\vec{\varphi}; s, \vec{\tau})$ may have some positive probability mass associated with phase configurations where a phase has reached the absorbing state. Absorption for a phase-type distribution represents triggering of an event, but we know whether an event has triggered or not. We therefore derive a normalized belief distribution $\hat{p}(\vec{\varphi}; s, \vec{\tau})$ that excludes any mass associated with the triggering of an event.

Action Selection

Given that we can compute a belief distribution over phase configurations from an observation $\langle s, \vec{\tau} \rangle$, we can use the Q_{MDP} value method to compute the expected value of an action set $B \subset A$ in any given situation:

$$Q(s, \vec{\tau}, B) = \sum_{\vec{\varphi}} \hat{p}(\vec{\varphi}; s, \vec{\tau}) \cdot Q_{\mathrm{MDP}}(s, \vec{\varphi}, B)$$

The action set with maximum Q-value, according to this formula, is the action set that should be selected for execution when the observation $\langle s, \vec{\tau} \rangle$ is made. Note that we use the normalized belief distribution to compute Q-values.

This is straightforward application of the Q_{MDP} value method. An observation includes time, however, which is a continuous quantity in our case. The action choice may therefore have to change continuously during execution, but this is impractical. We suggest that belief updates are made at a regular interval Δ. Belief updates should also be made every time an actual state transition occurs. This means that the current action choice is reconsidered at the time of state transitions, but also every Δ time units while remaining in a specific state. Note that $p_i(\varphi_i; s, t + \Delta) = \vec{\pi}_i e^{\mathbf{Q}_i t} e^{\mathbf{Q}_i \Delta} = p_i(\varphi_i; s, t) e^{\mathbf{Q}_i \Delta}$, so belief distributions can be updated at each stage by using transient analysis with $p_i(\varphi_i; s, t)$ as the initial-state distribution and Δ as time bound.

Empirical Evaluation

We now show, empirically, that phase tracking based on transient analysis notably outperforms the phase-tracking method proposed by Younes & Simmons (2004). We also demonstrate the effect of state filtering for solving continuous-time MDPs with phase transitions. The results have been generated on a 3 GHz Pentium 4 PC running Linux, and with an 800 MB memory limit set per process.

Our first test case is a variation of Howard's "the Foreman's Dilemma" (Howard 1960) and the preventive maintenance problem used by Younes & Simmons (2004). The model is a slight abstraction of the model shown in Figure 2, with the "stopped" and "running" states represented by a single "working" state. The service action and return event have exponential trigger-time distributions with rates 10 and 1, respectively. The trigger-time distribution for fail is $W(1.6x, 4.5)$, a Weibull distribution representing an increasing failure rate. We vary x between 1 and 40, with higher values meaning a higher expected failure time. The reward rate is 1 for the "working" state, -0.1 for the "serviced" state, and 0 for the "failed" state. Once failure occurs, no more reward can be earned. The optimal policy is to enable the service action after t_a time units in the "working" state, where t_a depends on the failure time distribution.

Because the problem is relatively simple, we can find the optimal value for t_a using numerical function optimization (and numerical integration over semi-infinite intervals for our failure time distribution). This gives us a reference point for the performance of the approximate solution technique that uses phase-type distributions. The problem can also be modeled and solved as an SMDP because no event can remain enabled across state transitions. Standard SMDP solution techniques (Howard 1971), however, permit an action to be enabled only immediately in a state (corresponding to $t_a = 0$) or not at all ($t_a = \infty$). Figure 3 shows the performance in terms of expected discounted reward ($\alpha = -\log 0.95$), relative to optimal, for the phase-type approximation technique and the standard SMDP solution technique. The policies generated by the latter perform far below optimal (as x increases, the mean time to failure increases and discounting makes failure less of a factor). By approximating the failure time distribution with an Erlang distribution, better performance is achieved. Note, however, that the reward earned by simulating phase transitions is still far from optimal. The performance is much improved by using the phase-tracking technique presented in this paper. Better performance than for phase simulation is achieved

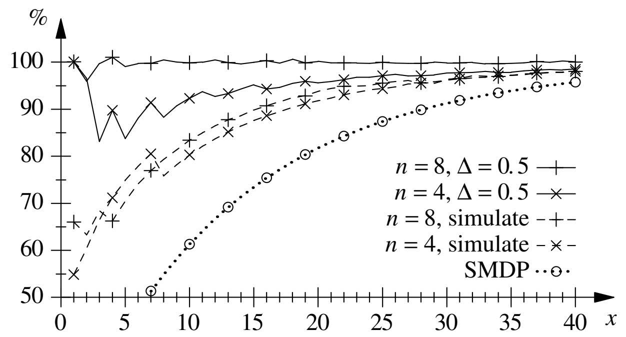

Figure 3: Percentage of optimal expected reward for maintenance problem with failure time distribution $W(1.6x, 4.5)$.

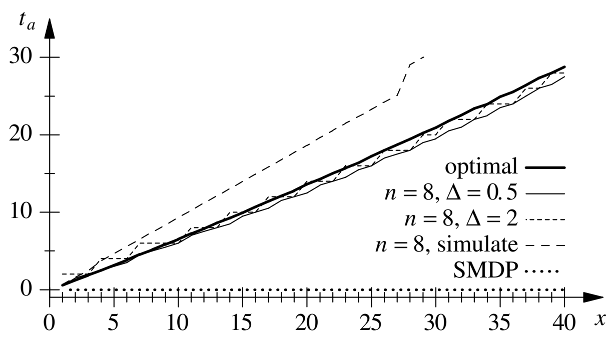

Figure 4: Average delay for enabling of the service action in the "working" state.

with fewer phases, and with 8 phases and $\Delta = 0.5$, the performance is near optimal. The solution time for the models with a phase-type approximation is only a few milliseconds.

Figure 4 plots the optimal value for t_a, as well as the value of t_a for the different approximate solution techniques. The standard SMDP solution results in $t_a = 0$, which means that failure is typically avoided, but reward is wasted in the "serviced" state. By simulating phase transitions, on the other hand, the service action is enabled too late on average. The phase-tracking method results in values of t_a much closer to optimal. We plot t_a when using 8 phases and two different values of Δ. Note that a lower value for Δ is not always better, as might otherwise be expected. This is because the phase-type distribution does not perfectly match the actual failure time distribution. Currently, we have no method for choosing the best Δ other than experimentation.

Our second test case is a system administration problem used by Younes & Simmons (2004). In this problem, there is a network of m computers, with each computer being either up or down. There is a crash event for each computer with trigger-time distribution $Exp(1)$. A reboot action with trigger-time distribution $U(0, 1)$ is available for each computer that is currently down. The reward rate for a state is equal to the number of computers that are up. We assume that at most one reboot action can be enabled at any time.

Unlike the previous test case, this is not an SMDP (except for $m = 1$) since a reboot action may remain enabled across state transitions, so we cannot solve this problem using standard SMDP techniques. The obvious solution is to reboot a computer whenever it goes down, and wait until rebooting is finished before going on to reboot another computer. We get this exact solution by using phase-type approximations of reboot time distributions and transient analysis to track phases. It is sufficient to perform phase tracking only at the time of state transitions ($\Delta = \infty$). Figure 5 plots the expected discounted reward for different number of phases and phase-tracking techniques. We see, again, that our proposed phase-tracking technique outperforms the technique based on simulation of phase transitions. We achieve optimal performance using only two phases.

Figure 6 shows the effect of state filtering on the solution time of structured policy iteration with ADDs for the system administration problem. We can see that state filtering helps reduce the planning time significantly, even when the number of phases is a power of 2. Furthermore, with $n = 5$ and without filtering, memory is exhausted already for $m = 8$. Note also that applying the filter in every iteration is better than to pre-filter reward vectors and transition probability matrices, because pre-filtering results in more complex ADD representations or vectors and matrices.

Discussion

We have provided a new technique for tracking phase configurations during execution of phase-dependent policies. The technique relies on efficient numerical methods for transient analysis of Markov chains, so the computational burden for phase tracking is small during execution. Furthermore, we need to perform transient analysis only on the Markov chains that represent phase-type distributions, which typically are very limited in size. We have demonstrated, through experimental evaluation, that the proposed phase-tracking method can result in near optimal performance (in terms of expected reward) even with a small number of phases. The new method clearly outperforms the method proposed by Younes & Simmons (2004), which simulates phase transitions during execution. These results are very general, as they are not tied to a particular solution method for MDPs. We have used structured value iteration in this paper, and we have shown that MDPs with phase transitions have specific structure that can be exploited to reduce the planning time for this method, but any solution method could be used to solve the MDP. For example, one could use the approximate solution method presented by Guestrin *et al.* (2003) to handle large state spaces.

MDPs with phase transitions arise, for example, when a GSMDP is approximated with an MDP by approximating each non-exponential trigger-time distribution with a phase-type distribution. We have seen that a GSMDP with finite state space can be modeled as a *general-state* (both discrete and continuous state features) SMDP. While there has been some interesting recent work on solving general-state MDPs (Feng *et al.* 2004; Guestrin, Hauskrecht, & Kveton 2004), these solution methods are not well suited for GSMDPs because they can only capture the state of the GSMDP at the time of state transitions (discrete dynamics). To act optimally in a GSMDP, it is important to account for what

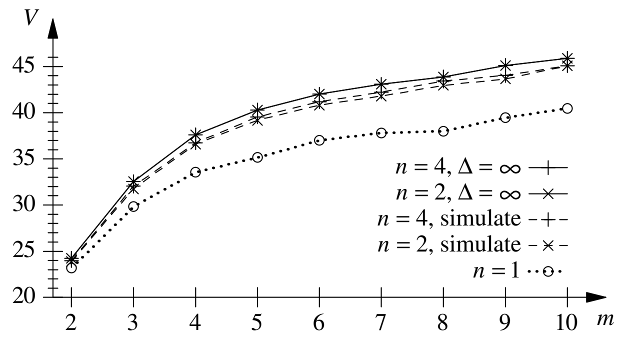

Figure 5: Expected discounted reward for system administration problem with m computers.

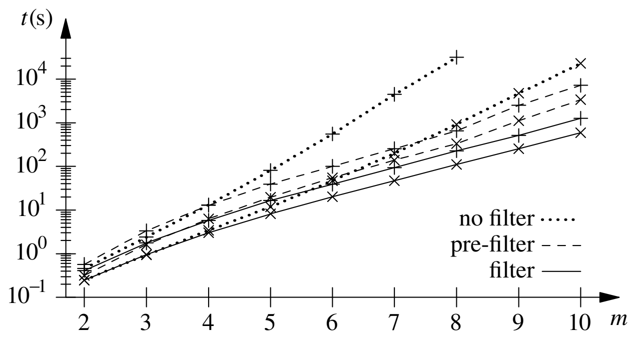

Figure 6: Effect of state filtering for system administration problem with $n = 4$ ($\times$) and $n = 5$ ($+$).

happens between state transitions. These methods could be combined with the use of phase-type distributions, however, to solve general-state GSMDPs.

Acknowledgments. Sponsored by the US Army Research Office (ARO) under contract no. DAAD190110485.

References

Aldous, D., and Shepp, L. 1987. The least variable phase type distribution is Erlang. *Comm. Statist. Stoch. Models* 3(3):467–473.

Asmussen, S.; Nerman, O.; and Olsson, M. 1996. Fitting phase-type distributions via the EM algorithm. *Scand. J. Statist.* 23(4):419–441.

Bahar, R. I.; Frohm, E. A.; Gaona, C. M.; Hachtel, G. D.; Macii, E.; Pardo, A.; and Somenzi, F. 1993. Algebraic decision diagrams and their applications. In *Proc. 1993 IEEE/ACM International Conference on Computer-Aided Design*, 188–191.

Boutilier, C.; Dearden, R.; and Goldszmidt, M. 2000. Stochastic dynamic programming with factored representations. *Artificial Intelligence* 121(1):49–107.

Clarke, E. M.; McMillan, K. L.; Zhao, X.; and Fujita, M. 1993. Spectral transforms for large Boolean functions with applications to technology mapping. In *Proc. 30th International Conference on Design Automation*, 54–60.

Erlang, A. K. 1917. Solution of some problems in the theory of probabilities of significance in automatic telephone exchanges. *The Post Office Electrical Engineers' Journal* 10:189–197.

Feng, Z.; Dearden, R.; Meuleau, N.; and Washington, R. 2004. Dynamic programming for structured continuous Markov decision processes. In *Proc. Twentieth Conference on Uncertainty in Artificial Intelligence*, 154–161.

Fox, B. L., and Glynn, P. W. 1988. Computing Poisson probabilities. *Comm. ACM* 31(4):440–445.

Glynn, P. W. 1989. A GSMP formalism for discrete event systems. *Proc. IEEE* 77(1):14–23.

Guestrin, C.; Koller, D.; Parr, R.; and Venkataraman, S. 2003. Efficient solution algorithms for factored MDPs. *J. Artificial Intelligence Res.* 19:399–468.

Guestrin, C.; Hauskrecht, M.; and Kveton, B. 2004. Solving factored MDPs with continuous and discrete variables. In *Proc. Twentieth Conference on Uncertainty in Artificial Intelligence*, 235–242.

Hoey, J.; St-Aubin, R.; Hu, A.; and Boutilier, C. 1999. SPUDD: Stochastic planning using decision diagrams. In *Proc. Fifteenth Conference on Uncertainty in Artificial Intelligence*, 279–288.

Howard, R. A. 1960. *Dynamic Programming and Markov Processes*. New York: John Wiley & Sons.

Howard, R. A. 1971. *Dynamic Probabilistic Systems*, volume II: Semi-Markov and Decision Processes. New York: John Wiley & Sons.

Jensen, A. 1953. Markoff chains as an aid in the study of Markoff processes. *Scand. Actuar. J.* 36:87–91.

Leland, W. E., and Ott, T. J. 1986. Load-balancing heiristics and process behavior. *ACM SIGMETRICS Perform. Eval. Rev.* 14(1):54–69.

Lippman, S. A. 1975. Applying a new device in the optimization of exponential queuing systems. *Oper. Res.* 23(4):687–710.

Littman, M. L.; Cassandra, A. R.; and Kaelbling, L. P. 1995. Learning policies for partially observable environments: Scaling up. In *Proc. Twelfth International Conference on Machine Learning*, 362–370.

Nelson, W. 1985. Weibull analysis of reliability data with few or no failures. *J. Qual. Tech.* 17(3):140–146.

Neuts, M. F. 1981. *Matrix-Geometric Solutions in Stochastic Models: An Algorithmic Approach*. Baltimore, Maryland: Johns Hopkins University Press.

Puterman, M. L. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York: John Wiley & Sons.

Stewart, W. J. 1994. *Introduction to the Numerical Solution of Markov Chains*. Princeton, New Jersey: Princeton University Press.

Younes, H. L. S., and Simmons, R. G. 2004. Solving generalized semi-Markov decision processes using continuous phase-type distributions. In *Proc. Nineteenth National Conference on Artificial Intelligence*, 742–747.