A Fast Analytical Algorithm for Solving Markov Decision Processes with Real-Valued Resources*

Janusz Marecki Sven Koenig Milind Tambe
Computer Science Department
University of Southern California
941 W 37th Place, Los Angeles, CA 90089
{marecki, skoenig, tambe}@usc.edu

Abstract

Agents often have to construct plans that obey deadlines or, more generally, resource limits for real-valued resources whose consumption can only be characterized by probability distributions, such as execution time or battery power. These planning problems can be modeled with continuous state Markov decision processes (MDPs) but existing solution methods are either inefficient or provide no guarantee on the quality of the resulting policy. We therefore present CPH, a novel solution method that solves the planning problems by first approximating with any desired accuracy the probability distributions over the resource consumptions with phase-type distributions, which use exponential distributions as building blocks. It then uses value iteration to solve the resulting MDPs by exploiting properties of exponential distributions to calculate the necessary convolutions accurately and efficiently while providing strong guarantees on the quality of the resulting policy. Our experimental feasibility study in a Mars rover domain demonstrates a substantial speedup over Lazy Approximation, which is currently the leading algorithm for solving continuous state MDPs with quality guarantees.

1 Introduction

Recent advances in robotics have made aerial, underwater and terrestrial unmanned autonomous vehicles possible. In many domains for which such unmanned vehicles are constructed, the vehicles have to construct plans that obey deadlines or, more generally, resource limits for real-valued resources whose consumption can only be characterized by probability distributions, such as execution time or battery power. It is a major challenge for AI research to construct good plans for these domains efficiently [Bresina *et al.*, 2002]. One cannot model the resulting planning problems with generalized semi-Markov decision processes [Younes and Simmons, 2004] since they can model actions that consume real-valued amounts of resources but not resource limits. One could model them with Markov decision processes (MDPs) by encoding the available amounts of resources in the states. The resulting continuous state MDP can then be solved approximately by discretizing the state space (for example, by discretizing the probability distributions) and then applying standard dynamic programming algorithms, such as value iteration [Boyan and Littman, 2000]. A major drawback of this approach is that discretizing the state space results in a combinatorial explosion of the number of states as the discretization becomes more and more fine-grained. In response, discretization-free approaches have been developed, some of which do not provide strong guarantees on the quality of the resulting policy [Lagoudakis and Parr, 2003; Nikovski and Brand, 2003; Hauskrecht and Kveton, 2003]. Lazy Approximation [Li and Littman, 2005] is currently the leading discretization-free algorithm for solving continuous state MDPs with quality guarantees. It solves the planning problems by approximating the probability distributions over the resource consumptions with piecewise constant functions. It then uses value iteration to solve the resulting MDPs by repeatedly approximating the value functions with piecewise constant value functions. Lazy Approximation has two problems that increase its runtime: First, complex piecewise constant functions are often necessary to approximate probability distributions closely. Second, the value functions need to be reapproximated after every iteration. We thus present CPH (**C**ontinuous state MDPs through **PH**ase-type probability distributions), a novel solution method that solves the planning problems by first approximating the probability distributions over the resource consumptions with phase-type distributions, which use exponential distributions as building blocks [Neuts, 1981]. It then uses value iteration to solve the resulting MDPs by exploiting properties of exponential distributions to calculate the value functions after every iteration accurately (= without reapproximations) and efficiently. CPH shares with Lazy Approximation that it provides strong guarantees on the quality of the resulting policy but runs substantially faster than Lazy Approximation in our experimental feasibility study in a Mars rover domain.

2 The Planning Problem

We use time as an example of a real-values resource and solve planning problems that are similar to the ones studied in [Li

*This material is based upon work supported by the DARPA/IPTO COORDINATORS program and the Air Force Research Laboratory under Contract No. FA875005C0030. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

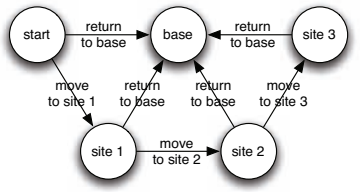


Figure 1: Mars rover domain.

and Littman, 2005]. They are modeled as MDPs with two sources of uncertainty: action durations and action outcomes. S denotes the finite set of states of the MDP and $A(s)$ the finite set of actions that can be executed in state $s \in S$. Assume that an agent is in state $s \in S$ with a deadline being $t > 0$ time units away (= with time-to-deadline t), after which execution stops. It executes an action $a \in A(s)$ of its choice. The execution of the action cannot be interrupted, and the action duration t' is distributed according to a given probability distribution $p_{s,a}(t')$ that can depend on both the state s and the action a. If $t' \geq t$, then the time-to-deadline $t - t' \leq 0$ after the action execution is non-positive, which means that the deadline is reached and execution stops. Otherwise, with probability $P(s'|s, a)$, the agent obtains reward $R(s, a, s') \geq 0$ and transitions to state $s' \in S$ with time-to-deadline $t - t'$ and repeats the process. The time-to-deadline at the beginning of execution is $\Delta > 0$. The objective of the agent is to maximize its expected total reward until execution stops.

We use a Mars rover domain (Figure 1) that is similar to one used in [Bresina *et al.*, 2002] as an example of our planning problems. A rover has to maximize its expected total reward with time-to-deadline $\Delta = 4$. The states of the MDP correspond to the locations of the rover: its start location, site 1, site 2, site 3 and its base station. The rover can perform two actions with deterministic action outcomes outside of its base station: (1) It can move to the next site and collect a rock probe from there. It receives rewards 4, 2 and 1 for collecting rock probes from sites 1, 2 and 3, respectively. (2) It can also move back to its base station to perform a communication task, which drains its energy completely and thus makes it impossible for the rover to perform additional actions. It receives reward 6 for performing the communication task. All action durations are characterized by the exponential distribution $p_{s,a}(t') = e^{-t'} = E(1)$. We also consider the case where all action durations are characterized by either Weibull or Normal distributions. The Mars rover domain might appear small with only 5 discrete states. However, if one discretizes the time-to-deadline into 1/100 time units and encodes it in the states, then there are already $5 * 400 = 2000$ states.

3 General Approach

We can solve our planning problems by encoding the time-to-deadline in the states, resulting in a continuous state MDP that can be solved with a version of value iteration [Boyan and Littman, 2000] as follows: Let $V^*(s)(t)$ denote the largest expected total reward that the agent can obtain until execution stops if it starts in state $s \in S$ with time-to-deadline $0 \leq t \leq \Delta$. The agent can maximize its expected total reward by executing the action

$$\arg\max_{a \in A(s)} \quad (\sum_{s' \in S} P(s'|s, a) \int_0^t p_{s,a}(t')(R(s, a, s') + V^*(s')(t - t'))\, dt')$$

in state $s \in S$ with time-to-deadline $0 \leq t \leq \Delta$, which can be explained as follows: When it executes action $a \in A(s)$ in state $s \in S$, it incurs action duration t' with probability $p_{s,a}(t')$. If $0 \leq t' \leq t$, then it transitions to state $s' \in S$ with probability $P(s'|s, a)$ and obtains an expected total future reward of $R(s, a, s') + V^*(s')(t - t')$.

Value iteration calculates the values $V^n(s)(t)$ using the following Bellman updates for all states $s \in S$, time-to-deadlines $0 \leq t \leq \Delta$, and iterations $n \geq 0$

$$V^0(s)(t) := 0$$

$$V^{n+1}(s)(t) := \begin{cases} 0 & \text{if } t \leq 0 \\ \max_{a \in A(s)} (\sum_{s' \in S} P(s'|s, a) & \text{otherwise} \\ \quad \int_0^t p_{s,a}(t')(R(s, a, s') + V^n(s')(t - t'))\, dt'). \end{cases}$$

It then holds that $\lim_{n \to \infty} V^n(s)(t) = V^*(s)(t)$ for all states $s \in S$ and times-to-deadline $0 \leq t \leq \Delta$. Unfortunately, value iteration cannot be implemented as stated since the number of values $V^n(s)(t)$ is infinite for each iteration n since the time-to-deadline t is a real-valued variable. We remedy this situation by viewing the $V^n(s)$ as value functions that map times-to-deadline t to the corresponding values $V^n(s)(t)$, similar to [Liu and Koenig, 2005]. In this context, we make the following contributions: First, we show after how many iterations n value iteration can stop so that the approximation error $\max_{s \in S, 0 \leq t \leq \Delta} |V^*(s)(t) - V^n(s)(t)|$ is no larger than a given constant $\epsilon > 0$. Second, we show that each value function $V^n(s)$ can be represented exactly with a vector of a small number of real values each. Finally, we show how the Bellman updates can efficiently transform the vectors of the value functions $V^n(s)$ to the vectors of the value functions $V^{n+1}(s)$. We present these three contributions in the following as three steps of CPH.

4 CPH: Step 1

CPH first approximates the probability distributions of action durations that are not exponential with phase-type distributions, resulting in chains of exponential distributions $\lambda e^{-\lambda t'} = E(\lambda)$ with potentially different constants $\lambda > 0$ [Neuts, 1981]. CPH then uses uniformization (i.e., creates self-transitions) to make all constants λ identical without changing the underlying stochastic process. We do not give details on how to implement these two well-known steps (see [Neuts, 1981; Younes and Simmons, 2004] for details) but rather provide an example. Figure 2 shows how an action with a deterministic action outcome and an action duration that is characterized by a Normal distribution $N(\mu = 2, \sigma = 1)$ is first approximated with a phase-type distribution with three

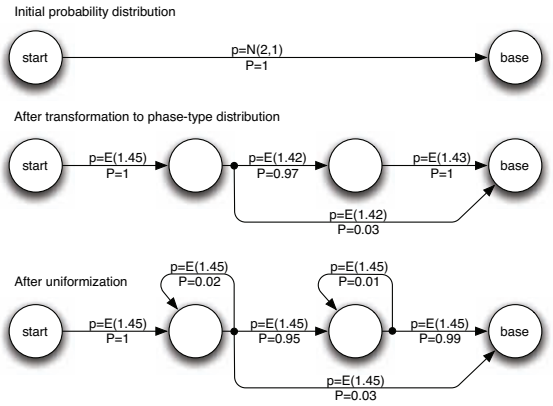


Figure 2: Transformation of a Normal distribution to a chain of exponential distributions $E(1.45)$. p and P denote the action duration and action outcome probabilities, respectively.

phases and then uniformized to yield the exponential distributions $E(1.45)$. From now on, we can therefore assume without loss of generality that the action durations t' of all actions are distributed according to exponential distributions $p_{s,a}(t') = p(t') = E(\lambda)$ with the same constant $\lambda > 0$. Value iteration cannot determine the value functions $V^*(s)$ with a finite number of iterations since the uniformized MDPs have self-transitions. However, Theorem 1 in the appendix shows that the approximation error is no larger than a given constant $\epsilon > 0$ if value iteration stops after at least

$$\log_{\frac{e^{\lambda\Delta}-1}{e^{\lambda\Delta}}} \frac{\epsilon}{R_{max}(e^{\lambda\Delta}-1)}$$

iterations, where $R_{max} := \max_{s\in S, a\in A(s), s'\in S} R(s,a,s')$. Running time of Step 1 of CPH is negligible since analytic phase-type approximation methods run in time $O(1)$.

5 CPH: Step 2

It follows directly from the considerations in the next section that there exist times-to-deadline $0 = t_{s,0} < t_{s,1} < \ldots < t_{s,m_s+1} = \Delta$ such that $V^n(s)(t) = V_i^n(s)(t)$ for all $t \in [t_{s,i}, t_{s,i+1})$, where

$$V_i^n(s)(t) = c_{s,i,1} - e^{-\lambda t}\left(c_{s,i,2} + \ldots + c_{s,i,n+1}\frac{(\lambda t)^{n-1}}{(n-1)!}\right) \quad (1)$$

for the parameters $c_{s,i,j}$ for all $s \in S$, $0 \le i \le m_s$ and $1 \le j \le n+1$. We refer to the times-to-deadline $t_{s,i}$ as breakpoints, the $[t_{s,i}, t_{s,i+1})$ as intervals, the expressions for the value functions $V_i^n(s)$ as gamma functions (which is a simplification since they are actually linear combinations of Euler's incomplete gamma functions), and the expressions for the value functions $V^n(s)$ as piecewise gamma functions. We represent each gamma function $V_i^n(s)$ as a vector $[c_{s,i,1}, \ldots, c_{s,i,n+1}] = [c_{s,i,j}]_{j=1}^{n+1}$ and each piecewise gamma function $V^n(s)$ as a vector of vectors $[t_{s,i}{:}V_i^n(s)]_{i=0}^{m_s} = [t_{s,i}{:}[c_{s,i,j}]_{j=1}^{n+1}]_{i=0}^{m_s}$. Figure 3, for example, shows the value function $V^*(\mathit{start})$ for our Mars rover domain.

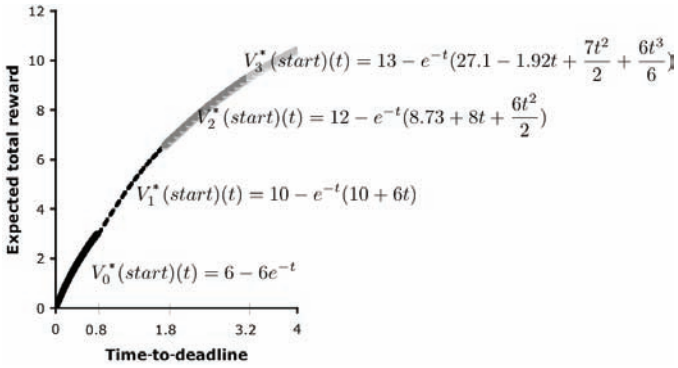


Figure 3: Value function $V^*(\mathit{start})$ consists of four gamma functions: $V^*(\mathit{start}) = [0{:}V_0^*(\mathit{start}),\ 0.8{:}V_1^*(\mathit{start}),\ 1.8{:}V_2^*(\mathit{start}),\ 3.2{:}V_3^*(\mathit{start})] = [0{:}[6,6],\ 0.8{:}[10,10,6],\ 1.8{:}[12,8.73,8,6],\ 3.2{:}[13,27.1,-1.92,7,6]]$.

6 CPH: Step 3

We now explain how CPH uses value iteration. For $n = 0$, the value functions $V^n(s) = 0$ satisfy $V^n(s) = [0{:}[0]]$ and thus are (piecewise) gamma. We show by induction that all value functions $V^{n+1}(s)$ are piecewise gamma if all value functions $V^n(s)$ are piecewise gamma. We also show how the Bellman updates can efficiently transform the vectors of the value functions $V^n(s)$ to the vectors of the value functions $V^{n+1}(s)$. Value iteration calculates

$$V^{n+1}(s)(t) := \max_{a\in A(s)} \sum_{s'\in S} P(s'|s,a) \int_0^t p(t')(R(s,a,s') + V^n(s')(t-t'))\, dt'.$$

We break this calculation down into four stages: First, $\overline{V}^n(s')(t-t') := R(s,a,s') + V^n(s')(t-t')$. Second, $\widetilde{V}^n(s')(t) := \int_0^t p(t')\overline{V}^n(s')(t-t')\, dt'$. Third, $\widehat{V}^n(s,a)(t) := \sum_{s'\in S} P(s'|s,a)\widetilde{V}^n(s')(t)$. Finally, $V^{n+1}(s)(t) := \max_{a\in A(s)} \widehat{V}^n(s,a)(t)$.[1]

Stage 1: Calculate $\overline{V}^n(s')(t-t') := R(s,a,s') + V^n(s')(t-t')$. Our induction assumption is that all value functions $V^n(s')$ are piecewise gamma, i.e., $V^n(s') = [t_{s',i}{:}[c_{s',i,j}]_{j=1}^{n+1}]_{i=0}^{m_{s'}}$. In Stage 1, CPH calculates $\overline{V}^n(s')(t-t') := R(s,a,s') + V^n(s')(t-t')$, which is the same as calculating $\overline{V}^n(s')(t) := R(s,a,s') + V^n(s')(t)$ since $R(s,a,s')$ is constant. Then,

$$\begin{aligned}\overline{V}^n(s') &= R(s,a,s') + V^n(s') \\ &= R(s,a,s') + [t_{s',i} : [c_{s',i,j}]_{j=1}^{n+1}]_{i=0}^{m_{s'}} \\ &= [t_{s',i} : [c'_{s',i,j}]_{j=1}^{n+1}]_{i=0}^{m_{s'}}.\end{aligned}$$

where $c'_{s',i,1} = R(s,a,s') + c_{s',i,1}$ and $c'_{s',i,j} = c_{s',i,j}$ for all $0 \le i \le m_{s'}$ and $2 \le j \le n+1$.

[1] We should really use $\overline{V}^n(s,a,s')(t-t')$ and $\widetilde{V}^n(s,a,s')(t)$ instead of $\overline{V}^n(s')(t-t')$ and $\widetilde{V}^n(s')(t)$, respectively, but this would make our notation rather cumbersome.

Stage 2: Calculate $\widetilde{V}^n(s')(t) := \int_0^t p(t')\overline{V}^n(s')(t-t')\,dt'$, which is a convolution of p and $\overline{V}^n(s')$ denoted as $p * \overline{V}^n(s')$. Consider the value functions $\overline{V}^n(s')$ defined in Stage 1. We now show by induction that

$$\begin{aligned}&\widetilde{V}_i^n(s')(t)\\ &= (p * \overline{V}_i^n(s'))(t) - e^{-\lambda t}(\sum_{i'=1}^{i} e^{\lambda t_{s',i'}}(p * \overline{V}_{i'}^n(s'))(t_{s',i'})\\ &\quad - \sum_{i'=0}^{i-1} e^{\lambda t_{s',i'+1}}(p * \overline{V}_{i'}^n(s'))(t_{s',i'+1})) \qquad (2)\end{aligned}$$

for all $t \in [t_{s',i}, t_{s',i+1})$. Equation (2) holds for $i = 0$ since $\widetilde{V}^n(s')(t) = (p * \overline{V}_0^n(s'))(t)$ for all $t \in [t_{s',0}, t_{s',1})$. Assume now that Equation (2) holds for some i. It then also holds for $i+1$ as shown in Figure 4. We use the following lemma to show that $\widetilde{V}_i^n(s')$ is a gamma function and convert it to vector notation.

Lemma 1. *Let* $p(t) = E(\lambda)$. *Then,* $p * [k_1, k_2, \ldots, k_n] = [k_1, k_1, k_2, \ldots, k_n]$.

Proof. By symbolic integration, $p * [k]_{j=1}^n = [k]_{j=1}^{n+1}$ for any constant k. Then, $p * [k_1, \ldots, k_n] = p * (\sum_{i=1}^{n-1}[k_i - k_{i+1}]_{j=1}^i + [k_n]_{j=1}^n) = \sum_{i=1}^{n-1}(p * [k_i - k_{i+1}]_{j=1}^i) + p * [k_n]_{j=1}^n = \sum_{i=1}^{n-1}[k_i - k_{i+1}]_{j=1}^{i+1} + [k_n]_{j=1}^{n+1} = [k_1, k_1, \ldots, k_n]$. □

We now transform Equation (2) to vector notation.

$$\begin{aligned}\widetilde{V}_i^n(s') &= [c''_{s',i,j}]_{j=1}^{n+2}\\ \text{where}\quad & c''_{s',i,1} := c'_{s',i,1} \qquad c''_{s',i,2} := c'_{s',i,1} + z_{s',i}\\ & c''_{s',i,j+1} := c'_{s',i,j} \quad \forall_{j=2,3,\ldots,n+1}\\ \text{with}\quad & z_{s',i} := \sum_{i'=1}^{i} e^{\lambda t_{s',i'}}(p * \overline{V}_{i'}^n(s'))(t_{s',i'})\\ & \qquad - \sum_{i'=0}^{i-1} e^{\lambda t_{s',i'+1}}(p * \overline{V}_{i'}^n(s'))(t_{s',i'+1}).\end{aligned}$$

Consequently, $\widetilde{V}^n(s') = [t_{s',i}{:}[c''_{s',i,j}]_{j=1}^{n+2}]_{i=0}^{m_{s'}}$.

Stage 3: Calculate $\widehat{V}^n(s,a)(t) := \sum_{s' \in S} P(s'|s,a)\widetilde{V}^n(s')(t)$. Consider the value functions $\widetilde{V}^n(s')$ defined in Stage 2. Since they might have different breakpoints $\{t_{s',i}\}_{i=0}^{m_{s'}}$ for different states $s' \in S$ with $P(s'|s,a) > 0$, CPH introduces the common breakpoints $\{t_{s,a,i}\}_{i=0}^{m_{s,a}} = \bigcup_{s' \in S: P(s'|s,a)>0}\{t_{s',i}\}_{i=0}^{m'_s}$ without changing the value functions $\widetilde{V}^n(s')$. Afterwards, $\widetilde{V}^n(s') = [t_{s,a,i}{:}[c'''_{s',i,j}]_{j=1}^{n+2}]_{i=0}^{m_{s,a}}$ where $c'''_{s',i,j} = c''_{s',i',j}$ for the unique i' with $[t_{s,a,i}, t_{s,a,i+1}) \subseteq [t_{s',i'}, t_{s',i'+1})$. Then,

$$\begin{aligned}\widehat{V}^n(s,a) &= \sum_{s' \in S} P(s'|s,a)\widetilde{V}^n(s')\\ &= \sum_{s' \in S} P(s'|s,a)[t_{s,a,i} : [c'''_{s',i,j}]_{j=1}^{n+2}]_{i=0}^{m_{s,a}}\\ &= [t_{s,a,i} : [\sum_{s' \in S} P(s'|s,a)c'''_{s',i,j}]_{j=1}^{n+2}]_{i=0}^{m_{s,a}}\\ &= [t_{s,a,i} : [c''''_{s,a,i,j}]_{j=1}^{n+2}]_{i=0}^{m_{s,a}}.\end{aligned}$$

Stage 4: Calculate $V^{n+1}(s)(t) := \max_{a \in A(s)} \widehat{V}^n(s,a)(t)$. Consider the value functions $\widehat{V}^n(s,a)$ defined in Stage 3. Since they might have different breakpoints $\{t_{s,a,i}\}_{i=0}^{m_{s,a}}$ for different actions $a \in A(s)$, CPH introduces common breakpoints $\bigcup_{a \in A(s)}\{t_{s,a,i}\}_{i=0}^{m_{s,a}}$ without changing the value functions $\widehat{V}^n(s,a)$. CPH then introduces additional dominance breakpoints at the intersections of the value functions to ensure that, over each interval, one of the value functions dominates the other ones. Let $\{t'_{s,i}\}_{i=0}^{m'_s}$ be the set of breakpoints afterwards. Then, $V^{n+1}(s) = [t'_{s,i}{:}[c''''_{s,i,j}]_{j=1}^{n+2}]_{i=0}^{m'_s}$ where $c''''_{s,i,j} = c''''_{s,a_{s,i},i',j}$ for actions $a_{s,i} \in A(s)$ and the unique i' with $[t'_{s,i}, t'_{s,i+1}) \subseteq [t_{s,a_{s,i},i'}, t_{s,a_{s,i},i'+1})$ and, for all $t \in [t'_{s,i}, t'_{s,i+1})$, $\widehat{V}^n(s,a)(t) \leq \widehat{V}^n(s,a_{s,i})(t)$. Furthermore, action $a_{s,i}$ should be executed according to the value function $V^{n+1}(s)$ if the current state is s and the time-to-deadline is $t \in [t'_{s,i}, t'_{s,i+1})$.

To summarize, the value functions $V^{n+1}(s)$ are piecewise gamma, and the vectors of the value functions $V^n(s)$ can be transformed automatically to the vectors of the value functions $V^{n+1}(s)$. The lengths of the vectors increase linearly in the number of iterations and, although the number of breakpoints (that are placed automatically during the transformations) can increase exponentially, in practice it stays small since one can merge small intervals after each iteration of value iteration to reduce the number of breakpoints The transformations require only simple vector manipulations and a numerical method that determines the dominance breakpoints approximately, for which CPH uses a bisection method. We now show experimentally that the transformations are both efficient and accurate.

7 Experiments

We performed an experimental feasibility study in our Mars rover domain that compares CPH and Lazy Approximation (LA) [Li and Littman, 2005], currently the leading algorithm for solving continuous state MDPs with quality guarantees (Figure 5). We always plot the error $\max_{0 \leq t \leq \Delta} |V^*(\textit{start})(t) - V(\textit{start})(t)|$ of the calculated value function $V(\textit{start})$ on the x-axis and the corresponding runtime (measured in milliseconds) in log scale on the y-axis.

Experiment 1: We determined how CPH and Lazy Approximation trade off between runtime and error for our Mars

$$\widetilde{V}^n_{i+1}(s')(t) = \int_0^t p(t')\overline{V}^n(s')(t-t')\,dt' = \widetilde{V}^n_{i+1}(s')(t) = \int_0^t p(t-t')\overline{V}^n(s')(t')\,dt'$$

We split the integral into two ranges. For $t_{s',i+1} \leq t' \leq t < t_{s',i+2}$ we know that $\overline{V}^n(s')(t') = \overline{V}^n_{i+1}(s')(t')$.

$$= \int_{t_{s',i+1}}^t p(t-t')\overline{V}^n_{i+1}(s')(t')\,dt' + \int_0^{t_{s',i+1}} p(t-t')\overline{V}^n(s')(t')\,dt'$$

It holds that $p(t-t') = \lambda e^{-\lambda(t-t')} = e^{-\lambda(t-t_{s',i+1})}\lambda e^{-\lambda(t_{s',i+1}-t')} = e^{-\lambda(t-t_{s',i+1})}p(t_{s',i+1}-t')$

$$= \int_0^t p(t-t')\overline{V}^n_{i+1}(s')(t')\,dt' - \int_0^{t_{s',i+1}} p(t-t')\overline{V}^n_{i+1}(s')(t')\,dt' + e^{-\lambda(t-t_{s',i+1})}\int_0^{t_{s',i+1}} p(t_{s',i+1}-t')\overline{V}^n(s')(t')\,dt'$$

$$= \int_0^t p(t-t')\overline{V}^n_{i+1}(s')(t')\,dt' - e^{-\lambda(t-t_{s',i+1})}\int_0^{t_{s',i+1}} p(t_{s',i+1}-t')\overline{V}^n_{i+1}(s')(t')\,dt' + e^{-\lambda(t-t_{s',i+1})}\widetilde{V}^n_i(s')(t_{s',i+1})$$

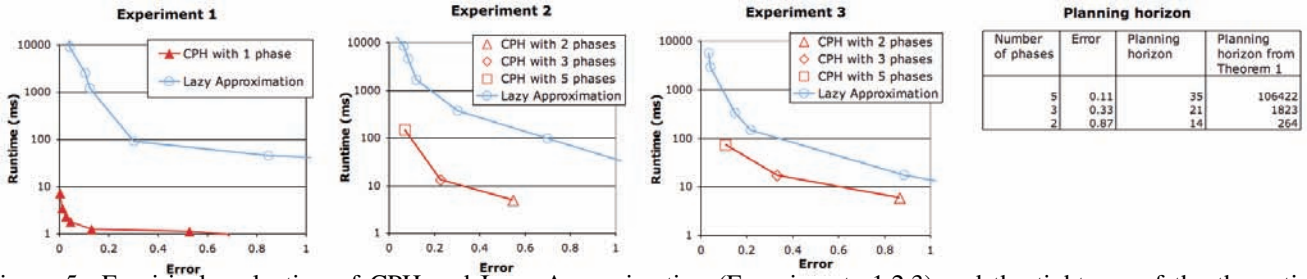
$$= (p * \overline{V}^n_{i+1}(s'))(t) - e^{-\lambda(t-t_{s',i+1})}(p * \overline{V}^n_{i+1}(s'))(t_{s',i+1}) + e^{-\lambda(t-t_{s',i+1})}\widetilde{V}^n_i(s')(t_{s',i+1})$$

We finally unroll the recursion by using the induction assumption.

$$= (p * \overline{V}^n_{i+1}(s'))(t) - e^{-\lambda(t-t_{s',i+1})}(p * \overline{V}^n_{i+1}(s'))(t_{s',i+1}) + e^{-\lambda(t-t_{s',i+1})}((p * \overline{V}^n_i(s'))(t_{s',i+1})$$
$$- e^{-\lambda t_{s',i+1}}\left(\sum_{i'=1}^{i} e^{\lambda t_{s',i'}}(p * \overline{V}^n_{i'}(s'))(t_{s',i'}) - \sum_{i'=0}^{i-1} e^{\lambda t_{s',i'+1}}(p * \overline{V}^n_{i'}(s'))(t_{s',i'+1})\right))$$

$$= (p * \overline{V}^n_{i+1}(s'))(t) - e^{-\lambda t}\left(\sum_{i'=1}^{i+1} e^{\lambda t_{s',i'}}(p * \overline{V}^n_{i'}(s'))(t_{s',i'}) - \sum_{i'=0}^{i} e^{\lambda t_{s',i'+1}}(p * \overline{V}^n_{i'}(s'))(t_{s',i'+1})\right)$$

Figure 4: Proof by induction of Equation (2).



Planning horizon

Number of phases	Error	Planning horizon	Planning horizon from Theorem 1
5	0.11	35	106422
3	0.33	21	1823
2	0.87	14	264

Figure 5: Empirical evaluation of CPH and Lazy Approximation (Experiments 1,2,3) and the tightness of the theoretical planning horizon calculated from Theorem 1.

rover domain. Since the action durations in the Mars rover domain are already distributed exponentially and thus phase-type with one phase, there are no errors introduced by approximating the probability distributions over the action durations with phase-type distributions. Also, since these distributions are equal, uniformization will not introduce self-transitions and value iteration can run for a finite number of iterations. We therefore varied for CPH the accuracy of the bisection method that determines the dominance breakpoints and for Lazy Approximation the accuracy of the piecewise constant approximations of the probability distributions over the action durations and value functions. Our results show that CPH is faster than Lazy Approximation with the same error, by three orders of magnitude for small errors. For example, CPH needed 2ms and Lazy Approximation needed 1000ms to compute a policy that is less than 1% off optimal, which corresponds to an error of 0.13 in the Mars rover domain.

Experiments 2 and 3: We then determined how CPH and Lazy Approximation trade off between runtime and error when all action durations in our Mars rover domain are characterized by either Weibull distributions $Weibull(\alpha = 1, \beta = 2)$ (Experiment 2) or Normal distributions $N(\mu = 2, \sigma = 1)$ (Experiment 3), and both methods thus need to approximate the value functions. We fixed for CPH the accuracy of the bisection method but varied the the number of phases used to approximate the probability distributions. Our results show that CPH is still faster than Lazy Approximation with the same error, by more than one order of magnitude for small errors. For example, CPH needed 149ms (with five phases) and Lazy Approximation needed 4471ms to compute a policy that is less than 1% off optimal for the Weibull distributions. In addition, we observed that CPH converged much faster than the theoretical planning horizon calculated from Theorem 1 suggests.

8 Conclusions

Planning with real-valued and limited resources can be modeled with continuous state MDPs. Despite recent advances in solving continuous state MDPs, state-of-the art solution methods are still either inefficient [Li and Littman, 2005]

or cannot provide guarantees on the quality of the resulting policy [Lagoudakis and Parr, 2003]. In this paper, we presented a solution method, called CPH, that avoids both of these shortcomings. The aim of this paper was to establish a sound theoretical framework for CPH and provide a first experimental feasibility study to demonstrate its potential. It is future work to study the theoretical properties of CPH in more depth (for example, analyze its complexity or extend its error analysis), evaluate CPH and its three approximations more thoroughly (for example, in larger domains, additional domains and against additional solution methods) and extend CPH (for example, to handle more than one resource, handle replenishable resources or, similarly to Lazy Approximation, approximate its value functions to be able to reduce its number of breakpoints).

References

[Boyan and Littman, 2000] J. Boyan and M. Littman. Exact solutions to time-dependent MDPs. In *NIPS*, pages 1026–1032, 2000.

[Bresina *et al.*, 2002] J. Bresina, R. Dearden, N. Meuleau, D. Smith, and R. Washington. Planning under continuous time and resource uncertainty: A challenge for AI. In *UAI*, pages 77–84, 2002.

[Hauskrecht and Kveton, 2003] M. Hauskrecht and B. Kveton. Linear programm approximations for factored continuous-state MDPs. In *NIPS*, pages 895–902, 2003.

[Lagoudakis and Parr, 2003] M. Lagoudakis and R. Parr. Least-squares policy iteration. *JMLR*, 4(6):1107–1149, 2003.

[Li and Littman, 2005] L. Li and M. Littman. Lazy approximation for solving continuous finite-horizon MDPs. In *AAAI*, pages 1175–1180, 2005.

[Liu and Koenig, 2005] Y. Liu and S. Koenig. Risk-sensitive planning with one-switch utility functions: Value iteration. In *AAAI*, pages 993–999, 2005.

[Neuts, 1981] M. Neuts. *Matrix-Geometric Solutions in Stochastic Models: An Algorithmic Approach*. John Hopkins University Press, Baltimore, 1981.

[Nikovski and Brand, 2003] D. Nikovski and M. Brand. Non-linear stochastic control in continuous state spaces by exact integration in Bellman's equations. In *ICAPS-WS2*, pages 91–95, 2003.

[Younes and Simmons, 2004] H. Younes and R. Simmons. Solving generalized semi-MDPs using continuous phase-type distributions. In *AAAI*, pages 742–747, 2004.

Appendix

The following theorem is a shortened version that does not take into account the errors introduced by approximating the probability distributions over the action durations with phase-type distributions and the dominance breakpoints with a numerical method. It takes only into account the error introduced by approximating the value functions due to running value iteration for only a finite number of iterations.

Theorem 1. *Let* $\epsilon > 0$ *be any positive constant and* $n \geq \log_{\frac{e^{\lambda\Delta}-1}{e^{\lambda\Delta}}} \frac{\epsilon}{R_{max}(e^{\lambda\Delta}-1)}$. *Then,* $\max_{s\in S, 0\leq t\leq\Delta} |V^*(s)(t) - V^{n^*}(s)(t)| \leq \epsilon$, *where* $R_{max} := \max_{s\in S, a\in A(s), s'\in S} R(s,a,s')$.

Proof. Let $\alpha := \lambda\Delta > 0$ and $b_i := \sum_{j=i}^{\infty} \frac{\alpha^j}{j!}$ for all $i \geq 0$. It holds that $b_0 = e^\alpha$ and $b_1 = b_0 - 1 = e^\alpha - 1$. We now provide a bound on the probability $p_i(t)$ that the sum of the action durations of a sequence of $i \geq 1$ actions is no more than $0 \leq t \leq \Delta$.

$$\begin{aligned} p_i(t) &\leq p_i(\Delta) = \int_0^\Delta (\underbrace{p * p * \ldots * p}_{i})(t')\, dt' = \int_0^\Delta e^{-\lambda t'} \frac{t'^{i-1}\lambda^i}{i!}\, dt' \\ &= \frac{1}{e^\alpha}\left(e^\alpha - \sum_{j=0}^{i-1}\frac{\alpha^j}{j!}\right) = \frac{1}{e^\alpha}\left(\sum_{j=0}^{\infty}\frac{\alpha^j}{j!} - \sum_{j=0}^{i-1}\frac{\alpha^j}{j!}\right) \\ &= \frac{1}{e^\alpha}\sum_{j=i}^{\infty}\frac{\alpha^j}{j!} = \frac{b_i}{e^\alpha}. \end{aligned}$$

The values $\frac{b_{i+1}}{b_i} = \frac{b_{i+1}}{\frac{\alpha^i}{i!}+b_{i+1}} = \frac{1}{\frac{\alpha^i}{i!\,b_{i+1}}+1}$ decrease strictly monotonically in i because the values

$$\frac{\alpha^i}{i!\,b_{i+1}} = \frac{\alpha^i}{i!\sum_{j=i+1}^{\infty}\frac{\alpha^j}{j!}} = \frac{1}{\frac{\alpha}{i+1} + \frac{\alpha^2}{(i+2)(i+1)} + \frac{\alpha^3}{(i+3)(i+2)(i+1)} + \ldots}$$

increase strictly monotonically in i. Consequently,

$$1 > \frac{e^\alpha - 1}{e^\alpha} = \frac{b_1}{b_0} > \frac{b_2}{b_1} > \frac{b_3}{b_2} > \ldots > 0.$$

Thus

$$\begin{aligned} b_i &< \frac{b_{i-1}}{b_{i-2}}b_{i-1} < \frac{b_1}{b_0}b_{i-1} < \frac{b_1}{b_0}\frac{b_{i-2}}{b_{i-3}}b_{i-2} < \left(\frac{b_1}{b_0}\right)^2 b_{i-2} \\ &< \ldots < \left(\frac{b_1}{b_0}\right)^{i-1} b_1. \end{aligned}$$

We now use these results to bound $|V^*(s)(t) - V^n(s)(t)|$ for all $s \in S$ and $0 \leq t \leq \Delta$. Assume that the agent starts in state $s \in S$ with time-to-deadline $0 \leq t \leq \Delta$. Value iteration with n iterations determines the highest expected total reward $V^n(s)(t)$ under the restriction that execution stops when the deadline is reached or n actions have been executed. The largest expected total reward $V^*(s)(t)$ does not have the second restriction and can thus be larger than $V^n(s)(t)$. In particular, only $V^*(s)(t)$ takes into account that the agent can execute the $(n+1)$st action with probability $p_{n+1}(t)$, the $(n+2)$nd action with probability $p_{n+2}(t)$, and so on, receiving a reward of at most R_{max} for each action execution. Additionally, $V^n(s)(t)$ is locally greedy, i.e., the reward for its first n actions exceeds the reward for the first n actions of $V^*(s)(t)$. Thus,

$$\begin{aligned} 0 &\leq V^*(s)(t) - V^n(s)(t) \leq \sum_{i=n+1}^{\infty} R_{max} p_i(t) \\ &\leq \frac{R_{max}}{e^\alpha}\sum_{i=n+1}^{\infty} b_i < \frac{R_{max}}{e^\alpha}\sum_{i=n+1}^{\infty}\left(\frac{b_1}{b_0}\right)^{i-1} b_1 \\ &= \frac{R_{max}b_1}{e^\alpha}\sum_{i=0}^{\infty}\left(\frac{b_1}{b_0}\right)^{i+n} = \frac{R_{max}b_1}{e^\alpha}\left(\frac{b_1}{b_0}\right)^n\sum_{i=0}^{\infty}\left(\frac{b_1}{b_0}\right)^i \\ &= \frac{R_{max}b_1}{e^\alpha}\left(\frac{b_1}{b_0}\right)^n\frac{1}{1-\frac{b_1}{b_0}}. \end{aligned}$$

We now bound this expression by ϵ.

$$\begin{aligned} \frac{R_{max}b_1}{e^\alpha}\left(\frac{b_1}{b_0}\right)^n\frac{1}{1-\frac{b_1}{b_0}} &\leq \epsilon \\ n &\geq \log_{\frac{e^{\lambda\Delta}-1}{e^{\lambda\Delta}}} \frac{\epsilon}{R_{max}(e^{\lambda\Delta}-1)}. \end{aligned}$$

□