$TiMDP_{poly}$: an Improved Method for Solving Time-dependent MDPs

Emmanuel Rachelson
Dept. of ECE
Technical University of Crete
73100 Chania, Greece
rachelson@intelligence.tuc.gr

Patrick Fabiani
ONERA-DCSD
2, avenue Edouard Belin
31055 Toulouse, France
patrick.fabiani@onera.fr

Frédérick Garcia
INRA-BIA
Chemin de Borde Rouge
31326 Castanet, France
fgarcia@toulouse.inra.fr

Abstract

We introduce $TiMDP_{poly}$, an algorithm designed to solve planning problems with durative actions, under probabilistic uncertainty, in a non-stationary, continuous-time context. Mission planning for autonomous agents such as planetary rovers or unmanned aircrafts often correspond to such time-dependent planning problems. Modeling these problems can be cast through the framework of Time-dependent Markov Decision Processes (TiMDPs). We analyze the TiMDP optimality equations in order to exploit their properties. Then, we focus on the class of piecewise polynomial models in order to approximate TiMDPs, and introduce several algorithmic contributions which lead to the $TiMDP_{poly}$ algorithm for TiMDPs. Finally, our approach is evaluated on an unmanned aircraft mission planning problem and on an adapted version of the well-known Mars rover domain.

1 Introduction

Taking into account both uncertainty and continuous time-dependency is a crucial issue in some planning domains such as operation planning for autonomous aerial vehicles or Mars rovers. While sensor noise or external disturbances are often modeled as stochastic outcomes in Markov Decision Processes(MDPs, [8]), these processes only model the stepwise evolution of the system.

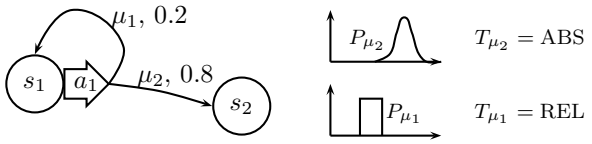
We focus on an extension of MDPs to continuous observable time: the set of decision epochs takes real values instead of successive integers and time is a continuous state variable among other discrete ones in the "hybrid" state space. Boyan & Littman [1] introduce the framework of Time-dependent Markov Decision Processes (TiMDPs). We build on their contribution to improve the resolution of TiMDPs and provide better insight on algorithms such as [3] or [5]. While the scope of this paper is on TiMDPs, many of the results presented here apply in broader frameworks for continuous variables and hybrid state spaces.

2 Continuous time in MDP planning

A Markov Decision Process is given by the tuple $\langle S, A, P, r\rangle$ where S is the set of possible states for the agent, A is a set of available actions among which the agent chooses at each decision epoch, $P(s'|s,a)$ is a Markovian transition model providing the probability of reaching state s' after undertaking action a in s, and $r(s,a)$ describes the reward obtained during transition (s,a). Solving an MDP boils down to finding a Markovian *control policy* π, mapping states to actions and optimizing a given criterion. A common criterion is the expected γ-discounted cumulative reward of applying policy π over an infinite horizon, starting in a given state s. An important issue illustrated by this criterion's definition is that a policy is optimized on the basis of a unit duration for all actions. In the problems we wish to consider, the uncertainty often affects both the action outcomes and the sojourn times in successive states.

Several extensions to discrete-event dynamic systems exist to take durative actions and time-dependency into account for decision optimization (*e.g.*[4, 11, 12]). We focus on TiMDPs [1], a straightforward way of including time in the state space of an MDP. A TiMDP is described by a set of discrete states S and a set of actions A, as in a standard MDPs. However, whenever one performs action a in s and *at time* t, an *outcome* μ, among the set M of outcomes, is triggered with probability $L(\mu|s,t,a)$. Each outcome is described by a destination state s_μ and a duration model P_μ characterizing the sojourn time before the transition to s_μ triggers. This duration model can either be *relative* — it provides the probability density function (pdf) on the sojourn time — or *absolute* — giving the pdf on the transition date. Figure 1 illustrates this definition and recalls the optimality equations for TiMDPs.

U is the expected value of outcome μ, while Q is the expected value of undertaking a in (s,t). Note that with TiMDPs, policy values have to be manipulated as functions of t in each state s, instead of simple scalar values as in the MDP case. $\overline{V}(s,t)$ is the best action's value in (s,t),s



$$U(\mu,t) = \begin{cases} \int_{-\infty}^{\infty} P_\mu(t')[R(\mu,t,t') + V(s'_\mu,t')]dt' \\ \int_{-\infty}^{\infty} P_\mu(t'-t)[R(\mu,t,t') + V(s'_\mu,t')]dt' \end{cases} \tag{1}$$

$$Q(s,t,a) = \sum_{\mu \in M} L(\mu|s,t,a) \cdot U(\mu,t) \tag{2}$$

$$\overline{V}(s,t) = \max_{a \in A} Q(s,t,a) \tag{3}$$

$$V(s,t) = \sup_{t' \geq t} \left(\int_t^{t'} K(s,\theta)d\theta + \overline{V}(s,t') \right) \tag{4}$$

Figure 1. Time-dependent MDP

given all Q functions, and, finally, $V(s,t)$ is the expected value function in s if one allows for a specific $wait$ action which leaves the discrete state unchanged and deterministically moves forward in time with a time-dependent reward rate $K(s,t)$. [1] illustrate that with piecewise constant (PWC) L functions, piecewise linear (PWL) reward models and discrete P_μ distributions, one could analytically perform the Bellman backups inspired by equations 1 to 4. [3] extends this idea to compute solutions to continuous state MDPs and [5] explore the practical resolution of value iteration using PWC functions with the Lazy Approximation algorithm. The approach we present in this paper for TiMDPs relates to the Lazy Approximation scheme. Our results complement and extend [5] in several ways.

3 Planning horizon vs. temporal horizon

In [3] and [5], whenever time is included as a state variable, the optimization process is presented as a finite horizon MDP where the value function is optimized for a limited number of consecutive decision epochs. However, this restriction can be avoided by distinguishing between *planning* horizon and *temporal* horizon. The planning horizon, as usually defined, is the number of sequential decision epochs of the agent. Deciding with a finite planning horizon restricts the number of steps an agent can perform. MDPs are usually optimized for an *unbounded* horizon. On the other hand, the *temporal* horizon T corresponds to the initial value of a non-replenishable time resource. Whenever this resource becomes depleted, the process enters an absorbing state providing zero reward and representing the end of the episode. Hence, every episode is defined between times 0 and T and we are mostly interested in the policy and value function between these two times. For real-life, observable time processes such as TiMDPs, the time-dependency of the problem is only known until a given bounded temporal horizon T. Thus, we consider the problem an infinite-horizon MDP on $[0,T]$. The optimal value function is then a fixed point of the Bellman operator for observable-time MDPs [10]. This brings us to performing value iteration-like Bellman backups, with a discount factor $\gamma = 1$, on the continuous $V(s,t)$ value functions[1].

4 Closed-form Bellman backups

Value iteration on TiMDPs updates $V_s(t) = V(s,t)$ every time the discrete state s is updated. Each of these Bellman backups can be performed analytically and in closed-form [1], if $P_\mu(t')$ and $P_\mu(\tau)$ are discrete distributions, $L(\mu|s,t,a)$ is a PWC function of t, $R(\mu,t,t') = r_t(t) + r_{t'}(t') + r_\tau(t'-t)$, and r_t, $r_{t'}$ and r_τ are PWL functions. In order to generalize on these hypotheses, we consider the case of piecewise polynomial (PWP) functions. We write $\mathcal{P}_m$ the set of PWP functions of maximum degree m and suppose that $P_\mu \in \mathcal{P}_A$, $r_t, r_{t'}, r_\tau \in \mathcal{P}_B$ and $L \in \mathcal{P}_C$.

Result 1 (Value function degree)**.** *The sequence of value functions issued by the application of the Bellman backups corresponding to equations 1 to 4 has the degree:*

$$d^\circ(V_n) = B + n(A + C + 1) \tag{5}$$

Proof. This follows from establishing that the convolution of two PWP functions of degree m and n yields a PWP of degree $m + n + 1$. Taking equations 1 to 4 step by step and observing the functions' degrees provides the result. □

Consequently, in order to have a closed-form solution of the Bellman equation throughout the value iterations, one needs to insure $A + C = -1$. While this is not possible for purely PWP functions, one needs to remember that A is indeed the degree of a PWP distribution (and not a PWP function). Analyzing equations 1 to 4 shows that if P_μ is a discrete distribution, then it behaves as a "$\mathcal{P}_{-1}$" distribution . Then, one can reach the $A + C = -1$ condition with $A = -1$ and $C = 0$. Hence, exact closed-form resolution of TiMDPs cannot be directly extended to PWP distributions, but one can still perform the Bellman backups, they just do not result in a closed-form solution anymore. Thus, any projection scheme on a lower PWP degree function space which provides error bounds on the approximation error could fit an approximate value iteration method for TiMDPs (and more generally for continuous state MDPs).

[1] See [10, 9] for a complete discussion on the mathematical assumptions necessary for a sound inclusion of time as a state variable in MDPs.

In practice, the convolution, multiplication, summation and intersection operations of equations 1 to 4 subdivide the definition intervals of the PWP functions to obtain the next V_{n+1} value function. We observed that Bellman backups on PWP representations result in a linear increase in the number of pieces necessary to describe the value function, even in the exact resolution case ($A + C = -1, B \leq 4$). So, to avoid numerical inconsistencies such as intervals of length tending to zero, one needs to make use of approximation at some point. Moreover, experience shows that these very small intervals often have very close values and, hence, can be easily merged into larger intervals if one allows for an L_∞-bounded approximation scheme.

5 Approximation method for value functions

Finding an optimal interpolation of a continuous function by a PWP in terms of number of intervals and degree is a difficult problem to solve. Thus, our method implements the sub-optimal (but efficient) following ϵ-approximation scheme. This methods proceeds in two steps: first it considers a single "piece" of the input function p_{in}, *ie.* an interval over which p_{in} has a continuous polynomial definition. Over this interval, which we write $\mathcal{I}$, it calls an interpolation method `interpolate`$(p_{in}, \mathcal{I}, l, \epsilon)$ where l is the maximum degree allowed and ϵ the L_∞ approximation tolerance. This method computes an interpolation polynomial of degree at most l over $\mathcal{I}$ and outputs it along with the largest approximation error e_{max} and the abscissa t_{max} where e_{max} is reached. If e_{max} is smaller than ϵ, then the output PWP p_{out} is set to the interpolation polynomial over $\mathcal{I}$ and the algorithm moves on to the second phase. Else, $\mathcal{I}$'s upper bound is shifted to t_{max} and the process restarts. Once a suitable interpolation has been found on the reduced $\mathcal{I}$, then a new $\mathcal{I}$ is defined by taking the uncovered part or the initial $\mathcal{I}$ and the same method is applied until the algorithm reaches the upper bound of the initial $\mathcal{I}$. Then the second phase of the approximation allows to keep the number of intervals low by trying to merge any consecutive intervals after the end of $\mathcal{I}$ into a larger interval using an interpolating PWP of degree l and an approximation error of ϵ at most. If it fails, it returns to the first phase. This procedure is repeated until T is reached and an interpolation PWP p_{out} is output. Since the `interpolate` method is free, it is easy to preserve the continuity of the function for $l \geq 1$ (and eventually its smoothness if l is large enough). This method always outputs a PWP function $p_{out} \in \mathcal{P}_l$ which has a suboptimal (but good) number of intervals. The approximation error ϵ is controllable and one has the guarantee that $\|p_{in} - p_{out}\|_\infty \leq \epsilon$. Experience showed that the output function was close to an optimal approximation in terms of intervals number with a significant reduction in computational effort.

6 Ordering Bellman backups

With the analytical computation of Bellman backups and the previous approximation method, one has a straightforward way of performing value iteration on TiMDPs. Since the value function we are looking for corresponds to the fixed point of an infinite horizon dynamic programming operator (section 3), we can avoid updating each state sequentially as in simple value iteration. Instead, ordering the states in which we perform the Bellman backups will accelerate convergence to an ϵ-optimal value function and thus reduce the computational effort due to by PWP operations. An efficient method for ordering Bellman backups in standard MDPs is *Prioritized Sweeping* [6]. Algorithm 1 adapts this method to TiMDPs.

Algorithm 1: Prioritized Sweeping for TiMDPs

```
Init: V ← 0
Init: priority_queue ← UnprioritizedVI()
Init: continue = true
while continue = true do
    while priority_queue ≠ ∅ do
        Remove the top state s' from priority_queue.
        V_{s'}(t).BellmanBackup()
        foreach (s, a) ∈ predecessors(s') do
            Q_{s,a}(t).BellmanUpdate()
            Prio(s, a) = ‖Q_{s,a}(t) − Q^{old}_{s,a}(t)‖_∞
            if Prio(s, a) > ε and Prio(s, a) > Prio(s)
            then
                Insert s in priority_queue with
                Prio(s) = Prio(s, a)
    priority_queue ← UnprioritizedVI()
    if max_priority(priority_queue) < ε then
        Either take a smaller ε or set continue = false.
```

A subtle but essential difference with [6] deals with priority computation: through difference of $Q(s, a, t)$ functions instead of $V(s)$ values. The `BellmanBackup` procedure applies equations 3 and 4 in order to update $\overline{V}_{s'}(t)$ and `BellmanUpdate` computes the result of equations 1 and 2 for parent transitions. Note that if memory is not an issue, one can also keep track of the U-functions to increase the calculation's efficiency. This priority queue can be initialized by hand if one has some prior knowledge about the problem's structure, or it can be built from a single pass of unprioritized value iteration through the state space. In order to insure that no states are left out during the optimization process, whenever the priority queue becomes empty, a new pass of unprioritized value iteration is performed. If this pass only generates priorities lower than ϵ, the algorithm terminates. Upon termination of the algorithm, the global value function $V(s, t)$ is guaranteed to be at least ϵ-optimal for the TiMDP problem.

7 Experimental results

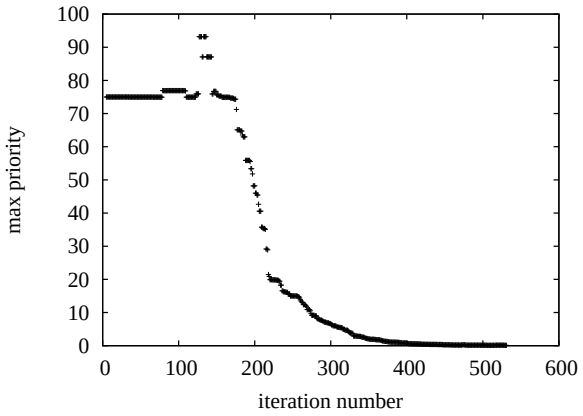


Figure 2. Maximum priority / iteration number

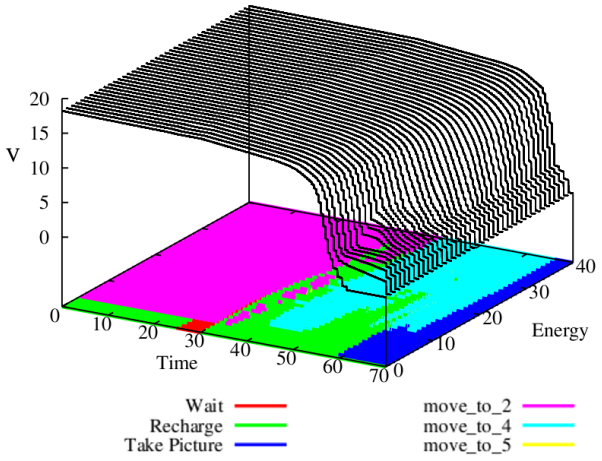


Figure 3. Rover's value function and policy

The $TiMDP_{poly}$ algorithm was implemented as a general purpose solver and tested on two benchmarks. The first is an original UAV mission planning problem where a drone needs to plan its movements in a windy area in order to monitor specific locations. This problem has 100 discrete states plus the continuous time variable. The second is an adapted version of the Mars rover domain presented in [2]. Both problems feature the hybrid state and action[2] spaces of TiMDPs. Our main conclusions were:

Prioritizing is useful. It reduced the Bellman backups number by a factor 62 for the UAV problem, taking it from 33000 to 531 and greatly decreasing computation time.

Impact of the PWP functions' maximum degree. While larger degrees imply less definition intervals in PWP, the calculation overhead can be a bad trade-off. Still, this evaluation is very implementation-dependent and further investigation is required for a final conclusion on this topic. So far, the best results were obtained with linear models.

Approximation is necessary. Even for very simple benchmarks, in the exact resolution case of [1], the number of intervals in value function definition increased steadily and eventually caused numerical problems before the value function converged to the optimal value function. This leads us to conclude – from experience – that even within the exact resolution conditions, for non-trivial problems, approximation through interval simplification is necessary.

Policy quality evolution. Figure 2 shows the evolution of the maximum priority with the Bellman backup number. One can use this maximum priority as an *asymptotic, approximate* measure of the current policy's quality since the priorities are related to the Bellman error.

[2] Actions are hybrid too because of the continuous "wait" action.

From one to several continuous variables. Figure 3 presents a value function in a specific state of the Mars rover domain. It is worth noting that the best option for representing partitions in the continuous part of the state space might not be plain hypercubes or kd-trees, but more flexible structures as the Kuhn triangulations used in [7].

References

[1] J. A. Boyan and M. L. Littman. Exact Solutions to Time Dependent MDPs. *NIPS*, 13:1026–1032, 2001.

[2] J. Bresina, R. Dearden, N. Meuleau, S. Ramakrishnan, and R. Washington. Planning under Continuous Time and Resource Uncertainty: a Challenge for AI. In *Proc. UAI*, 2002.

[3] Z. Feng, R. Dearden, N. Meuleau, and R. Washington. Dynamic Programming for Structured Continuous Markov Decision Problems. In *Proceedings UAI*, 2004.

[4] R. A. Howard. Semi-Markovian Decision Processes. In *34th Session of the International Statistical Institute*, 1963.

[5] L. Li and M. L. Littman. Lazy Approximation for Solving Continuous Finite-Horizon MDPs. In *Proc. AAAI*, 2005.

[6] A. W. Moore and C. G. Atkeson. Prioritized Sweeping: Reinforcement Learning with Less Data and Less Real Time. *Machine Learning Journal*, 13(1):103–105, 1993.

[7] R. Munos and A. W. Moore. Variable Resolution Discretization in Optimal Control. *MLJ*, 49(2-3):291–323, 2002.

[8] M. L. Puterman. *Markov Decision Processes*. John Wiley & Sons, Inc, 1994.

[9] E. Rachelson. *Temporal Markov Decision Problems — Formalization and Resolution*. PhD thesis, University of Toulouse, France, 2009.

[10] E. Rachelson, F. Garcia, and P. Fabiani. Extending the Bellman Equation for MDP to Continuous Actions and Continuous Time in the Discounted Case. In *ISAIM*, 2008.

[11] M. Wellman, M. Ford, and K. Larson. Path Planning under Time-Dependent Uncertainty. In *Proc UAI*, 1995.

[12] H. L. S. Younes and R. G. Simmons. Solving Generalized Semi-Markov Decision Processes using Continuous Phase-Type Distributions. In *AAAI*, 2004.