Real-Time Symbolic Dynamic Programming for Hybrid MDPs

Luis G. R. Vianna and **Leliane N. de Barros**
IME - USP
São Paulo, Brazil

Scott Sanner
NICTA & ANU
Canberra, Australia

Abstract

Recent advances in Symbolic Dynamic Programming (SDP) combined with the extended algebraic decision diagram (XADD) have provided exact solutions for expressive subclasses of finite-horizon Hybrid Markov Decision Processes (HMDPs) with mixed continuous and discrete state and action parameters. Unfortunately, SDP suffers from two major drawbacks: (1) it solves for all states and can be intractable for many problems that inherently have large optimal XADD value function representations; and (2) it cannot maintain compact (pruned) XADD representations for domains with nonlinear dynamics and reward due to the need for nonlinear constraint checking. In this work, we simultaneously address both of these problems by introducing real-time SDP (RTSDP). RTSDP addresses (1) by focusing the solution and value representation only on regions reachable from a set of initial states and RTSDP addresses (2) by using visited states as witnesses of reachable regions to assist in pruning irrelevant or unreachable (nonlinear) regions of the value function. To this end, RTSDP enjoys provable convergence over the set of initial states and substantial space and time savings over SDP as we demonstrate in a variety of hybrid domains ranging from inventory to reservoir to traffic control.

Introduction

Real-world planning applications frequently involve continuous resources (e.g. water, energy and speed) and can be modelled as Hybrid Markov Decision Processes (HMDPs). Problem 1 is an example of an HMDP representing the RESERVOIR MANAGEMENT problem, where the water levels are continuous variables. Symbolic dynamic programming (SDP) ((Zamani, Sanner, and Fang 2012)) provides an exact solution for finite-horizon HMDPs with mixed continuous and discrete state and action parameters, using the eXtended Algebraic Decision Diagram (XADD) representation. The XADD structure is a generalisation of the algebraic decision diagram (ADD) (Bahar et al. 1993) to compactly represent functions of both discrete and continuous variables. Decision nodes may contain boolean variables or inequalities between expressions containing continuous variables. Terminal nodes contain an expression on the continuous variables which could be evaluated to a real number.

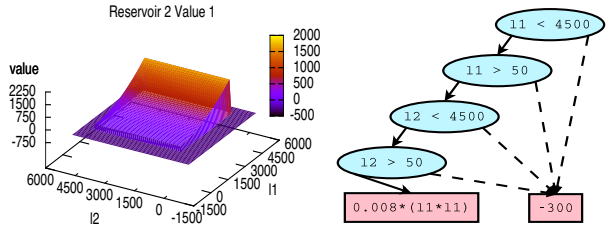


Figure 1: Nonlinear reward function for the RESERVOIR MANAGEMENT domain: graphical plot *(left)*; XADD representation *(right)*: blue ovals are internal nodes, pink rectangles are terminal nodes, $true$ branches are solid lines and $false$ are dashed.

An XADD representation of the reward function of RESERVOIR MANAGEMENT is shown in Figure 1.

Problem 1 - RESERVOIR MANAGEMENT (Yeh 1985; Mahootchi 2009). *In this problem, the task is to manage a water supply reservoir chain made up of k reservoirs each with a continuous water level L_i. The $drain_i$ and $no\text{-}drain_i$ actions determine whether water is drained from reservoir i to the next. The amount of drained water is linear on the water level, and the amount of energy produced is proportional to the product of drained water and water level. At every iteration, the top reservoir receives a constant spring replenishment plus a non-deterministic rain while the bottom reservoir has a part of its water used for city purposes. For example, the water level of the top reservoir, L_1, at the next iteration is:*

$$L_1' = \begin{cases} if\, rain \wedge drain_1 & : L_1 + 1000 - 0.5L_1 \\ if\, \neg rain \wedge drain_1 & : L_1 + 500 - 0.5L_1 \\ if\, rain \wedge \neg drain_1 & : L_1 + 1000 \\ if\, \neg rain \wedge \neg drain_1 & : L_1 + 500 \end{cases}$$

A **nonlinear** *reward is obtained by generating energy and there is a strong penalty if any reservoir level is outside the nominal minimum and maximum levels.*

$$R_{drain_i}(L_i) = \begin{cases} if\, 50 \leq L_i \leq 4500 & : 0.008 \cdot L_i^2 \\ else & : -300 \end{cases}$$

Unfortunately, SDP solutions for HMDPs suffers from two major drawbacks: (1) the overhead of computing a

complete optimal policy even for problems where the initial state is known; and (2) it cannot maintain compact (pruned) XADD representations of functions containing nonlinear decisions because it cannot verify their feasibility and remove or ignore infeasible regions. In this work, we address both of these problems simultaneously by proposing a new HMDP solver, named *Real-Time Symbolic Dynamic Programming (RTSDP)*. RTSDP addresses (1) by restricting the solution to states relevant to a specified initial state and addresses (2) by only expanding regions containing visited states, thus ignoring infeasible branches of nonlinear decisions. RTSDP solution is focused on the initial state by using simulation trials as done in Real-Time Dynamic Programming (RTDP).

We use the Real-Time Dynamic Programming (RTDP) algorithm because it is considered a state-of-the-art solver for MDPs ((Barto, Bradtke, and Singh 1995; Kolobov, Mausam, and Weld 2012)) that combines initial state information and value function heuristics with asynchronous updates to generate an optimal partial policy for the relevant states, i.e., states reachable from the initial state following an optimal policy. In order to be used in a hybrid state space, RTSDP modifies RTDP's single state update to a *region based* symbolic update. The XADD representation naturally factors the state-space into regions with the same continuous expressions, and thus allows efficient *region based updates* that improve the value function *on all states in the same region*. We claim that region updates promote better reusability of previous updates, improving the search efficiency. RTSDP is empirically tested on three challenging domains: INVENTORY CONTROL, with continuously parametrized actions; RESERVOIR MANAGEMENT, with a nonlinear reward model; and TRAFFIC CONTROL, with nonlinear dynamics. Our results show that, given an initial state, RTSDP can solve finite-horizon HMDP problems faster and using far less memory than SDP.

Hybrid Markov Decision Processes

An MDP (Puterman 1994) is defined by: a set of states $\mathcal{S}$, actions $\mathcal{A}$, a probabilistic transition function $\mathcal{P}$ and a reward function $\mathcal{R}$. In HMDPs, we consider a model factored in terms of variables (Boutilier, Hanks, and Dean 1999), i.e. a state $s \in \mathcal{S}$ is a pair of assignment vectors $(\vec{b}, \vec{x})$, where $\vec{b} \in \{0,1\}^n$ is boolean variables vector and each $\vec{x} \in \mathbb{R}^m$ is continuous variables vector, and the action set $\mathcal{A}$ is composed by a finite set of parametric actions, i.e. $\mathcal{A} = \{a_1(\vec{y}), \ldots, a_K(\vec{y})\}$, where $\vec{y}$ is the vector of parameter variables. The functions of state and action variables are compactly represented by exploiting structural independencies among variables in dynamic Bayesian networks (DBNs) (Dean and Kanazawa 1990).

We assume that the factored HMDP models satisfy the following: (i) next state boolean variables are probabilistically dependent on previous state variables and action; (ii) next state continuous variables are deterministically dependent on previous state variables, action and current state boolean variables; (iii) transition and reward functions are piecewise polynomial in the continuous variables. These assumptions allow us to write the probabilistic transition function $\mathcal{P}(s'|a(\vec{y}), s)$ in terms of state variables, i.e.:

$$\mathcal{P}(\vec{b'}, \vec{x'}|a(\vec{y}), \vec{b}, \vec{x}) = \prod_{i=1}^{n} \mathcal{P}(b'_i|a(\vec{y}), \vec{b}, \vec{x}) \prod_{j=1}^{m} \mathcal{P}(x'_j|\vec{b'}, a(\vec{y}), \vec{b}, \vec{x}).$$

Assumption (ii) implies the conditional probabilities for continuous variables are Dirac Delta functions, which correspond to deterministic transitions: $\vec{x'} \leftarrow \vec{T}_{a(\vec{y})}(\vec{b}, \vec{x}, \vec{b'})$. Though this restricts stochasticity to boolean variables, continuous variables depend on their current values, allowing the representation of general finite distributions, a common restriction in exact HMDP solutions (Feng et al. 2004; Meuleau et al. 2009; Zamani, Sanner, and Fang 2012).

In this paper, we consider finite-horizon HMDPs with an initial state $s_0 \in \mathcal{S}$ and horizon $H \in \mathbb{N}$. A non-stationary policy π is a function which specifies the action $a(\vec{y}) = \pi(s, h)$ to take in state $s \in \mathcal{S}$ at step $h \leq H$. The solution of an HMDP planning problem is an optimal policy π^* that maximizes the expected accumulated reward, i.e.:

$$V_H^{\pi^*}(s_0) = \mathbb{E}\left[\sum_{t=1}^{H} \mathcal{R}(s_t, a_t(\vec{y_t}))\Big| s_{t+1} \sim \mathcal{P}(s'|a_t(\vec{y_t}), s_t)\right],$$

where $a_t(\vec{y_t}) = \pi^*(s_t, t)$ is the action chosen by π^*.

Symbolic Dynamic Programming

The symbolic dynamic programming (SDP) algorithm (Sanner, Delgado, and de Barros 2011; Zamani, Sanner, and Fang 2012) is a generalisation of the classical *value iteration* dynamic programming algorithm (Bellman 1957) for constructing optimal policies for Hybrid MDPs. It proceeds by constructing a series of h stages-to-go optimal value functions $V_h(\vec{b}, \vec{x})$. The pseudocode of SDP is shown in Algorithm 1. Beginning with $V_0(\vec{b}, \vec{x}) = 0$ (line 1), it obtains the *quality* $Q_{h,a(\vec{y})}(\vec{b}, \vec{x})$ for each action $a(\vec{y}) \in \mathcal{A}$ (lines 4-5) in state $(\vec{b}, \vec{x})$ by regressing the expected reward for $h-1$ future stages $V_{h-1}(\vec{b}, \vec{x})$ along with the immediate reward $R(\vec{b}, \vec{x}, a(\vec{y}))$ as the following:

$$Q_{h,a(\vec{y})}(\vec{b}, \vec{x}) = R(\vec{b}, \vec{x}, a(\vec{y})) + \sum_{\vec{b'}} \int_{\vec{x'}} \left[\prod_i \mathcal{P}(b'_i|a(\vec{y}), \vec{b}, \vec{x}) \cdot \prod_j \mathcal{P}(x'_j|\vec{b'}, a(\vec{y}), \vec{b}, \vec{x}) \cdot V_{h-1}(\vec{b'}, \vec{x'}) \right]. \quad (1)$$

Note that since we assume deterministic transitions for continuous variables the integral over the continuous probabilistic distribution is actually a substitution:

$$\int_{\vec{x'}} \mathcal{P}(x'|\vec{b'}, a(\vec{y}), \vec{b}, \vec{x}) \cdot f(x') = subst_{x'=T_{a(\vec{y})}(\vec{b}, \vec{x}, \vec{b'})} f(x')$$

Given $Q_{h,a(\vec{y})}(\vec{b}, \vec{x})$ for each $a(\vec{y}) \in A$, we can define the h stages-to-go value function as the quality of the best (greedy) action in each state $(\vec{b}, \vec{x})$ as follows (line 6):

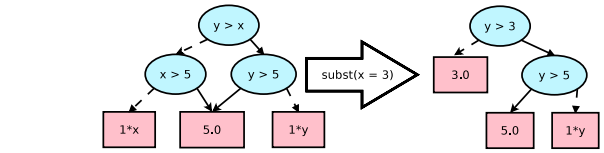
$$V_h(\vec{b}, \vec{x}) = \max_{a(\vec{y}) \in A} \left\{ Q_{h,a(\vec{y})}(\vec{b}, \vec{x}) \right\}. \quad (2)$$

The optimal value function $V_h(\vec{b}, \vec{x})$ and optimal policy π_h^* for each stage h are computed after h steps. Note that the policy is obtained from the maximization in Equation 2, $\pi_h^*(\vec{b}, \vec{x}) = \arg\max_{a(\vec{y})} Q_{h,a(\vec{y})}(\vec{b}, \vec{x})$ (line 7).

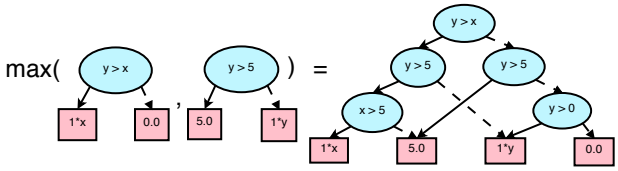
Algorithm 1: SDP(HMDP M, H) $\rightarrow (V^h, \pi_h^*)$
(Zamani, Sanner, and Fang 2012)

$V_0 := 0, h := 0$
while $h < H$ **do**
  $h := h + 1$
  foreach $a \in A$ **do**
    $Q_{a(\vec{y})} \leftarrow \texttt{Regress}\,(V_{h-1}, a(\vec{y}))$
  $V_h \leftarrow \max_{a(\vec{y})}\ Q_{a(\vec{y})}$ *// Maximize all* $Q_{a(\vec{y})}$
  $\pi_h^* \leftarrow \arg\max_{a(\vec{y})}\ Q_{a(\vec{y})}$
return (V_h, π_h^*)

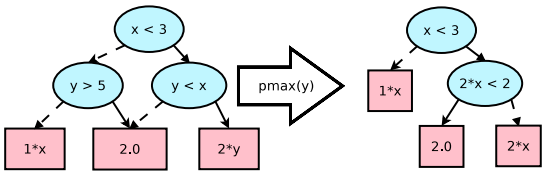
Procedure 1.1: $\texttt{Regress}\,(V_{h-1}, a(\vec{y}))$
$Q = \texttt{Prime}\,(V_{h-1})$ *// Rename all symbolic variables - // $b_i \rightarrow b_i'$ and all $x_i \rightarrow x_i'$*
foreach b_i' *in* Q **do**
  // Discrete marginal summation, also denoted $\sum_{b'}$
  $Q \leftarrow \left[Q \otimes P(b_i'|\vec{b}, \vec{x}, a(\vec{y}))\right]|_{b_i'=1}$
    $\oplus \left[Q \otimes P(b_i'|\vec{b}, \vec{x}, a(\vec{y}))\right]|_{b_i'=0}$
foreach x_j' *in* Q **do**
  //Continuous marginal integration
  $Q \leftarrow subst_{x_j'=T_{a(\vec{y})}(\vec{b},\vec{x},\vec{b'})} Q$
return $Q \oplus R(\vec{b}, \vec{x}, a(\vec{y}))$



(a) Substitution ($subst$).



(b) Maximization ($\max$)



(c) Parameter Maximization (pmax).

Figure 2: Examples of basic XADD operations.

XADD representation

XADDs (Sanner, Delgado, and de Barros 2011) are directed acyclic graphs that efficiently represent and manipulate piecewise polynomial functions by using decision nodes for separating regions and terminal nodes for the local functions. Internal nodes are decision nodes that contain two edges, namely *true* and *false* branches, and a decision that can be either a boolean variable or a polynomial inequality on the continuous variables. Terminal nodes contain a polynomial expression on the continuous variables. Every path in an XADD defines a partial assignment of boolean variables and a continuous region.

The SDP algorithm uses four kinds of XADD operations (Sanner, Delgado, and de Barros 2011; Zamani, Sanner, and Fang 2012): (i) *Algebraic operations* (sum $\oplus$, product $\otimes$), naturally defined within each region, so the resulting partition is a cross-product of the original partitions;(ii) *Substitution operations*, which replace or instantiate variables by modifying expressions on both internal and terminal nodes.(See Figure 2a); (iii) *Comparison operations* (maximization), which cannot be performed directly between terminal nodes so new decision nodes comparing terminal expressions are created (See Figure 2b); (iv) *Parameter maximization operations*, (*pmax* or $\max_{\vec{y}}$) remove parameter variables from the diagram by replacing them with values that maximize the expressions in which they appear. (See Figure 2c).

Thus, the SDP Bellman update (Equations 1 & 2) is performed synchronously as a sequence of XADD operations [1], in two steps:

- Regression of action a for all states:

$$Q_{h,a}^{DD(\vec{b},\vec{x},\vec{y})} = R_a^{DD(\vec{b},\vec{x},\vec{y})} \oplus \sum_{\vec{b'}} P_a^{DD(\vec{b},\vec{x},\vec{y},\vec{b'})} \otimes \left[subst_{(x'=T_a^{DD(\vec{b},\vec{x},\vec{y},\vec{b'})})} V_{h-1}^{\prime DD(b',x')} \right]. \quad (3)$$

- Maximization for all states w.r.t. the action set $\mathcal{A}$:

$$V_h^{DD(\vec{b},\vec{x})} = \max_{a \in \mathcal{A}} \left(\max_{\vec{y}}\ Q_{h,a}^{DD(\vec{b},\vec{x},\vec{y})} \right). \quad (4)$$

As these operations are performed the complexity of the value function increases and so does the number of regions and nodes in the XADD. A goal of this work is to propose an efficient solution that avoids exploring unreachable or unnecessary regions.

Real-Time Symbolic Dynamic Programming

First, we describe the RTDP algorithm for MDPs which motivates the RTSDP algorithm for HMDPs.

The RTDP algorithm

The RTDP algorithm (Barto, Bradtke, and Singh 1995) (Algorithm 2) starts from an initial admissible heuristic value function V ($V(s) > V^*(s), \forall s$) and performs trials to iteratively improve by monotonically decreasing the estimated value. A trial, described in Procedure 2.1, is a simulated execution of a policy interleaved with local Bellman updates on the visited states. Starting from a known initial state, an update is performed, the greedy action is taken, a next state

[1] $F^{DD(vars)}$ is a notation to indicate that F is an XADD symbolic function of $vars$.

is sampled and the trial goes on from this new state. Trials are stopped when their length reaches the horizon H.

The update operation (Procedure 2.2) uses the current value function V to estimate the future reward and evaluates every action a on the current state in terms of its quality Q_a. The greedy action, a_g, is chosen and the state's value is set to Q_{a_g}. RTDP performs updates on one state at a time and only on relevant states, i.e. states that are reachable under the greedy policy. Nevertheless, if the initial heuristic is admissible, there is an optimal policy whose relevant states are visited infinitely often in RTDP trials. Thus the value function converges to V^* on these states and an optimal partial policy closed for the initial state is obtained. As the initial state value depends on all relevant states' values, it is the last state to convergence.

Algorithm 2: `RTDP`(MDP M, s_0, H, V) $\longrightarrow V_H^*(s_0)$

while $\neg$ `Solved`$(M, s_0, V) \wedge \neg$`TimeOut`() **do**
  $V \leftarrow$ `MakeTrial`(M, s_0, V, H)
return V

Procedure 2.1: `MakeTrial` (*MDP M, s, V, h*)
if $s \in GoalStates$ *or* $h = 0$ **then**
  return V
$V_h(s), a_g \leftarrow$ `Update`(M, s, V, h)
// Sample next state from $P(s, a, s')$
$s^{new} \leftarrow$ `Sample`(s, a_g, M)
return `MakeTrial`$(M, s^{new}, V, h-1)$

Procedure 2.2: `Update` (*MDP M, s, V, h*)
foreach $a \in A$ **do**
  $Q_a \leftarrow R(s,a) + \sum_{s' \in S} P(s,a,s') \cdot [V_{h-1}(s')]$
$a_g \leftarrow \arg\max_a Q_a$
$V_h(s) \leftarrow Q_{a_g}$
return $V_h(s), a_g$

The RTSDP Algorithm

We now generalize the RTDP algorithm for HMDPs. The main `RTDP` (Algorithm 2) and `makeTrial` (Procedure 2.1) functions are trivially extended as they are mostly independent of the state or action structure, for instance, verifying if current state is a goal is now checking if the current state belongs to a goal region instead of a finite goal set. The procedure that requires a non-trivial extension is the `Update` function (Procedure 2.2). Considering the continuous state space, it is inefficient to represent values for individual states, therefore we define a new type of update, named *region update*, which is one of the contributions of this work.

While the synchronous SDP update (Equations 3 and 4) modifies the expressions for all paths of $V^{DD(\vec{b},\vec{x})}$, the region update only modifies a single path, the one that corresponds to the region containing the current state $(\vec{b_c}, \vec{x_c})$, denoted by Ω. This is performed by using a "mask", which restricts the update operation to region Ω. The "mask" is an XADD symbolic indicator of whether any state $(\vec{b}, \vec{x})$ belongs to Ω, i.e.:

$$I[(\vec{b},\vec{x}) \in \Omega] = \begin{cases} 0, & \text{if } (\vec{b},\vec{x}) \in \Omega \\ +\infty, & otherwise, \end{cases} \quad (5)$$

where 0 indicates valid regions and $+\infty$ invalid ones. The mask is applied in an XADD with a sum ($\oplus$) operation, the valid regions stay unchanged whereas the invalid regions change to $+\infty$. The mask is applied to probability, transition and reward functions restricting these operations to Ω, as follows:

$$T_{a,\Omega}^{DD(\vec{b},\vec{x},\vec{y},\vec{b'},\vec{x'})} = T_a^{DD(\vec{b},\vec{x},\vec{y},\vec{b'},\vec{x'})} \oplus I[(b,x) \in \Omega] \quad (6)$$

$$P_{a,\Omega}^{DD(\vec{b},\vec{x},\vec{y},\vec{b'})} = P_a^{DD(\vec{b},\vec{x},\vec{y},\vec{b'})} \oplus I[(b,x) \in \Omega] \quad (7)$$

$$R_{a,\Omega}^{DD(\vec{b},\vec{x},\vec{y})} = R_a^{DD(\vec{b},\vec{x},\vec{y})} \oplus I[(b,x) \in \Omega] \quad (8)$$

Finally, we define the *region update* using these restricted functions on symbolic XADD operations:

- Regression of action a for region Ω:

$$Q_{h,a,\Omega}^{DD(\vec{b},\vec{x},\vec{y})} = R_{a,\Omega}^{DD(\vec{b},\vec{x},\vec{y})} \oplus \sum_{\vec{b'}} P_{a,\Omega}^{DD(\vec{b},\vec{x},\vec{y},\vec{b'})} \otimes \left[subst_{(x'=T_{a,\Omega}^{DD(\vec{b},\vec{x},\vec{y},\vec{b'})})} V_{h-1}'^{DD(b',x')} \right]. \quad (9)$$

- Maximization for region Ω w.r.t. the action set $\mathcal{A}$:

$$V_{h,\Omega}^{DD(\vec{b},\vec{x})} = \max_{a \in \mathcal{A}} \left(\max_{\vec{y}} Q_{h,a,\Omega}^{DD(\vec{b},\vec{x},\vec{y})} \right). \quad (10)$$

- Update of the value function V_h for region Ω:

$$V_{h,new}^{DD(b,x)} \leftarrow \begin{cases} V_{h,\Omega}^{DD(b,x)}, & \text{if } (\vec{b},\vec{x}) \in \Omega \\ V_{h,old}^{DD(b,x)}, & otherwise, \end{cases} \quad (11)$$

  which is done as an XADD minimization $V_{h,new}^{DD(b,x)} \leftarrow \min(V_{h,\Omega}^{DD(b,x)}, V_{h,old}^{DD(b,x)})$. This minimum is the correct new value of V_h because $V_{h,\Omega} = +\infty$ for the invalid regions and $V_{h,\Omega} \leq V_{h,old}$ on the valid regions, thus preserving the V_h as an upper bound on V_h^*, as required to RTDP convergence.

The new update procedure is described in Algorithm 3. Note that the current state is denoted by $(\vec{b_c}, \vec{x_c})$ to distinguish it from symbolic variables $(\vec{b}, \vec{x})$.

Empirical Evaluation

In this section, we evaluate the RTSDP algorithm for three domains: RESERVOIR MANAGEMENT (Problem 1), INVENTORY CONTROL (Problem 2) and TRAFFIC CONTROL (Problem 3). These domains show different challenges for planning: INVENTORY CONTROL contains continuously parametrized actions, RESERVOIR MANAGEMENT has a nonlinear reward function, and TRAFFIC CONTROL has nonlinear dynamics. For all of our experiments RTSDP value function was initialised with an admissible max cumulated reward heuristic, that is $V_h(s) =$

Algorithm 3: `Region-update`(HMDP,$(\vec{b_c},\vec{x_c})$,V, h)$\longrightarrow$ $V_h, a_g, \vec{y}_g$

1 //`GetRegion` *extracts XADD path for* $(\vec{b_c},\vec{x_c})$)
2 $I[(\vec{b},\vec{x}) \in \Omega] \leftarrow$ `GetRegion`$(V_h, (\vec{b_c},\vec{x_c}))$
3 $V'^{DD(b',x')} \leftarrow$`Prime` $(V_{h-1}^{DD(b,x)})$ *//Prime variable names*
4 **foreach** $a(\vec{y}) \in A$ **do**
5 $\quad Q_{a,\Omega}^{DD(\vec{b},\vec{x},\vec{y})} \leftarrow R_{a,\Omega}^{DD(\vec{b},\vec{x},\vec{y})} \oplus \sum_{\vec{b'}} P_{a,\Omega}^{DD(\vec{b},\vec{x},\vec{y},\vec{b'})} \otimes \left[subst_{(x'=T_{a,\Omega}^{DD(\vec{b},\vec{x},\vec{y},\vec{b'})})} V'^{DD(b',x')} \right]$
6 $\quad \vec{y}_g{}^a \leftarrow \arg\max_{\vec{y}} Q_{a,\Omega}^{DD(\vec{b},\vec{x},\vec{y})}$ *//Maximizing parameter*
7 $V_{h,\Omega}^{DD(\vec{b},\vec{x})} \leftarrow \max_a \left(\text{pmax}_{\vec{y}}\ Q_{a,\Omega}^{DD(\vec{b},\vec{x},\vec{y})} \right)$
8 $a_g \leftarrow \arg\max_a \left(Q_{a,\Omega}^{DD(\vec{b},\vec{x})}(\vec{y}_g{}^a) \right)$
9 //The value is updated through a minimisation.
10 $V_h^{DD(b,x)} \leftarrow \min(V_h^{DD(b,x)}, V_{h,\Omega}^{DD(b,x)})$
11 **return** $V_h(b_c, x_c), a_g, \vec{y}_g{}^{a_g}$

$h \cdot \max_{s,a} \mathcal{R}(s,a) \forall s$. This is a very simple to compute and very little informative heuristic intended to show that RTSDP is an improvement to SDP even without relying on good heuristics. We show the efficiency of our algorithm by comparing it with synchronous SDP in solution time and value function size.

Problem 2 - INVENTORY CONTROL (Scarf 2002). *An inventory control problem, consists of determining what itens from the inventory should be ordered and how much to order of each. There is a continuous variable x_i for each item i and a single action $order$ with continuous parameters dx_i for each item. There is a fixed limit L for the maximum number of items in the warehouse and a limit l_i for how much of an item can be ordered at a single stage. The items are sold according to a stochastic demand, modelled by boolean variables d_i that represent whether demand is high (Q units) or low (q units) for item i. A reward is given for each sold item, but there are linear costs for items ordered and held in the warehouse. For item i, the reward is as follows:*

$$R_i(x_i, d_i, dx_i) = \begin{cases} \text{if } d_i \wedge x_i + dx_i > Q & : Q - 0.1x_i - 0.3dx_i \\ \text{if } d_i \wedge x_i + dx_i < Q & : 0.7x_i - 0.3dx_i \\ \text{if } \neg d_i \wedge x_i + dx_i > q & : q - 0.1x_i - 0.3dx_i \\ \text{if } \neg d_i \wedge x_i + dx_i < q & : 0.7x_i - 0.3dx_i \end{cases}$$

Problem 3 - *Nonlinear* TRAFFIC CONTROL (Daganzo 1994). *This domain describes a ramp metering problem using a cell transmission model. Each segment i before the merge is represented by a continuous car density k_i and a probabilistic boolean hi_i indicator of higher incoming car density. The segment c after the merge is represented by its continuous car density k_c and speed v_c. The actions correspond to traffic light cycles that give more time to each segment, so that the controller can give priority to segments with a greater density. The quantity of cars that pass the merge is given by:*

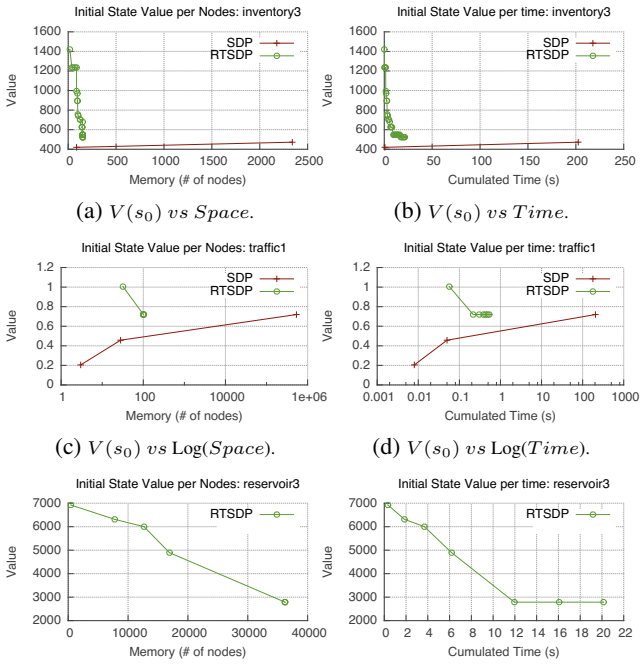
$$q_{i,a_j}(k_i, v_c) = \alpha_{i,j} \cdot k_i \cdot v_c,$$

where $\alpha_{i,j}$ is the fraction of cycle time that action a_j gives to segment i. The speed v_c in the segment after the merge is obtained as a **nonlinear** *function of its density k_c:*

$$v_c' = \begin{cases} \text{if } k_c \leq 0.25 & : 0.5 \\ \text{if } 0.25 < k_c \leq 0.5 & : 0.75 - k_c \\ \text{if } 0.5 < k_c & : k_c^2 - 2k_c + 1 \end{cases}$$

The reward obtained is the amount of car density that escapes from the segment after the merge, which is bounded.

In Figure 3 we show the convergence of RTSDP to the optimal initial state value $V_H(s_0)$ with each point corresponding to the current $V_H(s_0)$ estimate at the end of a trial. The SDP curve shows the optimal initial state value for different stages-to-go, i.e. $V_h(s_0)$ is the value of the *h-th* point in the plot. Note that for INVENTORY CONTROL and TRAFFIC CONTROL, RTSDP reaches the optimal value much earlier than SDP and its value function has far fewer nodes. For RESERVOIR MANAGEMENT, SDP could not compute the solution due to the large state space.



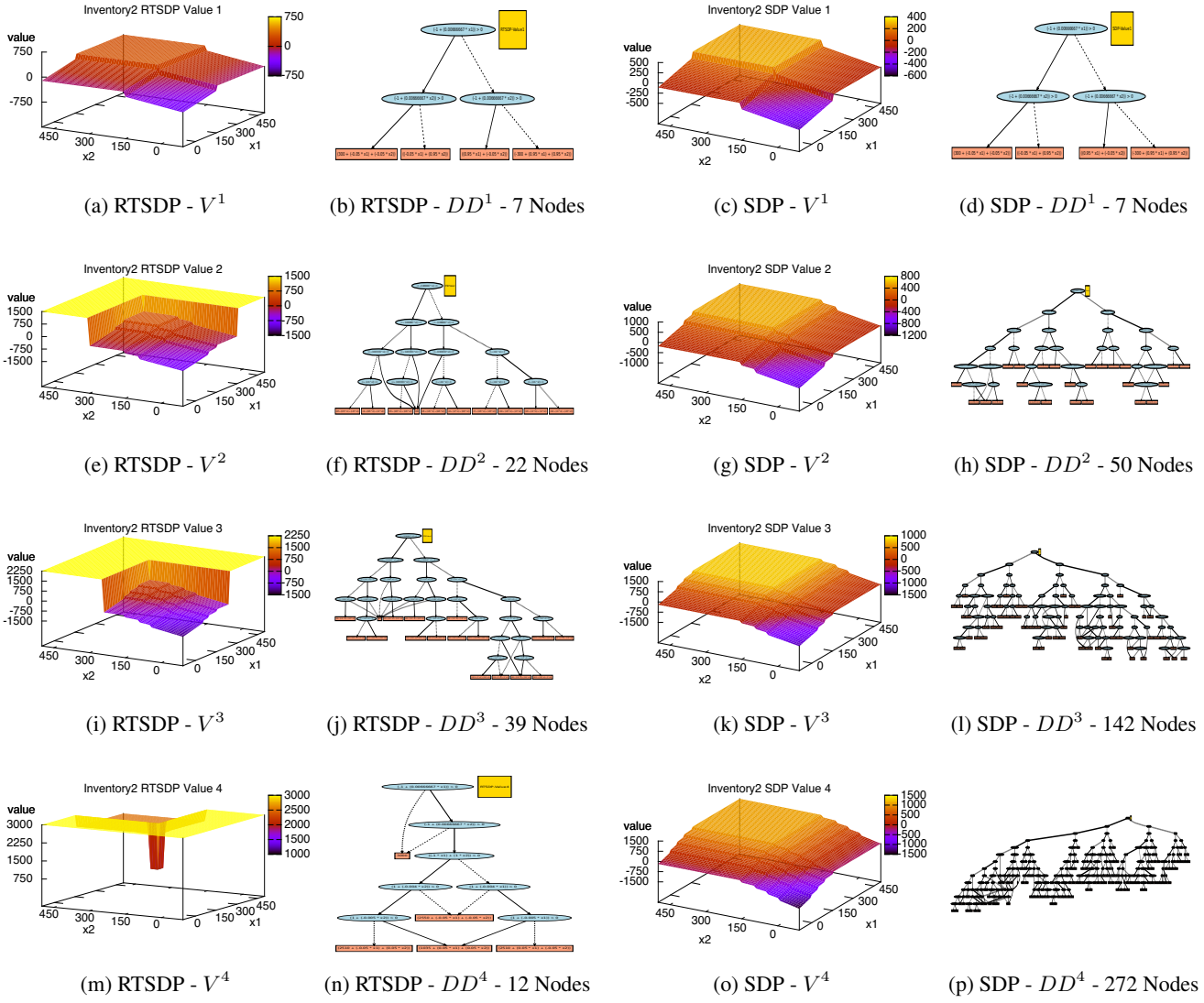
(a) $V(s_0)$ *vs Space.* (b) $V(s_0)$ *vs Time.*

(c) $V(s_0)$ *vs* Log(*Space*). (d) $V(s_0)$ *vs* Log(*Time*).

(e) $V(s_0)$ *vs Space.* (f) $V(s_0)$ *vs Time.*

Figure 3: RTSDP convergence and comparison to SDP on three domains: INVENTORY CONTROL (*top* - 3 continuous Vars.), TRAFFIC CONTROL (*middle* - 4 continuous Vars.) and RESERVOIR MANAGEMENT (*bottom* - 3 continuous Vars.). Plots of initial state value vs space ($left$) and time ($right$) at every as RTSDP trial or SDP iteration.

Figure 4 shows the value functions for different stages-to-go generated by RTSDP *(left)* and SDP *(right)* when solving INVENTORY CONTROL2, an instance with 2 continuous state variables and 2 continuous action parameters. The



(a) RTSDP - V^1 (b) RTSDP - DD^1 - 7 Nodes (c) SDP - V^1 (d) SDP - DD^1 - 7 Nodes

(e) RTSDP - V^2 (f) RTSDP - DD^2 - 22 Nodes (g) SDP - V^2 (h) SDP - DD^2 - 50 Nodes

(i) RTSDP - V^3 (j) RTSDP - DD^3 - 39 Nodes (k) SDP - V^3 (l) SDP - DD^3 - 142 Nodes

(m) RTSDP - V^4 (n) RTSDP - DD^4 - 12 Nodes (o) SDP - V^4 (p) SDP - DD^4 - 272 Nodes

Figure 4: Value functions generated by RTSDP and SDP on INVENTORY CONTROL, with $H = 4$.

value functions are shown as 3D plots (continuous state variables vs state value) with their corresponding XADDs. Since RTSDP runs all trials from the initial state, the value function for H stages-to-go, V_H *(bottom)* is always updated on the same region: *the one containing the initial state*. The updates insert new decisions to the XADD, splitting the region containing s_0 into smaller regions, only the small region that still contains the initial state will keep being updated (See Figure 4m).

An interesting observation is how the solution (XADD) size changes for the different stages-to-go value functions. For SDP (Figures 4d, 4h, 4l and 4p), the number of regions increases quickly with horizon (from top to bottom). However, RTSDP (Figures 4b, 4f, 4j and 4n) shows a different pattern: with few stages-to-go, the number of regions increases slowly with horizon, and for the target horizon (4n), the solution size is very small. This is explained because the number of reachable regions from the initial state in a few steps is very small and therefore the number of terminal nodes and overall XADD size is not very large for greater horizons. Note that this important difference in solution size is also reflected in the 3D plots, regions that are no longer relevant in a certain horizon stop being updated and remain with theirs last heuristical value (e.g. the plateau with value 2250 at V^3).

Table 1 presents performance results of RTSDP and SDP in various instances of all domains. Note that the performance gap between SDP and RTSDP grows impressively with the number of continuous variables, because restricting the solution to relevant states becomes necessary. SDP is unable to solve most nonlinear problems since it cannot prune nonlinear regions, while RTSDP restricts expansions to vis-

Instance (#Cont.Vars)	SDP		RTSDP	
	Time (s)	Nodes	Time (s)	Nodes
Inventory 1 (2)	0.97	43	1.03	37
Inventory 2 (4)	2.07	229	1.01	51
Inventory 3 (6)	202.5	2340	21.4	152
Reservoir 1 (1)	Did not finish		1.6	1905
Reservoir 2 (2)	Did not finish		11.7	7068
Reservoir 3 (3)	Did not finish		160.3	182000
Traffic 1 (4)	204	533000	0.53	101
Traffic 2 (5)	Did not finish		1.07	313

Table 1: Time and space comparison of SDP and RTSDP in various instances of the 3 domains.

ited states and has a much more compact value function.

Conclusion

We propose a new solution, the $RTSDP$ algorithm, for a general class of Hybrid MDP problems including linear dynamics with parametrized actions, as in INVENTORY CONTROL, or nonlinear dynamics, as in TRAFFIC CONTROL. RTSDP uses the initial state information efficiently to (1) visit only a small fraction of the state space and (2) avoid representing infeasible nonlinear regions. In this way, it overcomes the two major drawbacks of SDP and greatly surpasses its performance into a more scalable solution for linear and nonlinear problems that greatly expand the set of domains where SDP could be used. RTSDP can be further improved with better heuristics that may greatly reduce the number of relevant regions in its solution. Such heuristics might be obtained using bounded approximate HMDP solvers (Vianna, Sanner, and de Barros 2013).

Acknowledgment

This work has been supported by the Brazilian agency FAPESP (under grant 2011/16962-0). NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

References

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 Lightner, M. R., and Jess, J. A. G., eds., *ICCAD*, 188–191. IEEE Computer Society.

Barto, A.; Bradtke, S.; and Singh, S. 1995. Learning to act using real-time dynamic programming. *Artificial Intelligence* 72(1-2):81–138.

Bellman, R. E. 1957. *Dynamic Programming*. USA: Princeton University Press.

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

Daganzo, C. F. 1994. The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory. *Transportation Research Part B: Methodological* 28(4):269–287.

Dean, T., and Kanazawa, K. 1990. A model for reasoning about persistence and causation. *Computational Intelligence* 5(3):142–150.

Feng, Z.; Dearden, R.; Meuleau, N.; and Washington, R. 2004. Dynamic programming for structured continuous markov decision problems. In *Proceedings of the UAI-04*, 154–161.

Kolobov, A.; Mausam; and Weld, D. 2012. LRTDP Versus UCT for online probabilistic planning. In *AAAI Conference on Artificial Intelligence*.

Mahootchi, M. 2009. *Storage System Management Using Reinforcement Learning Techniques and Nonlinear Models*. Ph.D. Dissertation, University of Waterloo,Canada.

Meuleau, N.; Benazera, E.; Brafman, R. I.; Hansen, E. A.; and Mausam. 2009. A heuristic search approach to planning with continuous resources in stochastic domains. *Journal Artificial Intelligence Research (JAIR)* 34:27–59.

Puterman, M. L. 1994. *Markov decision processes*. New York: John Wiley and Sons.

Sanner, S.; Delgado, K. V.; and de Barros, L. N. 2011. Symbolic dynamic programming for discrete and continuous state MDPs. In *Proceedings of UAI-2011*.

Scarf, H. E. 2002. Inventory theory. *Operations Research* 50(1):186–191.

Vianna, L. G. R.; Sanner, S.; and de Barros, L. N. 2013. Bounded approximate symbolic dynamic programming for hybrid MDPs. In *Proceedings of the Twenty-Ninth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-13)*, 674–683. Corvallis, Oregon: AUAI Press.

Yeh, W. G. 1985. Reservoir management and operations models: A state-of-the-art review. *Water Resources research* 21,12:17971818.

Zamani, Z.; Sanner, S.; and Fang, C. 2012. Symbolic dynamic programming for continuous state and action MDPs. In Hoffmann, J., and Selman, B., eds., *AAAI*. AAAI Press.