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.

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, and no-drain, 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:

\[
L'_1 = \begin{cases} 
\text{if } \text{rain} \land \text{drain}_1 & : L_1 + 1000 - 0.5L_1 \\
\text{if } \neg\text{rain} \land \neg\text{drain}_1 & : L_1 + 500 - 0.5L_1 \\
\text{if } \text{rain} \land \neg\text{drain}_1 & : L_1 + 1000 \\
\text{if } \neg\text{rain} \land \text{drain}_1 & : L_1 + 500
\end{cases}
\]

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.

\[
R_{\text{drain}_i}(L_i) = \begin{cases} 
\text{if } 50 \leq L_i \leq 4500 & : 0.008 \cdot L_i^2 \\
\text{else} & : -300
\end{cases}
\]

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

\[
\Phi \begin{cases} 
\text{if } \text{rain} \land \text{drain}_1 & : \Phi - 0.5L_1 \\
\text{if } \neg\text{rain} \land \neg\text{drain}_1 & : \Phi - 0.5L_1 \\
\text{if } \text{rain} \land \neg\text{drain}_1 & : \Phi + 1000 \\
\text{if } \neg\text{rain} \land \text{drain}_1 & : \Phi + 500
\end{cases}
\]

(1)
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. The 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 $S$, actions $A$, a probabilistic transition function $P$ and a reward function $R$. In HMDPs, we consider a model factored in terms of variables (Boutilier, Hanks, and Dean 1999), i.e. a state $s \in S$ 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 $A$ is composed by a finite set of parametric actions, i.e. $A = \{a_1(y), \ldots, a_K(y)\}$, 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 $P(s'|a(y), s)$ in terms of state variables, i.e.:

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

Assumption (ii) implies the conditional probabilities for continuous variables are Dirac Delta functions, which correspond to conditional deterministic transitions: $x'_i \leftarrow T_{a(y)}(\vec{b}, \vec{x}, \vec{b}')$. An example of conditional deterministic transition is given in Problem 1, the definition of $L'_1$ in Equation 1. 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 $s_0 \in S$ and horizon $H \in \mathbb{N}$. A non-stationary policy $\pi$ is a function which specifies the action $a(y) = \pi(s, h)$ to take in state $s \in S$ at step $h \leq H$. The solution of an HMDP planning problem is an optimal policy $\pi^*$ that maximizes the expected accumulated reward, i.e.:

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

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

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 $V_1(\vec{b}, \vec{x}) = 0$ (line 1), it obtains the quality $Q_{h,a(y)}(\vec{b}, \vec{x})$ for each action $a(y) \in A$ (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 $R(\vec{b}, \vec{x}, a(y))$ as the following:

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

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

$$\int_{x'} P(x'_i|a(y), \vec{b}, \vec{x}) \cdot f(x'_i) = \text{sub}t_{x'_i=T_{a(y)}(\vec{b}, \vec{x}, \vec{b})} f(x'_i)$$

Given $Q_{h,a(y)}(\vec{b}, \vec{x})$ for each $a(y) \in A$, 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):

$$V_h(\vec{b}, \vec{x}) = \max_{a(y) \in A} \left\{ Q_{h,a(y)}(\vec{b}, \vec{x}) \right\}.$$
The optimal value function $V_h(\vec{b}, \vec{x})$ and optimal policy $\pi^*_h$ for each stage $h$ are computed after $h$ steps. Note that the policy is obtained from the maximization in Equation 3, $\pi^*_h(\vec{b}, \vec{x}) = \arg \max_{a(\vec{y})} Q_{h,a(\vec{y})}(\vec{b}, \vec{x})$ (line 7).

Algorithm 1: SDP (HMDP M, H) \to (V^b, \pi^*_h)
(Zamani, Sanner, and Fang 2012)

1. $V_0 := 0, h := 0$
2. while $h < H$ do
3. \hspace{1em} $h := h + 1$
4. \hspace{1em} foreach $a \in A$ do
5. \hspace{2em} $Q_{a(\vec{y})} \leftarrow \text{Regress (} V_{h-1}, a(\vec{y}) \text{)}$
6. \hspace{2em} $V_h \leftarrow \max_{a(\vec{y})} Q_{a(\vec{y})}$ // Maximize all $Q_{a(\vec{y})}$
7. \hspace{2em} $\pi_h \leftarrow \arg \max_{a(\vec{y})} Q_{a(\vec{y})}$
8. \hspace{1em} return $(V_h, \pi^*_h)$
9. Procedure 1.1: Regress $(V_{h-1}, a(\vec{y}))$
10. $Q := \text{Prime (} V_{h-1} \text{)}$ // Rename all symbolic variables -
11. \hspace{1em} foreach $b'_i$ in $Q$ do
12. \hspace{2em} // Discrete marginal summation, also denoted $\sum_{b'_i}$
13. \hspace{2em} $Q \leftarrow [Q \oplus P(b'_i, \vec{b}, \vec{x}, a(\vec{y}))]_{|b'_i=1}$
14. \hspace{2em} $\oplus [Q \oplus P(b'_i, \vec{b}, \vec{x}, a(\vec{y}))]_{|b'_i=0}$
15.\hspace{1em}foreach $x'_i$ in $Q$ do
16. \hspace{2em} //Continuous marginal integration
17. \hspace{2em} $Q \leftarrow \text{subst}_{x'_i = T_0(a(\vec{y}), \vec{x}, \vec{y})} Q$
18. \hspace{1em} return $Q \oplus R(\vec{b}, \vec{x}, a(\vec{y}))$

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.

Figure 2: Examples of basic XADD operations.

Thus, the SDP Bellman update (Equations 2 & 3) is performed synchronously as a sequence of XADD operations, in two steps:

- Regression of action $a$ for all states:
  \[ Q^{DD}_{h,a}(\vec{b}, \vec{x}, \vec{y}) = R^{DD}_{a}(\vec{b}, \vec{x}, \vec{y}) \oplus \sum_{\vec{y}'} P^{DD}_{h,a}(\vec{b}, \vec{x}, \vec{y}, \vec{y}') \otimes \text{subst}_{x'_j = T_a(\vec{b}, \vec{x}, \vec{y}, \vec{y}')} V^{DD}_{h-1}(x', y'). \]  \( (4) \)

- Maximization for all states w.r.t. the action set $\mathcal{A}$:
  \[ V^{DD}_{h,\vec{b},\vec{x}} = \max_{a(\vec{y})} \left( \max_{\vec{y}} Q^{DD}_{h,a}(\vec{b}, \vec{x}, \vec{y}) \right). \]  \( (5) \)

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 function.

Thus, the SDP Bellman update (Equations 2 & 3) is performed synchronously as a sequence of XADD operations, in two steps:

- Regression of action $a$ for all states:
  \[ Q^{DD}_{h,a}(\vec{b}, \vec{x}, \vec{y}) = R^{DD}_{a}(\vec{b}, \vec{x}, \vec{y}) \oplus \sum_{\vec{y}'} P^{DD}_{h,a}(\vec{b}, \vec{x}, \vec{y}, \vec{y}') \otimes \text{subst}_{x'_j = T_a(\vec{b}, \vec{x}, \vec{y}, \vec{y}')} V^{DD}_{h-1}(x', y'). \]  \( (4) \)

- Maximization for all states w.r.t. the action set $\mathcal{A}$:
  \[ V^{DD}_{h,\vec{b},\vec{x}} = \max_{a(\vec{y})} \left( \max_{\vec{y}} Q^{DD}_{h,a}(\vec{b}, \vec{x}, \vec{y}) \right). \]  \( (5) \)

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

Thus, the SDP Bellman update (Equations 2 & 3) is performed synchronously as a sequence of XADD operations, in two steps:

- Regression of action $a$ for all states:
  \[ Q^{DD}_{h,a}(\vec{b}, \vec{x}, \vec{y}) = R^{DD}_{a}(\vec{b}, \vec{x}, \vec{y}) \oplus \sum_{\vec{y}'} P^{DD}_{h,a}(\vec{b}, \vec{x}, \vec{y}, \vec{y}') \otimes \text{subst}_{x'_j = T_a(\vec{b}, \vec{x}, \vec{y}, \vec{y}')} V^{DD}_{h-1}(x', y'). \]  \( (4) \)

- Maximization for all states w.r.t. the action set $\mathcal{A}$:
  \[ V^{DD}_{h,\vec{b},\vec{x}} = \max_{a(\vec{y})} \left( \max_{\vec{y}} Q^{DD}_{h,a}(\vec{b}, \vec{x}, \vec{y}) \right). \]  \( (5) \)

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

Thus, the SDP Bellman update (Equations 2 & 3) is performed synchronously as a sequence of XADD operations, in two steps:

- Regression of action $a$ for all states:
  \[ Q^{DD}_{h,a}(\vec{b}, \vec{x}, \vec{y}) = R^{DD}_{a}(\vec{b}, \vec{x}, \vec{y}) \oplus \sum_{\vec{y}'} P^{DD}_{h,a}(\vec{b}, \vec{x}, \vec{y}, \vec{y}') \otimes \text{subst}_{x'_j = T_a(\vec{b}, \vec{x}, \vec{y}, \vec{y}')} V^{DD}_{h-1}(x', y'). \]  \( (4) \)

- Maximization for all states w.r.t. the action set $\mathcal{A}$:
  \[ V^{DD}_{h,\vec{b},\vec{x}} = \max_{a(\vec{y})} \left( \max_{\vec{y}} Q^{DD}_{h,a}(\vec{b}, \vec{x}, \vec{y}) \right). \]  \( (5) \)

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.
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 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 converge.

**Algorithm 2: RTDP (MDP $M, s_0, H, V) \rightarrow V^*_\phi(s_0)$**

```
1 while $\neg$ Solved$(M, s_0, V) \land \neg$ TimeOut() do
2     $V \leftarrow$ MakeTrial$(M, s_0, V, H)$
3     return $V$
4
5 Procedure 2.1: MakeTrial (MDP $M, s, V, h$)
6     if $s \in$ GoalStates or $h = 0$ then
7         return $V$
8     $V_h(s), a_g \leftarrow$ Update$(M, s, V, h)$
9     // Sample next state from $P(s, a, s')$
10    $s_{\text{new}} \leftarrow$ Sample$(s, a_g, M)$
11    return MakeTrial$(M, s_{\text{new}}, V, h - 1)$
12
13 Procedure 2.2: Update (MDP $M, s, V, h$)
14    for each $a \in A$ do
15        $Q_a \leftarrow R(s, a) + \sum_{s' \in S} P(s, a, s') \cdot V_{h-1}(s')$
16        $a_g \leftarrow \text{arg max}_a Q_a$
17        $V_h(s) \leftarrow Q_{a_g}$
18    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 4 and 5) 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 $\vec{s}$, denoted by $\Omega$. This is performed by using a “mask”, which restricts the update operation to region $\Omega$. The “mask” is an XADD symbolic indicator of whether any state $\vec{s}$ belongs to $\Omega$, i.e.:

$$I[\vec{s}] = \begin{cases} 0, & \text{if } \vec{s} \in \Omega \\ +\infty, & \text{otherwise} \end{cases}$$

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 $\Omega$, as follows:

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

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

$$R_{\alpha, \Omega}^{DD}(\vec{b}, \vec{x}, \vec{y}) = R_{\alpha}^{DD}(\vec{b}, \vec{x}, \vec{y}) \oplus I[(\vec{b}, \vec{x}) \in \Omega] \quad (9)$$

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

- **Regression of action $a$ for region $\Omega$:**

$$Q_{h, \alpha, \Omega}^{DD}(\vec{b}, \vec{x}, \vec{y}) = R_{h, \Omega}^{DD}(\vec{b}, \vec{x}, \vec{y}) \oplus \sum_{\vec{b}', \vec{x}, \vec{y}} P_{\alpha, \Omega}^{DD}(\vec{b}, \vec{x}, \vec{y}) \cdot \left[ \text{sub} \left( \vec{x}' = T_{\alpha, \Omega}^{DD}(\vec{b}, \vec{x}, \vec{y}), \vec{y}' \right) \right] V_{h-1}(\vec{x}') \quad (10)$$

- **Maximization for region $\Omega$ w.r.t. the action set $A$:**

$$V_{h, \Omega}^{DD}(\vec{b}, \vec{x}) = \max_{a \in A} \left( \max_{\vec{y}} Q_{h, \alpha, \Omega}^{DD}(\vec{b}, \vec{x}, \vec{y}) \right) \quad (11)$$

- **Update of the value function $V_h$ for region $\Omega$:**

$$V_{h, \text{new}}^{DD}(\vec{b}, \vec{x}) = \begin{cases} V_{h, \text{old}}^{DD}(\vec{b}, \vec{x}), & \text{if } \vec{s} \in \Omega \\ V_{h, \text{old}}^{DD}(\vec{b}, \vec{x}), & \text{otherwise} \end{cases} \quad (12)$$

which is done as an XADD minimization $V_{h, \text{new}}^{DD}(\vec{b}, \vec{x}) \leftarrow \min(V_{h,\Omega}^{DD}(\vec{b}, \vec{x}), V_{h,\text{old}}^{DD}(\vec{b}, \vec{x}))$. This minimum is the correct new value of $V_h$ because $V_{h,\Omega} \leq V_{h,\text{old}}$ 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 $(\vec{b}, \vec{x})$ 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),
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 $h_i$ indicator of higher incoming car density. The segment $c$ after the merge is represented by its continuous car density $k_c$ and speed $v_c$. Each action corresponds to a traffic light cycle that gives more time to one segment, so that the controller can give priority a segment with a greater density by choosing the action corresponding to it. The quantity of cars that pass the merge is given by:

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

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 $k_c$:

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

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.

Figure 4 shows the value functions for different stages-to-go generated by RTSDP (left) and SDP (right) when solving INVENTORY CONTROL, an instance with 2 continuous state variables and 2 continuous action parameters. The value functions are shown as 3D plots (value on the z axis as a function of continuous state variables on the x and y axes) with their corresponding XADDs. Note that the initial state is $(200, 200)$ and that for this state the value found by RTSDP is the same as SDP. One observation is that the z-axis scale changes on every plot because both the reward and the upper bound heuristic increases with stages to-go. 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, 4i and 4p), the number of regions increases quickly with the number of actions-to-go (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, but 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 small and therefore the number of terminal nodes and overall XADD size is not very large for greater stage-to-go, for instance, when the number of stages-to-go is the complete horizon the only reachable state (in 0 steps) is the initial state. 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 their upper bound heuristic 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 visited states and has a much more compact value function.

<table>
<thead>
<tr>
<th>Instance (#Cont.Vars)</th>
<th>SDP</th>
<th>RTSDP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time (s)</td>
<td>Nodes</td>
</tr>
<tr>
<td>Inventory 1 (2)</td>
<td>0.97</td>
<td>43</td>
</tr>
<tr>
<td>Inventory 2 (4)</td>
<td>2.07</td>
<td>229</td>
</tr>
<tr>
<td>Inventory 3 (6)</td>
<td>202.5</td>
<td>2340</td>
</tr>
<tr>
<td>Reservoir 1 (1)</td>
<td>Did not finish</td>
<td>1.6</td>
</tr>
<tr>
<td>Reservoir 2 (2)</td>
<td>Did not finish</td>
<td>11.7</td>
</tr>
<tr>
<td>Reservoir 3 (3)</td>
<td>Did not finish</td>
<td>160.3</td>
</tr>
<tr>
<td>Traffic 1 (4)</td>
<td>204</td>
<td>533000</td>
</tr>
<tr>
<td>Traffic 2 (5)</td>
<td>Did not finish</td>
<td>1.07</td>
</tr>
</tbody>
</table>

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

**Conclusion**

We propose a new solution, the RTSDP algorithm, for a general class of Hybrid MDP problems including linear dy-
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


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


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.