Bounded Approximate Symbolic Dynamic Programming for Hybrid MDPs

Luis G. R. Vianna\textsuperscript{1} \quad Scott Sanner\textsuperscript{2} \quad Leliane N. de Barros\textsuperscript{1}

\textsuperscript{1}University of São Paulo  \quad São Paulo, Brazil
\textsuperscript{2}Australian National University & NICTA  \quad Canberra, Australia
Hybrid Probabilistic Planning
Linear eXtended Algebraic Decision Diagram (XADD)
Symbolic Dynamic Programming (SDP)
XADD Compression
BASDP
Hybrid Probabilistic Planning

Linear eXtended Algebraic Decision Diagram (XADD)

Symbolic Dynamic Programming (SDP)

XADD Compression

BASDP
• Markov Decision Process (MDP) is an expressive model for sequential optimization.
Hybrid MDP

- Hybrid State: $\vec{s} = (\vec{b}, \vec{x})$
  - e.g. $b_1 = \text{AtGoal} \in \{0, 1\}$, $\vec{x} = (x_1, x_2) \in \mathbb{R}^2$
Hybrid MDP

- Hybrid State: \( \vec{s} = (\vec{b}, \vec{x}) \)
  
  e.g. \( b_1 = \text{AtGoal} \in \{0, 1\} \), \( \vec{x} = (x_1, x_2) \in \mathbb{R}^2 \)

- Parameterized Actions: \( a(\vec{y}) \), with \( \vec{y} \in \mathbb{R}^k \)
  
  e.g. \( a = \text{move}(y_1, y_2) \), with \( \vec{y} \in \mathbb{R}^2 \)
Hybrid MDP

• Hybrid State: $\vec{s} = (\vec{b}, \vec{x})$
  e.g. $b_1 = AtGoal \in \{0, 1\}$, $\vec{x} = (x_1, x_2) \in \mathbb{R}^2$

• Parameterized Actions: $a(\vec{y})$, with $\vec{y} \in \mathbb{R}^k$
  e.g. $a = move(y_1, y_2)$, with $\vec{y} \in \mathbb{R}^2$

• Transition Function: $T((\vec{b}, \vec{x}), a(\vec{y})) = (\vec{b}', \vec{x'})$
  e.g. $T((\neg AtGoal, x_1, x_2), move(y_1, y_2)) =$
  $x'_1 = x_1 + y_1,$
  $x'_2 = x_2 + y_2,$
  $AtGoal' = I[4 \leq x'_1 \leq 7 \land 2 \leq x'_2 \leq 4]$. 
Hybrid MDP

- **Hybrid State:** $\tilde{s} = (\tilde{b}, \tilde{x})$
  
  e.g. $b_1 = \text{AtGoal} \in \{0, 1\}$, $\tilde{x} = (x_1, x_2) \in \mathbb{R}^2$

- **Parameterized Actions:** $a(\tilde{y})$, with $\tilde{y} \in \mathbb{R}^k$
  
  e.g. $a = \text{move}(y_1, y_2)$, with $\tilde{y} \in \mathbb{R}^2$

- **Transition Function:** $T((\tilde{b}, \tilde{x}), a(\tilde{y})) = (\tilde{b}', \tilde{x}')$
  
  e.g. $T((\neg \text{AtGoal}, x_1, x_2), \text{move}(y_1, y_2)) =$

  $x_1' = x_1 + y_1,$
  $x_2' = x_2 + y_2,$
  $\text{AtGoal}' = I[4 \leq x_1' \leq 7 \land 2 \leq x_2' \leq 4].$

- **Reward Function:** $R((\tilde{b}, \tilde{x}), a(\tilde{y}), (\tilde{b}', \tilde{x}')) = r \in \mathbb{R}$
  
  e.g. $R((\neg \text{AtGoal}, \tilde{x}), a(\tilde{y}), (\text{AtGoal}, \tilde{x}')) = 1$, else $R(\cdot) = 0$
HMDP Solution

Optimal policy maximizes expected total reward:

\[ \pi^* = \arg \max_{a_t(\vec{y}_t)} \mathbb{E} \left[ \sum_{t=0}^{H} \gamma^t R(s_{t-1}, a_t(\vec{y}_t), s_t| r_t) \right]. \]
HMDP Solution

Optimal policy maximizes expected total reward:

$$\pi^* = \arg \max_{a_t(y_t)} \mathbb{E} \left[ \sum_{t=0}^{H} \gamma^t R(s_{t-1}, a_t(y_t), s_t) \right].$$

The maximal reward obtained from a state is its value function:

$$V^*(\vec{s}, h) = \mathbb{E} \left[ \sum_{t=0}^{h} \gamma^t r_t \mid s_0 = \vec{s}, a_t(y_t) = \pi^*(\vec{s}_t) \right].$$
Hybrid Probabilistic Planning

Linear eXtended Algebraic Decision Diagram (XADD)

Symbolic Dynamic Programming (SDP)

XADD Compression

BASDP
Linear XADD

XADDs are directed acyclic graphs with two kind of nodes:

- **Terminal (leaf) node**: A linear function, Ex: 0, 1.7, 7x₁ − 8x₂
- **Internal node**: A linear inequality or boolean variable.

- XADD use independencies and compute operations in piecewise functions efficiently.
XADD represent piecewise linear functions
XADD represent piecewise linear functions
XADD represent piecewise linear functions

Example of piecewise linear function in case and XADD form

\[
\begin{align*}
\phi_1 &= \theta_{11} \lor \theta_{12} \\
\theta_{11} &= x < 0 \\
\theta_{12} &= x > 0 \land x < -y \\
f_1 &= \frac{x}{2} \\
f_2 &= \frac{x}{5} - \frac{y}{2} \\
\phi_2 &= \theta_{21} \\
\theta_{21} &= x > 0 \land y > x
\end{align*}
\]
Hybrid Probabilistic Planning

Linear eXtended Algebraic Decision Diagram (XADD)

Symbolic Dynamic Programming (SDP)

XADD Compression

BASDP
The back propagation is performed using Bellman’s Equation [Sanner11, Zamani12]:

\[
Q^h_a(\vec{b}, \vec{x}, \vec{y}) = \sum_{\vec{b}'} \int_{\vec{x}'} \left[ \prod_{k=1}^{n+m} \mathcal{P}_{XADD}(v'_k|\vec{b}, \vec{x}, a, \vec{y}) \otimes \mathcal{R}_{XADD}(\vec{b}, \vec{x}, a, \vec{y}, \vec{b}', \vec{x}') \right]
\]

\[
V^h(\vec{b}, \vec{x}) = \max_{a \in A} \max_{\vec{y} \in \mathbb{R}^{\mid \vec{y} \mid}} \left\{ Q^h_a(\vec{b}, \vec{x}, \vec{y}) \right\}
\]
XADD Compression

(a) Value at $6^{th}$ iteration for exact SDP.

(b) Value at $6^{th}$ iteration for 5% approximate SDP.
XADD Compression

Size reduction by linear approximation and region merging
DD Leaf-based Compression

Based on ADD approximation [APRICODD], XADDs are approximated by successive leaf merging which removes internal nodes upon minimization.

ADD approximation by leaf merging
Successive Approximation

Original Function
Successive Approximation

- $f(x)$
- $x > 0$
- $x > 1$
- $x > 3$

First Merge

- $4.0$
- $1.5 + 0.5x$ (Error = 0.5)
- $1.0$
- $0.0$
Successive Approximation

![Graph of Successive Approximation](image)

- For $x > 0$, the function is represented by the red box with the value 4.0 and the error of 0.667.
- For $x > 3$, the function is represented by the red box with the value 0.0.
- The equation $1.0 + 0.667 \times x$ is given with an error of 0.667.

Second Merge
XADD Compression

Size reduction by linear approximation and region merging
Pairwise Leaf Merging

\[
\min_{\bar{c}^*} \max_{i \in \{1,2\}} \max_{\bar{x} \in S_{\phi_i}} \left| \bar{c}_i^T \begin{bmatrix} \bar{x} \\ 1 \end{bmatrix} - \bar{c}^*_i^T \begin{bmatrix} \bar{x} \\ 1 \end{bmatrix} \right|
\]

(1)
\[
\min_{\vec{c}^*, \epsilon} \epsilon \\
\text{s.t. } \epsilon \geq \left| \vec{c}_i^T \begin{bmatrix} \vec{x}_{ij}^k \\ 1 \end{bmatrix} - \vec{c}^*_i \begin{bmatrix} \vec{x}_{ij}^k \\ 1 \end{bmatrix} \right| \\
\forall i \in \{1,2\}, \forall \theta_{ij}, \forall k \in \{1 \ldots N_{ij}\}
\]
Constraint Generation Solution

1. Start with $C_S = \emptyset$ and an arbitrary solution $\vec{c}^*$.
2. For each polytope find optimal vertex $\vec{x}^k_{ij}$:

   $$\vec{x}^k_{ij} := \arg \max_{\vec{x}} \left( \vec{c}_i^T \begin{bmatrix} \vec{x} \\ 1 \end{bmatrix} - \vec{c}^*_T \begin{bmatrix} \vec{x} \\ 1 \end{bmatrix} \right)$$

   s.t. $\vec{x} \in \text{Polytope}(\theta_{ij})$.

3. Add constraints for these $\vec{x}^k_{ij}$ to the constraint set.
4. Solve the minimization step and find a new solution $\vec{c}$ and error $\epsilon$.
5. If $\vec{c}$ or $\epsilon$ is unchanged in the minimization, we are at an optimal solution, return it.
6. Otherwise return to the maximization step.
BASDP

- Return to SDP and Value Iteration setting.
- Add the following step between iterations.

\[
\hat{V}^h(\vec{b}, \vec{x}) = \text{XADDCOMPRESS} \, V^h(\vec{b}, \vec{x})
\]
Rover Linear 1D

Value function at iteration 6 for MARS ROVER1D, showing how different levels of approximation error (eps) lead to different compressions.
Performance: Nodes

![Performance plots for MARS ROVER2D: Space.](image)

Figure: Performance plots for MARS ROVER2D: Space.
Performance: Time

Figure: Performance plots for MARS ROVER 2D: Time.
Performance: Error

Figure: Performance plots for MARS ROVER2D: Maximum Error.
Rover 2D

(a) Value at 6\textsuperscript{th} iteration for exact SDP.

(b) Value at 6\textsuperscript{th} iteration for 5\% approximate SDP.

Value function at iteration 6 for the MARS ROVER2D domain;
Conclusions

- Introduced a compression method for XADDs;
- Solved a bilinear saddle point optimization by reduction to bi-level linear programming and constraint generation.
- Great time and space savings in exchange for small errors.
- Improved SDP scalability with bounded approximation.
Thanks for your attention!

Questions?