Bounded Approximate Symbolic Dynamic Programming for Hybrid MDPs

Luis G. R. Vianna[1] Scott Sanner[2] Leliane N. de Barros[1]

[1]University of São Paulo
São Paulo, Brazil

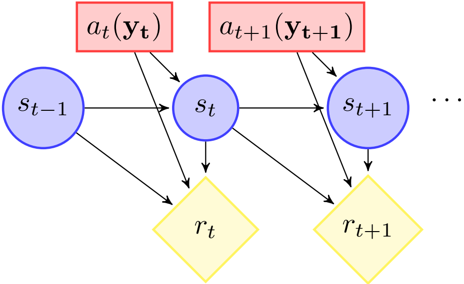
[2]Australian National University & NICTA
Canberra, Australia

IME - Instituto de
Matemática e Estatística

NICTA

MDP

- Markov Decision Process (MDP) is an expressive model for sequential optimization.



Hybrid MDP

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

Hybrid MDP

- Hybrid State: $\vec{s} = (\vec{b}, \vec{x})$
  e.g. $b_1 = AtGoal \in \{0, 1\}$, $\vec{x} = (x_1, x_2) \in \mathbb{R}^2$
- Parameterized Actions: $a(\vec{y})$, with $\vec{y} \in \mathbb{R}^k$
  e.g. $a = move(y_1, y_2)$, with $\vec{y} \in \mathbb{R}^2$

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. $T((\neg AtGoal, x_1, x_2), move(y_1, y_2)) =$

$$x_1' = x_1 + y_1,$$
$$x_2' = x_2 + y_2,$$
$$AtGoal' = I[4 \leq x_1' \leq 7 \wedge 2 \leq x_2' \leq 4].$$

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. $T((\neg AtGoal, x_1, x_2), move(y_1, y_2)) =$
  $$x_1' = x_1 + y_1,$$
  $$x_2' = x_2 + y_2,$$
  $$AtGoal' = I[4 \leq x_1' \leq 7 \wedge 2 \leq x_2' \leq 4].$$
- Reward Function: $R((\vec{b}, \vec{x}), a(\vec{y}), (\vec{b}', \vec{x}')) = r \in \mathbb{R}$
  e.g. $R((\neg AtGoal, \vec{x}), a(\vec{y}), (AtGoal, \vec{x}')) = 1, else\ R(\cdot) = 0$

HMDP Solution

Optimal policy maximizes expected total reward:

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

HMDP Solution

Optimal policy maximizes expected total reward:

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

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

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

Hybrid Probabilistic Planning

Linear eXtended Algebraic Decision Diagram (XADD)

Symbolic Dynamic Programming (SDP)

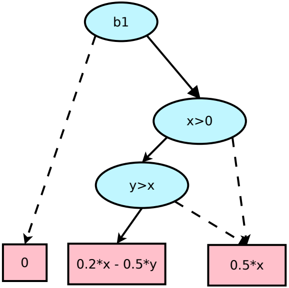
XADD Compression

BASDP

Linear XADD

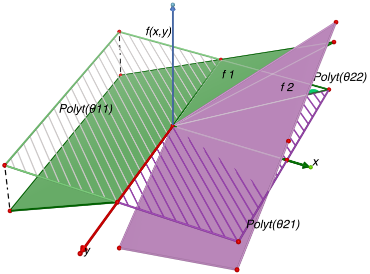
XADDs are directed acyclic graphs with two kind of nodes:

- *Terminal (leaf) node*: A linear function, Ex: $0, 1.7, 7x_1 - 8x_2$
- *Internal node*: A linear inequality or boolean variable.

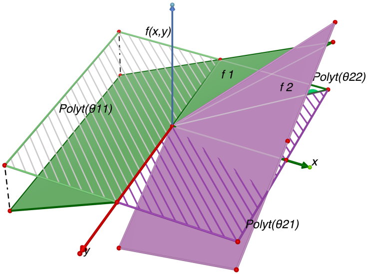


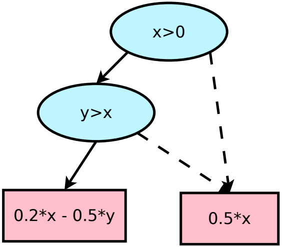
- XADD use independencies and compute operations in piecewise functions efficiently.

XADD represent piecewise linear functions

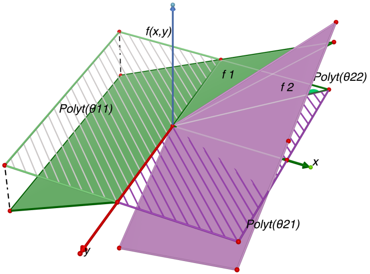


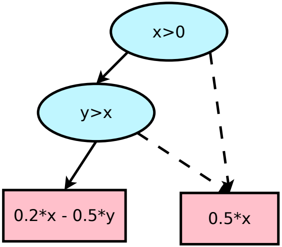
XADD represent piecewise linear functions





XADD represent piecewise linear functions





$$f(x,y) = \begin{cases} \phi_1 : & f_1 \\ \phi_2 : & f_2 \end{cases}$$

$$\phi_1 = \theta_{11} \vee \theta_{12}$$

$$\theta_{11} = x < 0$$

$$\theta_{12} = x > 0 \wedge x < -y$$

$$f_1 = \frac{x}{2}$$

$$\phi_2 = \theta_{21}$$

$$\theta_{21} = x > 0 \wedge y > x$$

$$f_2 = \frac{x}{5} - \frac{y}{2}$$

Example of piecewise linear function in case and XADD form

SDP

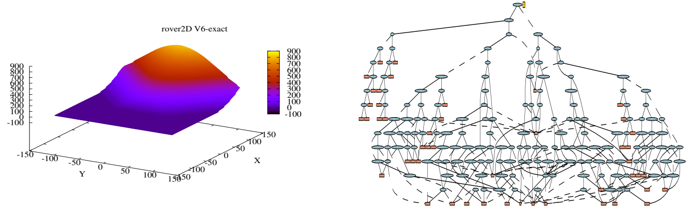
The back propagation is performed using Bellman's Equation [Sanner11, Zamani12]:

$$\underbrace{Q_a^h}_{XADD}(\vec{b},\vec{x},\vec{y}) = \sum_{\vec{b}'}\int_{\vec{x}'}\left[\prod_{k=1}^{n+m}\underbrace{P}_{XADD}(v_k'|\vec{b},\vec{x},a,\vec{y})\otimes\right.$$

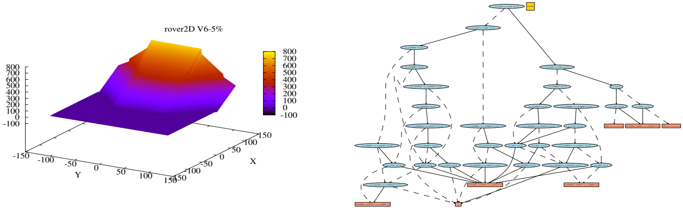
$$\left.\left(\underbrace{R}_{XADD}(\vec{b},\vec{x},a,\vec{y},\vec{b}',\vec{x}')\oplus\gamma\underbrace{V^{h-1}}_{XADD}(\vec{b}',\vec{x}')d\vec{x}'\right)\right]$$

$$\underbrace{V^h}_{XADD}(\vec{b},\vec{x}) = \max_{a\in A}\max_{\vec{y}\in\mathbb{R}^{|\vec{y}|}}\left\{Q_a^h(\vec{b},\vec{x},\vec{y})\right\}$$

XADD Compression

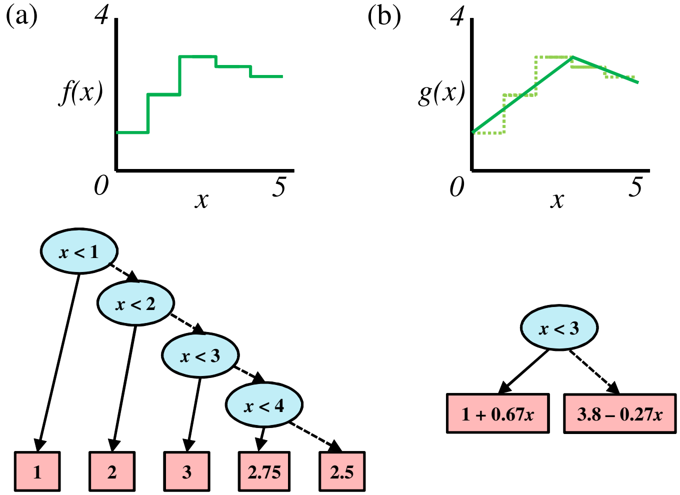


(a) Value at 6^{th} iteration for exact SDP.



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

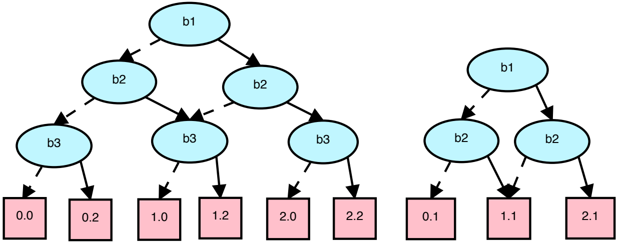
XADD Compression



Size reduction by linear approximation and region merging

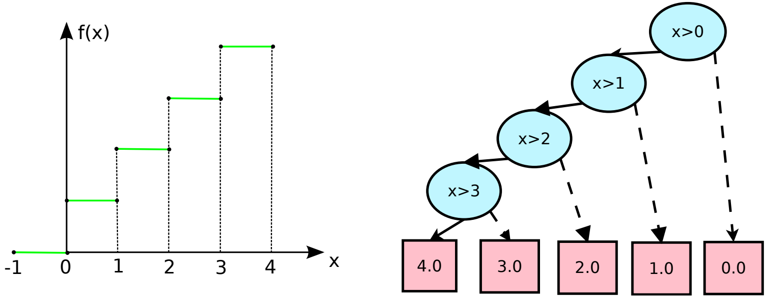
DD Leaf-based Compression

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



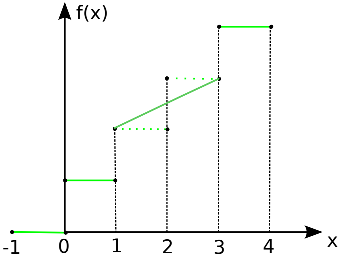
ADD approximation by leaf merging

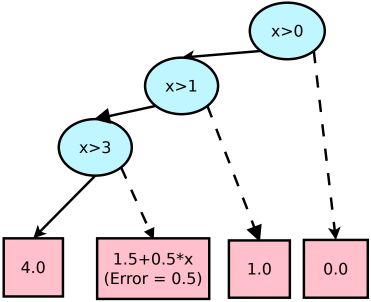
Successive Approximation



Original Function

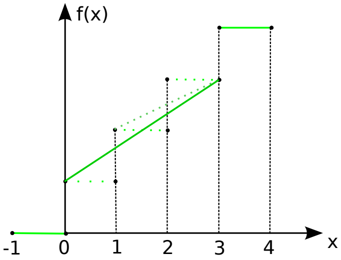
Successive Approximation

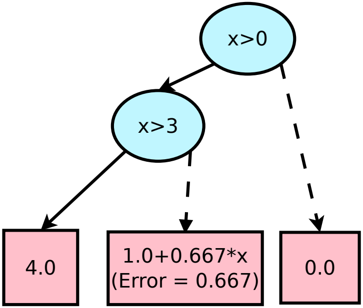




First Merge

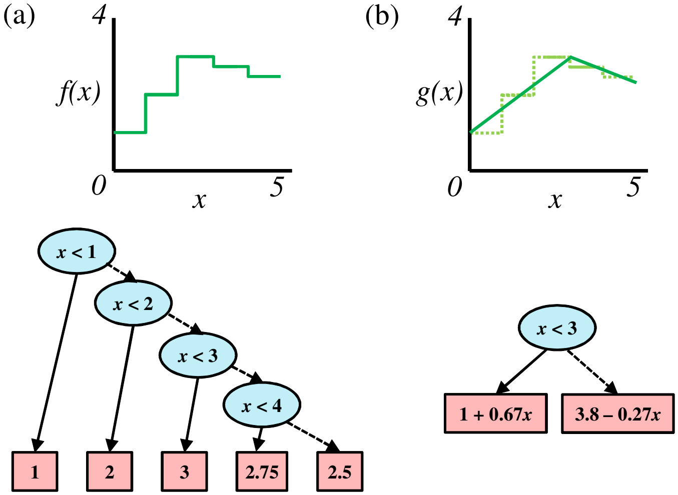
Successive Approximation





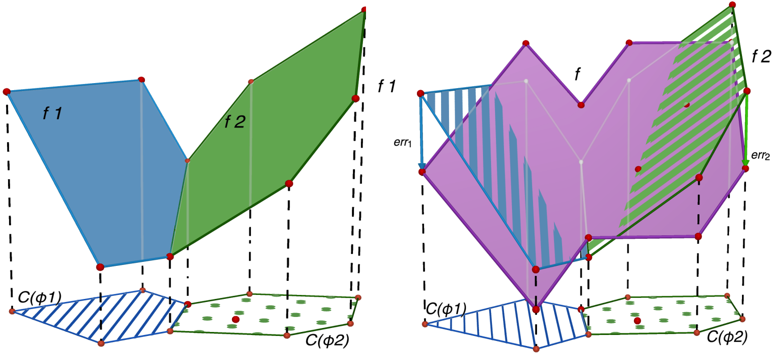
Second Merge

XADD Compression

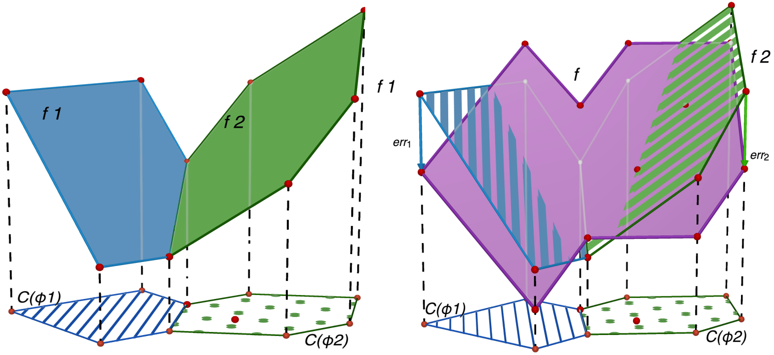


Size reduction by linear approximation and region merging

Pairwise Leaf Merging



$$\min_{\vec{c}^*} \max_{i\in\{1,2\}} \max_{\vec{x}\in S_{\phi_i}} \left| \underbrace{\vec{c_i}^T \begin{bmatrix} \vec{x} \\ 1 \end{bmatrix}}_{f_i} - \underbrace{\vec{c^*}^T \begin{bmatrix} \vec{x} \\ 1 \end{bmatrix}}_{f^*} \right| \quad (1)$$



$$\min_{\vec{c}^*,\epsilon} \epsilon \tag{2}$$

$$s.t.\ \epsilon \geq \left| \vec{c_i}^T \begin{bmatrix} \vec{x}_{ij}^k \\ 1 \end{bmatrix} - \vec{c^*}^T \begin{bmatrix} \vec{x}_{ij}^k \\ 1 \end{bmatrix} \right| ; \quad \begin{array}{l} \forall i \in \{1,2\}, \forall \theta_{ij}, \\ \forall k \in \{1 \ldots N_{ij}\} \end{array}$$

Constraint Generation Solution

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

$$\vec{x_{ij}^k} := \arg\max_{\vec{x}} \left(\vec{c_i}^T \begin{bmatrix} \vec{x} \\ 1 \end{bmatrix} - \vec{c^*}^T \begin{bmatrix} \vec{x} \\ 1 \end{bmatrix} \right)$$
$$\text{s.t. } \vec{x} \in Polytope(\theta_{ij})$$

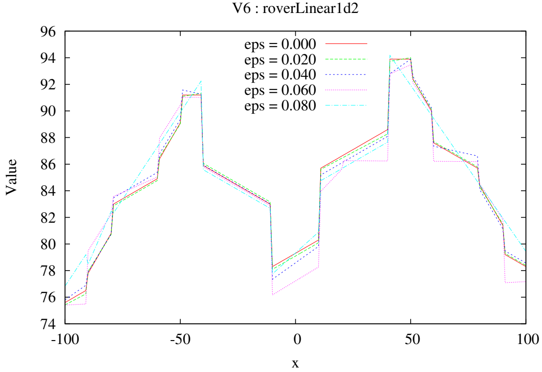
3. Add constraints for these $\vec{x_{ij}^k}$ to constraint set
4. Solve the minimization step and find a new solution $\vec{c}$ and error ϵ
5. If $\vec{c}$ or ϵ is unchanged in the minimization, we are at an optimal solution, return it.
6. Otherwise return to the maximization step

BASDP

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

$$\underbrace{V^h}_{XADD}(\vec{b}, \vec{x}) = \text{XADDCOMPRESS}\, V^h(\vec{b}, \vec{x})$$

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

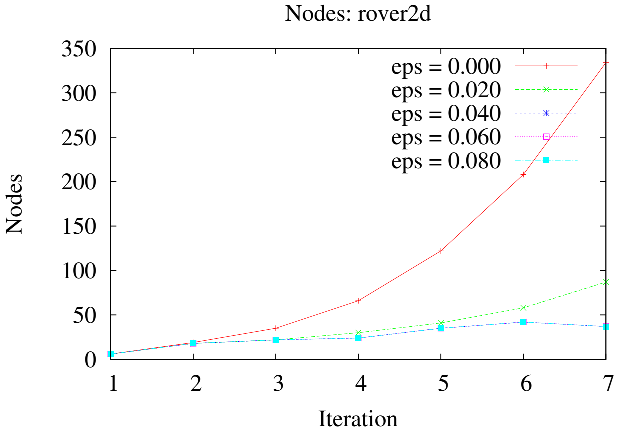


Figure: Performance plots for MARS ROVER2D: Space.

Performance: Time

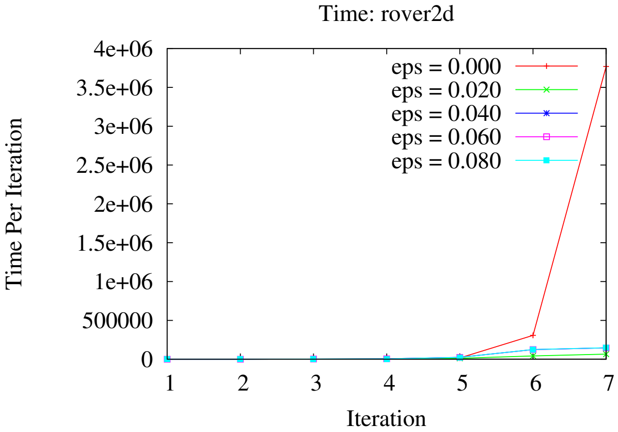


Figure: Performance plots for MARS ROVER2D: Time.

Performance: Error

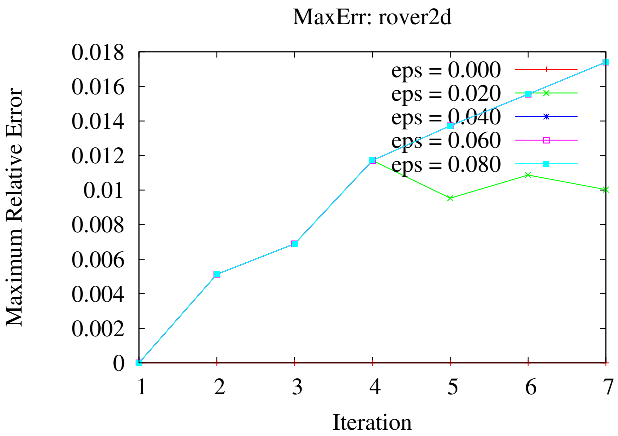
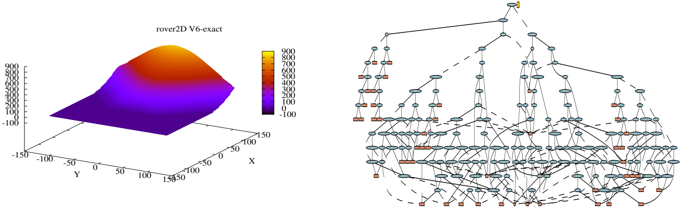
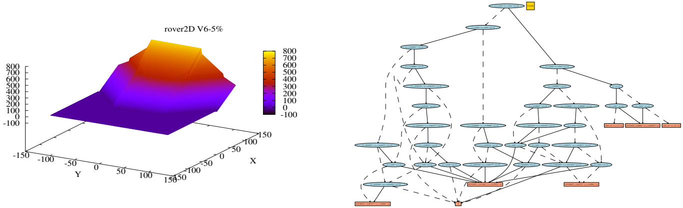


Figure: Performance plots for MARS ROVER2D: Maximum Error.

Rover 2D



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



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

Value function at iteration 6 for the MARS ROVER2Ddomain;

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?