Symbolic Dynamic Programming for Continuous State and Action MDPs

NICTA

Zahra Zamani (NICTA & the ANU), Scott Sanner (NICTA & the ANU), Cheng Fang (M.I.T.)

Highlight

Goal: Exact **dynamic programming** for **continuous state & action MDPs**:

$$Q_a^h(\mathbf{b},\mathbf{x},\mathbf{y}) = \left[R(\mathbf{b},\mathbf{x},a,\mathbf{y}) + \gamma \cdot \sum_{\mathbf{b}'} \int P(\mathbf{b}',\mathbf{x}'|\mathbf{b},\mathbf{x},a,\mathbf{y}) V^{h-1}(\mathbf{b}',\mathbf{x}') d\mathbf{x}' \right] \quad (1)$$

$$V^h(\mathbf{b},\mathbf{x}) = \max_{a\in A} \max_{\mathbf{y}\in\mathbb{R}^{|\mathbf{y}|}} \left\{ Q_a^h(\mathbf{b},\mathbf{x},\mathbf{y}) \right\} \quad (2)$$

Tools: 1: **Symbolic dynamic programming (SDP)** approach.
2: Use efficient **extended ADD (XADD)** data structure to compute SDP.

Discrete and Continuous State & Action MDPs

- **Discrete and Continuous State Space:** $(\mathbf{b},\mathbf{x})$ where $b_i \in \{0,1\}$ and $x_j \in \mathbb{R}$.
- **Continuous Action Space:** $A = \{a_1(\mathbf{y}_1),\ldots,a_p(\mathbf{y}_p)\}$, with parameter $\mathbf{y}_k \in \mathbb{R}^{|\mathbf{y}_k|}$.
- **Transition Model:** Joint DBN of **conditional probability functions (CPFs)** and **piecewise linear equations (PLEs)**:

$$P(\mathbf{b}',\mathbf{x}'|\mathbf{b},\mathbf{x},a,\mathbf{y}) = \prod_{i=1}^{n} P(b_i'|\mathbf{b},\mathbf{x},a,\mathbf{y}) \prod_{j=1}^{m} P(x_j'|\mathbf{b},\mathbf{b}',\mathbf{x},a,\mathbf{y})$$

$$P(b'=1|x,b) = \begin{cases} b \vee (x \geq -2 \wedge x \leq 2) : 1.0 \\ \neg b \wedge (x < -2 \vee x > 2) : 0.0 \end{cases} \quad P(x'|x,y) = \delta\left(x' - \begin{cases} y \geq -10 \wedge y \leq 10 : x+y \\ y < -10 \vee y > 10 : x \end{cases} \right)$$

- **Reward Model:** Any piecewise linear or univariate quadratic function:

$$R(x,b) = \begin{cases} \neg b \wedge x \geq -2 \wedge x \leq 2 : & 4 - x^2 \\ b \vee x < -2 \vee x > 2 : & 0 \end{cases}$$

Case Statement and Operators

Case supports unary and binary operations $c \cdot f$, $-f$, $\oplus$, $\ominus$, $\otimes$:

$$\begin{cases} \phi_1 : & f_1 \\ \phi_2 : & f_2 \end{cases} \oplus \begin{cases} \psi_1 : & g_1 \\ \psi_2 : & g_2 \end{cases} = \begin{cases} \phi_1 \wedge \psi_1 : & f_1 + g_1 \\ \phi_1 \wedge \psi_2 : & f_1 + g_2 \\ \phi_2 \wedge \psi_1 : & f_2 + g_1 \\ \phi_2 \wedge \psi_2 : & f_2 + g_2 \end{cases}$$

Case supports symbolic maximization of two cases:

$$\mathrm{casemax}\left(\begin{cases} \phi_1 : & f_1 \\ \phi_2 : & f_2 \end{cases}, \begin{cases} \psi_1 : & g_1 \\ \psi_2 : & g_2 \end{cases} \right) = \begin{cases} \phi_1 \wedge \psi_1 \wedge f_1 > g_1 : & f_1 \\ \phi_1 \wedge \psi_1 \wedge f_1 \leq g_1 : & g_1 \\ \phi_1 \wedge \psi_2 \wedge f_1 > g_2 : & f_1 \\ \phi_1 \wedge \psi_2 \wedge f_1 \leq g_2 : & g_2 \\ \vdots & \vdots \end{cases}$$

Theoretical Contribution: SDP

Symbolic Continuous Q-Value Regression

Evaluate (1): using **symbolic regression** via **case operations**:

$Q_a = Prime(V)$ [$\forall \mathbf{b_i} \rightarrow \mathbf{b_i'}$, $\forall \mathbf{x_i} \rightarrow \mathbf{x_i'}$]

For all x_j' in Q_a

$$Q_a := \int Q_a \otimes P(x_j'|\mathbf{b},\mathbf{b}',\mathbf{x},a,\mathbf{y})\, d_{x_j'} \quad \text{[Symbolic Substitution]}$$

For all b_i' in Q_a

$$Q_a := \left[Q_a \otimes P(b_i'|\mathbf{b},\mathbf{x},a,\mathbf{y}) \right] |_{b_i'=1} \oplus \left[Q_a \otimes P(b_i'|\mathbf{b},\mathbf{x},a,\mathbf{y}) \right] |_{b_i'=0} \quad [\text{Case } \oplus]$$

Compute final Q-Value (discount and add reward):

$$Q_a := R(\mathbf{b},\mathbf{x},a,\mathbf{y}) \oplus (\gamma \otimes Q_a)$$

Note that $\int f(x_j') \otimes \delta[x_j' - h(\mathbf{z})] dx_j' = f(x_j')\{x_j'/h(\mathbf{z})\}$ where the latter operation indicates that any occurrence of x_j' in $f(x_j')$ is *symbolically substituted* with the case statement $h(\mathbf{z})$ (Sanner, Delgado, de Barros, UAI 2011).

Symbolic Continuous Action Maximization

Evaluate (2): using casemax **for** $\max_{a\in A}$ and **symbolic optimization for** $\max_{\mathbf{y}}$:

Expand case into casemax of partitions, exploit commutativity of max:

$$\max_y \mathrm{casemax}_i\, \phi_i(\mathbf{b},\mathbf{x},y) f_i(\mathbf{b},\mathbf{x},y) = \mathrm{casemax}_i \boxed{\max_y \phi_i(\mathbf{b},\mathbf{x},y) f_i(\mathbf{b},\mathbf{x},y)}$$

Hence, just need to compute $\boxed{\max_y \phi_i(\mathbf{b},\mathbf{x},y) f_i(\mathbf{b},\mathbf{x},y)}$ for each partition i, e.g.

$$\phi_i(x,b,y) := \mathbb{I}[\neg b \wedge (x \geq 2) \wedge (y \leq 10) \wedge (y \geq -10) \wedge (y \leq 2-x) \wedge (y \geq -2-x)]$$

$$f_i(x,y) := 4 - (x+y)^2$$

In ϕ_i find upper (UB) and lower (LB) bounds for y given by $\phi_i(x,b,y)$:

$LB = \mathrm{casemax}(-10, -2-x)$ [Maximum of all constraints where $y \geq \cdots$]

$UB = \mathrm{casemin}(10, 2-x)$ [Minimum of all constraints where $y \leq \cdots$]

$Ind = \neg b \wedge (x \geq 2)$ [Constraints independent of y]

In f_i find roots of function derivative w.r.t. y:

$$\frac{\partial}{\partial y} f_i = -2y - 2x = 0 \implies Root = -x$$

Given *potential* maxima points(UB, LB, $Root$) find which yields maximum value:

$$Max = \mathrm{casemax}\Bigg(f_i\{y/Root\} = 4 - (x + -x)^2 = 4 , \quad \text{[Using substitution operator]}$$

$$f_i\{y/LB\} = \begin{cases} x \leq 8 : & 4 - (x + [-2-x])^2 = 0 \\ x > 8 : & 4 - (x + [-10])^2 = -x^2 + 20x - 96 \end{cases},$$

$$f_i\{y/UB\} = \begin{cases} x > -8 : & 4 - (x + [2-x])^2 = 0 \\ x \leq -8 : & 4 - (x + [10])^2 = -x^2 - 20x - 96 \end{cases} \Bigg)$$

Roots must lie within partition intervals: $LB \leq Root \leq UB$:

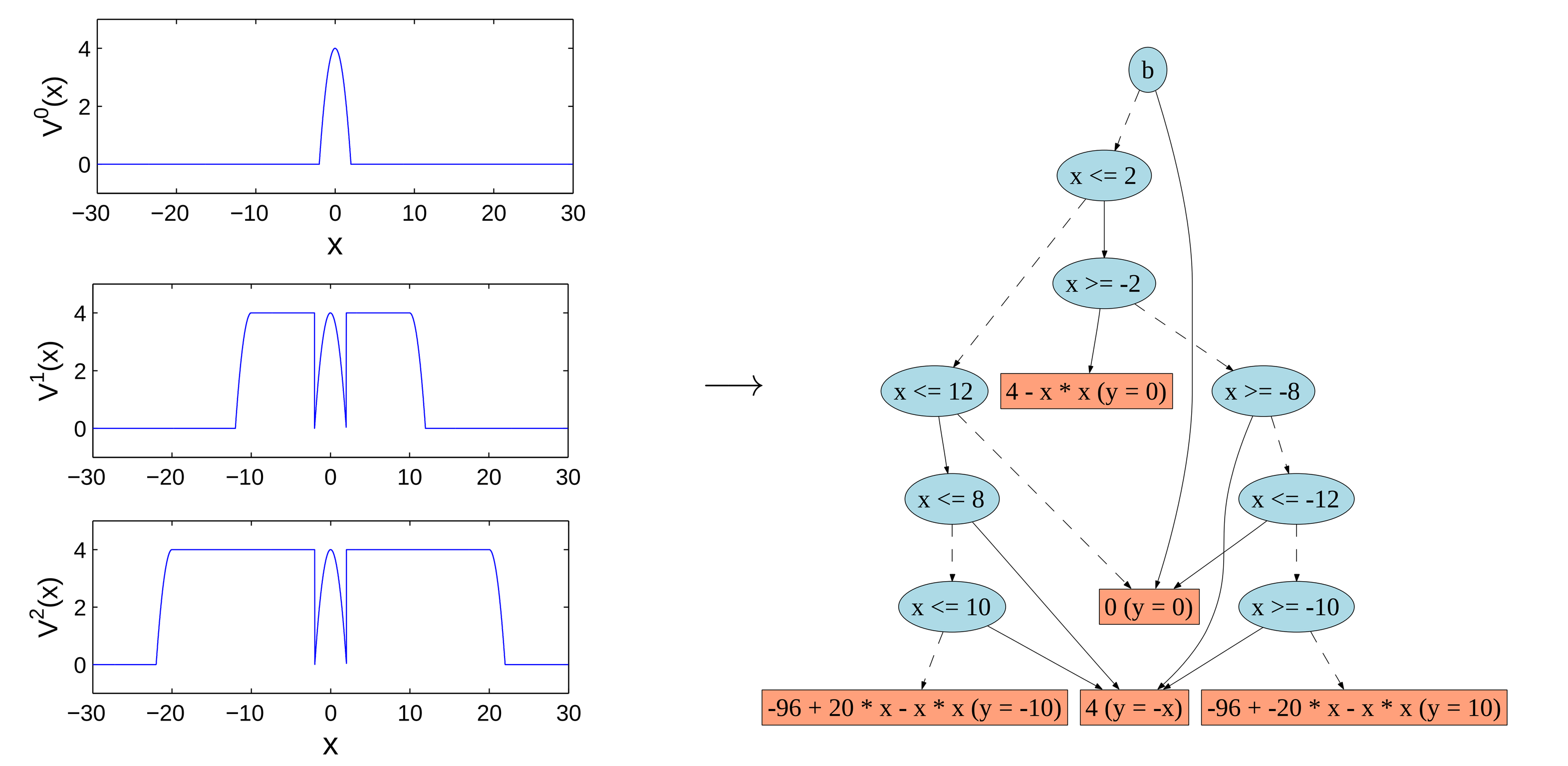
$$Cons = \underbrace{[-2-x \leq -x] \wedge [-10 \leq -x]}_{LB \leq Root} \wedge \underbrace{[-x \leq 2-x] \wedge [-x \leq 10]}_{Root \leq UB}$$

Result of $\boxed{\max_y \phi_i(\mathbf{b},\mathbf{x},y) f_i(\mathbf{b},\mathbf{x},y)}$ is a case statement:

$$\max_y \phi_i(\mathbf{b},\mathbf{x},y) f_i(\mathbf{b},\mathbf{x},y) = \{ Cons \wedge Ind : Max$$

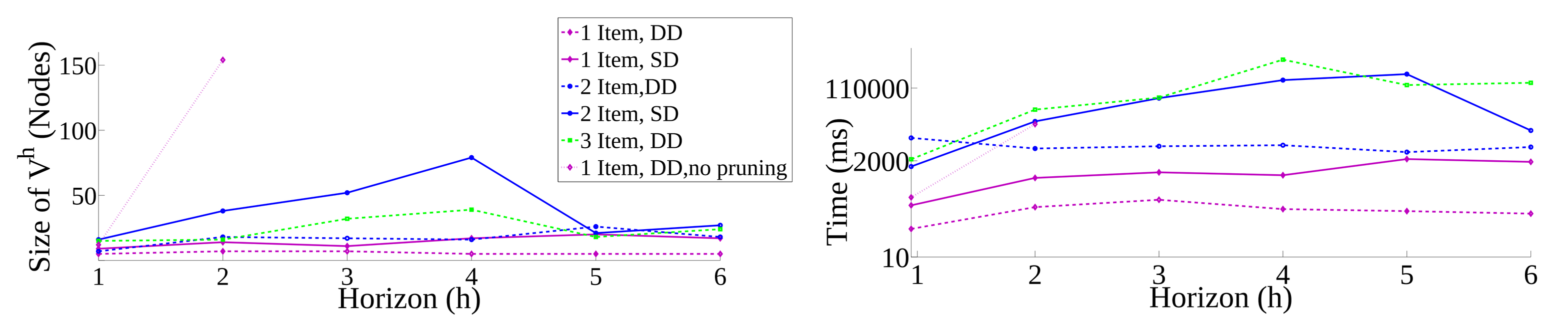
Extended ADDs (XADDs)

Question: How to avoid blow-up in case statements during SDP operations?
Answer: Extend algebraic decision diagram (ADD) to represent cases $\longrightarrow$ XADD:



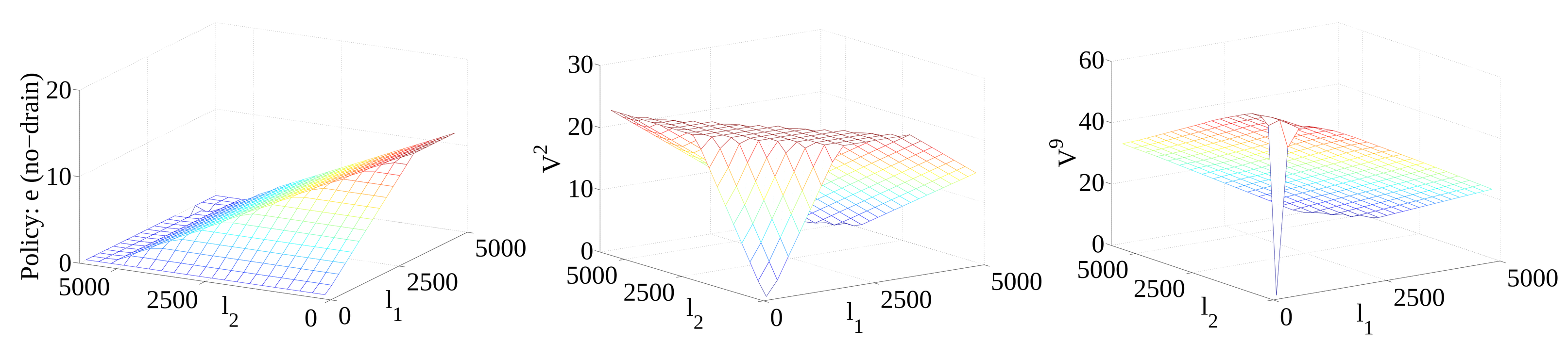
Empirical Results

Inventory Control: First exact policy for **multi-item joint capacitated problem**. Compare #items, stochastic (SD) vs deterministic demand (DD), XADD pruning:



Space and elapsed time (between current and previous horizon) vs. horizon.

Water Reservoir: **continuous reservoir levels** (l_1, l_2) **and time** (t), objective to control drain time-intervals of l_2 to l_1 to produce energy, avoid overflow/underflow:



(left) Policy $no\text{-}drain(e) = \pi^{2,*}(l_1,l_2)$ (optimal elapsed time e to perform for $no\text{-}drain$); (middle) $V^2(l_1,l_2)$; (right) $V^9(l_1,l_2)$.

Summary

- **Key insight:** **Dynamic programming operations implemented symbolically.**
- **Key contribution:** **Symbolic optimization for continuous action $\max_{\mathbf{y}}$.**
- **Key result:** **First exact solution to multi-variate continuous state and action MDPs with discrete noise and piecewise linear dynamics.**