AAAI-12

Symbolic Dynamic Programming for Continuous State and Action MDPs

Zahra Zamani
Scott Sanner

NICTA
THE AUSTRALIAN NATIONAL UNIVERSITY

Cheng Fang

Massachusetts Institute of Technology

Continuous State and Action MDPs: e.g., Inventory Control

Integer

Continuous

- Continuous state and actions
  - **State:** inventory quantities
  - **Action:** how much of each item to reorder

This work: optimal closed-form policies for multivariate continuous state and action MDPs

- Inventory closed-form optimal policy?
  - Scarf's solution (1958) for 1D inventory

First optimal policies for multi-item inventory control on 50+ years!

Where do we start?

Previous Work on Continuous State and **Discrete** Actions

Hybrid State, Discrete Action MDPs

- Hybrid discrete / continuous state

$$(\vec{b}, \vec{x}) = (b_1, \ldots, b_n, x_1, \ldots, x_m) \in \{0,1\}^n \times \mathbb{R}^m$$

- Discrete action set $a \in \mathcal{A}$
- DBN factored transition model

$$P(\vec{b}', \vec{x}' | \vec{b}, \vec{x}, a) =$$

$$\underbrace{\left(\prod_{i=1}^{n} P(b_i' | \vec{b}, \vec{x}, a) \right)}_{\text{discrete}} \underbrace{\left(\prod_{j=1}^{m} P(x_j' | \vec{b}, \vec{b}', \vec{x}, a) \right)}_{continuous}$$

- Arbitrary action-dependent reward

$$R_a(\vec{b}, \vec{x}) = x_1^2 + x_1 x_2$$

Value Iteration for Hybrid MDPs

- Value of policy in state is expected sum of rewards
- Want optimal value $V^{h,*}$ over horizons $h \in 0..H$
  - Implicitly provides optimal horizon-dependent policy
- Compute inductively via **Value Iteration** for $h \in 0..H$
  - **Regression step:**

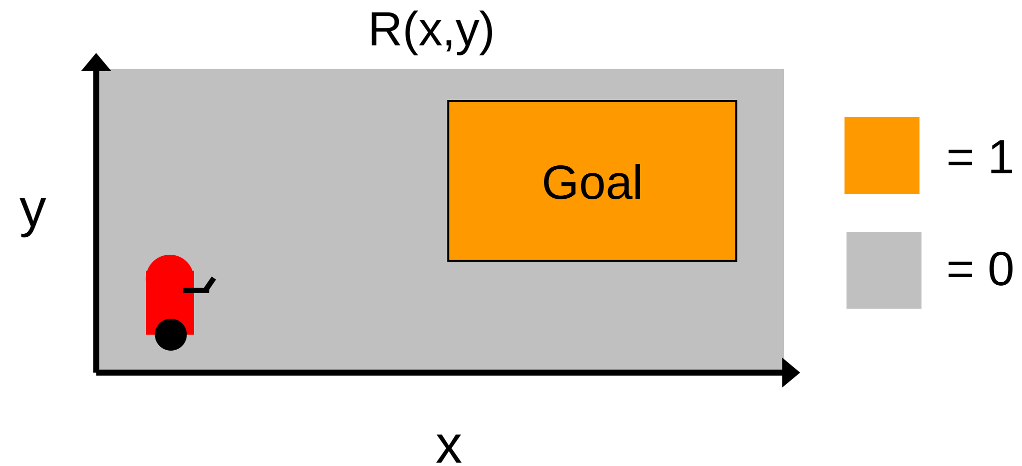
$$Q_a^{h+1}(\vec{b},\vec{x}) = R_a(\vec{b},\vec{x}) + \gamma \cdot$$
$$\sum_{\vec{b}'} \int_{\vec{x}'} \left(\prod_{i=1}^{n} P(b_i'|\vec{b},\vec{x},a) \prod_{j=1}^{m} P(x_j'|\vec{b},\vec{b}',\vec{x},a) \right) V^h(\vec{b}',\vec{x}') d\vec{x}'$$

  - **Maximization step:**

$$V_{h+1} = \max_{a \in A} Q_a^{h+1}(\vec{b},\vec{x})$$

Exact Solutions to Hybrid MDPs: Domain

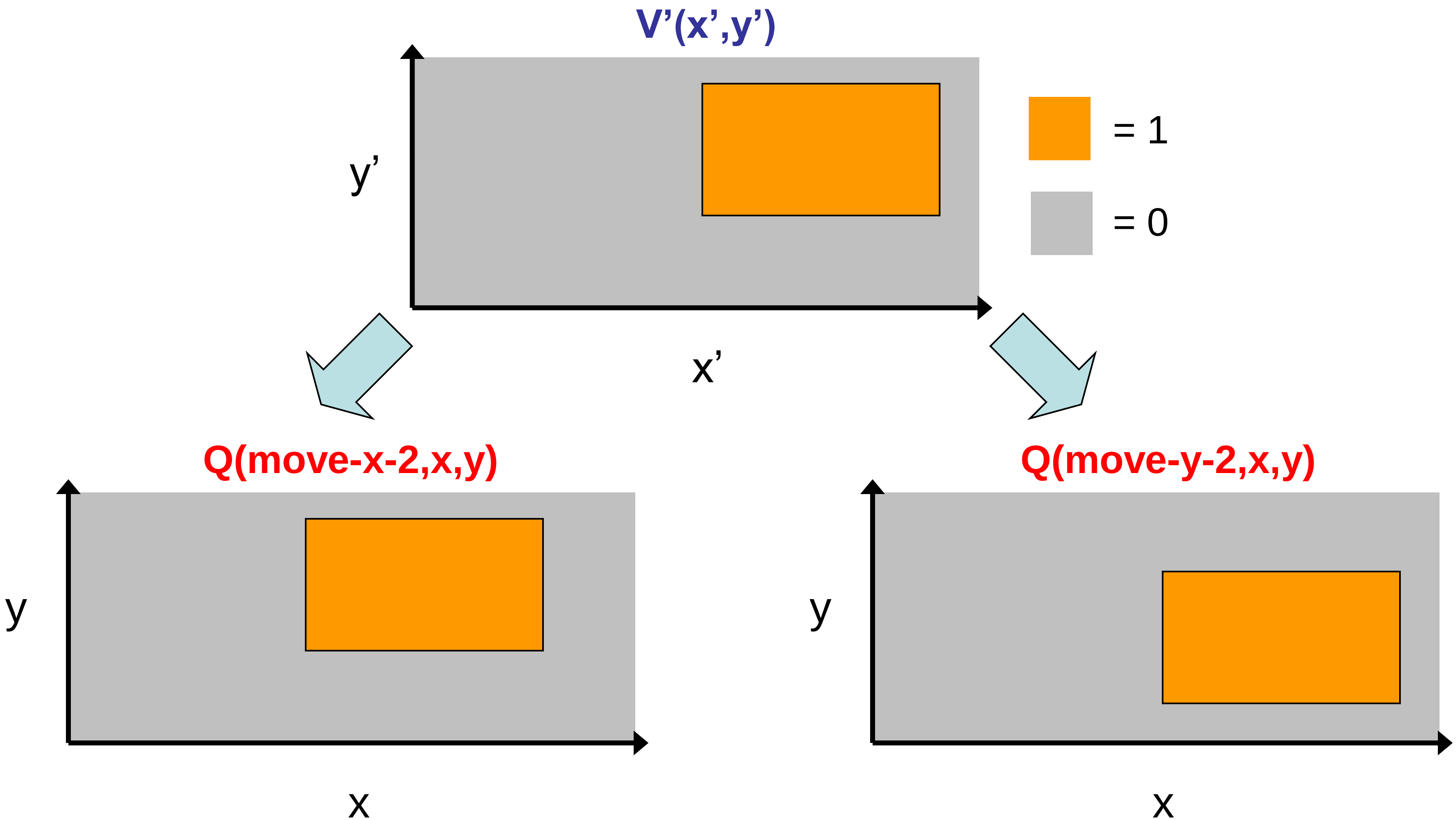
- 2-D Navigation
- State: $(x,y) \in \mathbb{R}^2$
- Actions:
  - move-x-2
    - x' = x + 2
    - y' = y
  - move-y-2
    - x' = x
    - y' = y + 2
- Reward:
  - R(x,y) = I[(x > 5) ^ (x < 10) ^ (y > 2) ^ (y < 5)]



Assumptions:
1. Continuous transitions are deterministic and linear
2. Discrete transitions can be stochastic
3. Reward is **piecewise rectilinear**

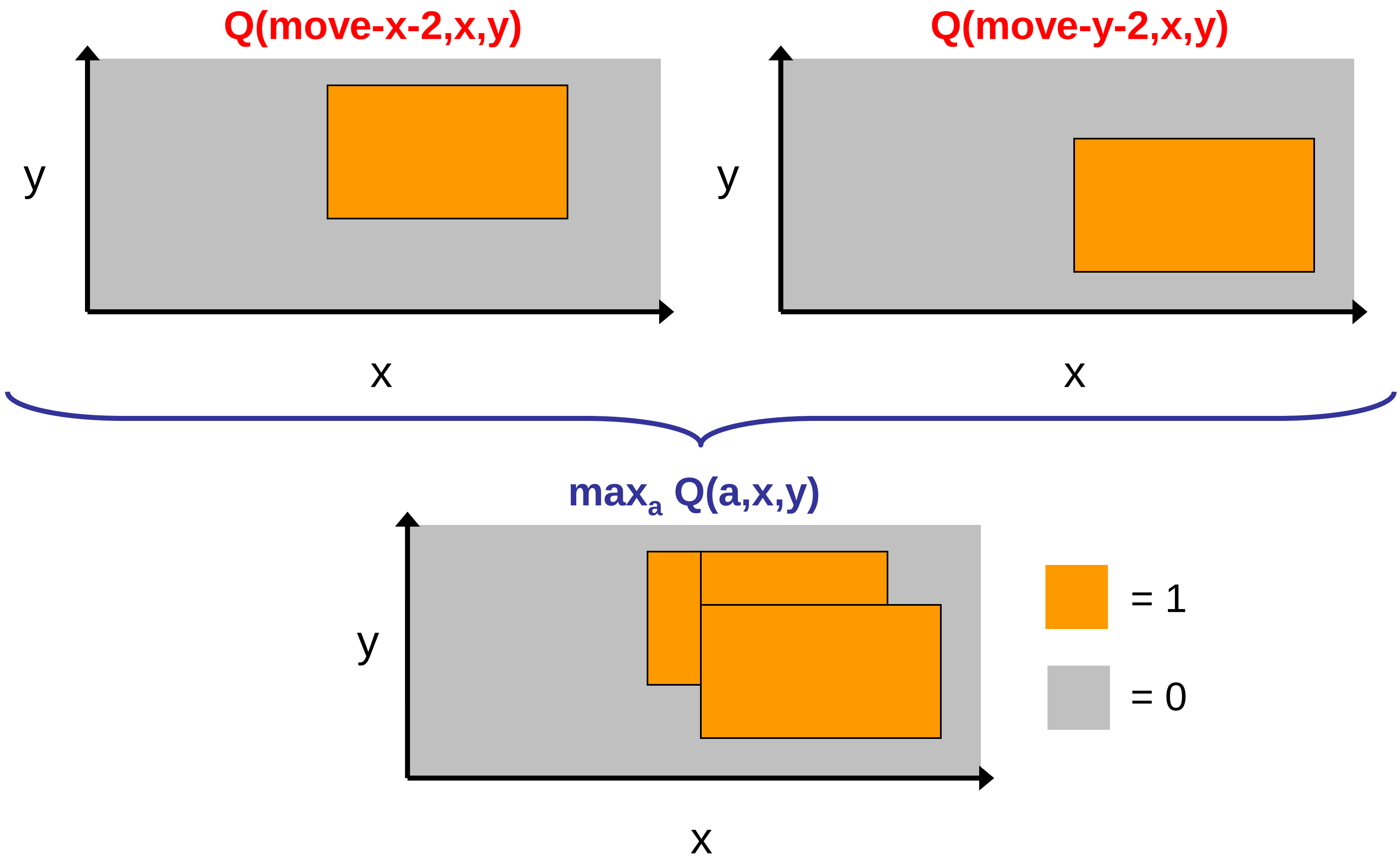
Exact Solutions to Hybrid MDPs: Regression

- Continuous regression is just translation of “pieces”



Exact Solutions to Hybrid MDPs: Maximization

- Q-value maximization yields piecewise rectilinear solution



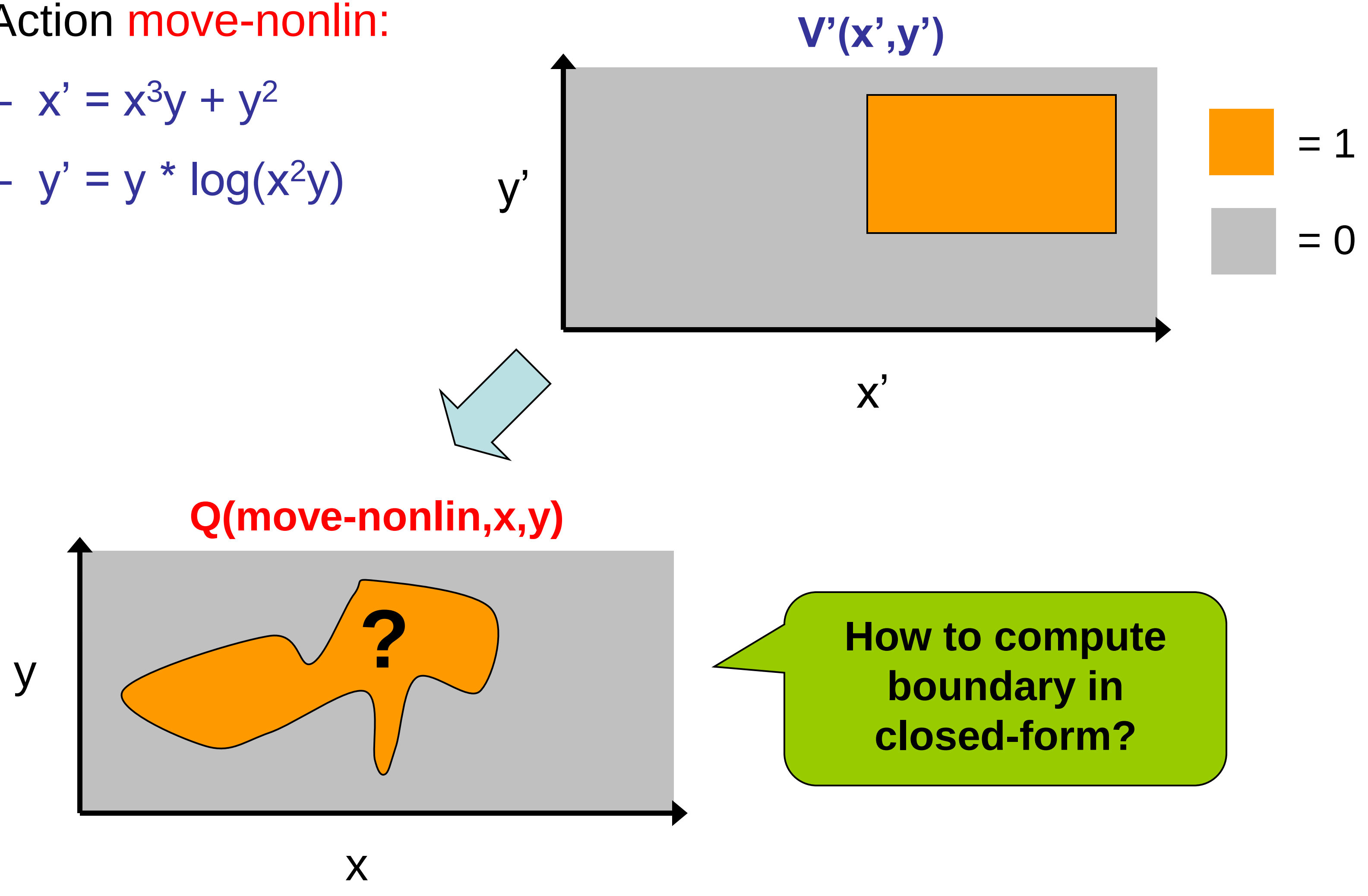
Previous Work Limitations I

- Exact regression when transitions nonlinear?

Action move-nonlin:

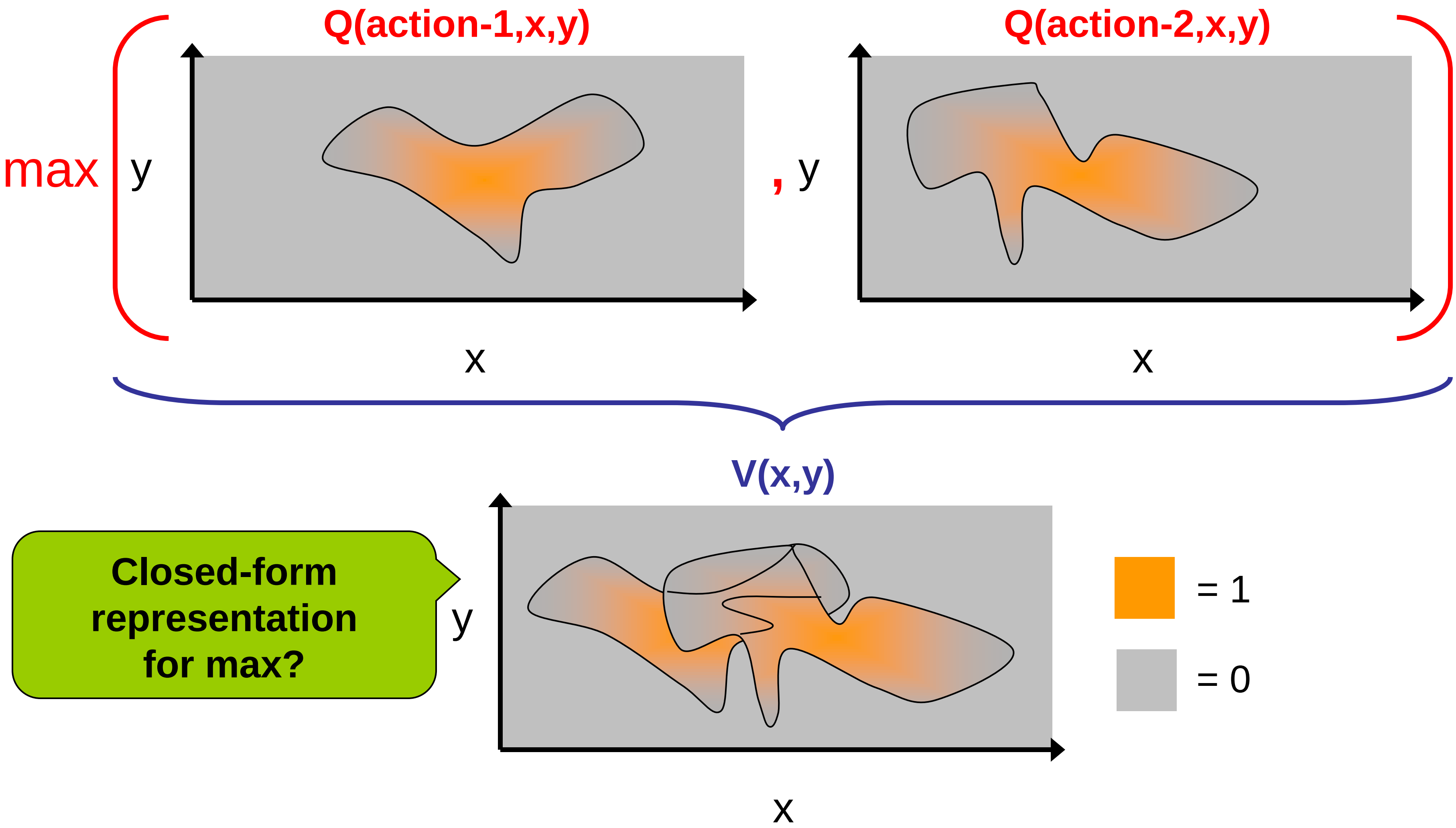
– $x' = x^3 y + y^2$

– $y' = y * \log(x^2 y)$



Previous Work Limitations II

- max(.,.) when reward/value arbitrary piecewise?



A solution to previous limitations:

Symbolic Dynamic Programming (SDP)

n.b., motivated by SDP from Boutilier *et al* (IJCAI-01) but here continuous instead of relational

Piecewise Functions (Cases)

$$z = f(x,y) = \begin{cases} \boxed{(x > 3) \wedge (y \cdot \;\; x) : \quad x + y} \\ (x \cdot \;\; 3) \vee (y > x) : \quad x^2 + xy^3 \end{cases}$$

Partition

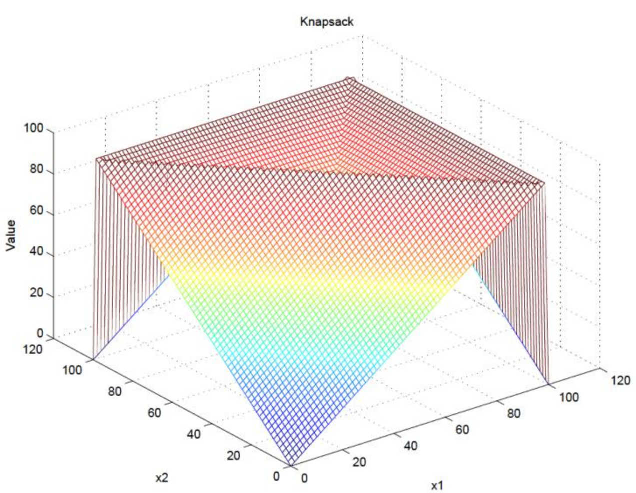
Constraint

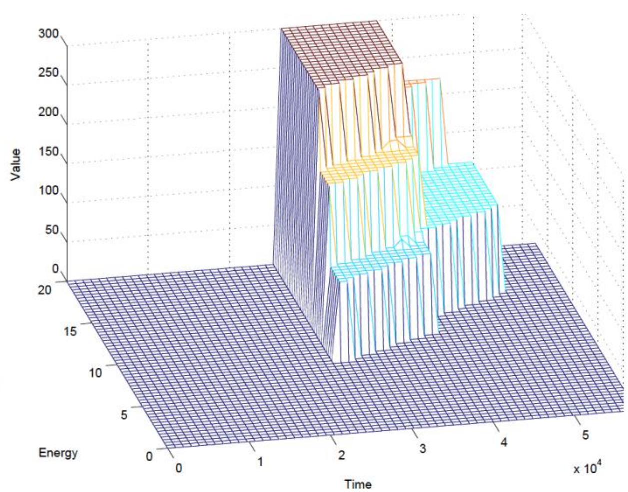
Value

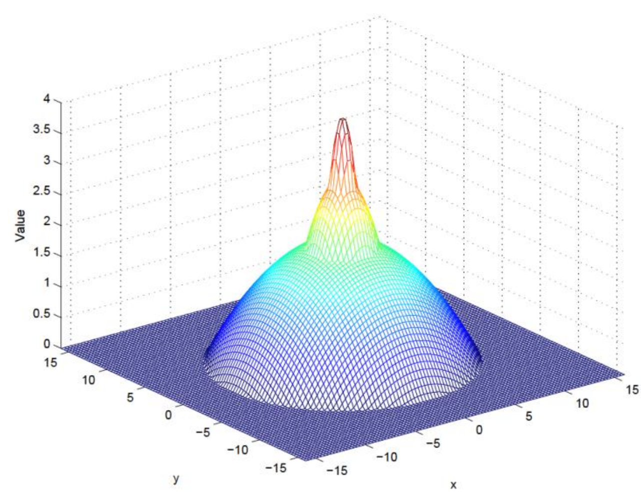
Linear constraints and value

Linear constraints, constant value

Quadratic constraints and value







Case Operations: ⊕, ⊗

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

Case Operations: ⊕, ⊗

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

- Similarly for ⊗
  - Expressions trivially closed under +, *
- What about max?
  - max(f_1, g_1) not pure arithmetic expression ☹

Case Operations: max

$$\max\left(\begin{cases}\phi_1 : & f_1\\ \phi_2 : & f_2\end{cases}, \begin{cases}\psi_1 : & g_1\\ \psi_2 : & g_2\end{cases}\right) =$$

?

Case Operations: max

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

Key point: still in case form!

Size blowup? We'll get to that…

Symbolic Dynamic Programming

- In a nutshell
  - $R(.)$, $P(.|.)$ defined as case statements
  - Value iteration uses case operations
    - $\oplus$, $\otimes$, max
  - If all VI operations maintain case, then
    - $V^h(.)$ is in case form for all horizons h!
- Almost there: we still need to define $\int_x$

SDP Regression Step

- **Continuous variables** x_j
  - $\int_x \delta[x - y] f(x) dx = f(y)$ triggers symbolic *substitution*
  - e.g., $\int_{x'_j} \delta[x'_j - g(\vec{x})] V' dx'_j \;=\; V'\{x'_j / g(\vec{x})\}$

$$\int_{x'_1} \delta[x'_1 - (x_1^2 + 1)] \left(\begin{cases} \underline{x'_1} < 2 : & \underline{x'_1} \\ \underline{x'_1} \geq 2 : & \underline{x'^2_1} \end{cases} \right) dx'_1 \;=\; \begin{cases} \underline{x_1^2 + 1} < 2 : & \underline{x_1^2 + 1} \\ \underline{x_1^2 + 1} \geq 2 : & (\underline{x_1^2 + 1})^2 \end{cases}$$

  - If g is *case*: need *conditional substitution*
    - see Sanner *et al* (UAI 2011)

UAI 2011

That's Discrete Action SDP!

- Value Iteration for h$\in$0..H
  - Regression step:

$$Q_a^{h+1}(\vec{b},\vec{x}) = R_a(\vec{b},\vec{x}) + \gamma \cdot \sum_{\vec{b}'} \int_{\vec{x}'} \left(\prod_{i=1}^{n} P(b_i'|\vec{b},\vec{x},a) \prod_{j=1}^{m} P(x_j'|\vec{b},\vec{b}',\vec{x},a) \right) V^h(\vec{b}',\vec{x}')d\vec{x}'$$

  - Maximization step:

$$V_{h+1} = \max_{a \in A} Q_a^{h+1}(\vec{b},\vec{x})$$

Continuous Actions

- Inventory control
  - Reorder based on stock, future demand
  - Action: $a(\vec{\Delta}); \vec{\Delta} \in \mathbb{R}^{|a|}$

- Need $\max_{\Delta}$ in Bellman backup

$$V_{h+1} = \max_{a \in A} \max_{\vec{\Delta}} Q_a^{h+1}(\vec{b}, \vec{x}, \vec{\Delta})$$

- How to compute?

Max-out: $\max_x f(x)$

- How to compute for case?

$$\max_x \left\{ \begin{array}{ll} \phi_1 : & f_1 \\ \vdots & \vdots \\ \phi_k : & f_k \end{array} \right. = \max_x \max_{i=1\ldots k} [\phi_i] \cdot f_i$$

$$= \max_{i=1\ldots k} \boxed{\max_x [\phi_i] \cdot f_i}$$

- Just $\max_x$ case partitions, case-max results!

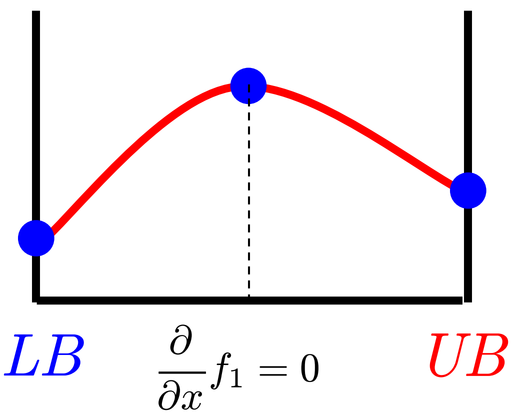
Example of Partition Max-out

$$\max_x [\phi_1] \cdot f_1$$

Consider function -∞ when constraints do not hold

$$\phi_1 := [x > -1] \wedge [x > y - 1] \wedge [x \cdot \; z] \wedge [x \cdot \; y + 1] \wedge [y > 0]$$

$$f_1 := x^2 - xy$$



$$LB := \begin{cases} y - 1 > -1 : & y - 1 \\ y - 1 \cdot \; -1 : & -1 \end{cases} \quad UB := \begin{cases} z < y + 1 : & z \\ z \geq y + 1 : & y + 1 \end{cases}$$

$$\frac{\partial}{\partial x} f_1 = 0 : Der0_x = \frac{y}{2}$$

Now an unconstrained max!

$$[\phi_{cons}] \max_{x \in \{LB, UB, Der0\}} f_1$$

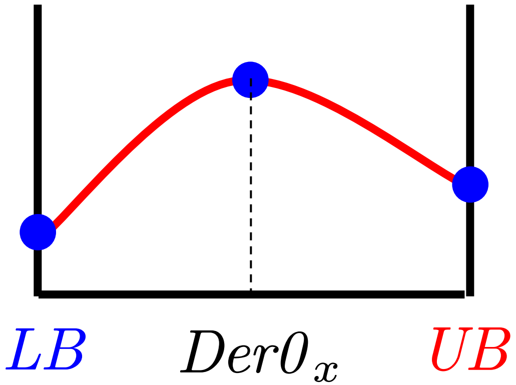
But how to evaluate?

What constraints here?

- those independent of x
- pairwise UB > Der0 > LB

Max-out Case Operation

- $\max_x$ case(x)
  - Reduced to partition max …*max* w.r.t. critical points
    - LB, UB
    - $Der0_x$
    - max(case(x/LB),
case(x/UB),
case(x/$Der0_x$))



See UAI 2011 paper for efficient substitutions into cases

  - Can even track substitutions through max to recover function of maximizing assignments!

Case $\rightarrow$ XADD

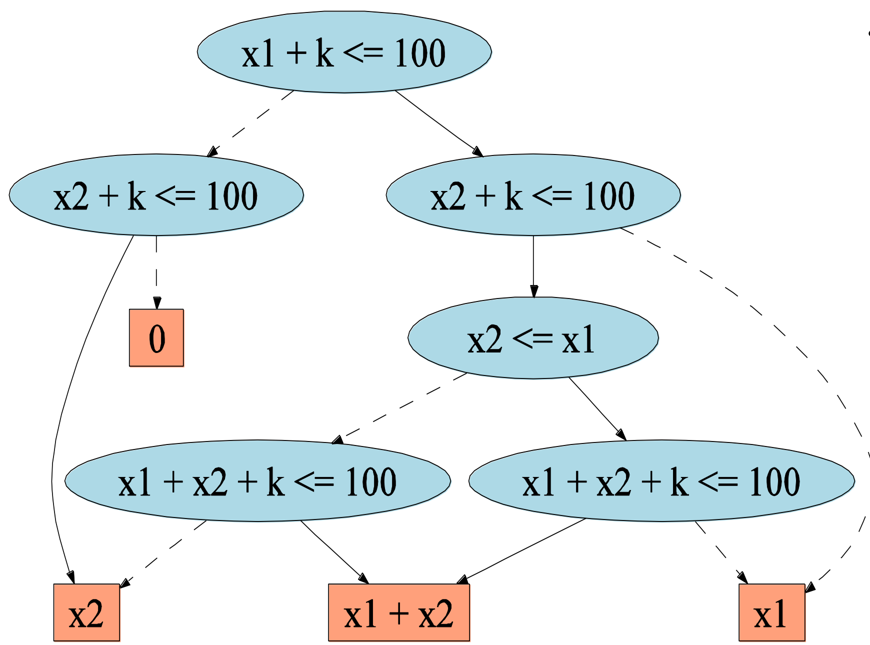
SDP needs an efficient data structure for

- compact, minimal case representation
- efficient case operations

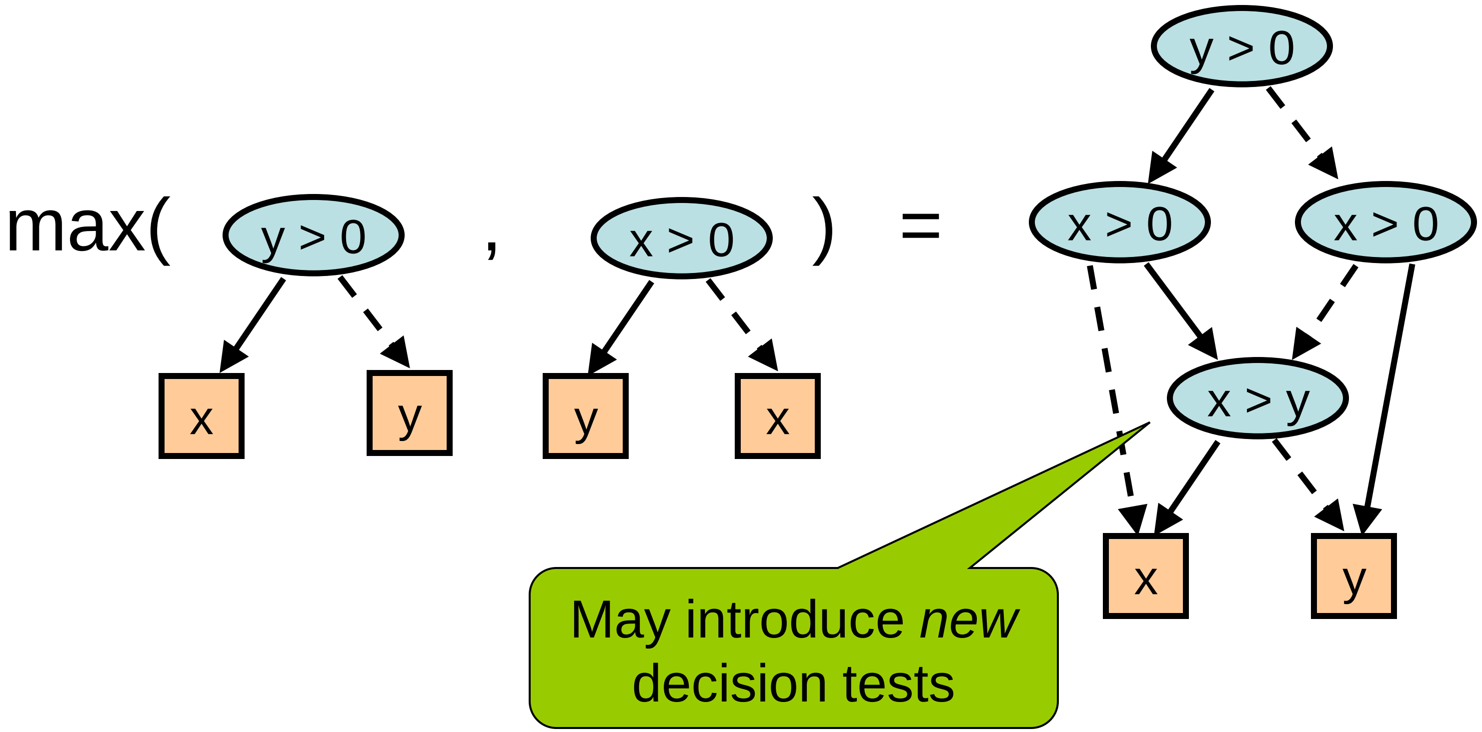
XADDs

- Extended ADD representation of case statements

$$V = \begin{cases} x_1 + k > 100 \wedge x_2 + k > 100 : & 0 \\ x_1 + k > 100 \wedge x_2 + k \cdot \; 100 : & x_2 \\ x_1 + k \cdot \; 100 \wedge x_2 + k > 100 : & x_1 \\ x_1 + x_2 + k > 100 \wedge x_1 + k \cdot \; 100 \wedge x_2 + k \cdot \; 100 \wedge x_2 > x_1 : & x_2 \\ x_1 + x_2 + k > 100 \wedge x_1 + k \cdot \; 100 \wedge x_2 + k \cdot \; 100 \wedge x_2 \cdot \; x_1 : & x_1 \\ x_1 + x_2 + k \cdot \; 100 : & x_1 + x_2 \end{cases}$$



XADD Maximization

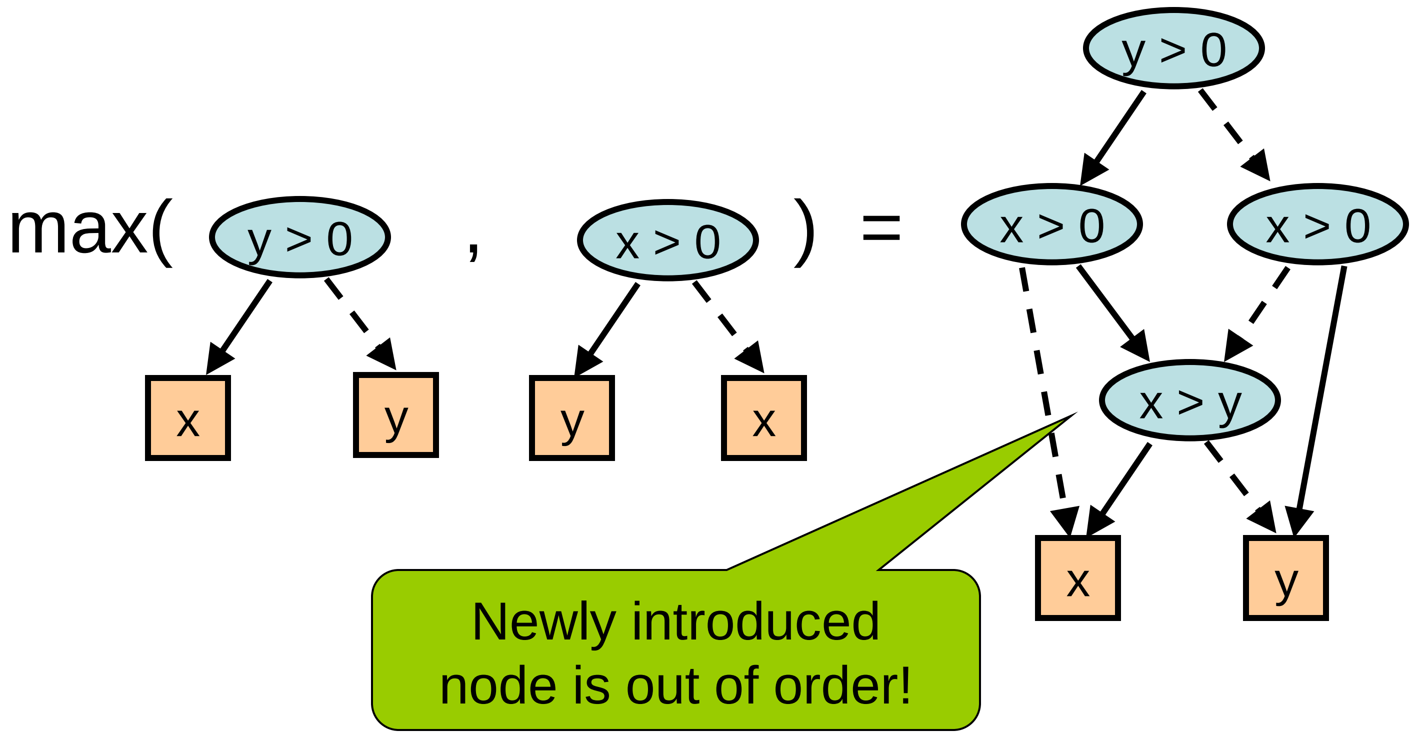


Maintaining XADD Orderings I

- Max may get variables out of order

Decision ordering (root→leaf)

- x > y
- y > 0
- x > 0

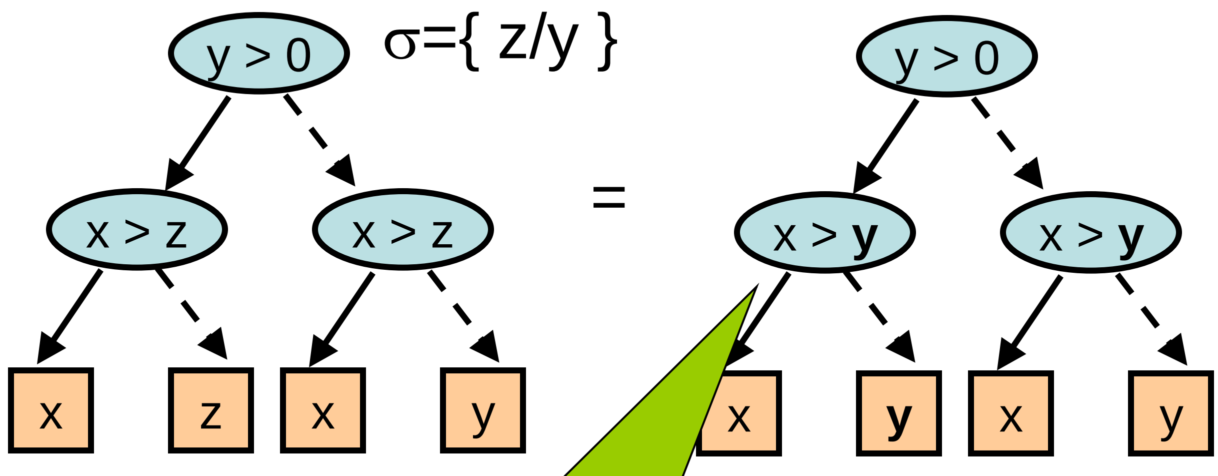


Maintaining XADD Orderings II

- Substitution may get vars out of order

Decision ordering (root→leaf):

- x > y
- y > 0
- x > z



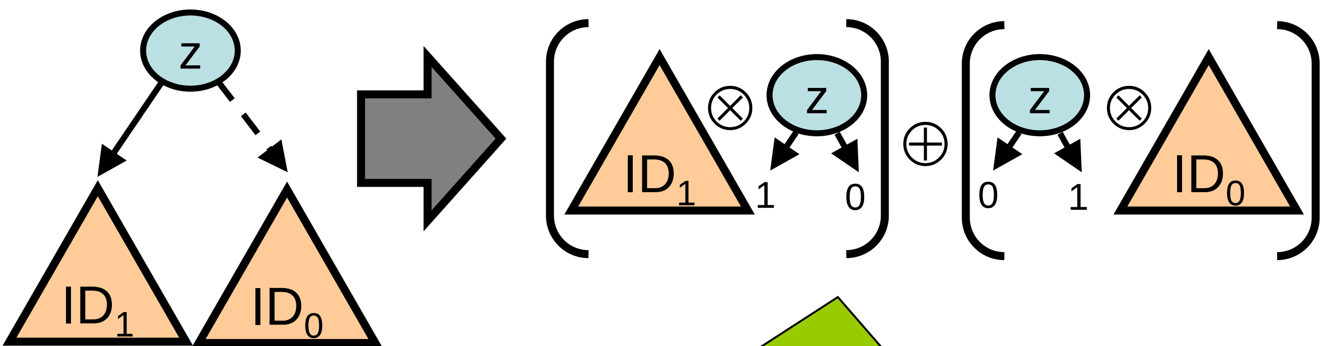
Substituted nodes are now out of order!

Correcting XADD Ordering

- Obtain *ordered* XADD from *unordered* XADD
  - key idea: binary operations maintain orderings

z is out of order

result will have z in order!

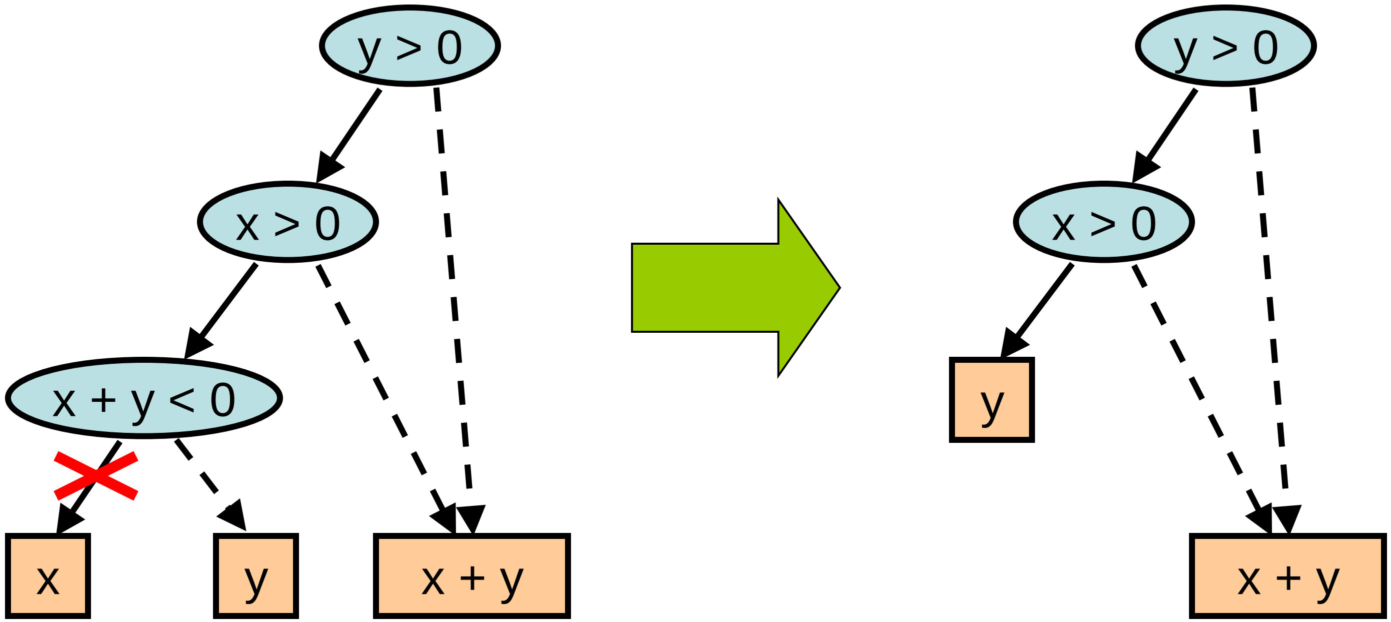


Inductively assume ID_1 and ID_0 are ordered.

All operands ordered, so applying ⊗, ⊕ produces ordered result!

Similar to Penberthy & Weld, AAAI-94

XADD Pruning

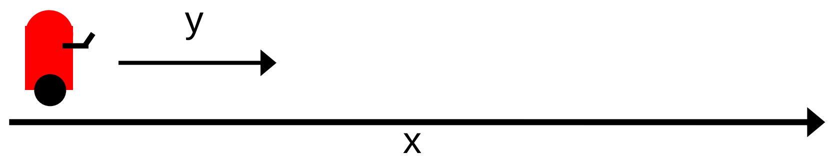


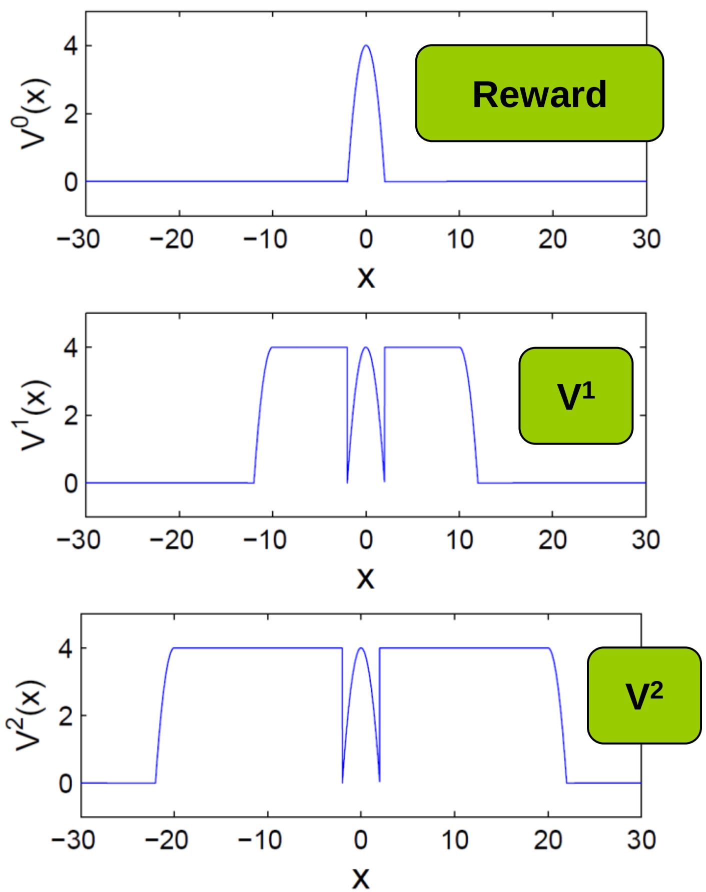
Node unreachable –
x + y < 0 always
false if x > 0 & y > 0

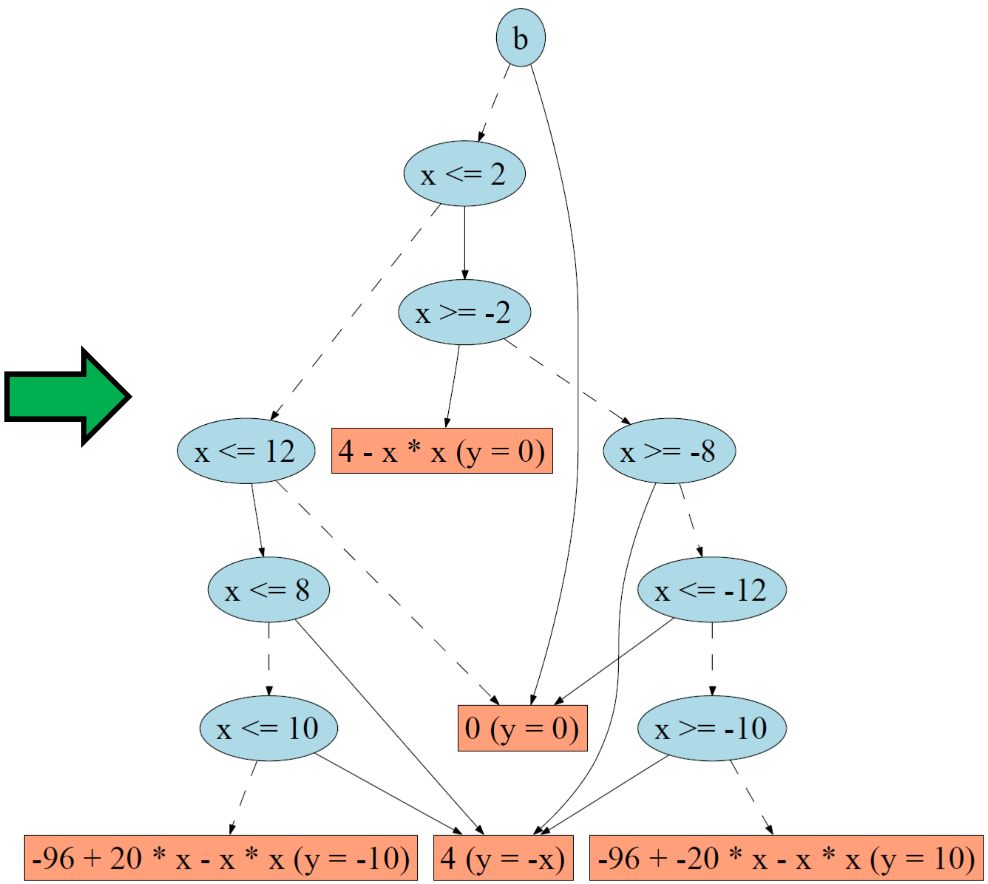
If **linear**, can detect with
feasibility checker of LP
solver & prune

Empirical Results

Illustrative Example



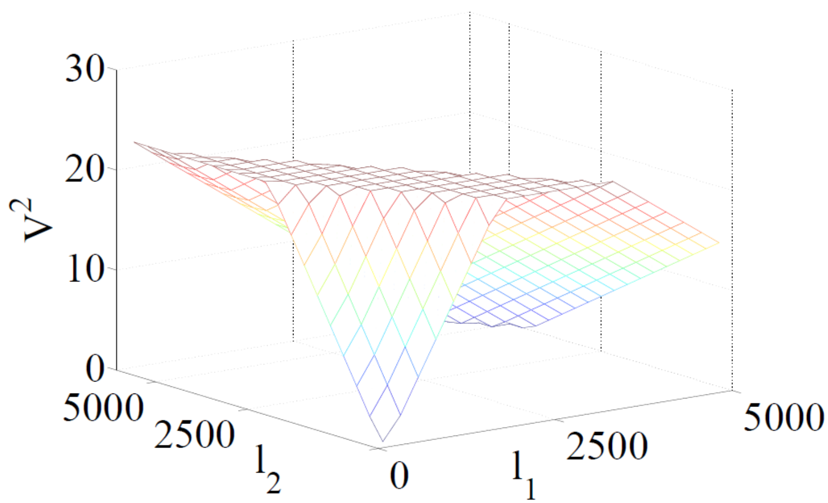


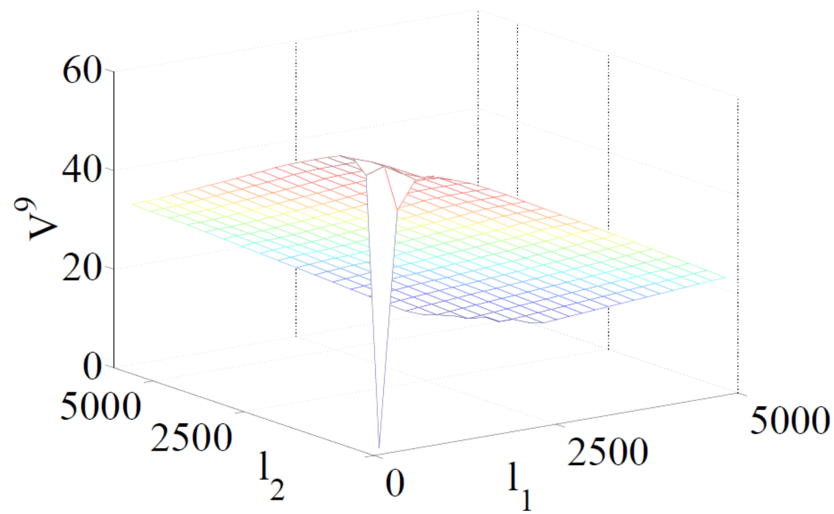


Symbolic Value (Symbolic Policy: y=…)

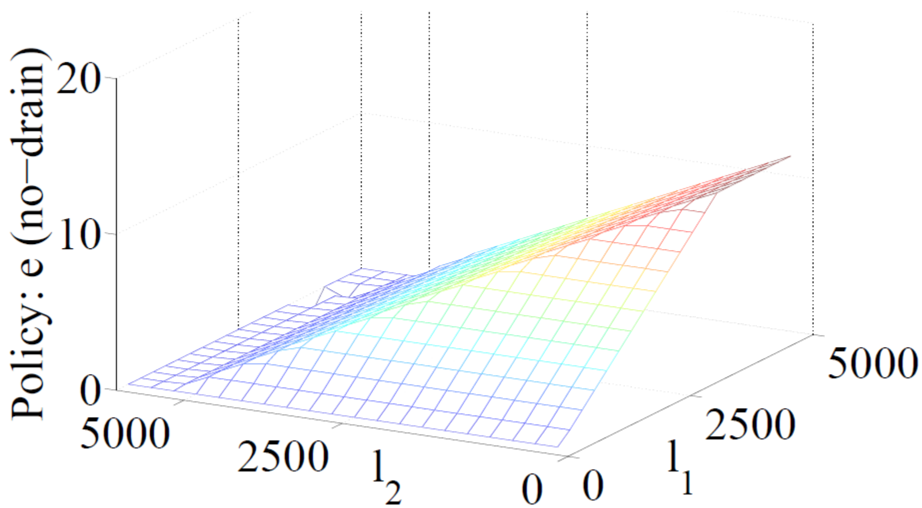
Reservoir Control

- Value Functions (vs level in each reservoir)

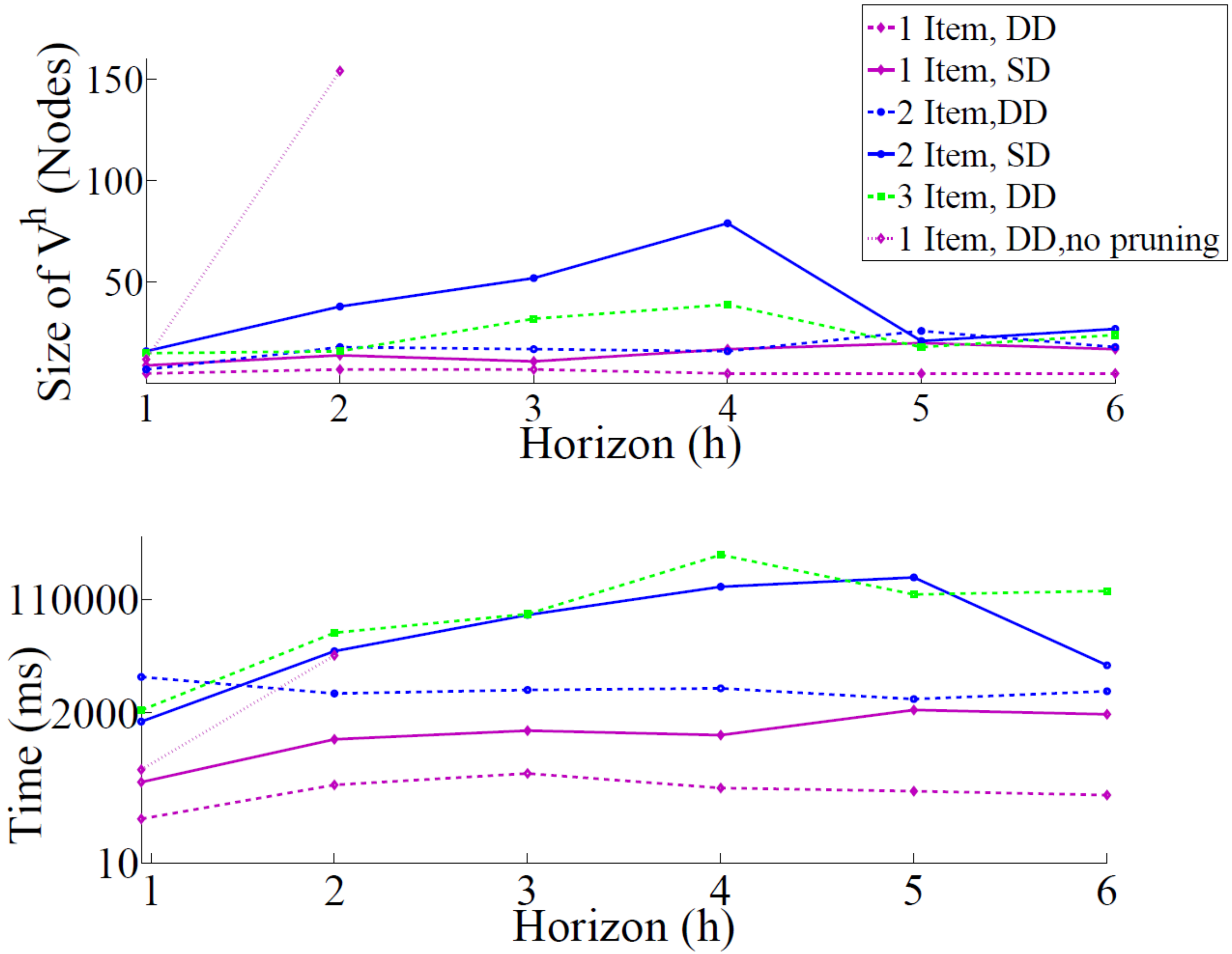




- Policy
(time to hold drain
 vs. reservoir levels)



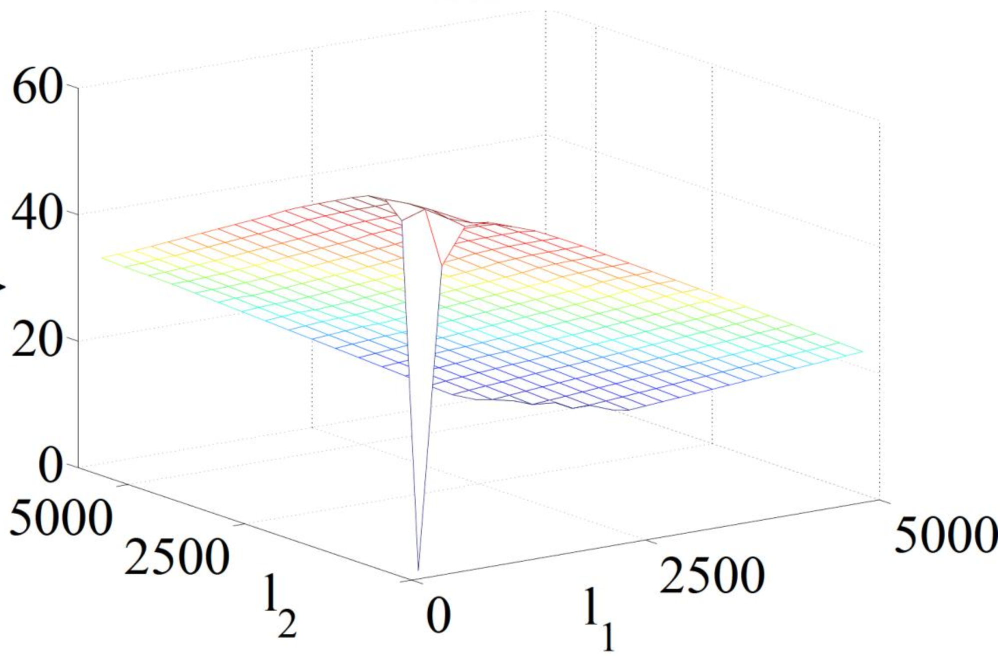
Inventory Control



Open Problems

- Nonlinear constraints
  - Optimal solutions for restricted cases

    e.g., quadratic, multilinear



- Bounded (interval) approximation
  - This XADD has > 1000 nodes!

Conclusions

- Key novel insights over Sanner *et* al (UAI 2011):
  - Introduced continuous actions
  - Showed how to compute $\max_x f(x)$ in closed form
  - All operations remains closed for value iteration
- Need compact case, efficient operations
  - Case $\rightarrow$ Extended ADD (XADD)
  - Extend to handle $\max_x f(x)$
- First exact, closed-form solutions to subset of **n-D continuous state-action MDPs**

First exact policies for continuous variant of multivariate inventory control… unsolved for 50+ years!