Jesse Hoey
Robert St-Aubin
Alan Hu
Craig Boutilier

Department of Computer Science
University of British Columbia

SPUDD
Stochastic Planning using Decision Diagrams

Motivation

MDP solution techniques must use structured representations for large problems

The more **compact** the representation, the better

ADDs provide a **compact** representation

SPUDD implements **VALUE ITERATION** using an **ADD** representation of the **MDP**

MDPs
Markov Decision Processes

Assumptions:

- Domain modeled as a fully observable MDP
- Finite set of states, $\mathcal{S}$, and actions, $\mathcal{A}$.
- The actions induce stochastic state transitions.
  $Pr(s,a,t)$: probability that action a, performed in state s, will result in a transition to state t.
- Real-valued reward function $\mathcal{R}$. $\mathcal{R}(s)$: immediate value of being in state s.
- Stationary policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$.
  π is a mapping from states to actions such that the optimality criterion is satisfied
- Optimality criterion: Expected total discounted reward (infinite horizon).
- Future rewards are discounted at a rate b

Value Iteration:

$$V^{n+1}(s) = R(s) + \max_{a \in \mathcal{A}} \left\{ \beta \sum Pr(s,a,t)\, V^{n}(t) \right\}$$

Stopping Criterion:

$$\| V^{n+1} - V^{n} \| < \frac{e(1-b)}{2b}$$

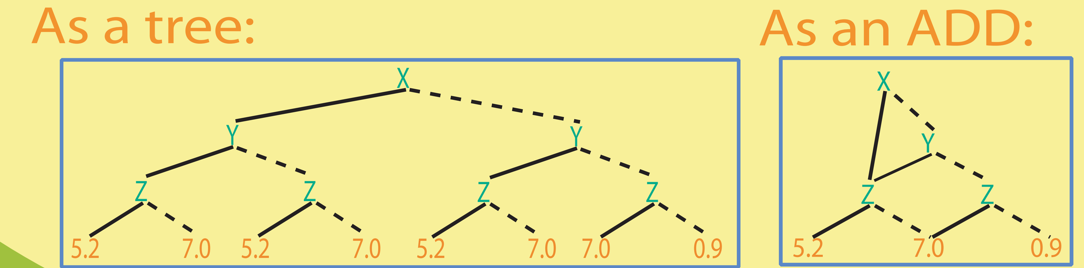
ADDs
Algebraic Decision Diagrams

What are they?

- Generalization of Bryant's Binary Decision Diagrams (BDDs)
- **Represent real-valued boolean functions $\mathcal{B}^n \rightarrow \mathcal{R}$ (may have multiple terminal nodes)**
- Canonical: each distinct function has a unique fully-reduced representation
- Requires a total ordering of the variables
- Can yield a much more compact structured representation as compared to the equivalent tree structure

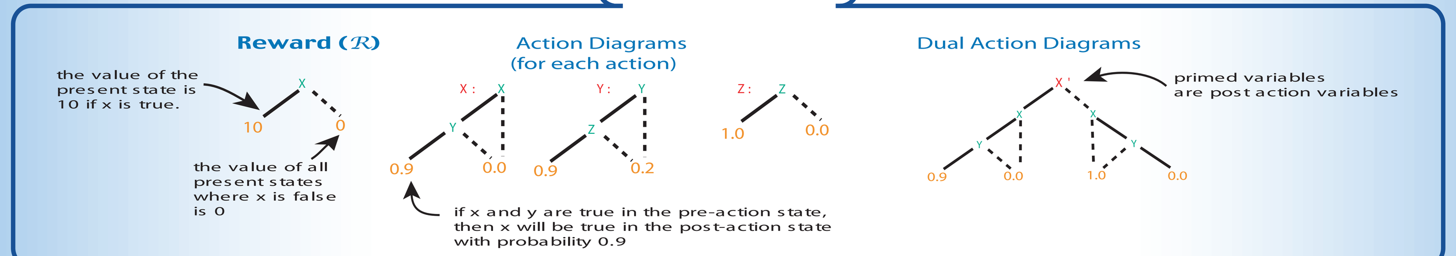
Example:

As a tree: As an ADD:



SPUDD Algorithm

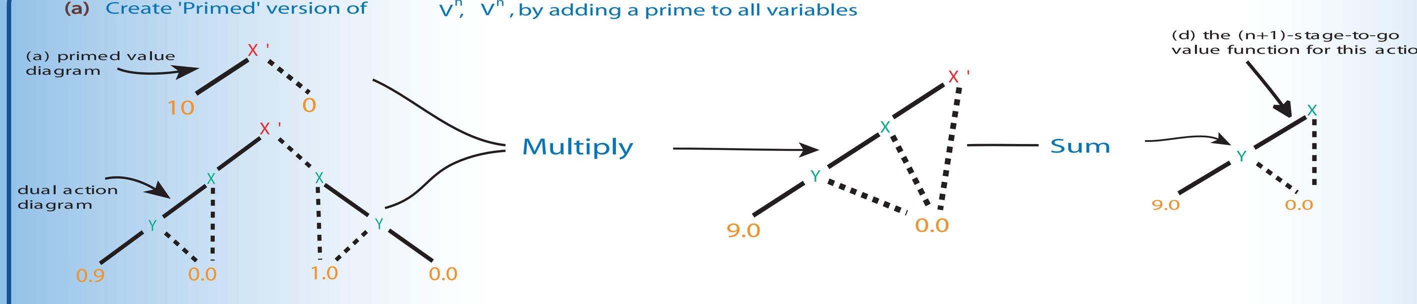
(1) Start with:



(2) Set $V^0 = \mathcal{R}$ (the 0–stage to go value function)
Set n=0

(3) Repeat until $\| V^{n+1} - V^{n} \| < \frac{e(1-b)}{2b}$

(a) Create 'Primed' version of V^n, $V^{n'}$, by adding a prime to all variables



do this for each action $a \in \mathcal{A}$

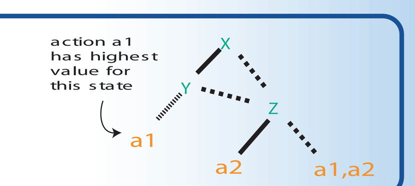
(b) Multiply $V^{n'}$ by dual action diagrams for each primed variable in $V^{n'}$

(c) Sum over sumdiagrams of primed varables

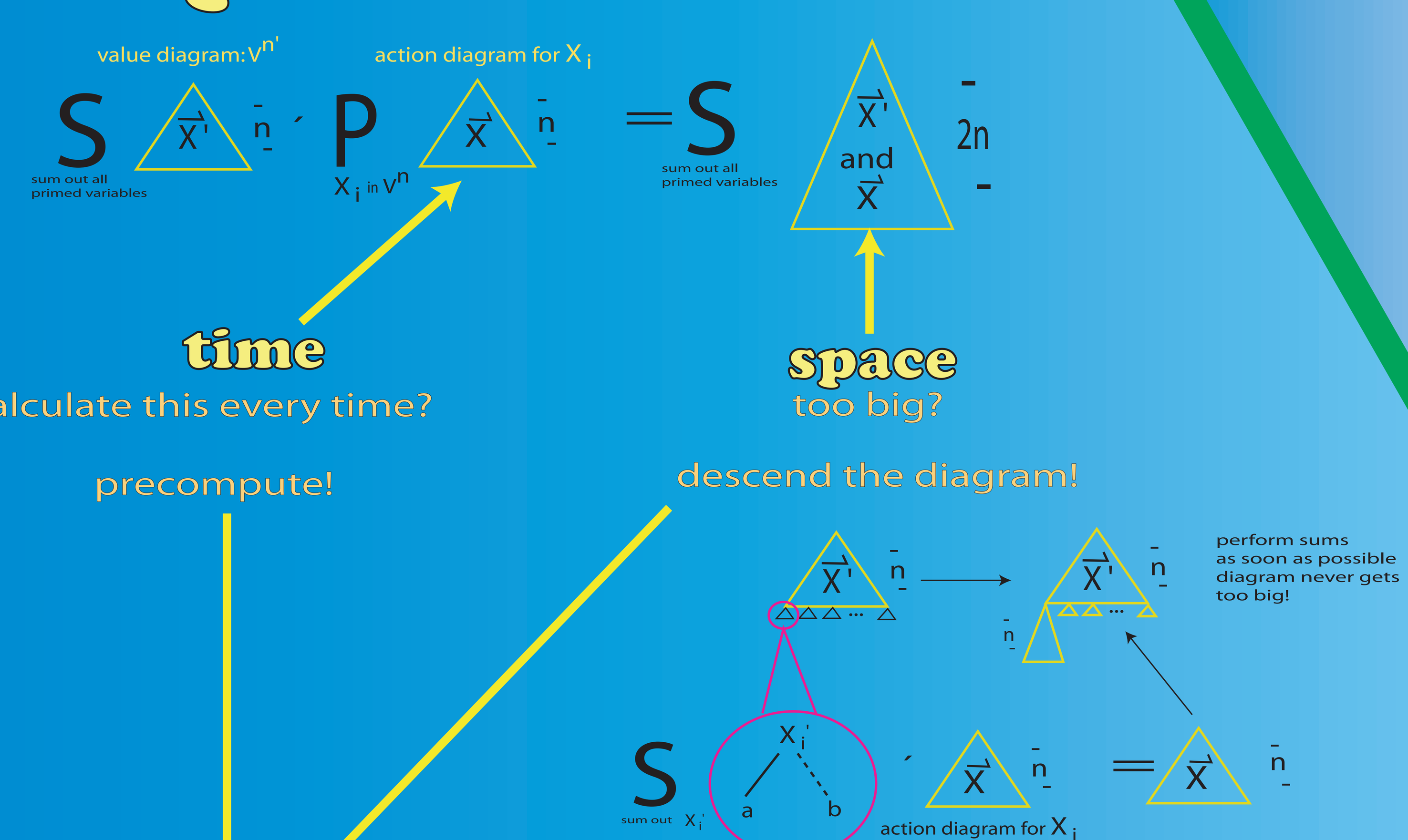
(d) V_a^{n+1}

(e) Maximize over all actions to create V^n

(4) perform final value iteration step → **assign to terminal nodes actions a which contributed in the maximization** → e-optimal policy diagram:



optimizations



tradeoff space/time

Break variables into subsets
precompute over subsets
perform space optimization
over subsets

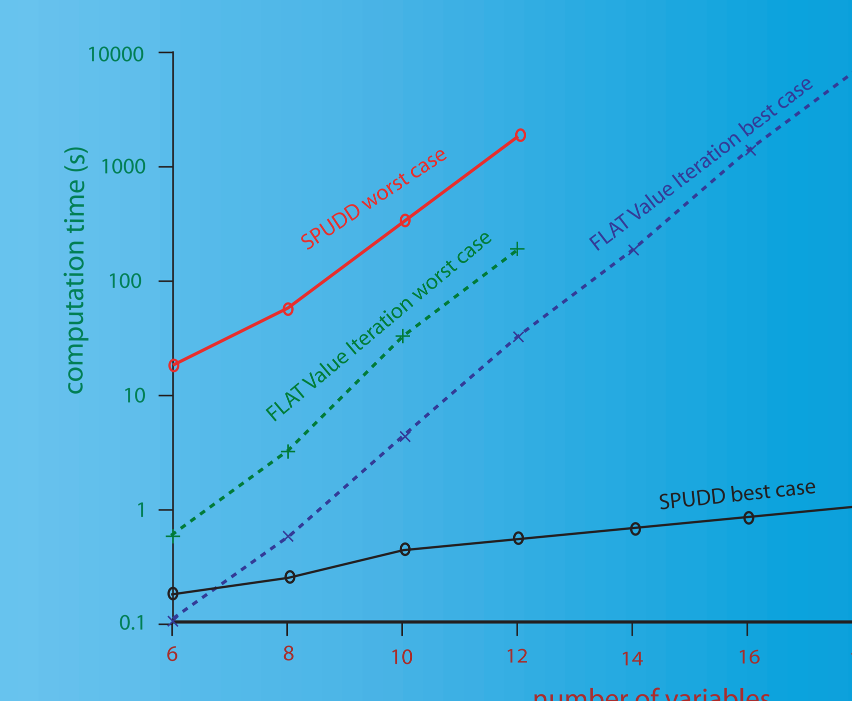
"tuning knob": number of subsets

performance

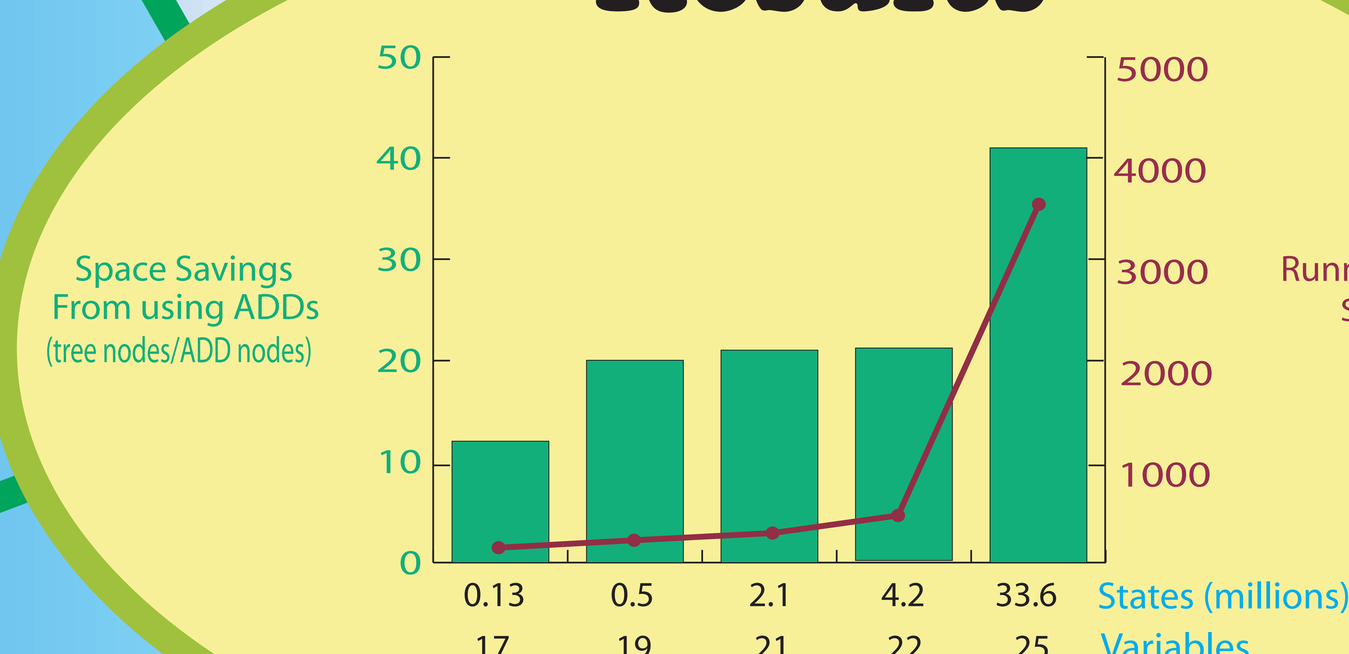
worst case:
every state has a unique value,
value function grows EXPONENTIALLY
in the number of variables

best case:
value function grows LINEARLY
in the number of variables

the best case for SPUDD actually involves a problem in which all variables are irrelevant to the value function. This LINEAR problem represents a best case in which all variables are required in the prediction of a state value



Results



Our 268 million state (28 variable) example completed in 14651 seconds (4 hours).
Tree-structured data was not obtainable for comparison.

Future Work

- ASPUDD: Approximation methods using ADDs
  ADDs are appropriate because they group similar states.
- Dynamic Variable Re-ordering
  Tries to overcome the limitation of a static variable ordering.
  Can reduce space requirements.
- Space/Time optimizations
  Further investigations into the capabilities of our space/time tuning knob.
  Dynamically choose subsets for further optimization.
- Other dynamic programming algorithms

SPUDD on the Web

http://www.cs.ubc.ca/spider/staubin/Spudd/index.html

RUN SPUDD ON YOUR OWN DATA

BROWSE PROBLEM EXAMPLES