Structured Representations

CMPSCI 686 Lecture #
Siddharth Srivastava,
Zhengzhu Feng
Today’s lecture

Exploiting Domain Structure to Improve the Efficiency of Inference

Introduction: MDP assumptions, representations etc.

Problems with CPT representations

Decision Diagrams and MDP representations

The Stochastic Planning Using Decision Diagrams (SPUDD) Algorithm

Symbolic Heuristic Search for Factored MDPs (Zhengzhu)
**Introduction: Notations and Assumptions**

Fully observable MDP

State space implicitly characterized by a set of variables

Actions affect the variables to create transitions between “states”.

Probabilities for each variable being true as a result of the action, as a function of its parents.

\[
X = \{ X_1, \ldots, X_n \} \\

P^a_{X_i}(X_1', \ldots, X_n') \\
X' = \{ X_1', \ldots, X_n' \}
\]

\[
P^a(X_i', X_j' \mid X_1, \ldots, X_n) = P^a_{X_i}(X_1, \ldots, X_n) P^a_{X_j}(X_1, \ldots, X_n), i \neq j
\]
Notations (ctd)

Reward function $R(\bar{x})$

Value function $V(\bar{x})$
Problems with CPT representations

Maintain a CPT for every possible action from every state, for resulting value of every variable
CPT will have all possible combinations of parent values
Similar blow up for Value function
Systems usually don’t have so much unique information in the CPT/Value function (Don’t-Cares, Similar Values)
Scope for merging
Decision Trees

Graphs for computing functions on boolean (discrete) variables

DTs satisfy two constraints:

Along every path from root to a leaf, any variable occurs at most once.
Along every path from root to a leaf, variables occur in the same order.
Binary/Algebraic Decision Diagrams

Reduced versions of DTs:
- Merge nodes that are roots of isomorphic graphs
- Remove nodes whose left and right branches go to isomorphic graphs

Sensitive to Ordering
Universal ordering → easy search for isomorphic subgraphs → ease in reductions

Binary Decision Diagrams: $U=B$
Algebraic Decision Diagrams: $U=R$
Decision Trees to Decision Diagrams

Merge Similar Nodes
Remove Redundant Nodes
Binary/Algebraic Decision Diagrams: Operations

\[ D(f) = \text{Decision Diagram of function } f \]

Wish to compute \( D( f \circ op \ g) \) given \( D(f) \) and \( D(g) \)

Inductive Rules for computing \( D \)

Induction is based upon the fact:

\[
f(X_1, \ldots, X_n) = X_1 f(1, X_2, \ldots, X_n) + \overline{X}_1 f(0, X_2, \ldots, X_n)
\]

\[
f \circ g(X_1, \ldots, X_n) = X_1 f(1, X_2, \ldots, X_n) \circ g(1, X_2, \ldots, X_n) + \overline{X}_1 f(0, X_2, \ldots, X_n) \circ g(0, X_2, \ldots, X_n)
\]
BDD/ADDs: Operations

Inductive Rules for operations:

\[ \text{i} \langle \text{op} \rangle \text{j} \rightarrow \text{i op j} \]

\[ \text{i} \langle \text{op} \rangle \rightarrow \text{i} \langle \text{op} \rangle 1 2 \]

\[ \rightarrow \text{i} \langle \text{op} \rangle 1 \]

\[ \rightarrow \text{i} \langle \text{op} \rangle 2 \]
BDD/ADDs: Operations (ctd.)

\[ f \circ g(X_1, \ldots, X_n) = X_1 f(1, X_2, \ldots, X_n) \circ g(1, X_2, \ldots, X_n) + \bar{X}_1 f(0, X_2, \ldots, X_n) \circ g(0, X_2, \ldots, X_n) \]
BDD/ADDs: Operations (ctd.)

\[
\text{IF } v = v'
\]

\[
\begin{align*}
\text{IF } v = v' & \\
&
\end{align*}
\]
Now possible to perform any binary operation on the diagrams
Use this knowledge in the Value Iteration Algorithm
Each probability function $p_{x'_i}(x_1, \ldots, x_n)$ and the value function is represented via an ADD rooted at $x'_i$.

But we also need the $1-P$ value to get the false branch for ADD representation.

Compute the $Q_i$ function to get both sides of the branches representing the outcomes $x'_i=$true and $x'_i=$false.

Value function is also represented via an ADD

Key motive of the SPUDD algorithm: exploit structure in CPTs and Value function via ADDs to reduce complexity.
Figure 2: Small FACTORY example: (a) action network for action bolt; (b) ADD representation of CPTs (action diagrams); and (c) immediate reward network and ADD representation of the reward table.

[Optimal and Stochastic Planning Using Decision Diagrams, Hoey et al]
Add Representation: Example (ctd.)

Figure 4: First Bellman backup for the Value Iteration using ADDs algorithm. (a) 0-stage-to-go primed value diagram, and dual action diagram for variable $C'$, $Q_{C'}^{0}$. (b) Intermediate result after multiplying $V^{10}$ with $Q_{C'}^{0}$. (c) Intermediate result after quantifying over $C'$.

[Optimal and Stochastic Planning Using Decision Diagrams, Hoey et al]
The SPUDD Algorithm

Value Iteration formula:

\[ V^{n+1}(s) = R(s) + \max_{a \in A} \left\{ \beta \sum_{t \in S} Pr(s, a, t) \cdot V^n(t) \right\} \]

Look at \( V^i(X) \) as the value of the state with that configuration of \( X_i \)-s, with \( i \)-steps-to-go; \( V^0 = R \)

For value iteration, we need the expected value of an action:

\[ h(x_1, \ldots, x_n) = \sum_{x_1', \ldots, x_n'} V^i(x_1', \ldots, x_n') P(x_1' | x_1, \ldots, x_n) \ldots P(x_n' | x_1, \ldots, x_n) \]

Break this up as:
The SPUDD Algorithm (ctd.)

\[ h(x_1, \ldots, x_n) = \sum_{x_n'} \left( \sum_{x_1'} V^{i'}(x_1', \ldots, x_n') P(x_1'|x_1, \ldots, x_n) \right) \cdots P(x_n'|x_1, \ldots, x_n) \]

Starting from the innermost Sum, the calculation is now a repetition of alternating sums and products.

Can be done using the ADDs: each P is an ADD; get several intermediate V ADD’s.

Only new case: the sum over each variable: achieved by adding the “True” and “False” sub-diagrams in the intermediate Diagrams of \( V*P(x_j) \).

Complexity of these computations depends only on the diagrams: not the state space.
The SPUDD Algorithm (ctd.)

After every step $i$, add reward $R$ to $\beta h$ and compute the max of ADDs for all possible actions to get $V^{i+1}$.

Stopping Condition

The final Value function is also an ADD
Compute $V^i - V^{i-1}$
Stop if the max leaf of $V^i - V^{i-1}$ is less than $\epsilon$. 