Exact Reasoning:
AND/OR Search and Hybrids

COMPSCI 276, Fall 2009
Set 8, Rina dechter
Approximation Techniques
bounded inference

COMPSCI 276, Fall 2009
Set 6: Rina Dechter

Reading: Primary: Class Notes (7)
Probabilistic Inference Tasks

- Belief updating:
  \[ \text{BEL}(X_i) = P(X_i = x_i \mid \text{evidence}) \]

- Finding most probable explanation (MPE)
  \[ \bar{x}^* = \arg \max_{x} P(\bar{x}, e) \]

- Finding maximum a-posteriory hypothesis
  \[ (a_1^*, ..., a_k^*) = \arg \max_{\tilde{a}} \sum_{X/A} P(\bar{x}, e) \]
  \[ A \subseteq X : \text{hypothesis variables} \]

- Finding maximum-expected-utility (MEU) decision
  \[ (d_1^*, ..., d_k^*) = \arg \max_{\tilde{d}} \sum_{X/D} P(\bar{x}, e) U(\bar{x}) \]
  \[ D \subseteq X : \text{decision variables} \]
  \[ U(\bar{x}) : \text{utility function} \]
Belief Updating

\[ P(\text{lung cancer}=\text{yes} \mid \text{smoking}=\text{no}, \text{dyspnoea}=\text{yes}) = ? \]
Conditioning generates the probability tree

\[ P(a, e = 0) = P(a) \sum_b P(b \mid a) \sum_c P(c \mid a) \sum_b P(d \mid a, b) \sum_{e=0} P(e \mid b, c) \]

Complexity of conditioning: exponential time, linear space
Conditioning + Elimination

\[ P(a, e = 0) = P(a) \sum_b P(b \mid a) \sum_c P(c \mid a) \sum_d P(d \mid a, b) \sum_{e=0} P(e \mid b, c) \]

Idea: conditioning until \( W^* \) of a (sub)problem gets small
Loop-cutset decomposition

- You condition until you get a polytree

\[
P(B|F=0) = P(B, A=0|F=0) + P(B, A=1|F=0)
\]

Loop-cutset method is time exp in loop-cutset size
And linear space
OR search space

Ordering: A B E C D F
AND/OR search space

Primal graph

DFS tree
OR vs AND/OR

AND/OR

OR

AND

OR

AND

OR

AND

OR

AND

OR

AND

OR

AND
**AND/OR vs. OR**

AND/OR size: exp(4), OR size exp(6)
AND/OR vs. OR

No-goods
(A=1, B=1)
(B=0, C=0)
AND/OR vs. OR

(A=1, B=1)
(B=0, C=0)
# OR space vs. AND/OR space

<table>
<thead>
<tr>
<th>width</th>
<th>height</th>
<th>OR space</th>
<th>AND/OR space</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>time(sec.)</td>
<td>nodes</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>3.154</td>
<td>2,097,150</td>
</tr>
<tr>
<td>4</td>
<td>9</td>
<td>3.135</td>
<td>2,097,150</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>3.124</td>
<td>2,097,150</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>3.125</td>
<td>2,097,150</td>
</tr>
<tr>
<td>5</td>
<td>13</td>
<td>3.104</td>
<td>2,097,150</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>3.125</td>
<td>2,097,150</td>
</tr>
<tr>
<td>6</td>
<td>9</td>
<td>3.124</td>
<td>2,097,150</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>3.125</td>
<td>2,097,150</td>
</tr>
<tr>
<td>5</td>
<td>13</td>
<td>3.114</td>
<td>2,097,150</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>3.114</td>
<td>2,097,150</td>
</tr>
</tbody>
</table>
The AND/OR search tree of $R$ relative to a spanning-tree, $T$, has:
- Alternating levels of: OR nodes (variables) and AND nodes (values)

Successor function:
- The successors of OR nodes $X$ are all its consistent values along its path.
- The successors of AND $<X,v>$ are all $X$ child variables in $T$

A solution is a consistent subtree

Task: compute the value of the root node
From DFS trees to pseudo-trees (Freuder 85, Bayardo 95)

(a) Graph

(b) DFS tree
depth=3

(c) pseudo-tree
depth=2

(d) Chain
depth=6
From DFS trees to Pseudo-trees

DFS tree
 depth = 3

pseudo- tree
 depth = 2
Finding min-depth backbone trees

Finding min depth DFS, or pseudo tree is NP-complete, but:

- Given a tree-decomposition whose tree-width is $w^*$, there exists a pseudo tree $T$ of $G$ whose depth satisfies (Bayardo and Mirankar, 1996, Bodlaender and Gilbert, 91):

  $$m \leq w^* \log n,$$
Generating pseudo-trees from Bucket trees

**d:** A B C E D F

Bucket-tree based on d

Induced graph

Bucket-tree

Bucket-tree used as pseudo-tree

AND/OR search tree
AND/OR Search-tree properties

\((k = \text{domain size}, \ m = \text{pseudo-tree depth}, \ n = \text{number of variables})\)

- **Theorem:** Any AND/OR search tree based on a pseudo-tree is sound and complete (expresses all and only solutions)

- **Theorem:** Size of AND/OR search tree is \(O(n \ k^m)\)
  - Size of OR search tree is \(O(k^n)\)

- **Theorem:** Size of AND/OR search tree can be bounded by \(O(\exp(w \times \log n))\)

- **Related to:** (Freuder 85; Dechter 90, Bayardo et. al. 96, Darwiche 1999, Bacchus 2003)

- When the pseudo-tree is a chain we get an OR space
Tasks and value of nodes

- \( V(n) \) is the value of the tree \( T(n) \) for the task:
  - **Counting**: \( v(n) \) is number of solutions in \( T(n) \)
  - **Consistency**: \( v(n) \) is 0 if \( T(n) \) inconsistent, 1 otherwise.
  - **Optimization**: \( v(n) \) is the optimal solution in \( T(n) \)
  - **Belief updating**: \( v(n) \), probability of evidence in \( T(n) \).
  - **Partition function**: \( v(n) \) is the total probability in \( T(n) \).

- **Goal**: compute the value of the root node recursively using dfs search of the AND/OR tree.

- **Theorem**: Complexity of AO dfs search is
  - **Space**: \( O(n) \)
  - **Time**: \( O(n \cdot k^m) \)
  - **Time**: \( O(exp(w \cdot \log n)) \)
A Bayesian Network

**Winter? (A)**

- **Sprinkler? (B)**
- **Rain? (C)**
- **Wet Grass? (D)**
- **Slippery Road? (E)**

**Tables:**

<table>
<thead>
<tr>
<th>A</th>
<th>( \Theta_A )</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>.6</td>
</tr>
<tr>
<td>false</td>
<td>.4</td>
</tr>
</tbody>
</table>

| A   | B   | \( \Theta_{B|A} \) |
|-----|-----|----------------|
| true| true| .2             |
| true| false| .8            |
| false| true| .75           |
| false| false| .25         |

| B   | C   | \( \Theta_{C|A} \) |
|-----|-----|----------------|
| true| true| .8             |
| true| false| .2           |
| false| true| .1            |
| false| false| .9         |

| C   | D   | \( \Theta_{D|BC} \) |
|-----|-----|----------------|
| true| true| .95            |
| true| false| .05         |
| false| true| .9            |
| false| false| .1          |

| C   | E   | \( \Theta_{E|C} \) |
|-----|-----|----------------|
| true| true| .7             |
| true| false| .3            |
| false| true| 0              |
| false| false| 1            |
Belief-updating on example

A Bayesian Network
A Bayesian Network

<table>
<thead>
<tr>
<th>A</th>
<th>( \Theta_A )</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>.6</td>
</tr>
<tr>
<td>false</td>
<td>.4</td>
</tr>
</tbody>
</table>

| A  | B  | \( \Theta_{B|A} \) |
|----|----|------------------|
| true | true | .2 |
| true | false | .8 |
| false | true | .75 |
| false | false | .25 |

| B  | C  | \( \Theta_{D|BC} \) |
|----|----|------------------|
| true | true | true | .95 |
| true | true | false | .05 |
| true | false | true | .9 |
| true | false | false | .1 |
| false | true | true | .8 |
| false | true | false | .2 |
| false | false | true | .1 |
| false | false | false | 1 |

| C  | E  | \( \Theta_{E|C} \) |
|----|----|------------------|
| true | true | true | 7 |
| true | false | true | .3 |
| false | true | true | 0 |
| false | true | false | 1 |

\[
P(A=0) \quad B \quad E \quad D \quad C
\]

\[
P(B=0|A=0) \quad P(B=1|A=0)
\]

\[
P(E=0|A=0,B=0) \quad P(E=1|A=0,B=0)
\]

\[
P(D=0|B=0,C=0) \times P(C=0|A=0)
\]

\[
P(D=1|B=0,C=1) \times P(C=1|A=0)
\]

\[
P(D=0|B=0,C=0) \times P(C=0|A=0)
\]

\[
P(D=1|B=0,C=1) \times P(C=1|A=0)
\]

\[
P(D=0|B=1,C=0) \times P(C=0|A=0)
\]

\[
P(D=1|B=1,C=1) \times P(C=1|A=0)
\]

\[
P(D=0|B=1,C=0) \times P(C=0|A=0)
\]

\[
P(D=1|B=1,C=1) \times P(C=1|A=0)
\]

\[
P(D=0|B=0,C=1) \times P(C=1|A=0)
\]

\[
P(D=1|B=0,C=1) \times P(C=1|A=0)
\]

\[
P(D=0|B=1,C=0) \times P(C=0|A=0)
\]

\[
P(D=1|B=1,C=1) \times P(C=1|A=0)
\]

\[
P(D=0|B=0,C=1) \times P(C=1|A=0)
\]

\[
P(D=1|B=0,C=1) \times P(C=1|A=0)
\]

\[
P(D=0|B=1,C=0) \times P(C=0|A=0)
\]

\[
P(D=1|B=1,C=1) \times P(C=1|A=0)
\]
A Bayesian Network

(d)
**AND/OR Tree DFS Algorithm (Belief Updating)**

**Value** of node = updated belief for sub-problem below

**AND node:** Combination operator (product)

**OR node:** Marginalization operator (summation)

---

### Evidence: E = 0

### Evidence: D = 1

---

### Result: P(D=1, E=0)
## Complexity of AND/OR Tree Search

<table>
<thead>
<tr>
<th></th>
<th>AND/OR tree</th>
<th>OR tree</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Space</strong></td>
<td>$O(n)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td><strong>Time</strong></td>
<td>$O(n \times k^m)$</td>
<td>$O(k^n)$</td>
</tr>
<tr>
<td></td>
<td>$O(n \times k^{w* \log n})$</td>
<td></td>
</tr>
</tbody>
</table>


$k$ = domain size  
$m$ = depth of pseudo-tree  
$n$ = number of variables  
$w^*$ = treewidth
From Search Trees to Search Graphs

- Any two nodes that root identical subtrees (subgraphs) can be **merged**
Any two nodes that root identical subtrees (subgraphs) can be merged.
AND/OR Tree
An AND/OR graph
Merging Based on Context

One way of recognizing nodes that can be merged:

context \( (X) \) = ancestors of \( X \) in pseudo tree that are connected to \( X \), or to descendants of \( X \)
AND/OR Search Graph

Constraint Satisfaction – Counting Solutions

pseudo tree

context minimal graph
AND/OR Tree DFS Algorithm (Belief Updating)

**Value of node**: Updated belief for sub-problem below

**OR node**: Marginalization operator (summation)

**AND node**: Combination operator (product)

---

### Evidence: E = 0

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>E=0</th>
<th>E=1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>.4</td>
<td>.6</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>.5</td>
<td>.5</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>.7</td>
<td>.3</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>.2</td>
<td>.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>B = 0</th>
<th>B = 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>.2</td>
<td>.8</td>
</tr>
<tr>
<td>1</td>
<td>.7</td>
<td>.3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>P(A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>.6</td>
</tr>
<tr>
<td>1</td>
<td>.4</td>
</tr>
</tbody>
</table>

### Evidence: D = 1

<table>
<thead>
<tr>
<th>B</th>
<th>C</th>
<th>D=0</th>
<th>D=1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>.2</td>
<td>.8</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>.1</td>
<td>.9</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>.3</td>
<td>.7</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>.5</td>
<td>.5</td>
</tr>
</tbody>
</table>

### Context

**Result**: $P(D=1, E=0)$
AND/OR Graph DFS Algorithm (Belief Updating)

Evidence: $E = 0$

Cache table for $D$

Result: $P(D=1, E=0)$
AND/OR context minimal graph
How Big Is the Context?

Theorem: *The maximum context size for a pseudo tree is equal to the treewidth of the graph along the pseudo tree.*
Treewidth vs. Pathwidth

**Tree**

- **treewidth** = 3
- \( = (\text{max cluster size}) - 1 \)

**Chain**

- **pathwidth** = 4
- \( = (\text{max cluster size}) - 1 \)
# Complexity of AND/OR Graph Search

<table>
<thead>
<tr>
<th></th>
<th>AND/OR graph</th>
<th>OR graph</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Space</strong></td>
<td>$O(n , k^{w^*})$</td>
<td>$O(n , k^{p,w^*})$</td>
</tr>
<tr>
<td><strong>Time</strong></td>
<td>$O(n , k^{w^*})$</td>
<td>$O(n , k^{p,w^*})$</td>
</tr>
</tbody>
</table>

$k = \text{domain size}$

$n = \text{number of variables}$

$w^* = \text{treewidth}$

$\text{pw}^* = \text{pathwidth}$

$w^* \leq \text{pw}^* \leq w^* \log n$
AND/OR search algorithms

- **AO(i)**
  - $i =$ the max size of a cache table (i.e. number of variables in a context)

Space:  
- $i = 0$: $O(n)$
- $i = w^*$: $O(n \exp w^*)$

Time:  
- $i = 0$: $O(\exp (w^* \log n))$
- $i = w^*$: $O(n \exp w^*)$
Searching AND/OR Graphs

- **AO**(i): searches depth-first, cache i-context
  - i = the max size of a cache table (i.e. number of variables in a context)

**Space:** $O(n)$
**Time:** $O(\exp(w* \log n))$

**Space:** $O(\exp(i))$
**Time:** $O(\exp(m_i+i))$

**Space:** $O(\exp w^*)$
**Time:** $O(\exp w^*)$
Caching

context(D)={D}
context(F)={F}
All four search spaces

Full OR search tree

Context minimal OR search graph

Full AND/OR search tree

Context minimal AND/OR search graph
All four search spaces

Full OR search tree

Full AND/OR search tree

Context minimal OR search graph

Context minimal AND/OR search graph
Available code

- http://graphmod.ics.uci.edu/group/Software