Reasoning with graphical models

Slides Set 6:
AND/OR search for Probabilistic Networks

Rina Dechter

(Dechter chapters 6 and 7)

slides6 W2020
Overall Perspective

- **Class 1: Introduction and Inference**
  
- **Class 2: Search**
  
- **Class 3: Variational Methods and Monte-Carlo Sampling**
### Types of queries

<table>
<thead>
<tr>
<th>Type of Inference</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max-Inference</td>
<td>$f(x^*) = \max_x \prod_{\alpha} f_\alpha(x_\alpha)$</td>
</tr>
<tr>
<td>Sum-Inference</td>
<td>$Z = \sum_x \prod_{\alpha} f_\alpha(x_\alpha)$</td>
</tr>
<tr>
<td>Mixed-Inference</td>
<td>$f(x_M^*) = \max_{x_M} \sum_{x_S} \prod_{\alpha} f_\alpha(x_\alpha)$</td>
</tr>
</tbody>
</table>

- **NP-hard**: exponentially many terms
- We will focus on **approximation** algorithms
  - **Anytime**: very fast & very approximate! Slower & more accurate

slides6 W2020
Outline: Search for Graphical Models

- Review Graphical Modes
  - AND/OR search spaces, pseudo-trees
    - AND/OR search trees
    - AND/OR search graphs
    - Generating good pseudo-trees
    - Brute-force AND/OR
  - Heuristic search (HS) for AND/OR spaces
    - Basic Heuristic search (Depth and Best)
    - AND/OR Depth-first HS (branch and bound)
    - AND/OR Best-first heuristic search
    - The Guiding MBE heuristic
    - Marginal Map (max-sum-product)

- Hybrids of search and Inference
- Summary and Class 2

Context minimal AND/OR search graph
18 AND nodes
Outline: Search for Graphical Models

• Review Graphical Modes

• AND/OR search spaces, pseudo-trees
  – AND/OR search trees
  – AND/OR search graphs
  – Generating good pseudo-trees
  – Brute-force AND/OR

• Heuristic search (HS) for AND/OR spaces
  – Basic Heuristic search (Depth and Best)
  – AND/OR Depth-first HS (branch and bound)
  – AND/OR Best-first heuristic search
  – The Guiding MBE heuristic
  – Marginal Map (max-sum-product)

• Hybrids of search and Inference
• Summary and Class 2
Conditioning - the Probability Tree

\[ P(D = 1, G = 0) = \sum_a P(a) \sum_c P(c|a) \sum_b P(b|a) \sum_f P(f|b,c) P(d = 1|b,a) P(g = 0|f) \]

**Complexity of conditioning:** exponential time, linear space
The Classic OR Search Space

Ordering: A B E C D F
AND/OR Search Space

Primal graph

DFS tree

Primal graph DFS tree
AND/OR vs. OR

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

- **Size of tree** $O(nk^h)$
- **Can be traversed in**
  - **Time** $O(nk^h)$, **Space** $O(n)$
- All solution trees = all configurations
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)
Arc weights
Cost of a solution tree
The value function
Arc Weights for AND/OR Trees

OR to AND arc weight \( <X,x> \) is the product of factors that all their arguments are just assigned at AND node \( X=x \) but not before.
**Cost of a Solution Tree**

A solution tree includes the root and has a single child for any OR node, and all children of any of its AND nodes.

**Cost of the solution tree**: the product of weights on its arcs

Cost of \((A=0, B=1, C=1, D=1, E=0)\) = \(0.6 \cdot 0.6 \cdot 0.5 \cdot 0.8 \cdot 0.5 = 0.0720\)
Arc Weights for AND/OR Trees

A Bayesian Network

OR to AND arc weight \((X,x)\) is the product of factors \(f\) that are instantiated at the AND node \(X=x\) but not before.
The Value Function (Probability of Evidence)

\[ P(E \mid A, B) \]

\[
\begin{array}{c|cc}
A & B & E=0 \quad E=1 \\
0 & 0 & .4 \quad .6 \\
0 & 1 & .5 \quad .5 \\
1 & 0 & .7 \quad .3 \\
1 & 1 & .2 \quad .8 \\
\end{array}
\]

Evidence: E=0

\[ P(B \mid A) \]

\[
\begin{array}{c|cc}
A & B=0 \quad B=1 \\
0 & .4 \quad .6 \\
1 & .7 \quad .3 \\
\end{array}
\]

\[ P(C \mid A) \]

\[
\begin{array}{c|cc}
A & C=0 \quad C=1 \\
0 & .2 \quad .8 \\
1 & .6 \quad .4 \\
\end{array}
\]

\[ P(A) \]

\[
\begin{array}{c|cc}
A & P(A) \\
0 & .6 \\
1 & .4 \\
\end{array}
\]

\[ P(D=1, E=0) = ? \]

\[ P(D \mid B, C) \]

\[
\begin{array}{c|cc}
B & C & D=0 \quad D=1 \\
0 & 0 & .2 \quad .8 \\
0 & 1 & .1 \quad .9 \\
1 & 0 & .3 \quad .7 \\
1 & 1 & .5 \quad .5 \\
\end{array}
\]

Evidence: D=1

Value of node = updated belief for sub-problem below

**AND node:** product

\[
\prod_{n' \in \text{children}(n)} v(n')
\]

**OR node:** Marginalization by summation

\[
\sum_{n \in \text{children}(n)} w(n, n') v(n')
\]

slides6 W2020
The Value Function

\[ P(E \mid A, B) \] \hspace{1cm} \[ P(B \mid A) \] \hspace{1cm} \[ P(C \mid A) \] \hspace{1cm} \[ P(A) \]

<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>

**Evidence:** E=0

\[ P(D \mid B, C) \]

<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>

**Evidence:** D=1, E=0

- \( V(n) \) is dictated by the query of interest
- \( V(n) \) the value of the sub-problem represented by \( T(n) \)
- For sum-inference it is the probability mess below \( n \)
- Can be computed recursively based on child values.
The Value Function for Optimization

Objective function:  

\[ F^* = \min_x \sum_{\alpha} f_\alpha(x_\alpha) \]

Can you find the Errors on the weights?

Node Value (bottom-up evaluation)

OR – minimization
AND – summation

slides6 W2020
The Value Function for Optimization

Objective function: \( F^* = \min_x \sum_\alpha f_\alpha(x_\alpha) \)

**AND node** = Combination operator (summation)

**OR node** = Marginalization operator (minimization)
The AND/OR Counting Value (#CSP)

AND node: Combination operator (product)

OR node: Marginalization operator (summation)

Value of node = number of solutions below it
Summary: AND/OR Search Tree for GMs

• The AND/OR search tree of R relative to a pseudo-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
  – Arc-weight are assigned from the model factors

• A solution is a consistent subtree. Its cost, the product of the weights.
• Query: compute the value of the root node
## Size and Traversal of AND/OR Search Tree

<table>
<thead>
<tr>
<th>AND/OR tree</th>
<th>OR tree</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Space</strong></td>
<td>O(n)</td>
</tr>
<tr>
<td><strong>Size=</strong></td>
<td>O(n (k^h))</td>
</tr>
<tr>
<td><strong>Time</strong></td>
<td>(Freuder &amp; Quinn85), (Collin, Dechter &amp; Katz91), (Bayardo &amp; Miranker95), (Darwiche01)</td>
</tr>
</tbody>
</table>

\[ h \leq w^* \log n \]

- \(k\) = domain size
- \(h\) = height of pseudo-tree
- \(n\) = number of variables
- \(w^*\) = treewidth
## AND/OR vs. OR Spaces

<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>3.15</td>
<td>2,097,150</td>
</tr>
<tr>
<td>4</td>
<td>9</td>
<td>3.13</td>
<td>2,097,150</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>3.12</td>
<td>2,097,150</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>3.12</td>
<td>2,097,150</td>
</tr>
<tr>
<td>5</td>
<td>13</td>
<td>3.11</td>
<td>2,097,150</td>
</tr>
</tbody>
</table>

Random graphs with 20 nodes, 20 edges and 2 values per node
Pseudo-trees
Given undirected graph \( G = (V, E) \), a directed rooted tree \( T = (V, E') \) defined on all its nodes is a pseudo tree if any arc of \( G \) which is not included in \( E' \) is a back-arc in \( T \), namely it connects a node in \( T \) to an ancestor in \( T \). The arcs in \( E' \) may not all be included in \( E \).

Given a pseudo tree \( T \) of \( G \), the extended graph of \( G \) relative to \( T \) includes also the arcs in \( E' \) that are not in \( E \): as \( GT = (V, E \cup E') \).
A pseudo-tree of a graph is a tree spanning its nodes, where all arcs in the graph not in the tree are back-arcs.

\[ h \leq w^* \log n \]
From DFS-Trees to Pseudo-Trees
Finding Min-height Pseudo-Trees

• Finding a min height pseudo-tree is NP-complete, but:
  • Given a tree-decomposition with treewidth \( w^* \), there exists a pseudo-tree whose height satisfies
    – \( h \leq w^* \log n \)
  • Optimality of \( h \) and \( w^* \) cannot be achieved at once.

\[
\begin{aligned}
W^* &= 1 \\
h &= 7 \\
\end{aligned}
\quad \rightarrow \quad 
\begin{aligned}
w^* &= 2 \\
h &= 3 \\
\end{aligned}
\]
AND/OR Search-Tree Properties

\( (k = \text{domain size, } h = \text{pseudo-tree height. } n = \text{number of variables}) \)

• **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 \( O(n k^h) \)
  
  Size of OR search tree is \( O(k^n) \)

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

• When the pseudo-tree is a chain we get an OR space
Summary: Queries 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.
  – Max-Inference: v(n) is the optimal solution in T(n)
  – Sum-Inference: v(n) is probability of evidence in T(n).
  – Mixed-Inference: v(n) is the marginal map in T(n).

• Goal: compute the value of the root node recursively traversing the AND/OR tree.

Complexity of searching depth-first is
  – Space: $O(n)$
  – Time: $O(nk^h)$
  – Time: $O(k^{w*logn})$
Outline: Search

- Review Graphical Modes
  - AND/OR search spaces, pseudo-trees
    - AND/OR search trees
    - AND/OR search graphs
    - Generating good pseudo-trees
    - Brute-force AND/OR
  - Heuristic search (HS) for AND/OR spaces
    - Basic Heuristic search (Depth and Best)
    - AND/OR Depth-first HS (branch and bound)
    - AND/OR Best-first heuristic search
    - The Guiding MBE heuristic
    - Marginal Map (max-sum-product)
- Hybrids of search and Inference
- Summary and Class 2
From Search Trees to Search Graphs

- Any two nodes that root identical subtrees or subgraphs can be merged
From Search Trees to Search Graphs

- Any two nodes that root identical subtrees or subgraphs can be merged
Merging Based on Context

- context \( (X) \) = ancestors of \( X \) in pseudo tree, connected to \( X \), or to descendants of \( X \)
- context \( (X) \) = parents in the induced graph
- \( \max |\text{context}| \) = induced width = treewidth

pseudo tree

context(●) = [● ● ●]
Definition 7.2.13 (context minimal AND/OR search graph) The AND/OR search graph of $M$ guided by a pseudo-tree $T$ that is closed under context-based merge operator, is called the context minimal AND/OR search graph and is denoted by $C_T(R)$. 
AND/OR Tree DFS Algorithm (Value=Sum-Product)

\[
\begin{align*}
P(E | A, B) & \quad P(B | A) & \quad P(C | A) & \quad P(A) \\
\begin{array}{c|cc}
A & B & E=0 \quad E=1 \\
\hline
0 & 0 & .4 \quad .6 \\
0 & 1 & .5 \quad .5 \\
1 & 0 & .7 \quad .3 \\
1 & 1 & .2 \quad .8 \\
\end{array} & \\
\begin{array}{c|cc}
A & B=0 \quad B=1 \\
\hline
0 & .4 \quad .6 \\
1 & .7 \quad .3 \\
\end{array} & \quad \begin{array}{c|cc}
A & C=0 \quad C=1 \\
\hline
0 & .2 \quad .8 \\
1 & .6 \\
\end{array} & \quad \begin{array}{c|c}
A & P(A) \\
\hline
0 & .6 \\
1 & .4 \\
\end{array}
\end{align*}
\]

Evidence: E=0

```
0 0 .4 .6 
0 1 .5 .5 
1 0 .7 .3 
1 1 .2 .8 
```

Result: \( P(D=1, E=0) \)

```
P(D | B, C)
\begin{array}{c|cc}
B & C & D=0 \quad D=1 \\
\hline
0 & 0 & .2 \quad .8 \\
0 & 1 & .1 \quad .9 \\
1 & 0 & .3 \quad .7 \\
1 & 1 & .5 \quad .5 \\
\end{array}
```

Evidence: D=1, E=0

\( P(D | B, C) \)

\( P(E | A, B) \)

\( P(B | A) \)

\( P(C | A) \)

\( P(A) \)

\( P(D | B, C) \)
AND/OR Search Graph (Value=Sum-Product)

Result: $P(D=1,E=0)$

Cache table for D

Evidence: D=1, E=0
AND/OR Search Graph (Optimization)

Context minimal AND/OR search graph

Objective function: \[ F^* = \min_x \sum_{\alpha} f_\alpha(x_\alpha) \]
Merging Based on Context

- context (X) = ancestors of X in pseudo tree, connected to X, or to descendants of X
- context (X) = parents in the induced graph
- max |context| = induced width = treewidth
How Big Is The Context?

• **Theorem:** The maximum context-size of a pseudo-tree equals the **treewidth** along the pseudo tree.

\[
\text{max context size} = \text{treewidth}
\]

(C K H A B E J L N O D P M F G)

slides6 W2020
**Treewidth vs. Pathwidth**

- Treewidth: $3 = (\text{max cluster size}) - 1$
- Pathwidth: $4 = (\text{max cluster size}) - 1$

*slides6 W2020*
All Four Search Spaces

- **Full OR search tree**
  - 126 nodes

- **Full AND/OR search tree**
  - 54 AND nodes

- **Context minimal OR search graph**
  - 28 nodes

- **Context minimal AND/OR search graph**
  - 18 AND nodes

Any query is best computed over the context-minimal AND/OR space

- \( k \) = domain size
- \( n \) = number of variables
- \( w^* \) = treewidth
- \( \text{pw}^* \) = pathwidth
All Four Search Spaces

<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^{pw*}))</td>
</tr>
<tr>
<td><strong>Time size</strong></td>
<td>O(n (k^{w*+1}))</td>
<td>O(n (k^{pw*+1}))</td>
</tr>
</tbody>
</table>

Computes any query:
- Constraint satisfaction
- Max-Inference: Optimization
- Sum-Inference: Weighted counting
- Mixed-Inference: Marginal Map,
- Maximum expected utility

**K** = domain size  
**n** = number of variables  
**w** = treewidth  
**pw** = pathwidth

Any query is best computed over the context-minimal AND/OR space
AND/OR Search and Variable Elimination

\[(C\ K\ H\ A\ B\ E\ J\ L\ N\ O\ D\ P\ M\ F\ G)\]
AND/OR Search and Variable Elimination

Variable Elimination

(C K H A B E J L N O D P M F G)
AND/OR Search and Variable Elimination

(C K H A B E J L N O D P M F G)
Road Map: Search

- Review Graphical Modes
  - AND/OR search spaces, pseudo-trees
    - AND/OR search trees
    - AND/OR search graphs
    - Generating good pseudo-trees
    - Brute-force AND/OR

- Heuristic search for AND/OR spaces
  - Basic Heuristic search (Depth and Best)
  - Depth-first AND/OR branch and bound
  - Best-first AND/OR search
  - The Guiding MBE heuristic
  - Marginal Map (max-sum-product)

- Hybrids of search and Inference
- Summary and Class 2
Finding Good Pseudo-Trees
Finding Min-Height Pseudo-Trees

- Finding min height 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 \( h \leq w^* \log n \),
Constructing Pseudo-Trees

- **Min-Fill** [Kjaerulff, 1990]
  - Depth-first traversal of the induced graph obtained along the min-fill elimination order
  - Variables ordered according to the smallest “fill-set”

- **Hypergraph Partitioning** [Karypis and Kumar, 2000]
  - Functions are vertices in the hypergraph and variables are hyperedges
  - Recursive decomposition of the hypergraph while minimizing the separator size at each step
  - Using state-of-the-art software package hMeTiS
Variable Orderings and Pseudo-Trees

Bucket-tree = pseudo-tree

\[ d: \text{A B C E D F} \]

Bucket-tree based on \( d \)

Induced graph

Bucket-tree

**Finding small height or small width pseudo-trees is NP-hard**

So, which orderings would give good pseudo-trees?

Bucket-tree used as pseudo-tree

AND/OR search tree
# Quality of Pseudo-Trees

<table>
<thead>
<tr>
<th>Network</th>
<th>hypergraph</th>
<th>min-fill</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>width</td>
<td>depth</td>
</tr>
<tr>
<td>barley</td>
<td>7</td>
<td>13</td>
</tr>
<tr>
<td>diabetes</td>
<td>7</td>
<td>16</td>
</tr>
<tr>
<td>link</td>
<td>21</td>
<td>40</td>
</tr>
<tr>
<td>mildew</td>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td>munin1</td>
<td>12</td>
<td>17</td>
</tr>
<tr>
<td>munin2</td>
<td>9</td>
<td>16</td>
</tr>
<tr>
<td>munin3</td>
<td>9</td>
<td>15</td>
</tr>
<tr>
<td>munin4</td>
<td>9</td>
<td>18</td>
</tr>
<tr>
<td>water</td>
<td>11</td>
<td>16</td>
</tr>
<tr>
<td>pigs</td>
<td>11</td>
<td>20</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Network</th>
<th>hypergraph</th>
<th>min-fill</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>width</td>
<td>depth</td>
</tr>
<tr>
<td>spot5</td>
<td>47</td>
<td>152</td>
</tr>
<tr>
<td>spot28</td>
<td>108</td>
<td>138</td>
</tr>
<tr>
<td>spot29</td>
<td>16</td>
<td>23</td>
</tr>
<tr>
<td>spot42</td>
<td>36</td>
<td>48</td>
</tr>
<tr>
<td>spot54</td>
<td>12</td>
<td>16</td>
</tr>
<tr>
<td>spot404</td>
<td>19</td>
<td>26</td>
</tr>
<tr>
<td>spot408</td>
<td>47</td>
<td>52</td>
</tr>
<tr>
<td>spot503</td>
<td>11</td>
<td>20</td>
</tr>
<tr>
<td>spot505</td>
<td>29</td>
<td>42</td>
</tr>
<tr>
<td>spot507</td>
<td>70</td>
<td>122</td>
</tr>
</tbody>
</table>

Bayesian Networks Repository

For more see [Dechter 2013]

SPOT5 Benchmarks

slides6 W2020
Finding Min-height Pseudo-Trees

- Finding a min height pseudo-tree is NP-complete, but:
- Given a tree-decomposition with treewidth $w^*$, there exists a pseudo-tree whose height satisfies
  - $h \leq w^* \log n$
- Optimality of $h$ and $w^*$ cannot be achieved at once.

Which pseudo-tree if no caching?
Which with caching?

$W^*=1$
$h=7$  

$w^*=2$
$h=3$
The Impact of the Pseudo-Tree

- Choose pseudo-tree with a minimal search graph
- But determinism and pruning for optimization is unpredictable
Road Map: Search

• Review Graphical Modes

• AND/OR search spaces, pseudo-trees
  – AND/OR search trees
  – AND/OR search graphs
  – Generating good pseudo-trees
  – **Brute-force AND/OR**

• Heuristic search (HS) for AND/OR spaces
  – Basic Heuristic search (Depth and Best)
  – AND/OR Depth-first HS (branch and bound)
  – AND/OR Best-first heuristic search
  – The Guiding MBE heuristic
  – Marginal Map (max-sum-product)

• Hybrids of search and Inference
• Summary and Class 2
### Types of queries

<table>
<thead>
<tr>
<th>Query Type</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max-Inference</td>
<td>$f(x^*) = \max_x \prod_{\alpha} f_\alpha(x_\alpha)$</td>
</tr>
<tr>
<td>Sum-Inference</td>
<td>$Z = \sum_x \prod_{\alpha} f_\alpha(x_\alpha)$</td>
</tr>
<tr>
<td>Mixed-Inference</td>
<td>$f(x^*<em>M) = \max</em>{x_M} \sum_{x_S} \prod_{\alpha} f_\alpha(x_\alpha)$</td>
</tr>
</tbody>
</table>

- All solved by AND/OR Depth-first search,
  - Linear memory, $\exp(h)$ time or
  - $\exp(w^*)$ memory and time
- But, we can do better by:
  - Pruning while searching
  - Generating upper and lower bounds anytime
AND/OR Tree DFS Algorithm (Belief Updating)

- **AND node**: Combination operator (product)
- **OR node**: Marginalization operator (summation)

Value of node = updated belief for sub-problem below

Evidence: D=1

Evidence: E=0

Searching the AND/OR tree
Dfs is straightforward

\[
P(A) = \begin{array}{c|c|c}
A & B & C \\
0 & 0.2 & 0.6 \\
1 & .5 & .9 \\
\end{array}
\]

\[
P(B | A) = \begin{array}{c|c|c}
A & B=0 & B=1 \\
0 & .4 & 0.6 \\
1 & .7 & .3 \\
\end{array}
\]

\[
P(C | A) = \begin{array}{c|c|c}
A & C=0 & C=1 \\
0 & .2 & 0.8 \\
1 & 0.7 & .3 \\
\end{array}
\]

\[
P(E | A, B) = \begin{array}{c|c|c|c|c|c|c}
A & B & E=0 & E=1 \\
0 & 0 & 0.4 & 0.6 \\
0 & 1 & 0.5 & 0.5 \\
1 & 0 & 0.7 & 0.3 \\
1 & 1 & 0.2 & 0.8 \\
\end{array}
\]

\[
P(D | B, C) = \begin{array}{c|c|c|c|c|c|c}
B & C & D=0 & D=1 \\
0 & 0 & 0.2 & 0.8 \\
0 & 1 & 0.1 & 0.9 \\
1 & 0 & 0.3 & 0.7 \\
1 & 1 & 0.5 & 0.5 \\
\end{array}
\]

Result: \(P(D=1, E=0) = 0.24408\)
AND/OR Graph DFS Algorithm (Belief Updating)

\[
P(E | A, B) \quad P(B | A) \quad P(C | A) \quad P(A)
\]

<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>

Evidence: E=0

\[
\text{Evidence: } D=1
\]

\[
P(AB | D) = BCD = 0 \quad D = 1
\]

\[
P(AB | C) = 0.352
\]

Cache table for D

\[
P(D | B, C)
\]

\[
\begin{array}{c|cc}
B & C & D=0 & D=1 \\
---&---&---&---
0 & 0 & .2 & .8 \\
0 & 1 & .1 & .9 \\
1 & 0 & .3 & .7 \\
1 & 1 & .5 & .5 \\
\end{array}
\]

Evidence: D=1

Context: [AB]

Searching the AND/OR graph should avoid dead caches, less simple
AND/OR Search Graph (Optimization)

Objective function: $F^* = \min_x \sum_{\alpha} f_\alpha(x_\alpha)$

Context minimal AND/OR search graph
**Dead Caches**

**Definition 8.1.9 (dead cache)** If $X$ is the parent of $Y$ in pseudo-tree $T$, and $\text{context}(X) \subseteq \text{context}(Y)$, then $\text{context}(Y)$ represents a dead cache.

(Darwiche 2000)
Dead Caches

Definition 8.1.9 (dead cache) If $X$ is the parent of $Y$ in pseudo-tree $T$, and $\text{context}(X) \subset \text{context}(Y)$, then $\text{context}(Y)$ represents a dead cache.
Searching AND/OR Graphs

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

\[
\begin{align*}
\text{Space: } & O(n) & \text{Space: } & O(\exp w^*) \\
\text{Time: } & O(\exp(w^* \log n)) & \text{Time: } & O(\exp w^*) \\
\end{align*}
\]

\[
\text{Space: } O(\exp(i)) & \quad \text{Time: } O(\exp(m_i+i))
\]

\( m_i \) is related to the size of the i-cutset.
Different Levels of Caching

Figure 7.11: AOC(2) graph (adaptive caching).

Figure 7.12: AOCutset(2) graph (AND/OR Cutset).
Search for Mixed Deterministic and Probabilistic Graphical Models
AND/OR Search for Mixed Networks

Definition 8.2.1 (backtrack-free AND/OR search tree) Given graphical model $M$ and given an AND/OR search tree $S_T(M)$, the backtrack-free AND/OR search tree of $M$ based on $T$, denoted $BF_T(M)$, is obtained by pruning from $S_T(M)$ all inconsistent subtrees, namely all nodes that root no consistent partial solution.

- Graph-based No-good and good learning are automatically performed by AND/OR (backjumping) and by caching.
AND/OR Backtrack-Free

(a) A constraint tree  
(b) Search tree  
(c) Backtrack-free search tree

Figure 8.1: AND/OR search tree and backtrack-free tree
Figure 8.2: Mixed network defined by the query \( \varphi = (A \lor C) \land (B \lor \neg E) \land (B \lor D) \)

**Example 8.2.6** We refer back to the example in Figure 7.4. Consider a constraint network that is defined by the CNF formula \( \varphi = (A \lor C) \land (B \lor \neg E) \land (B \lor D) \). The trace of algorithm AND-OR-cpe without caching is given in Figure 8.2. Notice that the clause \( A \lor C \) is not satisfied if \( A = 0 \) and \( C = 0 \), therefore the paths that contain this assignment cannot be part of a solution of the mixed network. The value of each node is shown to its left (the leaf nodes assume a dummy value of 1, not shown in the figure). The value of the root node is the probability of \( \varphi \). Figure 8.2 is similar to Figure 7.4. In Figure 7.4 the evidence can be modeled as the CNF formula with unit clauses \( D \land \neg E \). \( \square \)
AND/OR CPE (Constraint Probability Evaluation)

Figure 8.2: Mixed network defined by the query $\varphi = (A \lor C) \land (B \lor \neg E) \land (B \lor D)$

$P(D=0,E=1)$
The Effect of Constraint Propagation in AND/OR

Domains are \{1,2,3,4\}

CONSTRAINTS ONLY

FORWARD CHECKING

MAINTAINING ARC CONSISTENCY
Outline: Search

- Review Graphical Modes

- AND/OR search spaces, pseudo-trees
  - AND/OR search trees
  - AND/OR search graphs
  - Generating good pseudo-trees
  - Brute-force search

- Heuristic search (HS) for AND/OR spaces
  - Basic Heuristic search (Depth and Best)
  - AND/OR Depth-first HS (branch and bound)
  - AND/OR Best-first heuristic search
  - The Guiding MBE heuristic
  - Marginal Map (max-sum-product)

- Hybrids of search and Inference
- Summary and Class 2
Basic Heuristic Search

Heuristic function $\tilde{f}(\hat{x}_p)$ computes a lower bound on the best extension of partial configuration $\hat{x}_p$ and can be used to guide heuristic search. We focus on:

1. Branch-and-Bound
   - Use heuristic function $\tilde{f}(\hat{x}_p)$ to prune the depth-first search tree
   - Linear space

2. Best-First Search
   - Always expand the node with the lowest heuristic value $\tilde{f}(\hat{x}_p)$
   - Needs lots of memory

We assume min-sum problems in the following slides.
Basic Heuristic Search; **Best-First**

Task: compute \(v\) (root): MPE, MAP(MMAP)
Each node is a sub-problem
(defined by current conditioning)

- **Best-First Algorithms, \((A^*)\)**
  - Expand nodes in OPEN list in order of \(\min f(n)\)
  - Terminates with first full solution (for mpe)

- **Properties**
  - Optimal, if \(h(n) \leq v(n)\)
  - Expands least set of nodes
  - Exponential memory
  - **Not anytime solution for MPE**
  - But, yields lower bounds on value, anytime

\[
f(n) = g(n) + h(n) \leq g(n) + v(n) = f^*(n)\]
f(n) is a lower bound on best cost through n
Basic Heuristic Search; Depth-First

- **Depth-First (B&B for MAP)**
  - Expand in dfs order
  - Update UB with each solution
  - Prunes if $f(n) \geq UB$

- **Properties**
  - Can use only linear memory
  - Yields upper bounds anytime

---

Prunes if $f(n) \geq UB$

(UB) Upper Bound = best solution so far
Partial Solution Tree for AND/OR

Pseudo tree

Extension(T’) – solution trees that extend T’
g(T’) = conditioned value of a node
V(T’) = the combined value below T’
f*(T’) = conditioned value through T’
Exact Evaluation Function

Conditioned value of a node

\[ f^*(T') = w(A,0) + w(B,1) + w(C,0) + w(D,0) + v(D,0) + v(F) \]
Heuristic Evaluation Function

\[ f(T') = w(A,0) + w(B,1) + w(C,0) + w(D,0) + h(D,0) + h(F) = 12 \leq f^*(T') \]
Depth-First AND/OR Branch-and-Bound

- Associate each node $n$ with a heuristic lower bound $h(n)$ on $v(n)$

Algorithm AOBB:

- **EXPAND** (top-down)
  - Evaluate $f(T')$ and prune search if $f(T') \geq UB$
  - If not in cache, generate successors of the tip node $n$

- **PROPAGATE** (bottom-up)
  - Update value of the parent $p$ of $n$
    - OR nodes: minimization
    - AND nodes: summation
  - Cache value of $n$ based on context only if fully explored
Outline: Search

- Review Graphical Modes
  - AND/OR search spaces, pseudo-trees
    - AND/OR search trees
    - AND/OR search graphs
    - Generating good pseudo-trees
    - Brute-force search

- Heuristic search (HS) for AND/OR spaces
  - Basic Heuristic search (Depth and Best)
  - AND/OR Depth-first HS (branch and bound)
  - AND/OR Best-first heuristic search
  - The Guiding MBE heuristic
  - Marginal Map (max-sum-product)

- Hybrids of search and Inference
- Summary and Class 2
Conditioning versus Elimination

The main target in conditioning is to trade memory for time. It is not really possible to do better than elimination.

k “sparser” problems

1 “denser” problem
A Cycle-Cutset

Cycle cutset = \{A,B,C\}

1-cutset = \{A,B,C\}, size 3
Loop-Cutset, q-Cutset, Cycle-Cutset

- A loop-cutset is a subset of nodes of a directed graph that when removed the remaining graph is a poly-tree.
- A q-cutset is a subset of nodes of an undirected graph that when removed the remaining graph has an induced-width of q or less.
- A cycle-cutset is a q-cutset such that q=1.
Definition 7.3  *q-cutset, minimal.* Given a graph $G$, a subset of nodes is called a *$q$-cutset* for an integer $q$ iff when removed, the resulting graph has an induced-width less than or equal to $q$. A minimal *$q$-cutset* of a graph has a smallest size among all *$q$-cutsets* of the graph. A cycle-cutset is a 1-cutset of a graph.

Finding a minimal *$q$-cutset* is clearly a hard task [A. Becker and Geiger, 1999; Bar-Yehuda et al., 1998; Becker et al., 2000; Bidyuk and Dechter, 2004]. However, like in the special case of a cycle-cutset we can settle for a non-minimal *$q$-cutset* relative to a given variable ordering. Namely,

Example 7.4  Consider as another example the constraint graph of a graph coloring problem given in Figure 7.3a. The search space over a 2-cutset, and the induced-graph of the conditioned instances are depicted in 7.3b.
Example: 2-cutset conditioning (VEC(2))

- Inference may require too much memory
- **Condition** on some of the variables

Graph Coloring problem
VEC(q) : Variable Elimination with Conditioning;

- VEC for Probability of evidence:
- Identify a q-cutset, $C$, of size $|C|$ of the network
- For each assignment to $C=c$ solve by CTE or BE the conditioned sub-problem.
- Accumulate probability.
- Time complexity: $nk^{|c|+q+1}$
- Space complexity: $nk^q$
Time vs Space for w-cutset

- Random Graphs (50 nodes, 200 edges, average degree 8, $w^* \approx 23$)

$W$-cutset time $O(\exp(w+\text{cutset-size}))$

Space $O(\exp(w))$

(Dechter and El-Fatah, 2000)
(Larrosa and Dechter, 2001)
(Rish and Dechter 2000)
• Inference may require too much memory
• **Condition** on some of the variables
AND/OR w-cutset

graphical model

pseudo tree

1-cutset tree
AND/OR w-cutset

3-cutset

2-cutset

1-cutset
Summary: AND/OR Cutset-Conditioning

• Trade memory for time.

• We never improve time: cycle-cutset size is larger of equal to treewidth+1

• Sometime we do not worsen the time and memory can be much better (e.g., when the induced-width is high)
Software

- **aolib**
  - [http://graphmod.ics.uci.edu/group/Software](http://graphmod.ics.uci.edu/group/Software)
  (standalone AOBB, AOBF solvers)

- **daoopt**
  - [https://github.com/lotten/daoopt](https://github.com/lotten/daoopt)
  (distributed and standalone AOBB solver)

- **merlin**
  (standalone WMB, AOBB, AOBF, RBFAOO solvers)
  open source, BSD license
UAI Probabilistic Inference Competitions

- **2006**
  - (aolib)

- **2008**
  - (aolib)

- **2012**
  - (daoopt)

- **2014**
  - (daoopt)
  - (daoopt)
  - (merlin)

Marginal Map
Summary of Search

• AND/OR search spaces, pseudo-trees
  – AND/OR search trees
  – AND/OR search graphs
  – Generating good pseudo-trees
  – Brute-force search

• Heuristic search for AND/OR spaces
  – Depth-first AND/OR branch and bound
  – Best-first AND/OR search
  – The Guiding MBE heuristic
  – Marginal Map (max-sum-product)

• Hybrids of search and Inference
• Summary and Class 2