Exploiting Tree-Decomposition in Search: The AND/OR Paradigm

Rina Dechter
Bren School of ICS
University of California, Irvine

Collaborators:
Kalev Kask
Radu Marinescu
Robert Mateescu
Vibhav Gogate
Outline

- Background: Search and inference in Graphical models
- AND/OR search trees
  - Pseudo spanning trees, dfs trees
- AND/OR search graphs
  - Tree-width and path-width bounds
- Relationship to algorithmic and compilation schemes.

Vienna, Dec 2004
A constraint network is a triple
\[ R = \langle X, D, C \rangle \]
where:
- \( X = \{X_1, \ldots, X_n\} \) is a set of variables
- \( D = \{D_1, \ldots, D_n\} \) is the set of their domains
- \( C = \{C_1, \ldots, C_t\} \), \( C_i = (S_i, R_i) \) are the constraints, 
  \( S_i \) being the scope of the relation \( R_i \).

The main task:
- Determine if the problem has a solution (an assignment that satisfies all the relations);
- If yes, find one or all of them.
Probabilistic Networks

\[ P(S, C, B, X, D) = P(S) \, P(C|S) \, P(B|S) \, P(X|C,S) \, P(D|C,B) \]

- \( P(S|d) = ? \)
- MPE: \( \text{argmax} \, P(S) \, P(C|S) \, P(B|S) \, P(X|C,S) \, P(D|C,B) \)

Vienna, Dec 2004
Graphical models

A graphical model \((X,D,C)\):
- \(X = \{X_1, \ldots X_n\}\) variables
- \(D = \{D_1, \ldots D_n\}\) domains
- \(C = \{F_1, \ldots, F_t\}\) functions
  (constraints, CPTs, cnfs)

\[F_i := P(F | A, C)\]
\[F_i := F = A + C\]
Graphical models

A graphical model (X,D,C):
- X = \{X_1, \ldots X_n\} variables
- D = \{D_1, \ldots D_n\} domains
- C = \{F_1, \ldots, F_t\} functions
  (constraints, CPTs, cnfs)

Operators: combine and project:
- **MPE**: \( \max_x \prod_j P_j \)
- **CSP**: \( \prod_x \times_j C_j \)
- **Max-CSP**: \( \min_x \sum_j F_j \)
- **Belief updating**: \( \sum_{x,y} \prod_j P_i \)

All these tasks are NP-hard
- \( \rightarrow \) identify special cases
- \( \rightarrow \) approximate

Vienna, Dec 2004
Solution Techniques

Search: Conditioning

Complete
- Dfs search,
- Branch and bound, A*

Incomplete
- Simulated Annealing
- Gradient Descent

Inference: Elimination
Vienna, Dec 2004
Solution Techniques

Search: Conditioning

Complete
- Simulated Annealing
- Gradient Descent

Incomplete
- Time: $\exp(n)$
- Space: linear

Hybrids
- Time: $\exp(w^*)$
- Space: $\exp(w^*)$

Complete
- Adaptive Consistency
- Tree Clustering
- Dynamic Programming
- Resolution

Incomplete
- Local Consistency
- Unit Resolution mini-bucket(i)

Inference: Elimination
Vienna, Dec 2004
Solution Techniques

AND/OR search
Time: $\exp(w^* \log n)$
Space: linear

Search: Conditioning

Hybrids: AND-OR(i)
Space: $\exp(i)$
Time: $\exp(C_i)$

Complete
Adaptive Consistency
Tree Clustering
Dynamic Programming
Resolution

Incomplete
Simulated Annealing
Gradient Descent

Incomplete
Local Consistency
Unit Resolution
mini-bucket(i)

Inference: Elimination
Vienna, Dec 2004
Bucket Elimination
Adaptive Consistency (Dechter & Pearl, 1987)

\[ \text{Bucket}(E) : E \neq D, \; E \neq C, \; E \neq B \]
\[ \text{Bucket}(D) : D \neq A \; \parallel \; R_{DCB} \]
\[ \text{Bucket}(C) : C \neq B \; \parallel \; R_{ACB} \]
\[ \text{Bucket}(B) : B \neq A \; \parallel \; R_{AB} \]
\[ \text{Bucket}(A) : \; R_A \]

\[ \text{Bucket}(A) : A \neq D, \; A \neq B \]
\[ \text{Bucket}(D) : D \neq E \; \parallel \; R_{DB} \]
\[ \text{Bucket}(C) : C \neq B \; \parallel \; R_{BE}, \; R_{CE} \]
\[ \text{Bucket}(B) : B \neq E \; \parallel \; R_{BE}^C, \; R_{BE}^D \]
\[ \text{Bucket}(E) : \; \parallel R_E \]

**Complexity**: \( O(n \exp(w^*(d))) \),

\( w^*(d) \) - induced width along ordering \( d \)

Vienna, Dec 2004
Outline

- Background: Search and inference in Graphical models
- AND/OR search trees
  - Pseudo spanning trees, dfs trees
- AND/OR search graphs
  - Tree-width and path-width bounds
- Relationship to algorithmic and compilation schemes.

Vienna, Dec 2004
OR search space

Ordering: A B E C D F

Vienna, Dec 2004
AND/OR search space

Primal graph

DFS tree

Vienna, Dec 2004
OR vs AND/OR

Vienna, Dec 2004
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)

AND/OR

OR

AND

OR

AND

OR

AND

OR

AND

AND

OR

AND

OR

AND

A

B

E

C

D

F

OR
### 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>

Vienna, Dec 2004
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

Vienna, Dec 2004
From DFS trees to Pseudo-trees

DFS tree
depth = 3

pseudo- tree
depth = 2

Vienna, Dec 2004
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 Miranker, 1996, bodlaender and Gilbert, 91):

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

Bucket-tree used as pseudo-tree
Vienna, Dec 2004
## Pseudo Trees vs. DFS Trees

<table>
<thead>
<tr>
<th>Model (DAG)</th>
<th>$w^*$</th>
<th>Pseudo Tree avg. depth</th>
<th>DFS Tree avg. depth</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(N=50, P=2)$</td>
<td>9.54</td>
<td>16.82</td>
<td>36.03</td>
</tr>
<tr>
<td>$(N=50, P=3)$</td>
<td>16.1</td>
<td>23.34</td>
<td>40.6</td>
</tr>
<tr>
<td>$(N=50, P=4)$</td>
<td>20.91</td>
<td>28.31</td>
<td>43.19</td>
</tr>
<tr>
<td>$(N=100, P=2)$</td>
<td>18.3</td>
<td>27.59</td>
<td>72.36</td>
</tr>
<tr>
<td>$(N=100, P=3)$</td>
<td>30.97</td>
<td>41.12</td>
<td>80.47</td>
</tr>
<tr>
<td>$(N=100, P=4)$</td>
<td>40.27</td>
<td>50.53</td>
<td>86.54</td>
</tr>
</tbody>
</table>

N = number of nodes, P = number of parents. MIN-FILL ordering. 100 instances.

Vienna, Dec 2004
AND/OR search tree for graphical models

- 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

Vienna, Dec 2004
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 \( O(\exp(w \cdot \log n)) \)

- **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

Vienna, Dec 2004
Tasks and value of nodes

- **V(n)** is the value of the tree T(n) for the task:
  - **Consistency:** v(n) is 0 if T(n) inconsistent, 1 otherwise.
  - **Counting:** v(n) is number of solutions in T(n)
  - **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 k^m)
  - **Time:** O(exp(w * log n))
DFS algorithm (#CSP example)

OR node: Marginalization operator (commutation)
AND node: Combination operator (product)

Value of node = number of solutions below it

Vienna, Dec 2004
From Search Trees to Search Graphs

- Any two nodes that root identical subtrees/subgraphs can be **merged**

- Minimal AND/OR search graph: closure under merge of the AND/OR search tree
  - Inconsistent subtrees can be pruned too.
  - Some portions can be collapsed or reduced.

Vienna, Dec 2004
Vienna, Dec 2004
An AND/OR graph

Vienna, Dec 2004
Context based caching

- Caching is possible when context is the same

- context = parent-separator set in induced pseudo-graph
  = current variable + parents connected to subtree below

context(B) = \{A, B\}
context(c) = \{A, B, C\}
context(D) = \{D\}
context(F) = \{F\}
Induced-width of pseudo-trees

The induced-width of a pseudo-tree is its induced-width along a dfs order that includes pseudo arcs.

- For pseudo-chains induced-width is path-width (yielding path-decomposition)
- Contexts are bounded by the induced-width of the pseudo tree.
- Min induced-pseudo-width = tree-width

DFS order of both pseudo trees: \(d = A \circ B \circ C \circ E \circ D \circ F\)

Vienna, Dec 2004
Induced-width of pseudo-trees

The **induced-width of a pseudo-tree** is its induced-width along a dfs order that includes pseudo arcs.

- For pseudo-chains induced-width is path-width (yielding path-decomposition)
- Contexts are bounded by the induced-width of the pseudo tree.
- \( w = 2 \)

DFS order of both pseudo trees:
\( d = A \leq B \leq C \leq E \leq D \leq F \)

Vienna, Dec 2004
Caching

context(D)={D}
context(F)={F}

Vienna, Dec 2004
Caching

context(D)={D}

context(F)={F}

Vienna, Dec 2004
Size of minimal AND/OR context graphs

**Theorem:**

- Minimal AND/OR context graph is bounded exponentially by its pseudo-tree induced-width.

- The tree-width of a pseudo-chain is path-width (pw)

- → Minimal OR search graph is $O(\exp(pw^*))$.
- → Minimal AND/OR graph is $O(\exp(w^*))$

- Always, $w^* \leq pw^*$, but $pw^* \leq w^* \log n$

Vienna, Dec 2004
OR tree vs. OR graph

OR tree

OR graph

Vienna, Dec 2004
AND/OR tree vs. AND/OR graph

Vienna, Dec 2004
All four search spaces

Full OR search tree

Context minimal OR search graph

Full AND/OR search tree

Context minimal AND/OR search graph

Vienna, Dec 2004
All four search spaces

Full OR search tree

Context minimal OR search graph

Full AND/OR search tree

Context minimal AND/OR search graph

Vienna, Dec 2004
**Theorem:** for balanced w-trees

\[ pw = m = \log n \cdot tw \]

1) and-orTree-size = orGraph-size

2) \[
\frac{\text{orGraph-size}}{aorGraph-size} = k = (w-1) \frac{\log n}{1 + \log w} - 2w - 3
\]

Vienna, Dec 2004
Uniqueness of minimal AND/OR graph

- **Theorem**: Given a pseudo-tree, the minimal AND/OR search graph is unique for all graph-models that are consistent with that pseudo tree.

- **Related to compilation schemes**:
  - Minimal OR – related to OBDDs (McMillan)
  - Minimal AND/OR – related to tree-OBDDs (McMillan 94),
  - AND/OR graphs related to d-DNNF (Darwiche et. Al. 2002)
Searching AND/OR Graphs

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

Space: $O(n)$  \hspace{1cm} Space: $O(\exp{w^*})$
Time: $O(\exp(w^* \log n))$  \hspace{1cm} Time: $O(\exp w^*)$

AO(i) time complexity?

Vienna, Dec 2004
AND/OR w-cutset

3-cutset

2-cutset

1-cutset

Vienna, Dec 2004
AND/OR w-cutset

grahpical model  pseudo tree  1-cutset tree

Vienna, Dec 2004
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)

Vienna, Dec 2004
Overview

- Introduction and background for graphical models: inference and search

- Inference:
  - Exact: Tree-clustering, variable elimination,
  - Approximate: mini-bucket, belief propagation

- AND/OR search spaces for graphical models
  - mixed networks

- **Empirical evaluation over mixed networks, counting**

Vienna, Dec 2004
Mixed Networks

Belief Network

Constraint Network

Moral mixed graph

\[
P_{B}(\overline{x} | \overline{x} \in \rho) = \frac{P_{B}(\overline{x})}{P_{B}(\overline{x} \in \rho)}, \text{ if } \overline{x} \in \rho
\]

\[
P_{M}(\overline{x}) = \begin{cases} 
P_{B}(\overline{x} | \overline{x} \in \rho) & \text{if } \overline{x} \in \rho \\
0 & \text{otherwise}
\end{cases}
\]

Vienna, Dec 2004
All domains are \{1,2,3,4\}
Using look-ahead

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

CONSTRANTS ONLY

FORWARD CHECKING

MAINTAINING ARC CONSISTENCY
Experiments – mixed networks

- **Parameters of the mixed networks:**
  - N = number of variables
  - K = number of values per variable

- Belief network - (N,K,R,P):
  - R = number of root nodes
  - P = number of parents

- Measures:
  - Time
  - Number of nodes
  - Number of dead-ends

- Constraint network - (N,K,C,S,t):
  - C = number of constraints
  - S = scope size of the constraints
  - t = tightness (no. of disallowed tuples)

- Algorithms:
  - AO-C  constraint checking only
  - AO-FC  forward checking
  - AO-RFC relational forward checking

---

<table>
<thead>
<tr>
<th></th>
<th>Time</th>
<th>Nodes</th>
<th>Dead-ends</th>
<th>Full space</th>
</tr>
</thead>
<tbody>
<tr>
<td>OR</td>
<td>0.15</td>
<td>44,895</td>
<td>9,095</td>
<td>152,858</td>
</tr>
<tr>
<td>AND/OR</td>
<td>11.81</td>
<td>3,147,577</td>
<td>266,215</td>
<td>67,108,862</td>
</tr>
</tbody>
</table>

Vienna, Dec 2004
# CSP N40, K3, C50, P3, 20 inst, \( w^* = 13 \), depth=20
Time (seconds)

<table>
<thead>
<tr>
<th>tightness</th>
<th>70%</th>
<th>60%</th>
<th>50%</th>
<th>40%</th>
</tr>
</thead>
<tbody>
<tr>
<td># solutions</td>
<td>0</td>
<td>0</td>
<td>46,582</td>
<td>147,898,575</td>
</tr>
<tr>
<td>i</td>
<td>BE</td>
<td>8.889</td>
<td>8.709</td>
<td>8.531</td>
</tr>
<tr>
<td>0</td>
<td>A/O FC</td>
<td>0.110</td>
<td>0.454</td>
<td>3.129</td>
</tr>
<tr>
<td>OR FC</td>
<td>0.113</td>
<td>0.511</td>
<td>14.615</td>
<td>9737.823</td>
</tr>
<tr>
<td>3</td>
<td>A/O FC</td>
<td>0.111</td>
<td>0.453</td>
<td>3.103</td>
</tr>
<tr>
<td>OR FC</td>
<td>0.112</td>
<td>0.509</td>
<td>14.474</td>
<td>9027.365</td>
</tr>
<tr>
<td>6</td>
<td>A/O FC</td>
<td>0.110</td>
<td>0.454</td>
<td>3.006</td>
</tr>
<tr>
<td>OR FC</td>
<td>0.113</td>
<td>0.508</td>
<td>13.842</td>
<td>7293.472</td>
</tr>
<tr>
<td>9</td>
<td>A/O FC</td>
<td>0.114</td>
<td>0.453</td>
<td>2.895</td>
</tr>
<tr>
<td>OR FC</td>
<td>0.111</td>
<td>0.509</td>
<td>12.336</td>
<td>5809.917</td>
</tr>
<tr>
<td>12</td>
<td>A/O FC</td>
<td>0.109</td>
<td>0.458</td>
<td>2.703</td>
</tr>
<tr>
<td>OR FC</td>
<td>0.114</td>
<td>0.496</td>
<td>9.678</td>
<td>2598.778</td>
</tr>
<tr>
<td>13</td>
<td>A/O FC</td>
<td>0.111</td>
<td>0.457</td>
<td>2.605</td>
</tr>
<tr>
<td>OR FC</td>
<td>0.123</td>
<td>0.494</td>
<td>8.703</td>
<td>1170.203</td>
</tr>
</tbody>
</table>

- AND/OR search is orders of magnitudes faster than OR search when problems are loose (consistent)
- AND/OR with full caching equivalent to BE: time and space \( O(\exp w^*) \)

Vienna, Dec 2004
## Mixed networks results (1)

### Table

<table>
<thead>
<tr>
<th>Tightness</th>
<th>AO-C</th>
<th>AO-FC</th>
<th>AO-RFC</th>
<th>AO-C</th>
<th>AO-FC</th>
<th>AO-RFC</th>
<th>AO-C</th>
<th>AO-FC</th>
<th>AO-RFC</th>
<th>Nodes</th>
<th>Dead-ends</th>
<th>#sol</th>
</tr>
</thead>
<tbody>
<tr>
<td>80%</td>
<td>0.671</td>
<td>0.056</td>
<td>0.022</td>
<td>153.073</td>
<td>4.363</td>
<td>1.068</td>
<td>95.197</td>
<td>3.298</td>
<td>96.2</td>
<td>1.0E+05</td>
<td></td>
<td></td>
</tr>
<tr>
<td>60%</td>
<td>0.479</td>
<td>0.055</td>
<td>0.021</td>
<td>75.337</td>
<td>3.213</td>
<td>0.906</td>
<td>57.906</td>
<td>3.168</td>
<td>94.0</td>
<td>7.7E+07</td>
<td></td>
<td></td>
</tr>
<tr>
<td>40%</td>
<td>0.103</td>
<td>0.044</td>
<td>0.016</td>
<td>16.579</td>
<td>2.273</td>
<td>0.683</td>
<td>2.636</td>
<td>1.037</td>
<td>396.0</td>
<td>6.2E+09</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Observations

- Caching helps more on loose problems
- Constraint propagation helps more on tight problems

Vienna, Dec 2004
Conclusion

- AND/OR search spaces are a unifying framework for search or compilation applicable to any graphical models.

- With caching AND/OR is similar to inference (minimal graphs)

- AND/OR time and space bounds are equal to state of the art algorithms

- Empirical results
  - AND/OR search spaces are always more effective than traditional OR spaces
  - AND/OR allows a flexible tradeoff between space and time

- Graphical models should always use AND/OR search with embedded inference.

- Current work: Hybrid of inference and search: Heuristic generation and Branch and Bound