Constraint Processing from the Graphical Model Perspective

Rina Dechter

Information and Computer Science, UC-Irvine
Overview and Road Map

- Graphical models
- Constraint networks Model
- Inference
- Search
- Probabilistic Networks
Road Map

- Graphical models
- Constraint networks Model
- Inference
- Search
- Probabilistic Networks
**Propositional Reasoning**

*Example: party problem*

- If Alex goes, then Becky goes: $A \rightarrow B$
- If Chris goes, then Alex goes: $C \rightarrow A$

*Question:*

Is it possible that Chris goes to the party but Becky does not?

Is the propositional theory

$$\varphi = \{ A \rightarrow B, C \rightarrow A, \neg B, C \}$$

satisfiable?
Sudoku – Constraint Satisfaction

- **Variables**: empty slots
- **Domains** = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}
- **Constraints**:
  - 27 all-different

Each row, column and major block must be alldifferent

“Well posed” if it has unique solution: 27 constraints
**Constraint Networks**

**Example: map coloring**

Variables - countries (A, B, C, etc.)

Values - colors (red, green, blue)

Constraints: \( A \neq B, \ A \neq D, \ D \neq E, \) etc.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>red</td>
<td>green</td>
</tr>
<tr>
<td>red</td>
<td>yellow</td>
</tr>
<tr>
<td>green</td>
<td>red</td>
</tr>
<tr>
<td>green</td>
<td>yellow</td>
</tr>
<tr>
<td>yellow</td>
<td>green</td>
</tr>
<tr>
<td>yellow</td>
<td>red</td>
</tr>
</tbody>
</table>

**Constraint graph**

Diagram showing a constraint graph with nodes A, B, C, D, E, F, G and edges between them.
Applications

- Planning and scheduling
  - Transportation scheduling, factory scheduling
- Configuration and design problems
  - floorplans
- Circuit diagnosis
- Scene labeling
- Spreadsheets
- Temporal reasoning, Timetabling
- Natural language processing
- Puzzles: crosswords, sudoku, cryptarithmetic
Probabilistic Reasoning

**Party example: the weather effect**

- Alex is **likely**-to-go in bad weather
- Chris **rarely**-goes in bad weather
- Becky is indifferent but **unpredictable**

Questions:

- **Given bad weather, which group of individuals is most likely to show up at the party?**
- **What is the probability that Chris goes to the party but Becky does not?**

\[
P(W,A,C,B) = P(B|W) \cdot P(C|W) \cdot P(A|W) \cdot P(W)
\]

\[
P(A,C,B|W=bad) = 0.9 \cdot 0.1 \cdot 0.5
\]

\[
P(W|A=1) = .9
\]

\[
P(C|W=bad) = .1
\]

\[
P(B|W=bad) = .5
\]

\[
\begin{array}{|c|c|c|}
\hline
W & A & P(A|W) \\
\hline
\text{good} & 0 & 0.01 \\
\text{good} & 1 & 0.99 \\
\text{bad} & 0 & 0.1 \\
\text{bad} & 1 & 0.9 \\
\hline
\end{array}
\]
Bayesian Networks: Representation
(Pearl, 1988)

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

**Belief Updating:**

\[
P (\text{lung cancer}=\text{yes} | \text{smoking}=\text{no}, \text{dyspnoea}=\text{yes} ) = ?
\]
The “alarm” network - 37 variables, 509 parameters (instead of $2^{37}$)
Sample Domains

- Web Pages and Link Analysis
- Battlespace Awareness
- Epidemiological Studies
- Citation Networks
- Communication Networks (Cell phone Fraud Detection)
- Intelligence Analysis (Terrorist Networks)
- Financial Transactions (Money Laundering)
- Computational Biology
- Object Recognition and Scene Analysis
- Natural Language Processing (e.g. Information Extraction and Semantic Parsing)
Complexity of Reasoning Tasks

- Constraint satisfaction
- Counting solutions
- Combinatorial optimization
- Belief updating
- Most probable explanation
- Decision-theoretic planning

**Reasoning is computationally hard**

**Complexity is**
**Time and space(memory)**
Tree-solving is easy

**Belief updating**
*(sum-prod)*

**CSP – consistency**
*(projection-join)*

**MPE (max-prod)**

**#CSP (sum-prod)**

*Trees are processed in linear time and memory*
Transforming into a Tree

- By Inference (thinking)
  - Transform into a single, equivalent tree of sub-problems

- By Conditioning (guessing)
  - Transform into many tree-like sub-problems.
Inference and Treewidth

Inference algorithm:
Time: \( \exp(\text{tree-width}) \)
Space: \( \exp(\text{tree-width}) \)

\( \text{treewidth} = 4 - 1 = 3 \)
\( \text{treewidth} = (\text{maximum cluster size}) - 1 \)
Conditioning and Cycle cutset

Cycle cutset = \{A, B, C\}
Search over the Cutset

Graph Coloring problem

• Inference may require too much memory
• Condition (guessing) on some of the variables
Search over the Cutset (cont)

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

Graph Coloring problem
Inference vs. Conditioning

- **By Inference (thinking)**

Exponential in treewidth
Time and memory

- **By Conditioning (guessing)**

Exponential in cycle-cutset
Time-wise, linear memory
Graphical Models Reasoning

**Search: Conditioning**

- Complete
  - Depth-first search
  - Branch and Bound
  - A*

- Incomplete
  - Simulated Annealing
  - Gradient Descent

**Inference: Elimination**

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

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

Hybrids:

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

Time: \(\exp(n)\)
Space: linear
Road Map

- Graphical models
- Constraint networks Model
- Inference
- Search
- Probabilistic Networks
**Constraint Networks**

**Example: map coloring**

*Variables* - countries (A,B,C, etc.)

*Values* - colors (red, green, blue)

*Constraints:* \( A \neq B, \ A \neq D, \ D \neq E, \ etc. \)

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>red</td>
<td>green</td>
</tr>
<tr>
<td>red</td>
<td>yellow</td>
</tr>
<tr>
<td>green</td>
<td>red</td>
</tr>
<tr>
<td>green</td>
<td>yellow</td>
</tr>
<tr>
<td>yellow</td>
<td>green</td>
</tr>
<tr>
<td>yellow</td>
<td>red</td>
</tr>
</tbody>
</table>

Constraint graph
**Constraint Satisfaction Tasks**

**Example: map coloring**

Variables - countries (A, B, C, etc.)
Values - colors (e.g., red, green, yellow)

Constraints: \[ A \neq B, \ A \neq D, \ D \neq E, \ etc. \]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>red</td>
<td>green</td>
<td>red</td>
<td>green</td>
<td>blue</td>
</tr>
<tr>
<td>red</td>
<td>blue</td>
<td>gre</td>
<td>green</td>
<td>blue</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>green</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>red</td>
</tr>
<tr>
<td>red</td>
<td>blue</td>
<td>red</td>
<td>green</td>
<td>red</td>
</tr>
</tbody>
</table>

**Are the constraints consistent?**

**Find a solution, find all solutions**

**Count all solutions**

**Find a good solution**
Information as Constraints

- I have to finish my talk in 30 minutes
- 180 degrees in a triangle
- Memory in our computer is limited
- The four nucleotides that makes up a DNA only combine in a particular sequence
- Sentences in English must obey the rules of syntax
- Susan cannot be married to both John and Bill
- Alexander the Great died in 333 B.C.
A constraint network is: \( R=(X,D,C) \)

- **X variables** \( X = \{X_1, \ldots, X_n\} \)
- **D domain** \( D = \{D_1, \ldots, D_n\}, D_i = \{v_1, \ldots, v_k\} \)
- **C constraints** \( C = \{C_1, \ldots, C_l\}, \ldots, C_i = (S_i, R_i) \)

\( R \) expresses allowed tuples over scopes

- **A solution** is an assignment to all variables that satisfies all constraints (join of all relations).
- **Tasks:** consistency?, one or all solutions, counting, optimization
Crossword puzzle

Variables: $x_1, \ldots, x_{13}$

Domains: letters

Constraints: words from

{HOSES, LASER, SHEET, SNAIL, STEER, ALSO, EARN, HIKE, IRON, SAME, EAT, LET, RUN, SUN, TEN, YES, BE, IT, NO, US}
Figure 2.1: The 4-queens constraint network. The network has four variables, all with domains $D_i = \{1, 2, 3, 4\}$. (a) The labeled chess board. (b) The constraints between variables.

$R_{12} = \{(1,3), (1,4), (2,4), (3,1), (4,1), (4,2)\}$

$R_{13} = \{(1,2), (1,4), (2,1), (2,3), (3,2), (3,4), (4,1), (4,3)\}$

$R_{14} = \{(1,2), (1,3), (2,1), (2,3), (2,4), (3,1), (3,2), (3,4), (4,2), (4,3)\}$

$R_{23} = \{(1,3), (1,4), (2,4), (3,1), (4,1), (4,2)\}$

$R_{24} = \{(1,2), (1,4), (2,1), (2,3), (3,2), (3,4), (4,1), (4,3)\}$

$R_{34} = \{(1,3), (1,4), (2,4), (3,1), (4,1), (4,2)\}$
Figure 2.2: Not all consistent instantiations are part of a solution: (a) A consistent instantiation that is not part of a solution. (b) The placement of the queens corresponding to the solution (2, 4, 1, 3). (c) The placement of the queens corresponding to the solution (3, 1, 4, 2).
Figure 2.3: Constraint graphs of (a) the crossword puzzle and (b) the 4-queens problem.
Figure 1.9: Example of selection, projection, and join operations on relations.

(a) Relation $R$

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>c</td>
</tr>
<tr>
<td>b</td>
<td>b</td>
<td>c</td>
</tr>
<tr>
<td>c</td>
<td>b</td>
<td>c</td>
</tr>
<tr>
<td>c</td>
<td>b</td>
<td>s</td>
</tr>
</tbody>
</table>

(b) Relation $R'$

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>b</td>
<td>c</td>
</tr>
<tr>
<td>c</td>
<td>b</td>
<td>c</td>
</tr>
<tr>
<td>c</td>
<td>n</td>
<td>n</td>
</tr>
</tbody>
</table>

(c) Relation $R''$

<table>
<thead>
<tr>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a</td>
<td>1</td>
</tr>
<tr>
<td>b</td>
<td>c</td>
<td>2</td>
</tr>
<tr>
<td>b</td>
<td>c</td>
<td>3</td>
</tr>
</tbody>
</table>

(a) $\sigma_{x_3=c}(R')$

<table>
<thead>
<tr>
<th>$x_2$</th>
<th>$x_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>b</td>
</tr>
<tr>
<td>c</td>
<td>b</td>
</tr>
<tr>
<td>c</td>
<td>n</td>
</tr>
</tbody>
</table>

(b) $\pi_{(x_2,x_3)}(R')$

<table>
<thead>
<tr>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>b</td>
<td>c</td>
</tr>
<tr>
<td>c</td>
<td>b</td>
<td>c</td>
</tr>
<tr>
<td>c</td>
<td>b</td>
<td>c</td>
</tr>
</tbody>
</table>

(c) $R' \Join R''$
Constraint's representations

- Relation: allowed tuples

- Algebraic expression:

- Propositional formula:

- Semantics: by a relation
A (primal) constraint graph: a node per variable arcs connect constrained variables.
A dual constraint graph: a node per constraint's scope, an arc connects nodes sharing variables = hypergraph
Graph Concepts Reviews:
Hyper Graphs and Dual Graphs

- A hypergraph
- Dual graphs
- A primal graph
Propositional Satisfiability

\[ \varphi = \{ (\neg C), (A \lor B \lor C), (\neg A \lor B \lor E), (\neg B \lor C \lor D) \}. \]
Constraint graphs of 3 instances of the Radio frequency assignment problem in CELAR’s benchmark
Road Map

- Graphical models
- Constraint networks Model
- Inference
  - Variable elimination:
  - Tree-clustering
  - Constraint propagation
- Search
- Probabilistic Networks
Bucket Elimination

Adaptive Consistency (Dechter & Pearl, 1987)

\[
\begin{align*}
\text{Bucket E:} & \quad E \neq D, \ E \neq C \\
\text{Bucket D:} & \quad D \neq A \\
\text{Bucket C:} & \quad C \neq B \\
\text{Bucket B:} & \quad B \neq A \\
\text{Bucket A:} & \quad \text{contradiction}
\end{align*}
\]

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

\( w^* - \text{induced width} \)
The Idea of Elimination

\[ R_{DBC} = \prod_{DBC} R_{ED} \bowtie R_{EB} \bowtie R_{EC} \]

Eliminate variable E \(\Leftrightarrow\) join and project
Bucket Elimination
Adaptive Consistency (Dechter & Pearl, 1987)

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

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

Complexity : \( O(n \exp(w^*(d))) \), 
\( w^*(d) \) - induced width along ordering \( d \)
The induced-width

- **Width along** $d$, $w(d)$:
  - max # of previous parents

- **Induced width** $w^*(d)$:
  - The width in the ordered *induced graph*

- **Induced-width** $w^*$:
  - Smallest induced-width over all orderings

- **Finding** $w^*$
  - NP-complete (Arnborg, 1985) but greedy heuristics (min-fill).
Properties of bucket-elimination (Adaptive consistency)

- Adaptive consistency generates a backtrack-free representation.

- The time and space complexity of adaptive consistency along ordering $d$ is $\exp(w^*(d))$. Therefore, problems having bounded induced width are tractable (solved in polynomial time).

- Examples:
  - trees ($w^*=1$),
  - series-parallel networks ($w^*=2$)
  - partial $k$-trees ($w^*=k$).
Adaptive consistency is linear for trees and equivalent to enforcing directional arc-consistency (recording only unary constraints)
Tree Solving is Easy
Tree Solving is Easy

[Diagram of a tree with nodes labeled 1,2,3 and arrows indicating relationships.]
Tree Solving is Easy
Tree Solving is Easy
Tree Solving is Easy
Tree Solving is Easy
Tree Solving is Easy
Tree Solving is Easy

Constraint propagation

Solves trees in linear time
Tree Decomposition

- Each function in a cluster
- Satisfy running intersection property
Cluster Tree Elimination

1. \[ A \ B \ C \ R(a), R(b,a), R(c,a,b) \]
2. \[ B \ C \ D \ F \ R(d,b), R(f,c,d), h_{(1,2)}(b,c) \]
3. \[ B \ E \ F \ R(e,b,f), h_{(2,3)}(b,f) \]
4. \[ E \ F \ G \ R(g,e,f) \]

\[ h_{(1,2)}(b,c) = \downarrow_a R(a) \otimes R(b,a) \otimes R(c,a,b) \]

\[ h_{(2,3)}(b,f) = \downarrow_{c,d} R(d,b) \otimes R(f,c,d) \otimes h_{(1,2)}(b,c) \]

\[ \text{sep}(2,3) = \{B,F\} \]
\[ \text{elim}(2,3) = \{C,D\} \]
CTE: Cluster Tree Elimination

Time: \( O(\exp(w^*+1)) \)
Space: \( O(\exp(sep)) \)

\[
\begin{align*}
    h_{(1,2)}(b,c) & = \downarrow_a R(a) \otimes R(b,a) \otimes R(c,a,b) \\
    h_{(2,1)}(b,c) & = \downarrow_{d,f} R(d,b) \otimes R(f,c,d) \otimes h_{(3,2)}(b,f) \\
    h_{(2,3)}(b,f) & = \downarrow_{c,d} R(d,b) \otimes R(f,c,d) \otimes h_{(1,2)}(b,c) \\
    h_{(3,2)}(b,f) & = \downarrow_e R(e,b,f) \otimes h_{(4,3)}(e,f) \\
    h_{(3,4)}(e,f) & = \downarrow_b R(e,b,f) \otimes h_{(2,3)}(b,f) \\
    h_{(4,3)}(e,f) & = \downarrow_g R(g,e,f)
\end{align*}
\]
A tree decomposition for a belief network $BN = \langle X,D,G,P \rangle$ is a triple $\langle T, \chi, \psi \rangle$, where $T = (V,E)$ is a tree and $\chi$ and $\psi$ are labeling functions, associating with each vertex $v \in V$ two sets, $\chi(v) \subseteq X$ and $\psi(v) \subseteq P$ satisfying:

1. For each function $p_i \in P$ there is exactly one vertex such that $p_i \in \psi(v)$ and $\text{scope}(p_i) \subseteq \chi(v)$
2. For each variable $X_i \in X$ the set $\{v \in V | X_i \in \chi(v)\}$ forms a connected subtree (running intersection property)
Induced-width and Tree-width

**Induced-width of ordering**

- \( W^*(D) = 3 \)
- \( W^*(d) = 3 \)
- \( W^*(D) = 2 \)
- \( W^*(d) = 2 \)

**Tree-width**

- \( \text{Tree-width} = 3 \)
- \( \text{Tree-width} = 2 \)

**Tree-width of a graph** = smallest cluster in a cluster-tree

**Path-width of a graph** = smallest cluster in a cluster-path
Adaptive Consistency, Bucket-elimination

**Initialize**: partition constraints into $\text{bucket}_1, \ldots, \text{bucket}_n$

For $i=n$ down to 1 along d // process in reverse order

for all relations $R_1, \ldots, R_m \in \text{bucket}_i$ do

// join all relations and “project-out”

$$R_{\text{new}} \leftarrow \prod_{X_i}((-X_i) \bigwedge_j R_j)$$

**If** $R_{\text{new}}$ is not empty, add it to $\text{bucket}_{k}, k < i$, where $k$ is the largest variable index in $R_{\text{new}}$

**Else** problem is unsatisfiable

**Return** the set of all relations (old and new) in the buckets
The Idea of Elimination

\[ R_{DBC} = \prod_{DBC} R_{ED} \bowtie R_{EB} \bowtie R_{EC} \]

Eliminate variable E \leftrightarrow join and project
Properties of bucket-elimination (adaptive consistency)

- Adaptive consistency generates a constraint network that is backtrack-free (can be solved without deadends).

- The time and space complexity of adaptive consistency along ordering $d$ is \[ \text{complexity} \].

- Therefore, problems having bounded induced width are tractable (solved in polynomial time).

- Examples of tractable problem classes: trees ($w^* = 1$), series-parallel networks ($w^* = 2$), and in general $k$-trees ($w^* = k$).
Road Map

- Graphical models
- Constraint networks Model
- Inference
  - Variable elimination:
  - Tree-clustering
  - Constraint propagation
- Search
- Probabilistic Networks
Approximating Inference: Local Constraint Propagation

- **Problem**: bucket-elimination/tree-clustering algorithms are intractable when \( \textit{induced width} \) is large

- **Approximation**: bound the size of recorded dependencies, i.e. perform \( \textit{local constraint propagation} \) (\textit{local inference})
From Global to Local Consistency
A binary constraint $R(X,Y)$ is **arc-consistent** w.r.t. $X$ if every value in $x$’s domain has a match in $y$’s domain.

$$R_X = \{1,2,3\}, \quad R_Y = \{1,2,3\}, \quad \text{constraint} \quad X < Y$$

*Only domains are reduced:*

$$R_X \leftarrow \prod_X R_{XY} \boxtimes D_Y$$
Arc-consistency

\[ 1 \leq X, Y, Z, T \leq 3 \]
\[ X < Y \]
\[ Y = Z \]
\[ T < Z \]
\[ X \leq T \]
$I \leq X, Y, Z, T \leq 3$

$X < Y$

$Y = Z$

$T < Z$

$X \leq T$

$R_X \leftarrow \prod_X R_{XY} \otimes D_Y$
AC-3

AC-3(\(\mathcal{R}\))

\textbf{input:} a network of constraints \(\mathcal{R} = (X, D, C')\)

\textbf{output:} \(\mathcal{R}'\) which is the largest arc-consistent network equivalent to \(\mathcal{R}\)

1. \textbf{for} every pair \(\{x_i, x_j\}\) that participates in a constraint \(R_{ij} \in \mathcal{R}\)
2. \hspace{1em} \textbf{queue} \leftarrow \textbf{queue} \cup \{(x_i, x_j), (x_j, x_i)\}
3. \textbf{endfor}
4. \textbf{while} queue \(\neq\) \{\}
5. \hspace{1em} select and delete \((x_i, x_j)\) from queue
6. \hspace{1em} \textbf{Revise}((x_i), x_j)
7. \hspace{1em} \textbf{if} Revise((x_i), x_j) \textbf{causes a change in} D_i
8. \hspace{2em} \textbf{then} queue \leftarrow queue \cup \{(x_k, x_i), i \neq k\}
9. \hspace{1em} \textbf{endif}
10. \textbf{endwhile}

Figure 3.5: Arc-consistency-3 (AC-3)

- Complexity: \(O(ek^3)\)
- Best case \(O(ek)\), since each arc may be processed in \(O(2k)\)

Ijcai 2011
**Arc-consistency Algorithms**

- **AC-1**: brute-force, distributed $O(nek^3)$
- **AC-3**: queue-based $O(ek^3)$
- **AC-4**: context-based, optimal $O(ek^2)$
- **AC-5,6,7,...** Good in special cases

**Important**: applied at every node of search

- (n number of variables, e=#constraints, k=domain size)
Path-consistency

A pair \((x, y)\) is path-consistent relative to \(Z\), if every consistent assignment \((x, v)\) has a consistent extension to \(z\).

Figure 3.8: (a) The matching diagram of a 2-value graph coloring problem. (b) Graphical picture of path-consistency using the matching diagram.
Example: path-consistency

Figure 3.12: A graph-coloring graph (a) before path-consistency (b) after path-consistency
Apply Revise-3 \((O(k^3))\) until no change

\[ R_{ij} \leftarrow R_{ij} \cap \pi_{ij} (R_{ik} \bigotimes D_k \bigotimes R_{kj}) \]

Path-consistency (3-consistency) adds binary constraints.

- PC-1: \(O(n^5 k^5)\)
- PC-2: \(O(n^3 k^5)\)
- PC-4 optimal: \(O(n^3 k^3)\)
**Local $i$-consistency**

*i-consistency*: Any consistent assignment to any $i-1$ variables is consistent with at least one value of any $i$-th variable

---

**Figure 3.17**: The scope of consistency enforcing: (a) arc-consistency, (b) path-consistency, (c) $i$-consistency
Linear inequalities

\[ x + y + z \leq 15, \ z \geq 13 \Rightarrow \]
\[ x \leq 2, \ y \leq 2 \]

Boolean constraint propagation, unit resolution

\[ (A \lor B \lor \neg C), (\neg B) \Rightarrow \]
\[ (A \lor \neg C) \]
Is propositional theory

\[ \varphi = \{ \neg A \lor B, \neg C \lor A, \neg B, C \} \] satisfiable?

A is not arc-consistent relative to B

Enforce arc-consistency by resolution:

\[ \text{res}(\neg A \lor B, \neg B) \Rightarrow \neg A \]
\[ \text{res}(\neg C \lor A, C) \Rightarrow A \]
\[ \text{res}(A, \neg A) \Rightarrow \Phi \]

Given also \((B \lor C)\), path-consistency:

\[ \text{Res}((A \lor \neg B), (B \lor C)) = (A \lor C) \]

Relational arc-consistency rule = unit-resolution
Directional Resolution ⇔ Adaptive Consistency

\[ \text{Width } w = 3 \]
\[ \text{Induced width } w^* = 3 \]

\[ |\text{bucket}_i| = O(\exp(w^*)) \]

DR time and space: \( O(n \exp(w^*)) \)
Directional i-consistency

E: E ≠ D, E ≠ C, E ≠ B
D: D ≠ C, D ≠ A
C: C ≠ B
B: A ≠ B
A:

Adaptive

R_{DCB}

d-path

R_{DC}, R_{DB}
R_{CB}

d-arc

R_{D}
R_{C}
R_{D}
Arc-consistency

- Sound
- Incomplete
- Always converges (polynomial)
Relational Distributed Arc-Consistency

**Primal**

- **A < B**: A < B
- **A < D**: A < D
- **B = C**: B = C
- **D < C**: D < C

**Dual**

- **AB**: AB
- **BC**: BC
- **AD**: AD
- **DC**: DC

Ijcai 2011
Road Map

- Graphical models
- Constraint networks Model
- Inference
  - Variable elimination:
  - Tree-clustering
  - Constraint propagation
- Search
- Probabilistic Networks
Road Map:
Search in Graphical Models

- Improving search by bounded-inference in branching ahead
- Improving search by looking-back
- The alternative AND/OR search space
Road Map:
Search in Graphical Models

- Improving search by bounded-inference in branching ahead
- Improving search by looking-back
- The alternative AND/OR search space
Backtracking Search for a Solution
Backtracking Search for a Solution
Backtracking Search for All Solutions
The Search Space
Before Arc-Consistency

X
Y
T
Z

D={1,2,3}
The Search Space
After Arc-Consistency
The Effect of Variable Ordering

z divides x, y and t
The Effect of Consistency Level

- After arc-consistency $z=5$ and $l=5$ are removed

- After path-consistency
  - $R'_{zx}$
  - $R'_{zy}$
  - $R'_{zl}$
  - $R'_{xy}$
  - $R'_{xl}$
  - $R'_{yl}$

*Tighter networks yield smaller search spaces*
Improving Backtracking $O(\exp(n))$

- **Before search:** (reducing the search space)
  - Arc-consistency, path-consistency, i-consistency
  - Variable ordering (fixed)

- **During search:**
  - **Look-ahead schemes:**
    - value ordering/pruning (*choose a least restricting value*),
    - variable ordering (*Choose the most constraining variable*)
  - **Look-back schemes:**
    - Backjumping
    - Constraint recording
    - Dependency-directed backtracking
Looking-Ahead:
Constraint Propagation in Search

- Apply some level of constraint propagation at each node,
  - Forward-checking (FC)
  - Arc-consistency (MAC)

- Then:
  - Value pruning or ordering: prune values that lead to deadend
  - Variable ordering: choose a variable that leaves least options open
Forward-Checking for Value Ordering

**FC overhead:**

**MAC overhead:**
Forward-Checking for Value Ordering

**FW overhead:**

**MAC overhead:**

Not searched by forward checking
Forward-Checking, Variable Ordering

**FW overhead:**

**MAC overhead:**
After $X1 = \text{red}$ choose $X3$ and not $X2$

**FW overhead:**

**MAC overhead:**
Forward-Checking, Variable Ordering

After $X_1 = \text{red}$ choose $X_3$ and not $X_2$

FW overhead:

MAC overhead:
Forward-Checking, Variable Ordering

After \( X_1 = \text{red} \) choose \( X_3 \) and not \( X_2 \)

**FW overhead:**

**MAC overhead:**
Arc-consistency for Value Ordering

FW overhead:

MAC overhead:
Arc-Consistency for Value Ordering

Arc-consistency prunes $x_1=\text{red}$
Prunes the whole tree

FW overhead:

MAC overhead:
Constraint Programming

- Constraint solving embedded in programming languages
- Allows flexible modeling + with algorithms
- Logic programs + forward checking
- Eclipse, ILog, OPL
- Using only look-ahead schemes
Branching-Ahead for SAT: DLL

example: $(\sim AVB)(\sim CVA)(AVBVD)(C)$

(Davis, Logeman and Laveland, 1962)

Backtracking look-ahead with
Unit propagation=
Generalized arc-consistency

Only enclosed area will be explored with unit-propagation
Road Map:
Search in Graphical Models

- Improving search by bounded-inference in branching ahead
- Improving search by looking-back
- The alternative AND/OR search space
Look-back: Backjumping / Learning

- **Backjumping:**
  - In deadends, go back to the most recent culprit.

- **Learning:**
  - constraint-recording, no-good recording.
  - good-recording
Look-Back: Backjumping

- \((X_1=r,x_2=b,x_3=b,x_4=b,x_5=g,x_6=r,x_7=\{r,b\})\)
- \((r,b,b,b,g,r)\) conflict set of \(x_7\)
- \((r,-,b,b,g,-)\) c.s. of \(x_7\)
- \((r,-,b,-,g,-,\) minimal conflict-set
- Leaf deadend: \((r,b,b,b,g,r)\)
- Every conflict-set is a no-good

Figure 6.1: A modified coloring problem.
Example 6.3.1 In Figure 6.4, all of the backjumps illustrated lead to internal dead-ends, except for the jump back to $\langle x_1, \text{green} \rangle, \langle x_2, \text{blue} \rangle, \langle x_3, \text{red} \rangle, \langle x_4, \text{blue} \rangle$, because this is the only case where another value exists in the domain of the culprit variable.
Example 6.3.1 In Figure 6.4, all of the backjumps illustrated lead to internal dead-ends, except for the jump back to ⟨⟨x₁, green⟩, ⟨x₂, blue⟩, ⟨x₃, red⟩, ⟨x₄, blue⟩⟩, because this is the only case where another value exists in the domain of the culprit variable. □
Complexity of Backjumping

Graph-based and conflict-based backjumping

Simple: always jump back to parent in pseudo tree

Complexity for csp: \( \exp(w \times \log n) \)

From \( \exp(n) \) to \( \exp(w \times \log n) \) while linear space
Look-back: No-good Learning

Learning means recording conflict sets used as constraints to prune future search space.

- $(x_1=2, x_2=2, x_3=1, x_4=2)$ is a dead-end
- Conflicts to record:
  - $(x_1=2, x_2=2, x_3=1, x_4=2)$ 4-ary
  - $(x_3=1, x_4=2)$ binary
  - $(x_4=2)$ unary
Figure 6.9: The search space explicating by backtracking on the CSP from Figure 6.1, using the variable ordering \(x_6, x_3, x_4, x_2, x_7, x_1, x_5\) and the value ordering (blue, red, green, teal). Part (a) shows the ordered constraint graph, part (b) illustrates the search space. The cut lines in (b) indicate branches not explored when graph-based learning is used.
Complexity of Nogood-Learning for consistency

The complexity of learning along $d$ is time and space exponential in $w^*(d)$:

- The number of dead-ends is bounded by $O(nk^{w^*(d)})$.
- Number of constraint tests per dead-end are $O(e)$.

Space complexity is $O(nk^{w^*(d)})$.
Time complexity is $O(n^2e \cdot k^{w^*(d)})$.

**No-good Learning reduces time to $O(\exp(w^*))$ but requires $O(\exp(w^*))$ space.**
All Solutions and Counting

- For all solutions and counting we will see
  - The additional impact of Good learning
  - BFS makes sense with good learning
  - BFS and DFS time and space exp(path-width)
  - Good-learning doesn’t help consistency task
Value of node = number of solutions below it
#CSP - OR Search Tree

14 solutions
From search trees to search graphs

- Any two nodes that root identical subtrees (subgraphs) can be merged
From search trees to search graphs

- Any two nodes that root identical subtrees (subgraphs) can be merged.
#CSP - Searching the Graph by Good Caching

A  context(A) = [A]
B  context(B) = [AB]
C  context(C) = [ABC]
D  context(D) = [ABD]
E  context(E) = [AE]
F  context(F) = [F]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>R_{ABC}</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>R_{BCD}</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>E</th>
<th>R_{ABE}</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>E</th>
<th>F</th>
<th>R_{AEF}</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Good Caching

(A=0, E=0) is good
V(A=0, E=0) = 1
# CSP - Searching the Graph by Good Caching

No caching: $O(\exp(n))$

Good-caching: $O(\exp(pw))$
Summary: Search Principles

- Constraint propagation prunes search space
- Constraint propagation yields good advice for how to branch and where to go
- Backjumping and no-good learning helps prune search space and revise problem.
- Good learning cache results but helps only counting, enumeration
Road Map:
Search in Graphical Models

- Improving search by bounded-inference in branching ahead
- Improving search by looking-back
- The alternative AND/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

No-goods
(A=1, B=1)
(B=0, C=0)

AND/OR vs. OR

OR
AND
OR
AND
OR
AND
OR
AND

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

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

Space: linear
Time: \(O(\exp(m))\)
\(O(w \times \log n)\)

Linear space,
Time: \(O(\exp(n))\)
**AND/OR vs. OR Spaces**

<table>
<thead>
<tr>
<th>width</th>
<th>depth</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*
#CSP – AND/OR Search Tree

Ijcai 2011
#CSP – AND/OR Tree DFS

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

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

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>R_{ABC}</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>R_{BCD}</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>E</th>
<th>R_{ABE}</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>E</th>
<th>F</th>
<th>R_{AEF}</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Pseudo-Trees
(Freuder 85, Bayardo 95, Bodlaender and Gilbert, 91)

\[ m \leq w \ast \log n \]

(a) Graph

(b) DFS tree
    depth=3

(c) pseudo-tree
    depth=2

(d) Chain
    depth=6
(a) (b) (c)
The AND/OR search tree of R relative to a 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 Search A/O Trees to Search A/O 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 sub-trees can be pruned too.
  - Some portions can be collapsed or reduced.
From search trees to search graphs

- Any two nodes that root identical subtrees (subgraphs) can be merged.
From search trees to search graphs

- Any two nodes that root identical subtrees (subgraphs) can be merged.
AND/OR Tree
An AND/OR Graph: Caching Goods
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

\[
\begin{align*}
\text{context}(B) &= \{A, B\} \\
\text{context}(c) &= \{A,B,C\} \\
\text{context}(D) &= \{D\} \\
\text{context}(F) &= \{F\}
\end{align*}
\]
Complexity of AND/OR Graph

- **Theorem:** Traversing the AND/OR search graph is time and space exponential in the induced width/tree-width.

- If applied to the OR graph complexity is time and space exponential in the path-width.
#CSP – AND/OR Tree DFS

Ijcai 2011 137
#CSP – AND/OR Search Graph
(Caching Goods)
#CSP – AND/OR Search Graph
(Caching Goods)

Time and Space
\(O(\exp(w^*))\)
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
AND/OR vs. OR DFS Algorithms

- **AND/OR tree**
  - Space: \( O(n) \)
  - Time: \( O(n k^m) \)
  - \( O(n k^{w*} \log n) \)

- **OR tree**
  - Space: \( O(n) \)
  - Time: \( O(k^n) \)

- **AND/OR graph**
  - Space: \( O(n k^{w*}) \)
  - Time: \( O(n k^{w*}) \)

- **OR graph**
  - Space: \( O(n k^{pw*}) \)
  - Time: \( O(n k^{pw*}) \)

\[ k = \text{domain size} \]
\[ m = \text{tree depth} \]
\[ n = \# \text{of variables} \]
\[ w* = \text{induced width} \]
\[ pw* = \text{path width} \]

(Freuder85; Bayardo95; Darwiche01)
Properties of minimal AND/OR graphs

Theorem (Complexity):
- Minimal AND/OR context graph is bounded exponentially by the tree-width \( w^* \) of the graphical model along its pseudo-tree
- Minimal OR search graph is bounded exponentially by its path-width \( pw^* \) (\( w^* \leq pw^* \))

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

Related to compilation schemes:
- Minimal OR – related to OBDDs (Bryant, McMillan)
- Minimal AND/OR – related to tree-OBDDs (McMillan 94),
- d-DNNF (Darwiche et. Al. 2002)
- Case-factor diagrams (Mcallester, Collins, Pereira, 2005)
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)

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

AO(i) time complexity?
Impact of AND/OR for Constraint Processing

- Minor impact for Constraint-satisfaction
  - Search with backjumping or without backjumping
    - Space: linear, Time: $O(\exp(\log n \ w^*))$
  - Search with no-goods learning
    - time and space: $O(\exp(w^*))$
  - Variable-elimination
    - time and space: $O(\exp(w^*))$

- Counting, enumeration
  - Search with backjumping
    - Space: linear, Time: $O(\exp(n))$
    - Space: linear, Time: $O(\exp(\log n \ w^*))$
  - Search with no-goods caching only
    - space: $O(\exp(w^*))$ Time: $O(\exp(n))$
    - space: $O(\exp(w^*))$ Time: $O(\exp(\log n \ w^*))$
  - Search with goods and no-goods learning
    - Time and space: $O(\exp(\text{path-width}), O(\exp(\log n \ w^*)))$
    - Time and space: $O(\exp(\text{tree-width}), O(\exp(w^*)))$
  - Variable-elimination
    - Time and space: $O(\exp(w^*))$
Road Map

- Graphical models
- Constraint networks Model
- Inference
  - Variable elimination:
  - Tree-clustering
  - Constraint propagation
- Search
- Probabilistic Networks
Road Map: Bayesian Networks

- Bayesian networks definition
- Inference: Variable-elimination and tree-clustering
- Bounded-inference
  - Belied propagation
  - Mini-bucket, mini-clustering
  - Iterative join-graph propagation: a GBP scheme
- Search, conditioning
Road Map: Bayesian Networks

- Bayesian networks definition
- Inference: Variable-elimination and tree-clustering
- Bounded-inference
  - Belied propagation
  - Mini-bucket, mini-clustering
  - Iterative join-graph propagation: a GBP scheme
- Search, conditioning
- Sampling
Bayesian Networks: Representation
(Pearl, 1988)

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

**Belief Updating:**
\[ P \text{ (lung cancer=yes | smoking=no, dyspnoea=yes ) } = ? \]

\[ \text{BN} = (G, \Theta) \]
Linkage Analysis

- 6 individuals
- Haplotype: \{2, 3\}
- Genotype: \{6\}
- Unknown
Pedigree: 6 people, 3 markers
Graphical Models

A graphical model \((X,D,F)\):

- \(X = \{X_1, \ldots, X_n\}\) variables
- \(D = \{D_1, \ldots, D_n\}\) domains
- \(F = \{f_1, \ldots, f_m\}\) functions

Operators:
- combination
- elimination (projection)

Tasks:
- **Belief updating**: \(\sum_{x,y} \prod_j p_i\)
- **MPE**: \(\max_x \prod_j p_j\)
- **CSP**: \(\prod_x \times_j c_j\)
- **Max-CSP**: \(\min_x \sum_j f_j\)

All these tasks are NP-hard
- exploit problem structure
- identify special cases
- approximate
Probabilistic Inference Tasks

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

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

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

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

- Bayesian networks definition
- Inference:
  - Variable-elimination
  - tree-clustering
- Bounded-inference
  - Belied propagation
  - Mini-bucket, mini-clustering
  - Iterative join-graph propagation: a GBP scheme
- Search, conditioning
- Sampling
“Moral” Graph

\[ P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid \text{parents}(X_i)) \]

Conditioned Probability Distribution (CPD) Clique in moral graph ("family")

Moralize ("marry parents")
Belief updating: \( P(X|\text{evidence}) = ? \)

\[
P(a|e=0) \propto P(a,e=0) = \sum_{e=0,d,c,b} P(a)P(b|a)P(c|a)P(d|b,a)P(e|b,c) = P(a)\sum_{e=0} \sum_{d} \sum_{c} P(c|a) \sum_{b} P(b|a)P(d|b,a)P(e|b,c)
\]

Variable Elimination

\( h^B(a,d,c,e) \)
Bucket elimination

Algorithm $BE-bel$ (Dechter 1996)

$$P(A | E = 0) = \alpha \sum_{E=0,D,C,B} P(A) \cdot P(B | A) \cdot P(C | A) \cdot P(D | A, B) \cdot P(E | B, C)$$

$$\sum \Pi_b$$

Elimination operator

bucket $B$: $P(b | a)$

bucket $C$: $P(c | a)$

bucket $D$: $P(d | b, a)$

bucket $E$: $P(e | b, c)$

$W^* = 4$ (max clique size)

"induced width"

$P(a | e=0) = \frac{P(a, e=0)}{P(e=0)}$

$P(a | e=0) = P(e=0)$
Combination of Cost Functions

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>f(A,B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>b</td>
<td>0.4</td>
</tr>
<tr>
<td>b</td>
<td>g</td>
<td>0.1</td>
</tr>
<tr>
<td>g</td>
<td>b</td>
<td>0</td>
</tr>
<tr>
<td>g</td>
<td>g</td>
<td>0.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B</th>
<th>C</th>
<th>f(B,C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>b</td>
<td>0.2</td>
</tr>
<tr>
<td>b</td>
<td>g</td>
<td>0</td>
</tr>
<tr>
<td>g</td>
<td>b</td>
<td>0</td>
</tr>
<tr>
<td>g</td>
<td>g</td>
<td>0.8</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
A & \quad B & \quad C & \quad f(A,B,C) \\
\hline
b & b & b & 0.1 \\
\text{x} & b & g & 0 \\
\text{x} & g & b & 0 \\
\text{x} & g & g & 0.08 \\
g & b & b & 0 \\
g & b & g & 0 \\
g & g & b & 0 \\
g & g & g & 0.4 \\
\end{align*}
\]

\[= 0.1 \times 0.8\]
Factors: Sum-Out Operation

The result of summing out variable $X$ from factor $f(X)$

is another factor over variables $Y = X \setminus \{X\}$:

$$\left( \sum_X f \right)(y) \overset{\text{def}}{=} \sum_x f(x, y)$$

<table>
<thead>
<tr>
<th>$B$</th>
<th>$C$</th>
<th>$D$</th>
<th>$f_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>.95</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
<td>.05</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>true</td>
<td>.9</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
<td>.1</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
<td>.8</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>.2</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>true</td>
<td>0</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$B$</th>
<th>$C$</th>
<th>$\sum_D f_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>1</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>1</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>1</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>1</td>
</tr>
</tbody>
</table>

$$\sum_B \sum_C \sum_D f_1 = 4$$
Input: A belief network \( \{P_1, ..., P_n\}, d, e. \)

Output: belief of \( X_1 \) given \( e \).

1. Initialize:

2. Process buckets from \( p = n \) to 1
   for matrices \( \lambda_1, \lambda_2, ..., \lambda_j \) in bucket \( p \) do
   - If (observed variable) \( X_p = x_p \) assign
     \( X_p = x_p \) to each \( \lambda_i \).
   - Else, (multiply and sum)
     \( \lambda_p = \sum X_p \prod_{i=1}^{j} \lambda_i \).
     Add \( \lambda_p \) to its bucket.

3. Return \( Bel(x_1) = \alpha P(x_1) \cdot \prod_i \lambda_i(x_1) \)
Finding

Algorithm \textit{BE-mpe} (Dechter 1996)

\[ \text{MPE} = \max \ P(\overline{x}) \]

\[ \sum \text{ is replaced by } \max : \]

\[ \text{MPE} = \max_{a, e, d, c, b} P(a)P(c \mid a)P(b \mid a)P(d \mid a, b)P(e \mid b, c) \]

bucket B: \[ P(b \mid a) \]

bucket C: \[ h^B(a, d, c, e) \]

bucket D: \[ P(c \mid a) \]

bucket E: \[ e=0, h^D(a, e) \]

bucket A: \[ P(a) \]

W*=4

"induced width" (max clique size)
Generating the MPE-tuple

1. \( a' = \arg \max_a P(a) \cdot h^E(a) \)
2. \( e' = 0 \)
3. \( d' = \arg \max_d h^C(a', d, e') \)
4. \( c' = \arg \max_c P(c | a') \times h^B(a', d', c, e') \)
5. \( b' = \arg \max b P(b | a') \times P(d' | b, a') \times P(e' | b, c') \)

Return \( (a', b', c', d', e') \)
Complexity of Bucket-elimination

- Theorem: Bucket-elimination is $O(r \cdot k^{w^*+1})$ time and $O(nk^{w^*})$ space.
- When $w=1$ then $w^*=1 \rightarrow$ trees
- When we have a tree of functions $w=w^*$ and the hypertree width $hw = 1$.  

bucket-elimination
Sends messages
From leaves to root
Belief Updating Example

**SUM-PROD operators**

**POLY-TREE structure**

---

**P(h,f,r,m,b) = P(h) P(f) P(m|h,f) P(r|f) P(b|m)**

**P(F | B=1) = ?**

**P(B=1) = .19643**

**P(F=1|B=1) = .3725**

---

Ijcai 2011

*Probability of evidence*

*Updated belief*
Inference and Treewidth

Inference algorithm:
*Time:* \(\exp(\text{tree-width})\)
*Space:* \(\exp(\text{tree-width})\)

\[ \text{treewidth} = 4 - 1 = 3 \]
\[ \text{treewidth} = (\text{maximum cluster size}) - 1 \]
A tree decomposition for a belief network $BN =< X, D, G, P >$ is a triple $< T, \chi, \psi >$, where $T = (V, E)$ is a tree and $\chi$ and $\psi$ are labeling functions, associating with each vertex $v \in V$ two sets, $\chi(v) \subseteq X$ and $\psi(v) \subseteq P$ satisfying:

1. For each function $p_i \in P$ there is exactly one vertex such that $p_i \in \psi(v)$ and $\text{scope}(p_i) \subseteq \chi(v)$
2. For each variable $X_i \in X$ the set $\{v \in V | X_i \in \chi(v)\}$ forms a connected subtree (running intersection property)
Belief Propagation

\[ h(u, v) = \sum_{\text{elim}(u, v)} \prod_{f \in \text{cluster}(u) \setminus \{h(v, u)\}} f \]

\[ \text{cluster}(u) = \psi(u) \cup \{h(x_1, u), h(x_2, u), \ldots, h(x_n, u), h(v, u)\} \]

Compute the message:

\[ h(u, v) = \sum_{\text{elim}(u, v)} \prod_{f \in \text{cluster}(u) \setminus \{h(v, u)\}} f \]
Join-Tree Clustering

EXACT algorithm

Time and space:
exp(cluster size)=exp(treewidth)

\[
h_{(1,2)}(b, c) = \sum_a p(a) \cdot p(b \mid a) \cdot p(c \mid a, b)
\]

\[
h_{(2,1)}(b, c) = \sum_{d, f} p(d \mid b) \cdot p(f \mid c, d) \cdot h_{(3,2)}(b, f)
\]

\[
h_{(2,3)}(b, f) = \sum_{c, d} p(d \mid b) \cdot p(f \mid c, d) \cdot h_{(1,2)}(b, c)
\]

\[
h_{(3,2)}(b, f) = \sum_e p(e \mid b, f) \cdot h_{(4,3)}(e, f)
\]

\[
h_{(3,4)}(e, f) = \sum_b p(e \mid b, f) \cdot h_{(2,3)}(b, f)
\]

\[
h_{(4,3)}(e, f) = p(G = g_e \mid e, f)
\]
Cluster Tree Elimination - properties

- Correctness and completeness: Algorithm CTE is correct, i.e. it computes the exact joint probability of a single variable and the evidence.

- Time complexity:
  - $O (\ deg \times (n+N) \times d^{w^*+1})$

- Space complexity: $O (\ N \times d^{sep})$

where
- $deg$ = the maximum degree of a node
- $n$ = number of variables (= number of CPTs)
- $N$ = number of nodes in the tree decomposition
- $d$ = the maximum domain size of a variable
- $w^*$ = the induced width
- $sep$ = the separator size
Road Map: Bayesian Networks

- Bayesian networks definition
- Inference: Variable-elimination and tree-clustering
- Bounded-inference
  - Belief propagation
  - Mini-bucket, mini-clustering
  - Iterative join-graph propagation: a GBP scheme
- Search, conditioning
From Directional i-consistency to Mini-buckets

E: \( E \neq D, E \neq C, E \neq B \)
D: \( D \neq C, D \neq A \)
C: \( C \neq B \)
B: \( A \neq B \)
A: 

\[ R_{DCB} \]

Adaptive

\[ R_{DC}, R_{DB}, R_{CB} \]

\[ R_D, R_C, R_D \]

\( d \)-path

\( d \)-arc
The idea of Mini-bucket \cite{dechter1997}

Local computation: bound the size of recorded dependencies

\textit{Split a bucket into mini-buckets $\Rightarrow$ bound complexity}

\[
\text{bucket } (X) = \{ h_1, \ldots, h_r, h_{r+1}, \ldots, h_n \}
\]

\[
h^X = \max_X \bigcap_{i=1}^n h_i
\]

\[
\{ h_1, \ldots, h_r \} \quad \{ h_{r+1}, \ldots, h_n \}
\]

\[
g^X = (\max_X \bigcap_{i=1}^r h_i) \cdot (\max_X \bigcap_{i=r+1}^n h_i)
\]

\[
h^X \leq g^X
\]

Exponential complexity decrease: $O(e^n) \rightarrow O(e^r) + O(e^{n-r})$
Mini-Bucket Elimination

\[
\begin{align*}
\text{Bucket } B & : P(E|B,C) \quad P(B|A) \quad P(D|A,B) \\
\text{Bucket } C & : P(C|A) \quad h^B(C,E) \\
\text{Bucket } D & \\
\text{Bucket } E & : E = 0 \quad h^C(A,E) \\
\text{Bucket } A & : P(A) \quad h^E(A) \quad h^D(A)
\end{align*}
\]

\[\max_B \prod \]

MPE* is an upper bound on MPE --U
Generating a solution yields a lower bound--L
Semantics of Mini-Bucket: Splitting a Node

Variables in different buckets are renamed and duplicated
(Kask et. al., 2001), (Geffner et. al., 2007), (Choi, Chavira, Darwiche, 2007)

Before Splitting:
Network N

After Splitting:
Network N'
MBE(i) (Dechter and Rish 1997)

- **Input:** \( i \) – max number of variables allowed in a mini-bucket
- **Output:** [lower bound (P of a sub-optimal solution), upper bound]

**Example:** \texttt{approx-mpe(3)} versus \texttt{elim-mpe}

\[ w^* = 2 \quad \text{vs.} \quad w^* = 4 \]
Mini-Clustering (for sum-product)

Split a cluster into mini-clusters  =>  bound complexity

\[ \{h_1, \ldots, h_r, h_{r+1}, \ldots, h_n\} \]

\[ \sum_{\text{elim} \ i=1}^{n} h_i \leq \left( \sum_{\text{elim} \ i=1}^{r} h_i \right) \cdot \left( \sum_{\text{elim} \ i=r+1}^{n} h_i \right) \]

Exponential complexity decrease  
\[ O(e^n) \rightarrow O(e^{\text{var}(r)}) + O(e^{\text{var}(n-r)}) \]
MBE for likelihood computation

- Idea mini-bucket is the same:

\[
\sum_{X} f(x) \cdot g(x) \leq \sum_{X} f(x) \cdot \sum_{X} g(x) \\
\sum_{X} f(x) \cdot g(x) \leq \sum_{X} f(x) \cdot \max_{X} g(X)
\]

- So we can apply a sum in each mini-bucket, or better, one sum and the rest max, or min (for lower-bound)

- MBE-bel-max(i,m), MBE-bel-min(i,m) generating upper and lower-bound on beliefs approximates BE-bel

- MBE-map(i,m): max buckets will be maximized, sum buckets will be sum-max. Approximates BE-map.
Mini-Clustering, $i$-bound=3

\[ h_{(1,2)}^1(b, c) = \sum_a p(a) \cdot p(b \mid a) \cdot p(c \mid a, b) \]

\[ h_{(1,2)}^1(b) = \sum_{c,d} p(d \mid b) \cdot h_{(1,2)}^1(b, c) \]

\[ h_{(2,3)}^1(f) = \max_{c,d} p(f \mid c, d) \]

**Approximate Algorithm**

*Time and space:*

\[ \exp(i\text{-bound}) \]

*Number of variables in a mini-cluster*
Properties of MBE(i)/mc(I)

- **Complexity**: $O(r \exp(i))$ time and $O(\exp(i))$ space.
- Yields an upper-bound and a lower-bound.

- **Accuracy**: determined by upper/lower (U/L) bound.

- As $i$ increases, both accuracy and complexity increase.

- Possible use of mini-bucket approximations:
  - As anytime algorithms
  - As heuristics in search

- Other tasks: similar mini-bucket approximations for: belief updating, MAP and MEU (Dechter and Rish, 1997)
Anytime Approximation

\( \text{anytime} - \text{mpe}(\epsilon) \)

Initialize : \( i = i_0 \)
While time and space resources are available
\[
i \leftarrow i + i_{\text{step}}
\]
\[
U \leftarrow \text{upper bound computed by approx - mpe}(i)
\]
\[
L \leftarrow \text{lower bound computed by approx - mpe}(i)
\]
keep the best solution found so far
\[
\text{if } 1 \leq \frac{U}{L} \leq 1 + \epsilon, \text{ return solution}
\]
end
return the largest \( L \) and the smallest \( U \)
CPCS networks – medical diagnosis (noisy-OR CPD’s)

Test case: no evidence

Anytime-mpe(0.0001)
U/L error vs time

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>cpcs360</th>
<th>cpcs422</th>
</tr>
</thead>
<tbody>
<tr>
<td>elim-mpe</td>
<td></td>
<td></td>
</tr>
<tr>
<td>anytime-mpe(ε = 10^{-4})</td>
<td>70.3</td>
<td>505.2</td>
</tr>
<tr>
<td>anytime-mpe(ε = 10^{-1})</td>
<td>70.3</td>
<td>110.5</td>
</tr>
</tbody>
</table>

Time (sec)
MBE-mpe vs. IBP

approx - mpe is better on low - w * codes
IBP is better on randomly generated (high - w*) codes

Bit error rate (BER) as a function of noise (sigma):
Grid 15x15 - 10 evidence
Heuristics for partitioning
(Dechter and Rish, 2003, Rollon and Dechter 2010)

**Scope-based Partitioning Heuristic** (SCP) aims at minimizing the number of mini-buckets in the partition by including in each minibucket as many functions as respecting the $i$ bound is satisfied.

Use greedy heuristic derived from a distance function to decide which functions go into a single mini-bucket.
Road Map: Bayesian Networks

- Bayesian networks definition
- Inference: Variable-elimination and tree-clustering
- Bounded-inference
  - Belied propagation
  - Mini-bucket, mini-clustering
  - Iterative join-graph propagation: a GBP scheme
- Search, conditioning
Iterative Belief Propagation

- Belief propagation is exact for poly-trees
- IBP - applying BP iteratively to cyclic networks

No guarantees for convergence
Works well for many coding networks
Lets combine iterative-nature with anytime--IJ GP
IJ GP: a generalized Belief Propagation

- Apply belief propagation on a cluster-graph

- IJ GP uses join-graph clustering which is both *anytime* and *iterative*

- IJ GP applies message passing along a join-graph, rather than a join-tree

- Empirical evaluation shows that IJ GP is almost always superior to other approximate schemes (IBP, MC)
Belief network

Loopy BP graph
Arcs labeled with any single variable should form a TREE.
Collapsing Clusters
Message propagation

Minimal arc-labeled:
sep(1,2)={D,E}
elim(1,2)={A,B,C}

Non-minimal arc-labeled:
sep(1,2)={C,D,E}
elim(1,2)={A,B}
Constructing Join-Graphs

G: (GFE)
E: (EBF) (EF)
F: (FCD) (BF)
D: (DB) (CD)
C: (CAB) (CB)
B: (BA) (AB) (B)
A: (A)

a) schematic mini-bucket(i), i=3

b) arc-labeled join-graph decomposition
Linear Block Codes

Input bits

Parity bits

Received bits

Gaussian channel noise

$\sigma$

$Ijcai 2011$
Coding Networks – Bit Error Rate

N=400, 1000 instances, 30 it, \( w^* = 43 \), \( \sigma = .22 \)

\[
\begin{array}{c}
\text{IBP} \\
\text{UGP} \\
\text{MC}
\end{array}
\]

N=400, 500 instances, 30 it, \( w^* = 43 \), \( \sigma = .32 \)

\[
\begin{array}{c}
\text{IBP} \\
\text{UGP}
\end{array}
\]

N=400, 500 instances, 30 it, \( w^* = 43 \), \( \sigma = .51 \)

N=400, 500 instances, 30 it, \( w^* = 43 \), \( \sigma = .65 \)

\[
\begin{array}{c}
\text{IBP} \\
\text{UGP}
\end{array}
\]
CPCS 422 – KL Distance

CPCS 422, evid=0, w*=23, 1instance

CPCS 422, evid=30, w*=23, 1instance

KL distance

Ijcai 2011
CPCS 422 – KL vs. Iterations

CPCS 422, evid=0, w*=23, 1instance

KLDistance vs. Number of Iterations

CPCS 422, evid=30, w*=23, 1instance

KLDistance vs. Number of Iterations

Evidence = 0

Evidence = 30

IJcai 2011
More On the Power of Belief Propagation

- BP as local minima of KL distance
- BP’s power from constraint propagation perspective.
Optimizing the KL-Divergence

- IBP fixed points are stationary points of the KL-divergence: they may only be local minima, or they may not be minima.
- When IBP performs well, it will often have fixed points that are indeed minima of the KL-divergence.
- For problems where IBP does not behave as well, we will next seek approximations $Pr'$ whose factorizations are more expressive than that of the polytree-based factorization.

*These results also extend to generalized BP*
Belief and constraint networks

a) Belief network

\[
P(A) \\
P(B|A) \\
P(C|A) \\
P(D|A,B) \\
P(F|B,C) \\
P(G|D,F)
\]

\[
P(A) \rightarrow B \rightarrow C \\
A \rightarrow D \rightarrow F \\
B \rightarrow G \\
D \rightarrow F \\
A \rightarrow B \rightarrow D \\
C \rightarrow F \\
D \rightarrow B \rightarrow C \\
D \rightarrow B \rightarrow F \\
A \rightarrow C \rightarrow D \\
A \rightarrow C \rightarrow B \\
A \rightarrow D \rightarrow F \\
R(A) \rightarrow A \rightarrow B \\
R(A) \rightarrow A \rightarrow C \\
R(A) \rightarrow A \rightarrow B \rightarrow D \\
R(B) \rightarrow B \rightarrow C \rightarrow F \\
R(D) \rightarrow D \rightarrow F \rightarrow G \\
R(B) \rightarrow B \rightarrow D \\
R(C) \rightarrow C \rightarrow F \\
R(B) \rightarrow B \rightarrow C \\
R(C) \rightarrow C \rightarrow B \\
R(A,B) \rightarrow A \rightarrow B \\
R(A,C) \rightarrow A \rightarrow C \\
R(A,B,D) \rightarrow A \rightarrow B \rightarrow D \\
R(B,C,F) \rightarrow B \rightarrow C \rightarrow F \\
R(D,F,G) \rightarrow D \rightarrow F \rightarrow G \\
\]

b) Constraint network

\[
R(A) \\
R(A,B) \\
R(A,C) \\
R(A,B,D) \\
R(B,C,F) \\
R(D,F,G)
\]

\[
R(A) \rightarrow A \rightarrow B \\
R(A) \rightarrow A \rightarrow C \\
R(A) \rightarrow A \rightarrow B \rightarrow D \\
R(B) \rightarrow B \rightarrow C \rightarrow F \\
R(D) \rightarrow D \rightarrow F \rightarrow G \\
R(B) \rightarrow B \rightarrow D \\
R(C) \rightarrow C \rightarrow F \\
R(B) \rightarrow B \rightarrow C \\
R(C) \rightarrow C \rightarrow B \\
R(A,B) \rightarrow A \rightarrow B \\
R(A,C) \rightarrow A \rightarrow C \\
R(A,B,D) \rightarrow A \rightarrow B \rightarrow D \\
R(B,C,F) \rightarrow B \rightarrow C \rightarrow F \\
R(D,F,G) \rightarrow D \rightarrow F \rightarrow G \\
\]

c) Singleton dual join-graph

\[
R(A) \\
R(A,B) \\
R(A,C) \\
R(B,D) \\
R(B,C,F) \\
R(D,F,G)
\]

\[
R(A) \rightarrow A \rightarrow B \\
R(A) \rightarrow A \rightarrow C \\
R(A) \rightarrow A \rightarrow B \rightarrow D \\
R(B) \rightarrow B \rightarrow C \rightarrow F \\
R(D) \rightarrow D \rightarrow F \rightarrow G \\
R(B) \rightarrow B \rightarrow D \\
R(C) \rightarrow C \rightarrow F \\
R(B) \rightarrow B \rightarrow C \\
R(C) \rightarrow C \rightarrow B \\
R(A,B) \rightarrow A \rightarrow B \\
R(A,C) \rightarrow A \rightarrow C \\
R(A,B,D) \rightarrow A \rightarrow B \rightarrow D \\
R(B,C,F) \rightarrow B \rightarrow C \rightarrow F \\
R(D,F,G) \rightarrow D \rightarrow F \rightarrow G \\
\]
Distributed Relational Arc-Consistency

- Can be applied to the dual problem of any constraint network:

\[ h^i_i \leftarrow \pi_{l_{ij}}(R_i \Join \bigotimes_{k \in ne(i)} h^i_k) \]  (1)

\[ R_i \leftarrow R_i \cap \bigotimes_{k \in ne(i)} h^i_k \]  (2)
Flattening the Bayesian network

Belief network

Flat constraint network
Belief zero Propagation equals Arc-Consistency

\[ h_i^j = \sum_{\text{elim}(i,j)} (p_i \cdot (\prod_{k \in \text{ne}_j(i)} h_k^i)) \]

\[ h_i^j = \pi_{ij} (R_i \otimes (\otimes_{k \in \text{ne}(i)} h_k^i)) \]

Updated belief:

\[ Bel(A, B) = P(B | A) \cdot h_1^2 \cdot h_4^2 \cdot h_5^2 = \]

\[ = \]

Updated relation:

\[ R(A, B) = R(A, B) \otimes h_1^2 \otimes h_4^2 \otimes h_5^2 = \]

\[ = \]
Main theorem: IBP’s trace for zero beliefs is identical to arc-consistency on the flat network

Consequently,

1. **Soundness**: The inference of zero beliefs by IBP converges in a finite number of iterations, and all zero beliefs inferred are correct

2. **Incompleteness**: IBP may not infer all the true zero beliefs
Road Map: Bayesian Networks

- Bayesian networks definition
- Inference: Variable-elimination and tree-clustering
- Bounded-inference
  - Belied propagation
  - Mini-bucket, mini-clustering
  - Iterative join-graph propagation: a GBP scheme
- Search, conditioning
- Hybrid of Search and Inference
Conditioning generates the probability tree

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

\begin{figure}
\centering
\includegraphics[width=\textwidth]{probability_tree}
\caption{Complexity of conditioning: exponential time, linear space}
\end{figure}
AND/OR vs. OR Spaces

54 nodes

126 nodes
### 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>$O(n \cdot d^t)$</td>
<td>$O(d^n)$</td>
</tr>
<tr>
<td></td>
<td>$O(n \cdot d^{w^* \cdot \log n})$</td>
<td></td>
</tr>
</tbody>
</table>

*Freuder & Quinn85, (Collin, Dechter & Katz91), (Bayardo & Miranker95), (Darwiche01)*

\[d = \text{domain size}\]
\[t = \text{depth of pseudo-tree}\]
\[n = \text{number of variables}\]
\[w^* = \text{treewidth}\]
Pseudo-Trees
(Freuder 85, Bayardo 95, Bodlaender and Gilbert, 91)

\[ t \leq w \times \log n \]

(a) Graph
(b) DFS tree
  depth=3
(c) pseudo-tree
  depth=2
(d) Chain
  depth=6
Weighted AND/OR Tree

AND/OR node: Marginalization operator (summation)

AND node: Combination operator (product)

Value of node = updated belief for sub-problem below

Evidence: E=0

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

Evidence: D=1

**Ijcai 2011 209**
Merging based on **context**

\[
\text{context (X)} = \text{ancestors of X connected to descendants of X}
\]
How big is the context?

context (X) = ancestors of X in pseudo tree that are connected to X, or to descendants of X

context (X) = parents in the induced graph

max |context| = induced width = treewidth
AND/OR Tree DFS Algorithm
(Belief Updating)

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

Value of node = updated belief for sub-problem below

OR node: Marginalization operator (summation)
AND node: Combination operator (product)
**AND/OR Graph DFS Algorithm**

*(Belief Updating)*

Evidence: \( E = 0 \)

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

Context

Cache table for \( D \)

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

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

Evidence: \( D = 1 \)
# 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 , d^{w^*}))</td>
<td>(O(n , d^{pw^*}))</td>
</tr>
<tr>
<td><strong>Time</strong></td>
<td>(O(n , d^{w^*}))</td>
<td>(O(n , d^{pw^*}))</td>
</tr>
</tbody>
</table>

\(d = \text{domain size}\)
\(n = \text{number of variables}\)
\(w^* = \text{treewidth}\)
\(pw^* = \text{pathwidth}\)

\(w^* \leq pw^* \leq w^* \log n\)
Constructing Pseudo Trees

- AND/OR search algorithms are influenced by the quality of the pseudo tree

- Finding the minimal induced width / depth pseudo tree is NP-hard

- Heuristics
  - Min-Fill (min induced width)
  - Hypergraph partitioning (min depth)
### Quality of the 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*

*SPOT5 Benchmarks*
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
Theorem: The maximum context size for a pseudo tree is equal to the treewidth of the graph along the pseudo tree.

\[(C K H A B E J L N O D P M F G)\]
AND/OR Context Minimal
Graph

Variable Elimination

AND/OR Search

(C K H A B E J L N O D P M F G)
Searching AND/OR Graphs

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

\[
\begin{align*}
&i=0 & j & i=w^* \\
\text{Space:} & O(n) & & \text{Space:} O(\exp(w^*)) \\
\text{Time:} & O(\exp(w^* \log n)) & & \text{Time:} O(\exp(w^*))
\end{align*}
\]

\[
\begin{align*}
&j \\
\text{Space:} & O(\exp(j)) & \text{Space:} O(\exp(j)) \\
\text{Time:} & O(\exp(m_j+j)) & \text{Time:} O(\exp(m_j+j))
\end{align*}
\]
Searching the AND/OR space for MPE/MAP

Heuristic function \( f(x^p) \) computes a lower bound on the best extension of \( x^p \) and can be used to guide a heuristic search algorithm. We focus on:

1. **DF Branch-and-Bound**
   Use heuristic function \( f(x^p) \) to prune the depth-first search tree
   Linear space

2. **Best-First Search**
   Always expand the node with the highest heuristic value \( f(x^p) \)
   Needs lots of memory

Ijcai 2011
Road Map: Bayesian Networks

- Bayesian networks definition
- Inference: Variable-elimination and tree-clustering
- Bounded-inference
  - Belied propagation
  - Mini-bucket, mini-clustering
  - Iterative join-graph propagation: a GBP scheme
- Search, conditioning
- Hybrid of Search and Inference
- Reasoning with mixed networks
Conditioning generates the probability tree

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

Complexity of conditioning: exponential time, linear space
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. For each cutset we can do BP
Conditioning and Cycle cutset

Cycle cutset = \{A,B,C\}
Search over the Cutset (cont)

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

Graph Coloring problem

Ijcai 2011
Variable elimination with conditioning; w-cutset algorithms

- Identify an w-cutset of the network
- For each assignment to the cutset solve the conditioned sub-problem
- Aggregate the solutions over all assignments.
- Time complexity:
- Space complexity: $O(k^\omega)$
- What w should we use?
Time vs Space for \( w \)-cutset

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

\[ W + c(w) \text{ vs } w \]

\( W \)-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)
Road Map: Bayesian Networks

- Bayesian networks definition
- Inference: Variable-elimination and tree-clustering
- Bounded-inference
- Search, conditioning
- Hybrid of Search and Inference
- Reasoning with mixed networks
  - Inference
  - Search
  - Sampling solutions
Road Map: Bayesian Networks

- Bayesian networks definition
- Inference: Variable-elimination and tree-clustering
- Bounded-inference
- Search, conditioning
- Hybrid of Search and Inference
- Reasoning with mixed networks
  - Inference
  - Search
  - Sampling solutions
Mixed Networks

Bayesian Network

Constraint Network

Mix: Combine? Subsume?

Semantics?

Algorithms?

231
Uncertainty and Determinism

Belief Network (B)

Constraint Network (R)

Variables: $A, B, C, D, E, F$
Domains: $D_A = D_B = D_C = D_D = D_E = D_F = \{0, 1\}$
CPTS: $P(A), P(B | A), P(C | A), P(D | B, C)$
$P(E | A, B), P(F | A)$

Variables: $A, B, C, D, E, F$
Domains: $D_A = D_B = D_C = D_D = D_E = D_F = \{0, 1\}$
Relations: $R_1(ABC), R_2(ACF), R_3(BCD), R_4(A, E)$
Expresses the set of solutions: $\rho = R(ABCDEF)$
Mixed Networks

Theorem: The mixed graph is a minimal I-map (independency map) relative to dm-separation.
Tasks for Mixed Networks

- Given $M = (B, R)$:
  - **Belief updating**: Find the likelihood of $X$ given $e$ and assuming consistency: $P(x|R,e) = ?$
  - **MPE**: find probability of most likely solution of $R$
  - **MAP**: Find the probability of most likely consistent assignment to a subset of variables
  - **Constraint Probability Evaluation (CPE)**: Find the probability that an assignment is consistent,

- All these can be extended to influence diagrams, and
  - We can add constraints between control variables
  - The relevant probability distribution is conditional of $R$
Road Map: Bayesian Networks

- Bayesian networks definition
- Inference: Variable-elimination and tree-clustering
- Bounded-inference
- Search, conditioning
- Hybrid of Search and Inference
- Reasoning with mixed networks
  - Inference
  - Search
  - Sampling solutions
Belief network $P(g, f, d, c, b, a)$

$= P(g|f, d)P(f|c, b)P(d|b, a)P(b|a)P(c|a)P(a)$

$\varphi = (G \lor D) \land \neg (D \land \neg B) \land (B \lor C)$

$P(\varphi) = ?$

$P(x | \varphi) = ?$
Complexity

- Time and space exponential in the induced-width of the mixed graph, whose evidence node and unit literals are removed.

- Apply constraint propagation to the constraint portion and then solve the mixed network.
Elim-CPE-D on Insurance network
(Dechter and Larkin, UAI 2001)

19 instances with Insurance network. 20 relations, arity 3, tightness 25%, 5 evidence nodes.
Road Map: Bayesian Networks

- Bayesian networks definition
- Inference: Variable-elimination and tree-clustering
- Bounded-inference
- Search, conditioning
- Hybrid of Search and Inference
- Reasoning with mixed networks
  - Inference
  - Search
  - Sampling solutions
The AND/OR search tree of a constraint network R relative to a pseudo-tree, T, has alternating levels of: AND nodes (variables) and OR nodes (values).

The root is the root of T (OR node).

Successor function:
- The successors of an OR node X are all its consistent values along its path.
- The successors of an AND node <X,v> are all the children of X in T.
- AND nodes have labels.

A solution is a subtree.
AND/OR search tree properties

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

  Size of AND/OR search tree is \( O(n \, k^m) \)

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

- **Theorem:** A constraint network that has a tree-width \( w^* \) has an AND/OR search tree whose size is bounded by \( O(\exp(w^* \, \log n)) \)

  (similar to RC, Darwiche 01; Bacchus et.al 03; Freuder 85; Bayardo 95)

- **Result:** AND/OR search tree algorithms are
  - Space: \( O(n) \)
  - Time: \( O(\exp(w^* \, \log n)) \)

\( k = \text{domain size} \)
\( m = \text{pseudo-tree depth} \)
\( n = \text{number of variables} \)
AND/OR tree 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>

\[ A \quad B = 0 \quad B = 1 \]

<table>
<thead>
<tr>
<th>A</th>
<th>C=0</th>
<th>C=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>

\[ A \quad P(A) \]

<table>
<thead>
<tr>
<th>P(D,B,C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

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

\[ \text{Result: } P(D=1,E=0) \]

\[ \text{OR node: Marginalization operator (summation)} \]

\[ \text{AND node: Combination operator (product)} \]

\[ \text{Value of node = updated belief for subproblem below} \]
The Effect of Constraint Propagation

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

CONSTRANTS ONLY

FORWARD CHECKING

MAINTAINING ARC CONSISTENCY

Changes'05
Experimental results

- **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 allowed tuples)

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

---

**OR vs. AND/OR Spaces**

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

N=25, K=2, R=2, P=2, C=10, S=3, t=70%, 20 instances, w*=9, h=14
### AND/OR Search Algorithms (1)

<table>
<thead>
<tr>
<th>t</th>
<th>i</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>#sol</th>
</tr>
</thead>
<tbody>
<tr>
<td>20%</td>
<td>0</td>
<td>0.671</td>
<td>0.056</td>
<td>0.022</td>
<td>153,073</td>
<td>4,388</td>
<td>95,197</td>
<td>3,299</td>
<td>962</td>
<td>1.6E+05</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.619</td>
<td>0.053</td>
<td>0.019</td>
<td>101,268</td>
<td>3,476</td>
<td>95,197</td>
<td>3,299</td>
<td>962</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0.479</td>
<td>0.055</td>
<td>0.022</td>
<td>75,397</td>
<td>3,213</td>
<td>95,197</td>
<td>3,299</td>
<td>962</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0.297</td>
<td>0.053</td>
<td>0.019</td>
<td>51,746</td>
<td>3,152</td>
<td>95,197</td>
<td>3,299</td>
<td>962</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>0.103</td>
<td>0.044</td>
<td>0.016</td>
<td>16,579</td>
<td>2,273</td>
<td>95,197</td>
<td>3,299</td>
<td>962</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40%</td>
<td>0</td>
<td>2.877</td>
<td>0.791</td>
<td>1.094</td>
<td>774,697</td>
<td>167,921</td>
<td>239,991</td>
<td>40,069</td>
<td>36,119</td>
<td>7.7E+07</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2.426</td>
<td>0.663</td>
<td>0.894</td>
<td>329,686</td>
<td>57,023</td>
<td>239,991</td>
<td>40,069</td>
<td>36,119</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1.409</td>
<td>0.445</td>
<td>0.544</td>
<td>183,286</td>
<td>35,325</td>
<td>239,991</td>
<td>40,069</td>
<td>36,119</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0.739</td>
<td>0.301</td>
<td>0.338</td>
<td>119,125</td>
<td>23,655</td>
<td>239,991</td>
<td>40,069</td>
<td>36,119</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>0.189</td>
<td>0.142</td>
<td>0.149</td>
<td>27,848</td>
<td>9,148</td>
<td>239,991</td>
<td>40,069</td>
<td>36,119</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60%</td>
<td>0</td>
<td>6.827</td>
<td>4.717</td>
<td>7.427</td>
<td>1,974,952</td>
<td>1,158,544</td>
<td>622,791</td>
<td>158,968</td>
<td>622,791</td>
<td>6.2E+09</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>5.560</td>
<td>3.908</td>
<td>6.018</td>
<td>673,117</td>
<td>351,022</td>
<td>622,791</td>
<td>158,968</td>
<td>622,791</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>2.809</td>
<td>2.219</td>
<td>3.149</td>
<td>346,842</td>
<td>183,895</td>
<td>622,791</td>
<td>158,968</td>
<td>622,791</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>1.334</td>
<td>1.196</td>
<td>1.535</td>
<td>204,414</td>
<td>105,527</td>
<td>622,791</td>
<td>158,968</td>
<td>622,791</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>0.255</td>
<td>0.311</td>
<td>0.425</td>
<td>36,262</td>
<td>23,160</td>
<td>622,791</td>
<td>158,968</td>
<td>622,791</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>80%</td>
<td>0</td>
<td>14.181</td>
<td>14.199</td>
<td>21.791</td>
<td>4,282,678</td>
<td>3,703,920</td>
<td>370,314</td>
<td>277,250</td>
<td>370,314</td>
<td>1.1E+11</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>11.334</td>
<td>11.797</td>
<td>17.916</td>
<td>1,319,599</td>
<td>1,108,561</td>
<td>370,314</td>
<td>277,250</td>
<td>370,314</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>5.305</td>
<td>6.286</td>
<td>9.061</td>
<td>626,405</td>
<td>519,258</td>
<td>370,314</td>
<td>277,250</td>
<td>370,314</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>2.204</td>
<td>2.890</td>
<td>3.725</td>
<td>336,146</td>
<td>274,375</td>
<td>370,314</td>
<td>277,250</td>
<td>370,314</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>0.318</td>
<td>0.543</td>
<td>0.714</td>
<td>44,340</td>
<td>39,550</td>
<td>370,314</td>
<td>277,250</td>
<td>370,314</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100%</td>
<td>0</td>
<td>23.595</td>
<td>27.129</td>
<td>41.744</td>
<td>7,450,537</td>
<td>7,450,537</td>
<td>7,450,537</td>
<td>0</td>
<td>0</td>
<td>1.1E+12</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>19.050</td>
<td>22.842</td>
<td>34.800</td>
<td>2,161,401</td>
<td>2,161,401</td>
<td>2,161,401</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>8.325</td>
<td>11.528</td>
<td>16.636</td>
<td>956,965</td>
<td>956,965</td>
<td>956,965</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>3.153</td>
<td>4.863</td>
<td>6.255</td>
<td>483,921</td>
<td>483,921</td>
<td>483,921</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>0.366</td>
<td>0.681</td>
<td>0.884</td>
<td>50,616</td>
<td>50,616</td>
<td>50,616</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**N=40, K=2, R=2, P=2, C=10, S=4, 20 instances, w*=12, h=19**
AND/OR Search Algorithms (2)

<table>
<thead>
<tr>
<th>N=100, K=2, R=10, P=2, C=30, S=3, 20 instances, w*=28, h=38</th>
</tr>
</thead>
<tbody>
<tr>
<td>t</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>10%</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>20%</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>30%</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>N=100, K=2, R=5, P=2, C=40, S=3, 20 instances, w*=41, h=51</th>
</tr>
</thead>
<tbody>
<tr>
<td>t</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>10%</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>20%</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>30%</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
### AND/OR Search vs. Bucket Elimination

**N=70, K=2, R=5, P=2, C=30, S=3, 20 instances, w*=23, h=31**

<table>
<thead>
<tr>
<th>t</th>
<th>i</th>
<th>Time</th>
<th>Nodes</th>
<th>Dead-ends</th>
<th>#sol</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>BE</td>
<td>AO-FC</td>
<td>AO-RFC</td>
</tr>
<tr>
<td>40%</td>
<td>0</td>
<td>26.400</td>
<td>1.956</td>
<td>1.263</td>
<td>48,768</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>1.872</td>
<td>1.231</td>
<td>30,299</td>
<td>17,954</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>1.878</td>
<td>1.291</td>
<td>26,335</td>
<td>17,467</td>
</tr>
<tr>
<td>50%</td>
<td>0</td>
<td>30.652</td>
<td>35.570</td>
<td>2,883,284</td>
<td>2,707,820</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>18.583</td>
<td>18.872</td>
<td>557,478</td>
<td>511,931</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>12.444</td>
<td>12.110</td>
<td>244,635</td>
<td>215,801</td>
</tr>
<tr>
<td>60%</td>
<td>0</td>
<td>396.754</td>
<td>511.434</td>
<td>51,223,471</td>
<td>50,089,187</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>167.852</td>
<td>182.451</td>
<td>5,881,086</td>
<td>5,707,665</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>80.484</td>
<td>83.601</td>
<td>1,722,900</td>
<td>1,655,420</td>
</tr>
</tbody>
</table>

### N=60, K=2, R=5, P=2, C=40, S=3, 20 instances, w*=23, h=31

<table>
<thead>
<tr>
<th>t</th>
<th>i</th>
<th>Time</th>
<th>Nodes</th>
<th>Dead-ends</th>
<th>#sol</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>BE</td>
<td>AO-FC</td>
<td>AO-RFC</td>
</tr>
<tr>
<td>40%</td>
<td>0</td>
<td>66.871</td>
<td>0.655</td>
<td>0.603</td>
<td>9,126</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.630</td>
<td>0.568</td>
<td>5,732</td>
<td>5,117</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>0.632</td>
<td>0.566</td>
<td>5,175</td>
<td>4,560</td>
</tr>
<tr>
<td>50%</td>
<td>0</td>
<td>3.178</td>
<td>3.021</td>
<td>57,769</td>
<td>54,802</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>2.993</td>
<td>2.794</td>
<td>30,991</td>
<td>28,121</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>2.731</td>
<td>2.578</td>
<td>24,668</td>
<td>22,522</td>
</tr>
<tr>
<td>60%</td>
<td>0</td>
<td>65.171</td>
<td>70.242</td>
<td>2,302,068</td>
<td>2,291,538</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>54.101</td>
<td>56.419</td>
<td>791,391</td>
<td>780,861</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>39.606</td>
<td>40.718</td>
<td>459,131</td>
<td>448,659</td>
</tr>
</tbody>
</table>
Road Map: Bayesian Networks

- Bayesian networks definition
- Inference: Variable-elimination and tree-clustering
- Bounded-inference
- Search, conditioning
- Hybrid of Search and Inference
- Reasoning with mixed networks
  - Inference
  - Search
  - Sampling solutions
Generate Random Solutions

- **Motivation:** generating tests for hardware verification
- *Given a CSP, \( R = (X, D, C) \), generate solutions for \( R \) s.t. if \( \rho = sol(R) \):*

\[
\forall t \in \rho, \ P(t) = \frac{1}{|\rho|}
\]

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td></td>
<td>0.5</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td></td>
<td>0.5</td>
</tr>
</tbody>
</table>

*Brute-force: generate and list all solutions*
Modeling CN as BN on the same variables (Approach 2):

- Find a BN over same variables s.t.

\[ P(x_1,\ldots,x_n) = \frac{1}{\#\text{sol}} \text{ if } (x_1,\ldots,x_n) \text{ is a solution} \]

Conversion solved problem
Modeling CN as BN on the same variables (Approach 2):

- Find a BN over same variables s.t.

\[ P(x_1,\ldots,x_n) = \frac{1}{\#\text{sol}} \quad \text{if } (x_1,\ldots,x_n) \text{ is a solution} \]
A variable-elimination-based conversion

Complexity: $\exp(w^*)$

But the network is already easy
A variable-elimination-based conversion

\[ P(B \mid A, D) = 1 / \prod_{AB} (R_{AB} \land R_{DB}) \]

**Complexity: \( \exp(w^*) \)**

*But the network is already easy*
A conversion algorithm

\[ R \rightarrow P \]

\[ B : R_{AB} \rightarrow R_{BD} \]
\[ D : R_{DE} \rightarrow R'_{AD} \]
\[ A : R_{AC} \rightarrow R'_{AE} \]
\[ C : R_{CE} \rightarrow R'_{CE} \]
\[ E : R'_{E} \]

\[ P(B \mid A, D) = \frac{R(A, B) \cdot R(B, D)}{\sum_{B} R(A, B) \cdot R(B, D)} \]

\[ R(A, D) = \sum_{B} R(A, B) \cdot R(B, D) \]

Changes’05
A conversion algorithm

\[ R \rightarrow P \]

\[ P(B \| A, D) = \frac{1}{\prod_{AB}(R_{AB} \bowtie R_{DB})} \]
Thank You

- “Constraint Processing”, Morgan Kaufmann, 2003

- **Probabilistic networks:** Transferring these ideas to Probabilistic network, helping unifying the principles.

- **Current work:** Mixing probabilistic and deterministic network

Questions?