Reasoning with Bayesian Networks

Rina Dechter
Donald Bren School of Computer Science
University of California, Irvine, USA
Road Map

- Overview: Bayesian networks and algorithms
- Exact Inference
- Bounded-inference
- Search
- Sampling
- Hybrid of search and inference
- Modeling and learning
- Software
Road Map

- Overview: Bayesian networks and algorithms
- Exact Inference
- Bounded-inference
- Search
- Sampling
- Hybrid of search and inference
- Modeling and learning
- Software
Bayesian Networks (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:

$$P\left(\text{lung cancer}=\text{yes} \mid \text{smoking}=\text{no}, \text{dyspnoea}=\text{yes}\right) = ?$$

Most likely explanation (MPE):

$$\text{MPE} = \text{find } \arg\max \ P(S)\cdot P(C|S)\cdot P(B|S)\cdot P(X|C,S)\cdot P(D|C,B)$$
Bayesian Networks encode independencies

Causal relationship

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

\[ BN = (G, \Theta) \]
What are they good for?

- **Diagnosis:** $P(\text{cause} | \text{symptom}) = ?$
- **Prediction:** $P(\text{symptom} | \text{cause}) = ?$
- **Classification:** $\max_{\text{class}} P(\text{class} | \text{data})$
- **Decision-making** (given a cost function)

Diagram:
- **Cause**
  - $C_1$
  - $C_2$

**Applications:**
- **Medicine**
  - Speech recognition
- **Bio-informatics**
- **Stock market**
- **Text Classification**
- **Computer troubleshooting**
Monitoring Intensive-Care Patients

Alarm network

37 variables
509 parameters

\[ \lll 2^{37} \]
• 6 individuals
• Haplotype: \{2, 3\}
• Genotype: \{6\}
• Unknown
Pedigree: 6 people, 3 markers
Constraint Networks

Map coloring

Variables: countries (A B C etc.)

Values: colors (red green blue)

Constraints: $A \not= B, A \not= D, D \not= E, ...$
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: \(\Sigma_{X,Y} \Pi_j P_i\)
  - MPE: \(\max_X \Pi_j P_j\)
  - CSP: \(\Pi_X \times_j C_j\)
  - Max-CSP: \(\min_X \Sigma_j f_j\)

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

\[
\begin{array}{c|c|c|c}
A & C & F & P(F|A,C) \\
0 & 0 & 0 & 0.14 \\
0 & 0 & 1 & 0.96 \\
0 & 1 & 0 & 0.40 \\
0 & 1 & 1 & 0.60 \\
1 & 0 & 0 & 0.35 \\
1 & 0 & 1 & 0.65 \\
1 & 1 & 0 & 0.72 \\
1 & 1 & 1 & 0.68 \\
\end{array}
\]

Relation

\[
f_i := (F = A + C)
\]

Primal graph (interaction graph)
Type of CPD

- Discrete variable
  - Tables
  - Noisy-or, noisy-and,
  - Decision trees
  - If/then rules
  - multinomial

- Continuous variables
  - Linear Gaussian
Example of Networks

Using Bayesian Networks to analyze expression data (Friedman et al. 2001)

FIG. 2. An example of the graphical display of Markov features. This graph shows a “local map” for the gene SVS1. The width (and color) of edges corresponds to the computed confidence level. An edge is directed if there is a sufficiently high confidence in the order between the genes connected by the edge. This local map shows that CLN2 separates SVS1 from several other genes. Although there is a strong connection between CLN2 to all these genes, there are no other edges connecting them. This indicates that, with high confidence, these genes are conditionally independent given the expression level of CLN2.
Example of networks

Fig. 2. (a) Naive Bayes model over 3 classes, for an expression data set with 3 expression measurements for each gene. A multinomial distribution is associated with \( g.C \) (shown as a histogram). For each class \( g.C \), each experiment is associated with a Gaussian CPD (shown in CPD 1). (b) Protein interaction model for a dataset with 4 genes in which the interactions are between: \( g_1 \) and \( g_2 \); \( g_2 \) and \( g_3 \); \( g_2 \) and \( g_4 \); and \( g_3 \) and \( g_4 \). Shown is the resulting Markov network, with its two types of potentials: \( \phi_i(g_i.C) \) and \( \phi_e(g_i.C, g_j.C) \). (c) Resulting unified partially-directed model.

Segal, Wang and Koller, 2003 “Discovering molecular pathways from Protein interaction and gene expression
Sample Domains for Graphical Models

- Web Pages and Link Analysis
- Communication Networks (Cell phone Fraud Detection)
- Natural Language Processing (e.g. Information Extraction and Semantic Parsing)
- Battle-space Awareness
- Epidemiological Studies
- Citation Networks
- Intelligence Analysis (Terrorist Networks)
- Financial Transactions (Money Laundering)

**Computational Biology**
- RNA
- Linkage Analysis
- Association studies
- Object Recognition and Scene Analysis

...
Road Map

- **Overview:** Bayesian networks and algorithms
- Exact Inference
- Bounded-inference
- Search
- Sampling
- Hybrid of search and inference
- Modeling and learning
- Software
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 may require too much memory
- **Condition** on some of the variables
Inference vs. Conditioning

- **By Inference (thinking)**

  - Exponential in treewidth
  - Time and memory

- **By Conditioning (guessing)**

  - Exponential in cycle-cutset
  - Time-wise, linear memory
Solution Techniques, State of the art

**Search (Conditioning)**
- Complete: DFS search, Branch-and-Bound, A*
- Incomplete: Simulated Annealing, Gradient Descent, Stochastic Local Search

Hybrids
- Complete: Adaptive Consistency, Tree Clustering, Variable Elimination, Resolution
- Incomplete: Belief-propagation, Unit Resolution, Mini-bucket(i)

**AND/OR search**
- Complete: Time: $\exp(\text{treewidth} \times \log n)$, Space: linear
- Incomplete: Time: $\exp(\text{treewidth})$, Space: $\exp(\text{treewidth})$

**Inference (Elimination)**
- Time: $\exp(\text{treewidth})$
- Space: $\exp(\text{treewidth})$
I will present algorithms that are uniformly applicable to both likelihood and optimizations, first.

As time permits, will focus on specific tasks:
Likelihood: belief, probability of evidence

optimization: mpe vs map

Will focus on discrete variables and assume table representation
Road Map

- Overview: Bayesian networks and algorithms
- Exact Inference
  - Bounded-inference
  - Search
  - Sampling
  - Hybrid of search and inference
- Modeling and learning
- Software
Outline

- Introduction
- Inference
  - Exact: Variable elimination, bucket elimination
  - Exact: cluster-tree propagation (join/junction-trees)
- Approximate inference
- Search
- Compilation: AND/OR Decision Diagrams
- Software
"Moral" Graph

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

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

$\sum \prod_b$ Elimination operator

bucket B: $\lambda_B^{B}(a,d,c,e)$

bucket C: $\lambda_C^{C}(a,d,e)$

bucket D: $\lambda_D^{D}(a,e)$

bucket E: $e=0$

$W^* = 4$

“induced width” (max clique size)

$P(a \mid e=0) = \frac{P(a,e=0)}{P(e=0)}$
The operation in a bucket

- Multiplying functions
- Marginalizing (summing-out) functions
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>

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

\[= 0.1 \times 0.8\]
# Factors: Multiplication Operation

<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>$D$</th>
<th>$E$</th>
<th>$f_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>0.448</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>0.192</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>0.112</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>0.248</td>
</tr>
</tbody>
</table>

The result of multiplying the above factors:

<table>
<thead>
<tr>
<th>$B$</th>
<th>$C$</th>
<th>$D$</th>
<th>$E$</th>
<th>$f_1(B, C, D)f_2(D, E)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>$0.4256 = (.95)(.448)$</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>$0.1824 = (.95)(.192)$</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>$0.0056 = (.05)(.112)$</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>$0.2480 = (1)(.248)$</td>
</tr>
</tbody>
</table>

Thanks to Darwiche
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>
</tbody>
</table>
| false| false| 1             

$$
\sum_B \sum_C \sum_D f_1 = 4
$$

Thanks to Darwiche
Bucket Elimination and Induced Width

Ordering: a, e, d, c, b

\[
\begin{align*}
\text{bucket}(B) &= P(e|b,c), P(d|a,b), P(b|a) \\
\text{bucket}(C) &= P(c|a) \parallel \lambda_B(a,c,d,e) \\
\text{bucket}(D) &= \parallel \lambda_C(a,d,e) \\
\text{bucket}(E) &= e = 0 \parallel \lambda_D(a,e) \\
\text{bucket}(A) &= P(a) \parallel \lambda_E(a)
\end{align*}
\]
Bucket Elimination and Induced Width

Ordering: a, b, c, d, e
bucket(E) = \( P(e|b, c), \ e = 0 \)
bucket(D) = \( P(d|a, b) \)
bucket(C) = \( P(c|a) \ || \ P(e = 0|b, c) \)
bucket(B) = \( P(b|a) \ || \ \lambda_D(a, b), \lambda_C(b, c) \)
bucket(A) = \( P(a) \ || \ \lambda_B(a) \)

Ordering: a, e, d, c, b
bucket(B) = \( P(e|b, c), P(d|a, b), P(b|a) \)
bucket(C) = \( P(c|a) \ || \ \lambda_B(a, c, d, e) \)
bucket(D) = \( \ || \ \lambda_C(a, d, e) \)
bucket(E) = \( e = 0 \ || \ \lambda_D(a, c) \)
bucket(A) = \( P(a) \ || \ \lambda_E(a) \)
Induced-width

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

- **Induced width** along ordering $d$, $w^*(d)$:
  - The width in the ordered induced graph, obtained by connecting “parents” of each node $X$, recursively from top to bottom
Induced width (continued)

\[ w^*(d) \] – the induced width of the primal graph along ordering \( d \)

The effect of the ordering:

\[ w^*(d_1) = 4 \]
\[ w^*(d_2) = 2 \]

Finding smallest induced-width is hard!
Greedy algorithms (min-fill) works well.
Significant research area
Input: A belief network \( \{P_1, \ldots, P_n\} \), \( d, e \).
Output: belief of \( X_1 \) given \( e \).

1. Initialize:

2. Process buckets from \( p = n \) to 1
   for matrices \( \lambda_1, \lambda_2, \ldots, \lambda_j \) 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 \( Bel(x_1) = \alpha P(x_1) \cdot \prod_i \lambda_i(x_1) \)
Handling Observations

Observing \( b = 1 \)

Ordering: \( a, e, d, c, b \)

\[
\begin{align*}
\text{bucket}(B) &= P(c|b, c), P(d|a, b), P(b|a), b = 1 \\
\text{bucket}(C) &= P(c|a), \quad \parallel \quad P(e|b = 1, c) \\
\text{bucket}(D) &= \quad \parallel \quad P(d|a, b = 1) \\
\text{bucket}(E) &= e = 0 \quad \parallel \quad \lambda_{C}(e, a) \\
\text{bucket}(A) &= P(a), \quad \parallel \quad P(b = 1|a) \quad \lambda_{D}(a), \lambda_{E}(e, a)
\end{align*}
\]

Ordering: \( a, b, c, d, e \)

\[
\begin{align*}
\text{bucket}(E) &= P(e|b, c), \quad e = 0 \\
\text{bucket}(D) &= P(d|a, b) \\
\text{bucket}(C) &= P(c|a) \quad \parallel \quad \lambda_{E}(b, c) \\
\text{bucket}(B) &= P(b|a), b = 1 \quad \parallel \quad \lambda_{D}(a, b), \lambda_{C}(a, b) \\
\text{bucket}(A) &= P(a) \quad \parallel \quad \lambda_{B}(a)
\end{align*}
\]
Search vs. Inference

Search (conditioning)

A

B

C

D

E

F

G

A=1

A=k

\( A \)

B

D

C

F

G

E

\( k \) “sparser” problems

Inference (elimination)

A

B

D

C

F

G

E

1 “denser” problem

A

D

C

F

G

E

\[ \text{A} = k \]
Finding \( \text{MPE} = \max_x P(\bar{x}) \)

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

\[
\sum \text{ is replaced by } \max : \\
\text{MPE} = \max_{a,e,d,c,b} P(a)P(c | a)P(b | a)P(d | a,b)P(e | b, c)
\]
Finding \[ \text{MPE} = \max \frac{P(\overline{x})}{x} \]

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

\[
\sum \quad \text{is replaced by} \quad \max : \\
\text{MPE} = \max_{a,e,d,c,b} P(a)P(c \mid a)P(b \mid a)P(d \mid a,b)P(e \mid b,c)
\]

\[
\text{Elimination operator}
\]

bucket B: \quad P(b \mid a) \quad P(d \mid b,a) \quad P(e \mid b,c)

bucket C: \quad P(c \mid a) \quad h^B (a,d,c,e)

bucket D: \quad h^C (a,d,e)

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

bucket A: \quad P(a) \quad h^E (a)

\text{MPE}

\text{"induced width" (max clique size)}

W*=4
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. \( b' = \arg \max_b P(b | a') \times P(d' | b, a') \times P(e' | b, c') \)

\[ B: \quad P(b | a) \quad P(d | b, a) \quad P(e | b, c) \]

\[ C: \quad P(c | a) \quad h^B(a, d, c, e) \]

\[ D: \quad h^c(a, d, e) \]

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

\[ A: \quad P(a) \quad h^E(a) \]

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**

| H | P(H) | F | P(F) | M | P(M|H,F) |
|---|------|---|------|---|---------|
| 0 | .9   | 0 | .99  | 1 | .05     |
| 1 | .1   | 0 | .01  | 1 | .1      |

<table>
<thead>
<tr>
<th>F</th>
<th>P(F)</th>
<th>h_3(F)</th>
<th>F</th>
<th>h_4(F)</th>
<th>F</th>
<th>P(F,B=1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>.99</td>
<td>.1245</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>.123255</td>
</tr>
<tr>
<td>1</td>
<td>.01</td>
<td>.73175</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>.073175</td>
</tr>
</tbody>
</table>

\[
P(h,f,r,m,b) = P(h) P(f) P(m|h,f) P(r|f) P(b|m)
\]

\[
P(F | B=1) = ?
\]

P(B=1) = .19643

\[
P(F=1|B=1) = .3725
\]

Probability of evidence

Updated belief
Relationship with Pearl’s belief propagation on poly-trees (Pearl 1988)

Pearl’s belief propagation for single-root query

BE-bel using topological ordering

On a trees induced-width is 1: message-passing is linear.
On poly-tree width = induced-width, message-passing is linear.
But message propagation can go both ways
Propagation in both directions

- Messages can propagate both ways and we get beliefs for each variable
Outline

- Introduction
- Inference
  - Exact: Variable elimination, bucket elimination
  - cluster-tree propagation (join/junction-trees)
- Approximate:
- Search
- Compilation: AND/OR Decision Diagrams
- Software
From Bucket elimination to bucket-tree elimination

If we want the marginal on D?

Bucket G: $P(G|F)$
Bucket F: $P(F|B,C) \xrightarrow{\lambda^F_G(F)}$
Bucket D: $P(D|A,B)$
Bucket C: $P(C|A) \xrightarrow{\lambda^C_F(B,C)}$
Bucket B: $P(B|A) \xrightarrow{\lambda^B_D(A,B)} \xrightarrow{\lambda^B_C(A,B)}$
Bucket A: $P(A) \xrightarrow{\lambda^A_B(A)}$
BTE: allows messages both ways

Each bucket can compute its marginal probability

$$\frac{\lambda^A_F(B,C)}{\lambda^B_G(B,C)} \frac{\pi^D_D(A,B)}{\lambda^C_B(A,B)} \frac{\pi^F_F(F)}{\lambda^E_G(F)}$$
Same Message Passing rule up and down

\[
\begin{align*}
\text{bucket}(u) &= P(u) \cup \{ \lambda(x_1, u), \lambda(x_2, u), \ldots, \lambda(x_n, u), \lambda(v, u) \} \\
\lambda(u, v) &= \sum_{\text{elim}(u, v)} \prod_{f \in \text{bucket}(u) - \{\lambda(v, u)\}} f \\
\text{Elim}(u,v) &= \text{cluster}(u) - \text{sep}(u,v)
\end{align*}
\]
From a bucket-tree to a join-tree

- Merge non-maximal buckets into maximal clusters.
- Connect clusters into a tree: each cluster to one with which it shares a largest subset of variables.
- Separators are variable-intersection on adjacent clusters.

A super-bucket-tree is an i-map of the Bayesian network.
The general tree-decomposition

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

\[ \text{treewidth} = 4 - 1 = 3 \]
\[ \text{treewidth} = (\text{maximum cluster size}) - 1 \]
The general Message Passing on a general tree-decomposition

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

For max-product
Just replace \( \Sigma \) With max.

Compute the message:

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

\( \text{Elim}(u, v) = \text{cluster}(u) \setminus \text{sep}(u, v) \)
A tree decomposition for a belief network $BN = \langle X,D,G,P \rangle$ 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)
Tree Decomposition for belief updating

- **A**: $p(a)$
- **B**: $p(b|a)$
- **C**: $p(c|a,b)$
- **D**: $p(d|b)$
- **E**: $p(e|b,f)$
- **F**: $p(f|c,d)$
- **G**: $p(g|e,f)$

Parents:
- A: B, C
- B: A, C
- C: A, B
- D: B
- E: B, D
- F: C, D
- G: E

Children:
- A: None
- B: A
- C: B
- D: B
- E: B, D
- F: C, D
- G: E

Conditional probabilities:
- $p(a)$
- $p(b|a)$
- $p(c|a,b)$
- $p(d|b)$
- $p(e|b,f)$
- $p(f|c,d)$
- $p(g|e,f)$

Tree structure:
- A (Root)
  - B
    - C
      - D
        - E
          - F
            - G
CTE: Cluster Tree Elimination

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

For each cluster $P(X|e)$ is computed, also $P(e)$

1. ABC
   $h_{(1,2)}(b, c) = \sum_a p(a) \cdot p(b | a) \cdot p(c | a, b)$

2. BCDF
   $h_{(2,1)}(b, c) = \sum_{d,f} p(d | b) \cdot p(f | c,d) \cdot h_{(3,2)}(b, f)$
   $h_{(2,3)}(b, f) = \sum_{c,d} p(d | b) \cdot p(f | c,d) \cdot h_{(1,2)}(b, c)$
   $h_{(3,2)}(b, f) = \sum_e p(e | b, f) \cdot h_{(4,3)}(e, f)$

3. BEF
   $h_{(3,4)}(e, f) = \sum_b p(e | b, f) \cdot h_{(2,3)}(b, f)$
   $h_{(4,3)}(e, f) = p(G = g_e | e, f)$

4. EFG

Figure 1: Example of Cluster Tree Elimination
Algorithm cluster-tree elimination (CTE)

Input: A tree decomposition $< T, \chi, \psi >$ for a problem $M = < X, D, F, \Pi >$, $X = \{X_1, \ldots, X_n\}$, $F = \{f_1, \ldots, f_r\}$.

Output: An augmented tree whose vertices are clusters containing the original functions as well as messages received from neighbors. A solution computed from the augmented clusters.

Compute messages:

For every edge $(u, v)$ in the tree, do

- Let $m_{(u,v)}$ denote the message sent by vertex $u$ to vertex $v$.
- Let $\text{cluster}(u) = \psi(u) \cup \{m_{(i,u)}|(i, u) \in T\}$.
- If vertex $u$ has received messages from all adjacent vertices other than $v$, then compute and send to $v$,

$$ m_{(u,v)} = \sum_{\text{sep}(u,v)} \prod_{f \in \text{cluster}(u), f \neq m_{(v,u)}} f $$

Endfor

Note: functions whose scope does not contain elimination variables do not need to be processed, and can instead be directly passed on to the receiving vertex.

Return: A tree-decomposition augmented with messages, and for every $v \in T$
Let $C_i$ and $C_j$ two adjacent clusters and $\text{sep}(i,j)$ be their separator

$$\text{bel}(\text{sep}) = \sum_{e \in \text{lim}(i,j)} \prod_{f \in C_i} f = \sum_{e \in \text{lim}(j,i)} \prod_{f \in C_j} f = h_{(i,j)} \cdot h_{(j,i)}$$
Examples of tree-clustering
CTE - properties

- **Correctness and completeness**: Algorithm CTE is correct, i.e. it computes the exact joint probability in each cluster and therefore of every single variable and the evidence.

- **Time complexity**: \( O(\; \text{deg} \times (n+N) \times k^{w^*+1}\; ) \)

- **Space complexity**: \( O(\; N \times k^{\text{sep}}\; ) \)

where

- \( \text{deg} \) = the maximum degree of a node in the cluster-tree
- \( n \) = number of variables (= number of CPTs)
- \( N \) = number of nodes in the tree decomposition
- \( k \) = the maximum domain size of a variable
- \( w^* \) = the induced width
- \( \text{sep} \) = the separator size
Road Map

- Overview: Bayesian networks and algorithms
- Exact Inference
- **Bounded-inference**
- Search
- Sampling
- Hybrid of search and inference
- Modeling and learning
- Software
Road Map

- Overview: Bayesian networks and algorithms
- Exact Inference
- Bounded-inference
  - Mini-buckets, mini-clusters
  - Belief propagation, Generalized belief propagation

Search
- Sampling

Hybrid of Search and Inference

Modeling and Learning

Software
- Complete
- Incomplete
  - Simulated Annealing
  - Gradient Descent
  - Stochastic

Hybrids
- Complete
- Incomplete
  - Local Consistency
  - Unit Resolution
  - Mini-bucket(s)

Complete
- DFS search
- Branch-and-Bound
- A*

Hybrids
- Adaptive Consistency
- Tree Clustering
- Variable Elimination
- Resolution
The idea of Mini-bucket (Dechter and Rish 1997)

Local computation: bound the size of recorded dependencies

Split a bucket into mini-buckets => bound complexity

\[
bucket (X) = \{ h_1, \ldots, h_r, h_{r+1}, \ldots, h_n \} \\
h^X = \max_X \prod_{i=1}^n h_i \\
\{ h_1, \ldots, h_r \} \quad \{ h_{r+1}, \ldots, h_n \} \\
g^X = (\max_X \prod_{i=1}^r h_i) \cdot (\max_X \prod_{i=r+1}^n h_i) \\
\]

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

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

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

\[\max_B \prod \]

\[\max_B \prod\]

\[\max_B \prod\]

\[\max_B \prod\]

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: `approx-mpe(3)` versus `elim-mpe`

- **Example Diagram:**
  - Mini-buckets: $\max_B$ 
    - $P(\text{elb,c})$ 
    - $P(\text{dla,b})P(\text{bla})$ 
    - $P(\text{cla})$ 
    - $h^B(e,c)$ 
    - $h^B(d,a)$ 
    - $h^C(e,a)$ 
    - $P(\text{a})$ 
    - $h^E(a)$ 
    - $h^D(a)$
  - Max variables in a mini-bucket: $3$
  - $E = 0$
  - $h^C(e,a)$
  - $h^E(a)$
  - $h^D(a)$
  - $U = Upper\ bound\ (MPE)$

- **Approx-mpe(3):**
  - $w^* = 2$

- **Elim-mpe:**
  - $w^* = 4$
Properties of MBE(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

\[ \text{anytime mpe}(\varepsilon) \]

\textbf{Initialize} \( i = i_0 \)
\textbf{While} time and space resources are available
    \( i \leftarrow i + i_{\text{step}} \)
    \( U \leftarrow \text{upper bound computed by } \text{approx-mpe}(i) \)
    \( L \leftarrow \text{lower bound computed by } \text{approx-mpe}(i) \)
    keep the best solution found so far
    \textbf{if} \( 1 \leq \frac{U}{L} \leq 1 + \varepsilon \), return solution
\textbf{end}
\textbf{return} the largest \( L \) and the smallest \( U \)
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.
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>\textit{elim-mpe}</td>
<td>115.8</td>
<td>1697.6</td>
</tr>
<tr>
<td>anytime-mpe(\varepsilon), \varepsilon = 10^{-4}</td>
<td>70.3</td>
<td>505.2</td>
</tr>
<tr>
<td>anytime-mpe(\varepsilon), \varepsilon = 10^{-1}</td>
<td>70.3</td>
<td>110.5</td>
</tr>
</tbody>
</table>
Mini-Clustering (for sum-product)

Split a cluster into mini-clusters  =>  bound complexity

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

**APPROXIMATE algorithm**

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

Exponential complexity decrease

\[ O(e^n) \rightarrow O(e^{\text{var}(r)}) + O(e^{\text{var}(n-r)}) \]
Mini-Clustering, i-bound=3

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

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

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

**APPROXIMATE algorithm**

**Time and space:**

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

Number of variables in a mini-cluster
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

Grid 15x15, evid=10, w*=22, 10 instances

NHD

Absolute error

Relative error

Time (seconds)
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

- Overview: Bayesian networks and algorithms
- Exact Inference
- Bounded-inference
  - Mini-buckets, mini-clusters
  - Belief propagation, Generalized belief propagation

- Search
- Sampling
- Hybrid of search and inference
- Modeling and learning
- Software Complete
  - Simulated Annealing
  - Gradient Descent
  - Stochastic

- Incomplete
  - Local Consistency
  - Unit Resolution
  - Mini-bucket(s)
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--IJGP
BP works on dual graph

- Need a slide saying the belief propagation operates on the dual graph
IJGP - Example

Belief network

Loopy BP graph
Arcs labeled with any single variable should form a **TREE**
Collapsing Clusters
Join-Graphs

more accuracy

less complexity
Belief Propagation

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

Compute the message:

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

For max-product: IJGP replaces summation with maximization
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}

\[ h_{(1,2)}(de) = \sum_{a,b,c} p(a) p(c) p(b \mid ac) p(d \mid abe) p(e \mid bc) h_{(3,1)}(bc) \]

\[ h_{(1,2)}(cde) = \sum_{a,b} p(a) p(c) p(b \mid ac) p(d \mid abe) p(e \mid bc) h_{(3,1)}(bc) \]
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

σ

σ
Coding Networks – Bit Error Rate

N=400, 1000 instances, 30 it, \( w^* = 43 \), \( \sigma = 0.22 \)

N=400, 500 instances, 30 it, \( w^* = 43 \), \( \sigma = 0.32 \)

N=400, 500 instances, 30 it, \( w^* = 43 \), \( \sigma = 0.51 \)

N=400, 500 instances, 30 it, \( w^* = 43 \), \( \sigma = 0.65 \)
CPCS 422 – KL Distance

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

KL distance
0.0001
0.001
0.01
0.1

IJP 30 it (at convergence)
MC
IBP 10 it (at convergence)

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

KL distance
0.0001
0.001
0.01
0.1

IJP at convergence
MC
IBP at convergence

evidence=0
evidence=30
CPCS 422 – KL vs. Iterations

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

- KL distance
  - 0.0001
  - 0.001
  - 0.01
  - 0.1

- IJGP (3)
- IJGP(10)
- IBP

number of iterations: 0 5 10 15 20 25 30 35

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

- KL distance
  - 1

- IJGP(3)
- IJGP(10)
- IBP

number of iterations: 0 5 10 15 20 25 30 35

evidence=0
evidence=30
More On the Power of Belief Propagation

- BP as local minima of KL distance
- BP’s power from constraint propagation perspective.
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 $P_{\delta'}$ whose factorizations are more expressive than that of the polytree-based factorization.

These results also extend to generalized BP.
Summary of IJGP so far

A spectrum of approximations.

IBP: results from applying IJGP to the dual joinigraph.

Jointree algorithm: results from applying IJGP to a jointree (as a joinigraph).

In between these two ends, we have a spectrum of joinigraphs and corresponding factorizations, where IJGP seeks stationary points of the KL–divergence between these factorizations and the original distribution.
Constraint networks

Map coloring

Variables: countries (A B C etc.)
Values: colors (red green blue)
Constraints: \( A \neq B, A \neq D, D \neq E, \text{ etc.} \)
Arc-consistency

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

Primal

A

1  2  3

A < B

1  2  3

A < D

1  2

2  3

B

1  2  3

B = C

1  1

2  2

3  3

C

1  2  3

D

1  2  3

D < C

1  2

2  3

Dual

AB

1  2  2  2

2  3  3  3

BC

1  1

2  2

3  3

AD

1  2

2  3

DC

1  2  2  3

BC

2  2  1  2

3  3  2  3

DC

1  2  2  3

AB

1  2

2  3

AD

1  2

2  3

BC

1  1

2  2

3  3

D < C

1  2

2  3

B = C

1  1

2  2

3  3

A < B

1  2

2  3

A < D

1  2

2  3

D < C

1  2

2  3
Flattening the Bayesian Network

Belief network

Flat constraint network
**Theorem:**

Trace of zero beliefs of Iterative Belief Propagation = Trace of invalid tuples of arc-consistency on flat network

**Soundness:**

- The inference of zero beliefs by Loopy BP converges in a finite number of iterations
- All the inferred zero beliefs are correct

**Incompleteness:**

- Loopy BP may not infer all the true zero beliefs
Properties of \textit{ijgp}

- Properties of the sum-product algorithm
  - If/when the algorithm converges, the convergence is a stationary point of the KL distance to the posterior distribution

- Properties of the max-product algorithm
  - If the max-marginals agree...
IJGP summary

- IJGP borrows the iterative feature from IBP and the anytime virtues of bounded inference from MC

- Empirical evaluation showed the potential of IJGP, which improves with iteration and most of the time with i-bound, and scales up to large networks

- IJGP is almost always superior, often by a high margin, to IBP and MC

- Based on all our experiments, we think that IJGP provides a practical breakthrough to the task of belief updating
Exact Reasoning by Search

- Why consider search?
- Can we do any better in search?
- Can we combine search and inference?
Road Map

- Overview: Bayesian networks and algorithms
- Exact Inference
- Bounded-inference
- Search
- Sampling
- Hybrid of search and inference
- Modeling and learning
- Software
Conditioning generates the probability tree

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

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

![Diagram of a search space with nodes labeled A, B, C, D, E, and F. Each node has a binary value (0 or 1) attached to it. The diagram shows the possible paths through the search space, with each path representing a search.](image)
The AND/OR Search Tree

Pseudo tree (Freuder and Quinn, IJCAI'85)
A solution subtree is \((A=0, B=1, C=0, D=0, E=1, F=1)\)
Weighted AND/OR Tree

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

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

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

**Evidence: E=0**

**Evidence: D=1**
AND/OR vs. OR Spaces

54 nodes

126 nodes
### 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
### 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>


- $d$ = domain size
- $t$ = depth of pseudo-tree
- $n$ = number of variables
- $w^*$ = treewidth
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
**Pseudo-Trees**

(Freuder 85, Bayardo 95, Bodlaender and Gilbert, 91)

(a) Graph

(b) DFS tree
   depth=3

(c) pseudo-tree
   depth=2

(d) Chain
   depth=6

\[ t \leq w^* \log n \]
AND/OR tree search \( \{P(\text{evidence})\} \)

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

**Value of node** = updated belief for subproblem below

---

### Evidence: E=0

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

---

### Evidence: D=1

**Result:** \( P(E=0) \)

---

### Evidence: B=0

**Result:** \( P(A) \)

---

### Evidence: A=0

**Result:** \( P(B) \)

---

### Evidence: B=0

**Result:** \( P(A) \)

---

### Evidence: B=1

**Result:** \( P(A) \)

---

### Evidence: D=0

**Result:** \( P(C) \)

---

### Evidence: D=1

**Result:** \( P(C) \)

---

### Evidence: C=0

**Result:** \( P(A) \)

---

### Evidence: C=1

**Result:** \( P(A) \)

---

### Evidence: E=0

**Result:** \( P(D) \)

---

### Evidence: E=1

**Result:** \( P(D) \)

---

### Evidence: C=0

**Result:** \( P(B) \)

---

### Evidence: C=1

**Result:** \( P(B) \)

---

### Evidence: B=0

**Result:** \( P(A) \)

---

### Evidence: B=1

**Result:** \( P(A) \)

---

### Evidence: C=0

**Result:** \( P(B) \)

---

### Evidence: C=1

**Result:** \( P(B) \)

---

### Evidence: E=0

**Result:** \( P(D) \)

---

### Evidence: E=1

**Result:** \( P(D) \)

---

### Evidence: D=0

**Result:** \( P(C) \)

---

### Evidence: D=1

**Result:** \( P(C) \)

---

### Evidence: E=0

**Result:** \( P(D) \)

---

### Evidence: E=1

**Result:** \( P(D) \)

---

### Evidence: C=0

**Result:** \( P(B) \)

---

### Evidence: C=1

**Result:** \( P(B) \)

---

### Evidence: B=0

**Result:** \( P(A) \)

---

### Evidence: B=1

**Result:** \( P(A) \)

---

### Evidence: C=0

**Result:** \( P(B) \)

---

### Evidence: C=1

**Result:** \( P(B) \)

---

### Evidence: E=0

**Result:** \( P(D) \)

---

### Evidence: E=1

**Result:** \( P(D) \)

---

### Evidence: D=0

**Result:** \( P(C) \)

---

### Evidence: D=1

**Result:** \( P(C) \)

---

### Evidence: E=0

**Result:** \( P(D) \)

---

### Evidence: E=1

**Result:** \( P(D) \)

---

### Evidence: C=0

**Result:** \( P(B) \)

---

### Evidence: C=1

**Result:** \( P(B) \)

---

### Evidence: B=0

**Result:** \( P(A) \)

---

### Evidence: B=1

**Result:** \( P(A) \)

---

### Evidence: C=0

**Result:** \( P(B) \)

---

### Evidence: C=1

**Result:** \( P(B) \)
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**

From AND/OR Tree

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

AND

OR
An AND/OR Graph
Merging based on context

context (X) = 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)*

### P(E | A, B)

<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>

### P(B | A)

<table>
<thead>
<tr>
<th>A</th>
<th>B=0</th>
<th>B=1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>.4</td>
<td>.6</td>
</tr>
<tr>
<td>1</td>
<td>.1</td>
<td>.9</td>
</tr>
</tbody>
</table>

### P(C | A)

<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>

### P(A)

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

---

**Evidence:** E=0

---

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

---

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

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

---

**Context**

- [ ]
- [A]
- [AB]
- E
- C
- [AB]
- D
- [BC]

---

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

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

*Value of node = updated belief for sub-problem below*
AND/OR Graph DFS Algorithm
(Belief Updating)

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

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

\[ P(B|A) \]

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

\[ P(C|A) \]

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

\[ P(A) \]

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

Evidence: \( E=0 \)

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

Cache table for \( D \)

Context

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>$O(n \cdot d^{w*})$</td>
<td>$O(n \cdot d^{pw*})$</td>
</tr>
<tr>
<td><strong>Time</strong></td>
<td>$O(n \cdot d^{w*})$</td>
<td>$O(n \cdot d^{pw*})$</td>
</tr>
</tbody>
</table>

$d = \text{domain size}$  
$n = \text{number of variables}$  
$w^* = \text{treewidth}$  
$pw^* = \text{pathwidth}$  
$w^* \leq pw^* \leq w^* \log n$  

\[w^* \leq pw^* \leq w^* \log n\]
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>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>width</td>
<td>depth</td>
<td>width</td>
<td>depth</td>
</tr>
<tr>
<td>barley</td>
<td>7</td>
<td>13</td>
<td>7</td>
<td>23</td>
</tr>
<tr>
<td>diabetes</td>
<td>7</td>
<td>16</td>
<td>4</td>
<td>77</td>
</tr>
<tr>
<td>link</td>
<td>21</td>
<td>40</td>
<td>15</td>
<td>53</td>
</tr>
<tr>
<td>mildew</td>
<td>5</td>
<td>9</td>
<td>4</td>
<td>13</td>
</tr>
<tr>
<td>munin1</td>
<td>12</td>
<td>17</td>
<td>12</td>
<td>29</td>
</tr>
<tr>
<td>munin2</td>
<td>9</td>
<td>16</td>
<td>9</td>
<td>32</td>
</tr>
<tr>
<td>munin3</td>
<td>9</td>
<td>15</td>
<td>9</td>
<td>30</td>
</tr>
<tr>
<td>munin4</td>
<td>9</td>
<td>18</td>
<td>9</td>
<td>30</td>
</tr>
<tr>
<td>water</td>
<td>11</td>
<td>16</td>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>pigs</td>
<td>11</td>
<td>20</td>
<td>11</td>
<td>26</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Network</th>
<th>hypergraph</th>
<th>min-fill</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>width</td>
<td>depth</td>
<td>width</td>
<td>depth</td>
</tr>
<tr>
<td>spot5</td>
<td>47</td>
<td>152</td>
<td>39</td>
<td>204</td>
</tr>
<tr>
<td>spot28</td>
<td>108</td>
<td>138</td>
<td>79</td>
<td>199</td>
</tr>
<tr>
<td>spot29</td>
<td>16</td>
<td>23</td>
<td>14</td>
<td>42</td>
</tr>
<tr>
<td>spot42</td>
<td>36</td>
<td>48</td>
<td>33</td>
<td>87</td>
</tr>
<tr>
<td>spot54</td>
<td>12</td>
<td>16</td>
<td>11</td>
<td>33</td>
</tr>
<tr>
<td>spot404</td>
<td>19</td>
<td>26</td>
<td>19</td>
<td>42</td>
</tr>
<tr>
<td>spot408</td>
<td>47</td>
<td>52</td>
<td>35</td>
<td>97</td>
</tr>
<tr>
<td>spot503</td>
<td>11</td>
<td>20</td>
<td>9</td>
<td>39</td>
</tr>
<tr>
<td>spot505</td>
<td>29</td>
<td>42</td>
<td>23</td>
<td>74</td>
</tr>
<tr>
<td>spot507</td>
<td>70</td>
<td>122</td>
<td>59</td>
<td>160</td>
</tr>
</tbody>
</table>

Bayesian Networks Repository

SPOT5 Benchmarks
All Four Search Spaces

Full OR search tree
126 nodes

Context minimal OR search graph
28 nodes

Full AND/OR search tree
54 AND nodes

Context minimal AND/OR search graph
18 AND nodes
How Big Is The Context?

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

\[
\text{max context size} = \text{treewidth}
\]
AO(j): searches depth-first, cache i-context

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

Space: $O(n)$
Time: $O(\exp(w* \log n))$
Search for MPE/MAP problem

- Searching the AND?OR space by
  - Branch and bound
  - Best-first
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

![Diagram of DF Branch-and-Bound](attachment:image1.png)

![Diagram of Best-First Search](attachment:image2.png)
**AND/OR Branch-and-Bound (AOBB)**
(Marinescu & Dechter, IJCAI’05)

- **Bound (AOBB)**
- **OR Branch**
- **and**
- **Bound**

\[ \text{lb}(n) = g(n) + h(n) \]

Maintain
\[ \text{ub} = \text{best solution found so far} \]

Prune subtree below \( n \) if \( \text{lb}(n) \geq \text{ub} \)

- \( g(n) \)
- \( h(n) \)

Estimates the optimal cost below \( n \)
Split a bucket into mini-buckets => bound complexity

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

\[
h^X = \min_{X} \sum_{i=1}^{n} h_i
\]

\[
g^X = \left( \min_{X} \sum_{i=1}^{r} h_i \right) + \left( \min_{X} \sum_{i=r+1}^{n} h_i \right)
\]

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

Exponential complexity decrease: \( O(e^n) \rightarrow O(e^r) + O(e^{n-r}) \)
Mini-bucket Heuristics for BB search

(Kask and dechter AIJ, 2001, Kask, Dechter and Marinescu UAI 2003)

\[ f(a, e, D) = P(a) \cdot h^B(D, a) \cdot h^C(e, a) \]
AND/OR Branch-and-Bound (contd.)

\[ h(n) \geq ub(n) \]
AND/OR Branch and Bound for Constraint Optimization

- Search AND/OR Context-minimal graph
  - exploit decomposition and equivalence

- Prune irrelevance via mini-bucket heuristics, and constraint propagation

- Depth-first (AOBB) and best-first (AOBF)

- Dynamic variable orderings

- Applied to MPE and weighted CSPs

- Applied to Integer Programming
Experiments

- **Benchmarks**
  - Belief Networks (BN)
  - Weighted CSPs (WCSP)

- **Algorithms**
  - AOB-C – AND/OR Branch-and-Bound w/ caching
  - AOBF-C – Best-first AND/OR Search
  - Samlam
  - Superlink
  - Toolbar (DFBB+EDAC), Toolbar-BTD (BTD+EDAC)

- **Heuristics**
  - Mini-Bucket heuristics
## Genetic Linkage Analysis

<table>
<thead>
<tr>
<th>Pedigree</th>
<th>SamIam Superlink</th>
<th>BB-C+SMB(i)</th>
<th>BB-C+SMB(i)</th>
<th>BB-C+SMB(i)</th>
<th>BB-C+SMB(i)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>AOBB+SMB(i)</td>
<td>AOBB+SMB(i)</td>
<td>AOBB+SMB(i)</td>
<td>AOBB+SMB(i)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AOBB-C+SMB(i)</td>
<td>AOBB-C+SMB(i)</td>
<td>AOBB-C+SMB(i)</td>
<td>AOBB-C+SMB(i)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AOBF-C+SMB(i)</td>
<td>AOBF-C+SMB(i)</td>
<td>AOBF-C+SMB(i)</td>
<td>AOBF-C+SMB(i)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>i=12</td>
<td>i=14</td>
<td>i=16</td>
<td>i=18</td>
</tr>
<tr>
<td></td>
<td></td>
<td>time</td>
<td>time</td>
<td>time</td>
<td>time</td>
</tr>
<tr>
<td></td>
<td></td>
<td>nodes</td>
<td>nodes</td>
<td>nodes</td>
<td>nodes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pedigree</th>
<th>time</th>
<th>nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ped30</td>
<td>13095.83</td>
<td>93,233,570</td>
</tr>
<tr>
<td></td>
<td>10212.70</td>
<td>82,552,957</td>
</tr>
<tr>
<td>ped33</td>
<td>2804.61</td>
<td>34,229,495</td>
</tr>
<tr>
<td></td>
<td>1426.99</td>
<td>2,504,020</td>
</tr>
<tr>
<td>ped42</td>
<td>561.31</td>
<td>2364.67</td>
</tr>
<tr>
<td></td>
<td>22,595,247</td>
<td></td>
</tr>
</tbody>
</table>

Min-fill pseudo tree. Time limit 3 hours.
Algorithms for AND/OR Space are currently superior

- **Back-jumping** for CSPs  
  (Gaschnig 1977), (Dechter 1990), (Prosser, Bayardo and Mirankar, 1995)

- **Pseudo-search re-arrangement**, for any CSP task  
  (Freuder and Quinn 1985)

- **Pseudo-tree search for soft constraints**  
  (Larrosa, Meseguer and Sanchez, 2002)

- **Recursive Conditioning**  
  (Darwiche, 2001), explores the AND/OR tree or graph for any query

- **BTD: Searching tree-decompositions** for optimization  
  (Jeagou and Terrioux, 2004)

- **Value Elimination**  
  (Bacchus, Dalmao and Pittasi, 2003)
Road Map

- Overview: Bayesian networks and algorithms
- Exact Inference
- Bounded-inference
- Search
- Hybrid of search and inference
- Sampling
- Modeling and learning
- Software
 Conditioning generates the probability tree

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

Complexity of conditioning: exponential time, linear space
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)

Graph Coloring problem

- Inference may require too much memory
- **Condition** on some of the variables
Variable elimination with conditioning; w-cutset algorithms

- Identify an w-cutset $c_w$ of the network
- For each assignment to the cutset $c_w$ solve the conditioned sub-problem by $CTE$
- Aggregate the solutions over all $c_w$ assignments.
- Time complexity: $O(k^{c_w+w})$
- Space complexity: $O(k^\omega)$
- What w should we use?
  - $W=1$? $W=0$? $W=w^*$
  - Depends on the graph
  - Practice: use the largest $w$ allowed by space
- Alternate conditioning and elimination?
Time vs Space for w-cutset

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

Branch and bound

Bucket elimination

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

Space $O(\exp(w))$
Road Map

- Overview: Bayesian networks and algorithms
- Exact Inference
- Bounded-inference
- Search
- Sampling
- Hybrid of search and inference
- Modeling and learning
- Software
Sampling: Approximation of Search

1. Importance Sampling
2. Markov Chain Monte Carlo: Gibbs Sampling
3. Sampling in presence of Determinism
4. Rao-Blackwellisation
5. AND/OR importance sampling

See: Sampling Techniques for Probabilistic and Deterministic Graphical models PDF Tutorial, AAAI 2010, Atlanta, GA, July 12, 2010:
http://www.ics.uci.edu/~dechter/talks.html
Sampling for Probability Inference

- Logic Sampling
- Importance Sampling
  - Likelihood Sampling
  - Choosing a Proposal Distribution
- Markov Chain Monte Carlo (MCMC)
  - Metropolis-Hastings
  - Gibbs sampling
- Variance Reduction
Input: Bayesian network

\[ X = \{X_1, \ldots, X_N\}, \text{N- #nodes, T - # samples} \]

Output: T samples

Process nodes in topological order – first process the ancestors of a node, then the node itself:

1. For \( t = 0 \) to \( T \)
2. For \( i = 0 \) to \( N \)
3. \( X_i \leftarrow \) sample \( x_i^t \) from \( P(x_i \mid \text{pa}_i) \)
Logic sampling (example)

\[ P(X_1, X_2, X_3, X_4) = P(X_1) \times P(X_2 \mid X_1) \times P(X_3 \mid X_1) \times P(X_4 \mid X_2, X_3) \]

\[ \text{No Evidence} \]

// generates sample \( k \)

1. Sample \( x_1 \) from \( P(x_1) \)
2. Sample \( x_2 \) from \( P(x_2 \mid X_1 = x_1) \)
3. Sample \( x_3 \) from \( P(x_3 \mid X_1 = x_1) \)
4. Sample \( x_4 \) from \( P(x_4 \mid X_2 = x_2, X_3 = x_3) \)
Logic Sampling w/ Evidence

Input: Bayesian network
\[ X = \{X_1, \ldots, X_N\}, \text{ N- #nodes} \]
\[ E \text{ – evidence, T - # samples} \]

Output: T samples consistent with E

1. For t=1 to T
2. For i=1 to N
3. \[ X_i \leftarrow \text{sample } x_i^t \text{ from } P(x_i \mid pa_i) \]
4. If \( X_i \text{ in } E \) and \( X_i \neq x_i \), reject sample:
5. Goto Step 1.
Logic Sampling (example)

Evidence: $X_3 = 0$

// generates sample $k$
1. Sample $x_1$ from $P(x_1)$
2. Sample $x_2$ from $P(x_2 \mid x_1)$
3. Sample $x_3$ from $P(x_3 \mid x_1)$
4. If $x_3 \neq 0$, rejectsample and start from 1, otherwise
5. Sample $x_4$ from $P(x_4 \mid x_2, x_3)$
Monte Carlo Estimate

**Estimator:**
- An estimator is a function of the samples.
- It produces an estimate of the unknown parameter of the sampling distribution.

Given i.i.d. samples $S^1, S^2, \ldots S^T$ drawn from $P$, the Monte Carlo estimate of $E_P[g(x)]$ is given by:

$$\hat{g} = \frac{1}{T} \sum_{t=1}^{T} g(S^t)$$
Example: Monte Carlo estimate

- Given:
  - A distribution \( P(X) = (0.3, 0.7) \).
  - \( g(X) = 40 \) if \( X \) equals 0
    = 50 if \( X \) equals 1.
- Estimate \( E_P[g(x)] = (40 \times 0.3 + 50 \times 0.7) = 47 \).
- Generate \( k \) samples from \( P \): 0,1,1,1,0,1,1,0,1,0

\[
\hat{g} = \frac{40 \times \text{samples}(X = 0) + 50 \times \text{samples}(X = 1)}{\# \text{samples}}
\]
\[
= \frac{40 \times 4 + 50 \times 6}{10} = 46
\]
Importance sampling: Main idea

- Express query as the expected value of a random variable w.r.t. to a distribution Q.
- Generate random samples from Q.
- Estimate the expected value from the generated samples using a monte carlo estimator (average).
Importance sampling for $P(e)$

Let $Z = X \setminus E$,

Let $Q(Z)$ be a (proposal) distribution, satisfying

$$P(z, e) > 0 \Rightarrow Q(z) > 0$$

Then, we can rewrite $P(e)$ as:

$$P(e) = \sum_z P(z, e) = \sum_z P(z, e) \frac{Q(z)}{Q(z)} = E_Q \left[ \frac{P(z, e)}{Q(z)} \right] = E_Q[w(z)]$$

Monte Carlo estimate:

$$\hat{P}(e) = \frac{1}{T} \sum_{t=1}^{T} w(z^t), \text{ where } z^t \leftarrow Q(Z)$$
Likelihood Weighting
(Fung and Chang, 1990; Shachter and Peot, 1990)

Is an instance of importance sampling!

“Clamping” evidence + logic sampling + weighing samples by evidence likelihood

Works well for likely evidence!
Likelihood Weighting: Sampling

Sample in topological order over $\mathbf{X}$!

Clamp evidence, Sample $x_i \leftarrow P(X_i|\text{pa}_i)$, $P(X_i|\text{pa}_i)$ is a look-up in CPT!
Likelihood Weighting: Proposal Distribution

\[ Q(X \setminus E) = \prod_{x_i \in X \setminus E} P(X_i \mid pa_i, e) \]

Notice: \( Q \) is another Bayesian network

Example:

Given a Bayesian network: \( P(X_1, X_2, X_3) = P(X_1) \times P(X_2 \mid X_1) \times P(X_3 \mid X_1, X_2) \) and evidence \( X_2 = x_2 \).

\[ Q(X_1, X_3) = P(X_1) \times P(X_3 \mid X_1, X_2 = x_2) \]

Weights:

Given a sample: \( x = (x_1, \ldots, x_n) \)

\[ w = \frac{P(x, e)}{Q(x)} = \frac{\prod_{x_i \in X \setminus E} P(x_i \mid pa_i, e) \times \prod_{e_j \in E} P(e_j \mid pa_j)}{\prod_{X_i \in X \setminus E} P(x_i \mid pa_i, e)} \]

\[ = \prod_{E_j \in E} P(e_j \mid pa_j) \]
Likelihood Weighting: Estimates

Estimate $P(e)$:

$$\hat{P}(e) = \frac{1}{T} \sum_{t=1}^{T} w^{(t)}$$

Estimate Posterior Marginals:

$$\hat{P}(x_i \mid e) = \frac{\hat{P}(x_i, e)}{\hat{P}(e)} = \frac{\sum_{t=1}^{T} w^{(t)} g_{x_i}(x^{(t)})}{\sum_{t=1}^{T} w^{(t)}}$$

$$g_{x_i}(x^{(t)}) = 1 \text{ if } x_i = x_i^{(t)} \text{ and equals zero otherwise}$$
Likelihood Weighting

- Converges to exact posterior marginals
- Generates Samples Fast
- Sampling distribution is close to prior (especially if $E \subseteq$ Leaf Nodes)
- Increasing sampling variance
  $\Rightarrow$ Convergence may be slow
  $\Rightarrow$ Many samples with $P(x^{(t)})=0$ rejected
Avoid rejection

- **Gibbs Sampling**: An MCMC approach
- **Likelihood weighting**: An importance sampling approach
- **Exploit structure**
  - **Cutset-sampling** (likelihood and Gibbs)
  - **SamplingSearch** (avoid inconsistency)
Overview

1. Probabilistic Reasoning/Graphical models
2. Importance Sampling
3. Markov Chain Monte Carlo: Gibbs Sampling
4. Sampling in presence of Determinism
5. Rao-Blackwellisation
6. AND/OR importance sampling
Gibbs Sampling (Geman&Geman, 1984)

- **Gibbs sampler** is an algorithm to generate a sequence of samples from the **joint probability distribution** of two or more random variables.

- Sample new variable value one variable at a time from the variable’s conditional distribution:

  \[ P(X_i) = P(X_i \mid x_1^t, \ldots, x_{i-1}^t, x_{i+1}^t, \ldots, x_n^t) = P(X_i \mid x^t \setminus x_i) \]

- Samples from a Markov chain with stationary distribution \( P(X/e) \)
Ordered Gibbs Sampler

Generate sample $x_{t+1}^t$ from $x_t^t$:

$$X_1 = x_1^{t+1} \leftarrow P(X_1 \mid x_2^t, x_3^t, ..., x_N^t, e)$$
$$X_2 = x_2^{t+1} \leftarrow P(X_2 \mid x_1^{t+1}, x_3^t, ..., x_N^t, e)$$
$$...$$
$$X_N = x_N^{t+1} \leftarrow P(X_N \mid x_1^{t+1}, x_2^{t+1}, ..., x_{N-1}^{t+1}, e)$$

In short, for $i=1$ to $N$:

$$X_i = x_i^{t+1} \leftarrow \text{sampled from } P(X_i \mid x^t \setminus x_i, e)$$
Given Markov blanket (parents, children, and their parents), $X_i$ is independent of all other nodes.

Markov blanket:

$$markov(X_i) = pa_i \cup ch_i \cup \left( \bigcup_{X_j \in ch_i} pa_j \right)$$

$$P(X_i \mid x^t \setminus x_i) = P(X_i \mid markov_i) :$$

$$P(x_i \mid x^t \setminus x_i) \propto P(x_i \mid pa_i) \prod_{X_j \in ch_i} P(x_j \mid pa_j)$$

Computation is linear in the size of Markov blanket!
Input: $X$, $E=e$
Output: $T$ samples $\{x^t\}$

Fix evidence $E=e$, initialize $x^0$ at random

1. For $t = 1$ to $T$ (compute samples)
2. For $i = 1$ to $N$ (loop through variables)
3. $x_i^{t+1} \leftarrow P(X_i \mid markov_i^t)$
4. End For
5. End For
Gibbs Sampling Example - BN

\[ X = \{X_1, X_2, ..., X_9\}, E = \{X_9\} \]

\[ X_1 = x_1^0 \]
\[ X_6 = x_6^0 \]
\[ X_2 = x_2^0 \]
\[ X_7 = x_7^0 \]
\[ X_3 = x_3^0 \]
\[ X_8 = x_8^0 \]
\[ X_4 = x_4^0 \]
\[ X_5 = x_5^0 \]
Gibbs Sampling Example - BN

\[ X = \{ X_1, X_2, \ldots, X_9 \}, E = \{ X_9 \} \]

\[ X_1^1 \leftarrow P(X_1 \mid x_2^0, \ldots, x_8^0, x_9) \]

\[ X_2^1 \leftarrow P(X_2 \mid x_1^1, \ldots, x_8^0, x_9) \]

\[ \ldots \]
Answering Queries $P(x_i | e) = ?$

- **Method 1**: count # of samples where $X_i = x_i$ (*histogram estimator*):

\[
\bar{P}(X_i = x_i) = \frac{1}{T} \sum_{t=1}^{T} \delta(x_i, x^t)
\]

  - Dirac delta f-n

- **Method 2**: average probability (*mixture estimator*):

\[
\bar{P}(X_i = x_i) = \frac{1}{T} \sum_{t=1}^{T} P(X_i = x_i | \text{markov}_i^t)
\]

- Mixture estimator converges faster (*consider estimates for the unobserved values of $X_i$; prove via Rao-Blackwell theorem*)
AND/OR w-cutset

3-cutset

2-cutset

1-cutset
Cutset Sampling

Generate sample $c_{t+1}^t$ from $c_t^t$, $C \subset X$:

$$C_1 = c_1^{t+1} \leftarrow P(c_1 \mid c_2^t, c_3^t, \ldots, c_K^t, e)$$

$$C_2 = c_2^{t+1} \leftarrow P(c_2 \mid c_1^{t+1}, c_3^t, \ldots, c_K^t, e)$$

$$\vdots$$

$$C_K = c_K^{t+1} \leftarrow P(c_K \mid c_1^{t+1}, c_2^{t+1}, \ldots, c_K^{t+1}, e)$$

Queries:

$$\overline{P}(c_i \mid e) = \frac{1}{T} \sum_{t=1}^{T} P(c_i \mid c_{-i}^t, e)$$

$$\overline{P}(x_i \mid e) = \frac{1}{T} \sum_{t=1}^{T} P(x_i \mid c^t, e)$$
Cutset Sampling Example

Sample a new value for $X_2$:

$c^0 = \{x_2^0, x_5^0\}$

$BTE(x_2', x_5^0, x_9)$

$BTE(x_2'', x_5^0, x_9)$

$x_2^1 \leftarrow P(x_2 | x_5^0, x_9) = \frac{1}{\alpha} \left[ BTE(x_2', x_5^0, x_9) + BTE(x_2'', x_5^0, x_9) \right]$

W-cutset sampling, Bidyuk and Dechter JAIR 2007
CPCS360b Test Results

MSE vs. #samples (left) and time (right)

Ergodic, $|X| = 360$, $D(X_i) = 2$, $|C| = 21$, $|E| = 36$

Exact Time > 60 min using Cutset Conditioning

Exact Values obtained via Bucket Elimination
Coding Networks, MSE vs. $w$

Note:
1-cutset = All Code Bits
SampleSearch

- Combining importance sampling with backtracking search.
Road Map

- Overview: Bayesian networks and algorithms
- Exact Inference
- Bounded-inference
- Search
- Sampling
- Hybrid of search and inference
- **Software**
- Modeling and learning
Software & Competitions

- **How to use the software**
  - [http://graphmod.ics.uci.edu/group/Software](http://graphmod.ics.uci.edu/group/Software)
  - [http://mulcyber.toulouse.inra.fr/projects/toulbar2](http://mulcyber.toulouse.inra.fr/projects/toulbar2)

- **Reports on competitions**
  - UAI-2006, 2008, 2010 Competitions
    - PE, MAR, MPE tasks
  - CP-2006 Competition
    - WCSP task
Road Map

- Overview: Bayesian networks and algorithms
- Exact Inference
- Bounded-inference
- Search
- Sampling
- Hybrid of search and inference
- Modeling and learning
- Software
Modeling with Bayesian Networks

Bayesian networks will be constructed in three consecutive steps.

Step 1

Define the network variables and their values.

- A query variable is one which we need to ask questions about, such as compute its posterior marginal.
- An evidence variable is one which we may need to assert evidence about.
- An intermediary variable is neither query nor evidence and is meant to aid the modeling process by detailing the relationship between evidence and query variables.

The distinction between query, evidence and intermediary variables is not a property of the Bayesian network, but of the task at hand.
Bayesian networks will be constructed in three consecutive steps.

**Step 2**

Define the network structure (edges).

We will be guided by a causal interpretation of network structure.

The determination of network structure will be reduced to answering the following question about each network variable $X$: what set of variables we regard as the direct causes of $X$?

What about the boundary strata?
Step 3
Define the network CPTs.

- CPTs can sometimes be determined completely from the problem statement by objective considerations.
- CPTs can be a reflection of subjective beliefs.
- CPTs can be estimated from data.
Example

The flu is an acute disease characterized by fever, body aches and pains, and can be associated with chilling and a sore throat. The cold is a bodily disorder popularly associated with chilling and can cause a sore throat. Tonsillitis is inflammation of the tonsils which leads to a sore throat and can be associated with fever.

Our goal here is to develop a Bayesian network to capture this knowledge and then use it to diagnose the condition of a patient suffering from some of the symptoms mentioned above.

Variables? Arcs? Try it.
A naive Bayes structure has the following edges $C \rightarrow A_1, \ldots, C \rightarrow A_m$, where $C$ is called the class variable and $A_1; \ldots; A_m$ are called the attributes.

Variables are binary: values are either true or false. More refined information may suggest different degrees of body ache.
CPTs can be obtained from medical experts, who supply this information based on known medical statistics or subjective beliefs gained through practical experience.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>true</td>
<td>false</td>
<td>?</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>2</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>3</td>
<td>?</td>
<td>?</td>
<td>true</td>
<td>false</td>
<td>?</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

? indicates the unavailability of corresponding data for that patient.
Tools for Bayesian network inference can generate a network parameterization $\Theta$, which tries to maximize the probability of seeing the given cases.

If each case is represented by event $d_i$, such tools will generate a parametrization $\Theta$ which leads to a probability distribution $\Pr$ that attempts to maximize:

$$\prod_{i=1}^{N} \Pr(d_i).$$

Term $\Pr(d_i)$ represents the probability of seeing the case $i$.

The product represents the probability of seeing all $N$ cases (assuming the cases are independent).
Learning Bayesian networks

Data + Prior information

Inducer

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

\[ \begin{array}{cc}
E & B \\
- & e \ b & .9 & .1 \\
- & e \ b & .7 & .3 \\
- & - & .8 & .2 \\
- & e \ b & .99 & .01 \\
\end{array} \]
## The Learning Problem

<table>
<thead>
<tr>
<th></th>
<th>Known Structure</th>
<th>Unknown Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Complete Data</strong></td>
<td>Statistical parametric estimation (closed-form eq.)</td>
<td>Discrete optimization over structures (discrete search)</td>
</tr>
<tr>
<td><strong>Incomplete Data</strong></td>
<td>Parametric optimization (EM, gradient descent...)</td>
<td>Combined (Structural EM, mixture models…)</td>
</tr>
</tbody>
</table>
## Learning Problem

<table>
<thead>
<tr>
<th></th>
<th>Known Structure</th>
<th>Unknown Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Complete</strong></td>
<td>Statistical parametric estimation (closed-form eq.)</td>
<td>Discrete optimization over structures (discrete search)</td>
</tr>
<tr>
<td><strong>Incomplete</strong></td>
<td>Parametric optimization (EM, gradient descent...)</td>
<td>Combined (Structural EM, mixture models…)</td>
</tr>
</tbody>
</table>

### Complete

| E | B | P(A | E, B) |
|---|---|--------|
| e | b | ?      |
| e | b | ?      |
| e | b | ?      |
| e | b | ?      |
| e | b | ?      |

### Incomplete

- P(A | E, B) for different E, B, A configurations:
  - \( <Y,N,N> \)
  - \( <Y,Y,Y> \)
  - \( <N,N,Y> \)
  - \( <N,Y,Y> \)

### Diagram

- **Inducer**
  - E, B, A
  - P(A | E, B)

- Probabilities:
  - E: B: P(A | E, B) = 0.9: 0.1
  - E: B: P(A | E, B) = 0.7: 0.3
  - E: B: P(A | E, B) = 0.8: 0.2
  - E: B: P(A | E, B) = 0.99: 0.01
### Learning Problem

<table>
<thead>
<tr>
<th>Known Structure</th>
<th>Unknown Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Complete</strong></td>
<td></td>
</tr>
<tr>
<td>Statistical parametric estimation (closed-form eq.)</td>
<td>Discrete optimization over structures (discrete search)</td>
</tr>
<tr>
<td><strong>Incomplete</strong></td>
<td></td>
</tr>
<tr>
<td>Parametric optimization (EM, gradient descent...)</td>
<td>Combined (Structural EM, mixture models…)</td>
</tr>
</tbody>
</table>

| $E, B, A$     | $P(A | E, B)$   |
|---------------|---------------|
| $\langle Y, N, N \rangle$ |               |
| $\langle Y, ?, Y \rangle$ |               |
| $\langle N, N, Y \rangle$ |               |
| $\langle N, Y, ? \rangle$ |               |
| $\langle ?, Y, Y \rangle$ |               |
| $\cdot$       |               |
| $\cdot$       |               |

**Inducer**

| $E, B, P(A | E, B)$ |
|------------------|
| $e, b$           | .9 1 |
| $e, b$           | .7 3 |
| $e, b$           | .8 2 |
| $e, b$           | .99 0.01 |
## Learning Problem

<table>
<thead>
<tr>
<th></th>
<th>Known Structure</th>
<th>Unknown Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Complete</strong></td>
<td>Statistical parametric estimation (closed-form eq.)</td>
<td>Discrete optimization over structures (discrete search)</td>
</tr>
<tr>
<td><strong>Incomplete</strong></td>
<td>Parametric optimization (EM, gradient descent...)</td>
<td>Combined (Structural EM, mixture models…)</td>
</tr>
</tbody>
</table>

### E, B, A

| E, B, A          | P(A | E, B) |
|------------------|-----------|
| <Y, N, N>        | .         |
| <Y, Y, Y>        | .         |
| <N, N, Y>        | .         |
| <N, Y, Y>        | .         |

### Inducer

| E, B, P(A | E, B) |
|---------|
| e b    | .9 .1  |
| e b    | .7 .3  |
| e -    | .8 .2  |
| e b    | .99 .01|
Learning Problem

<table>
<thead>
<tr>
<th>Known Structure</th>
<th>Unknown Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Complete</strong></td>
<td>Statistical parametric estimation (closed-form eq.)</td>
</tr>
<tr>
<td><strong>Incomplete</strong></td>
<td>Parametric optimization (EM, gradient descent...)</td>
</tr>
</tbody>
</table>

**Inducer**

| E  | B  | P(A | E, B) |
|----|----|---------|
| e  | b  | ?       |
| e  | b  | ?       |
| e  | b  | ?       |
| e  | b  | ?       |
| e  | b  | ?       |

| E  | B  | P(A | E, B) |
|----|----|---------|
| e  | b  | .9 .1   |
| e  | b  | .7 .3   |
| e  | b  | .8 .2   |
| e  | b  | .99 .01 |
Learning Parameters: complete data

- **ML-estimate:**
  \[
  \max_{\Theta} \log P(D \mid \Theta) \text{ - decomposable!}
  \]

- **MAP-estimate** (Bayesian statistics)
  \[
  \max_{\Theta} \log P(D \mid \Theta) P(\Theta)
  \]

**Conjugate priors - Dirichlet**

\[
\text{MAP}(\theta_{x,\text{pa}_x}) = \frac{\sum N_{x,\text{pa}_x} + \alpha_{x,\text{pa}_x}}{\sum N_{x,\text{pa}_x} + \sum \alpha_{x,\text{pa}_x}}
\]

Equivalent sample size (prior knowledge)
Learning Parameters: incomplete data

Non-decomposable marginal likelihood (hidden nodes)

Current model \((G, \Theta)\)

Initial parameters

Data

Expectation

Inference:

\[ P(S|X=0, D=1, C=0, B=1) \]

Maximization

Update parameters (ML, MAP)

EM-algorithm: iterate until convergence
Learning graph structure

Find $\hat{G} = \arg \max_G \text{Score}(G)$

- **Heuristic search:**
  - Greedy local search
  - Best-first search
  - Simulated annealing

  Complete data – local computations

  Incomplete data (score non-decomposable): Structural EM

- **Constrained-based methods**
  - Data impose independence relations (constrains)
Scoring functions:

Minimum Description Length (MDL)

- Learning ⇔ data compression

\[
\text{MDL}(BN \mid D) = -\log P(D \mid \Theta, G) + \frac{\log N}{2} \left| \Theta \right|
\]

- DL(Data|model)
- DL(Model)

- Other: MDL = -BIC (Bayesian Information Criterion)
- Bayesian score (BDe) - asymptotically equivalent to MDL
Thank you