Bounded inference non-iteratively; Mini-bucket elimination

COMPSCI 276, Spring 2017
Set 8: Rina Dechter

Reading: Class Notes (8), Darwiche chapters 14
Agenda

• Mini-bucket elimination
• Weighted Mini-bucket
• Mini-clustering
• Iterative Belief propagation
• Iterative-join-graph propagation
Probabilistic Inference Tasks

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

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

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

- Finding maximum-expected-utility (MEU) decision
  \[ (d_1^*, \ldots, d_k^*) = \arg\max_d \sum_{X/D} P(\bar{x}, e)U(\bar{x}) \]  
  Here, \( D \subseteq X \) : decision variables
  \( U(\bar{x}) \) : utility function
Queries

• Probability of evidence (or partition function)

\[ P(e) = \sum_{X-\text{var}(e)} \prod_{i=1}^{n} P(x_i \mid pa_i) \mid_e \quad Z = \sum_{X} \prod_i \psi_i(C_i) \]

• Posterior marginal (beliefs):

\[ P(x_i \mid e) = \frac{P(x_i, e)}{P(e)} = \frac{\sum_{X-\text{var}(e)-X_i} \prod_{j=1}^{n} P(x_j \mid pa_j) \mid_e}{\sum_{X-\text{var}(e)} \prod_{j=1}^{n} P(x_j \mid pa_j) \mid_e} \]

• Most Probable Explanation

\[ \bar{x}^* = \arg\max_{\bar{x}} P(\bar{x}, e) \]
Bucket Elimination

Query: $P(a|e=0) \propto P(a,e=0)$  

Elimination Order: $d,e,b,c$

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

$$= P(a) \sum_c P(c|a) \sum_b P(b|a) \sum_{e=0} P(e|b,c) \sum_d P(d|a,b)$$

Original Functions

|    | D: $P(d|a,b)$ | E: $P(e|b,c)$ | B: $P(b|a)$ | C: $P(c|a)$ | A: $P(a)$ |
|----|--------------|--------------|-------------|-------------|----------|

Messages

- $f_D(a,b) = \sum_d P(d|a,b)$
- $f_E(b,c) = P(e=0|b,c)$
- $f_B(a,c) = \sum_b P(b|a)f_D(a,b)f_E(b,c)$
- $f_C(a) = \sum_c P(c|a)f_B(a,c)$

Bucket Tree

- $f_D(a,b)$
- $f_E(b,c)$
- $f_B(a,c)$
- $f_C(a)$

Time and space $\exp(w^*)$
Finding

**MPE = max P(\overline{x})**

Algorithm *elim-mpe* (Dechter 1996)

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

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

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

bucket C: \( P(c \mid a) \)

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

bucket E: \( e=0 \)

bucket A: \( P(a) \)

\( MPE \)

\( h^E(a) \)

\( h^D(a,e) \)

\( h^C(a,d,e) \)

\( \max_b \prod \) Elimination operator

"induced width" (max clique size)

\( W^* = 4 \)
Generating the MPE-tuple

5. $b' = \arg \max P(b | a') \times P(d' | b, a') \times P(e' | b, c')$

4. $c' = \arg \max P(c | a') \times h^B(a', d', c, e')$

3. $d' = \arg \max h^C(a', d, e')$

2. $e' = 0$

1. $a' = \arg \max P(a) \cdot h^E(a)$

B: $P(b|a) \ P(d|b,a) \ P(e|b,c)$

C: $P(c|a) \ h^B(a,d,c,e)$

D: $h^C(a,d,e)$

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

A: $P(a) \ h^E(a)$

Return $(a', b', c', d', e')$
Approximate Inference

• Metrics of evaluation

• **Absolute error**: given $e > 0$ and a query $p = P(x|e)$, an estimate $r$ has absolute error $e$ iff $|p - r| < e$

• **Relative error**: the ratio $r/p$ in $[1-e, 1+e]$.

• Dagum and Luby 1993: approximation up to a relative error is NP-hard.

• Absolute error is also NP-hard if error is less than .5
Mini-Buckets: “Local Inference”

• Computation in a bucket is time and space exponential in the number of variables involved

• Therefore, partition functions in a bucket into “mini-buckets” on smaller number of variables
Mini-Bucket Approximation: MPE task

Split a bucket into mini-buckets => bound complexity

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

\[ h^X = \max_X \prod_{i=1}^r h_i \]

\[ \{ h_1, ..., h_r \} \]

\[ \{ h_{r+1}, ..., h_n \} \]

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

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

Exponential complexity decrease: \( O(e^a) \rightarrow O(e^r) + O(e^{n-r}) \)
We can generate a solution $s$ going forward as before: $U = F(s)$. $L = \text{lower bound}$
Mini-Bucket Elimination

[Dechter and Rish, 1997; 2003]
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 $N'$
Relaxed Network Example

B1: $P(b1|a), P(d|b1,a)$
B2: $P(e|b2,c)$
C: $P(c|a)$
D:
E: $E=e$
A: $P(a)$

(a)

(b)
MBE-MPE(i): Algorithm MBE-mpe

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

**Example:** MBE-mpe(3) versus BE-mpe

\[
\begin{align*}
B &: f(a, b) f(b, c) f(b, d) f(b, e) \\
C &: \lambda_{B\rightarrow C}(a, c) f(c, a) f(c, e) \\
D &: f(a, d) \lambda_{B\rightarrow D}(d, e) \\
E &: \lambda_{C\rightarrow E}(a, e) \lambda_{D\rightarrow E}(a, e) \\
A &: f(a) \lambda_{E\rightarrow A}(a)
\end{align*}
\]

\[U = \text{Upper bound} \quad w^* = 2\]

\[
\begin{align*}
B &: f(a, b) f(b, c) f(b, d) f(b, e) \\
C &: \lambda_{B\rightarrow C}(a, c, d, e) f(c, a) f(c, e) \\
D &: f(a, d) \lambda_{C\rightarrow D}(a, d, e) \\
E &: \lambda_{D\rightarrow E}(a, e) \\
A &: f(a) \lambda_{E\rightarrow A}(a)
\end{align*}
\]

\[\text{OPT} \quad w^* = 4\]

[Dechter and Rish, 1997]
Mini-Bucket Decoding

\[ \hat{b} = \arg \min_b f(\hat{a}, b) + f(b, \hat{e}) + f(b, \hat{d}) + f(b, \hat{e}) \]

\[ \hat{c} = \arg \min_c \lambda_{B \rightarrow C}(\hat{a}, c) + f(c, \hat{a}) + f(c, \hat{e}) \]

\[ \hat{d} = \arg \min_d f(\hat{a}, d) + \lambda_{B \rightarrow D}(d, \hat{e}) \]

\[ \hat{e} = \arg \min_e \lambda_{C \rightarrow E}(\hat{a}, e) + \lambda_{D \rightarrow E}(\hat{a}, e) \]

\[ \hat{a} = \arg \min_a f(a) + \lambda_{E \rightarrow A}(a) \]

Greedy configuration = upper bound

L = lower bound

[Dechter and Rish, 2003]
(i,m)-Partitionings

Definition 7.1.1 ((i,m)-partitioning) Let \( H \) be a collection of functions \( h_1, ..., h_t \) defined on scopes \( S_1, ..., S_t \), respectively. We say that a function \( f \) is subsumed by a function \( h \) if any argument of \( f \) is also an argument of \( h \). A partitioning of \( h_1, ..., h_t \) is canonical if any function \( f \) subsumed by another function is placed into the bucket of one of those subsuming functions. A partitioning \( Q \) into mini-buckets is an \((i,m)\)-partitioning if and only if (1) it is canonical, (2) at most \( m \) non-subsumed functions are included in each mini-bucket, (3) the total number of variables in a mini-bucket does not exceed \( i \), and (4) the partitioning is refinement-maximal, namely, there is no other \((i,m)\)-partitioning that it refines.
MBE(i,m), MBE(i)

• Input: Belief network ( $P_1,\ldots,P_n$ )
• Output: upper and lower bounds
• Initialize: put functions in buckets along ordering
• Process each bucket from p=n to 1
  • Create (i,m)-partitions
  • Process each mini-bucket
• (For mpe): assign values in ordering d
• Return: mpe-configuration, upper and lower bounds
Algorithm MBE-mpe($i,m$)

Input: A belief network $B\leftarrow\langle X,D,G,P \rangle$, where $P = \{P_1, \ldots, P_n\}$; an ordering of the variables, $d = X_1, \ldots, X_n$; observations e.

Output: An upper bound $U$ and a lower bound $L$ on the most probable configuration given the evidence. A suboptimal solution $\hat{x}$ that provides the lower bound $L = P(\hat{x})$.

1. Initialize: Generate an ordered partition of the conditional probability function, $\text{bucket}_1, \ldots, \text{bucket}_n$, where $\text{bucket}_i$ contains all functions whose highest variable is $X_i$. Put each observed variable in its bucket. Let $\psi_i$ be the product of input function in a bucket and let $h_i$ be the messages in the bucket.

2. Backward: For $p \leftarrow n$ downto 1, do for all the functions $h_1, h_2, \ldots, h_j$ in $\text{bucket}_p$, do

   - If (observed variable) $\text{bucket}_p$ contains $X_p = x_p$, assign $X_p = x_p$ to each function and put each in appropriate bucket.

   - else, Generate an an $(i,m)$-partitioning, $Q' = \{Q_1, \ldots, Q_r\}$ of $h_1, h_2, \ldots, h_i$ in $\text{bucket}_p$.

   - for each $Q_i \in Q'$ containing $h_1, \ldots, h_i$, do

     $h_i \leftarrow \max_{X_p} \prod_{j=1}^{i} h_{ij}$ \hspace{1cm} (1.1)

     Add $h_i$ to the bucket of the largest-index in $\text{scope}(h_i)$. Put constants in $\text{bucket}_1$.

3. Forward:

   - Generate an mpe upper bound cost by maximizing over $X_1$, the product in $\text{bucket}_1$. Namely $U \leftarrow \max_{X_1} \psi_1 \prod_{j} h_{1j}$.

   - (Generate an approximate mpe tuple): Given $x_1^n, \ldots, x_{p-1}^n$, assign $x_p^n$ to $X_p$ that maximizes the product of all functions in $\text{bucket}_p$. $L \leftarrow P(x_1^n, \ldots, x_n^n)$

4. Return $U$ and $L$ and configuration: $\hat{x} = (x_1^n, \ldots, x_n^n)$

Figure 1.2: Algorithm MBE-mpe($i,m$).
Partitioning, Refinements

Clearly, as the mini-buckets get smaller, both complexity and accuracy decrease.

Definition 7.1.4 Given two partitionings $Q'$ and $Q''$ over the same set of elements, $Q'$ is a refinement of $Q''$ if and only if for every set $A \in Q'$ there exists a set $B \in Q''$ such that $A \subseteq B$.

It is easy to see that:

Proposition 7.1.5 If $Q''$ is a refinement of $Q'$ in bucket $p$, then $h^p \leq g^p_{Q'} \leq g^p_{Q''}$.

Remember that mbe-mpe computes the bounds on $MPE = \max_x P(\bar{x}, \bar{e})$, rather than on $M = \max_x P(\bar{x}|\bar{e}) = MPE/P(\bar{e})$. Thus

$$\frac{L}{P(\bar{e})} \leq M \leq \frac{U}{P(\bar{e})}$$
Properties of MBE-mpe(i)

• **Complexity:** $O(r \exp(i))$ time and $O(r \exp(i))$ space.

• **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 best-first search
Anytime Approximation

Algorithm anytime-mpe(ε)
Input: Initial values of $i$ and $m$, $i_0$ and $m_0$; increments $i_{\text{step}}$ and $m_{\text{step}}$; and desired approximation error $\epsilon$.
Output: $U$ and $L$
1. Initialize: $i = i_0, m = m_0$.
2. do
3. run $mbe-mpc(i,m)$
4. $U \leftarrow$ upper bound of $mbe-mpc(i,m)$
5. $L \leftarrow$ lower bound of $mbe-mpc(i,m)$
6. Retain best bounds $U$, $L$, and best solution found so far
7. if $1 \leq U/L \leq 1 + \epsilon$, return solution
8. else increase $i$ and $m$: $i \leftarrow i + i_{\text{step}}$ and $m \leftarrow m + m_{\text{step}}$
9. while computational resources are available
10. Return the largest $L$
    and the smallest $U$ found so far.
MBE for Belief Updating and for Probability of Evidence or Partition Function

• 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.
Algorithm MBE-bel-max(i,m)

Input: A belief network $\mathcal{B} = (X, D, \mathcal{P}_G, \mathcal{I})$, an ordering $d = (X_1, \ldots, X_n)$; evidence $e$

Output: an upper bound on $P(X_1, e)$ and an upper bound on $P(e)$.

1. Initialize: Partition $P = \{P_1, \ldots, P_n\}$ into buckets $\text{bucket}_1, \ldots, \text{bucket}_n$, where $\text{bucket}_k$ contains all CPTs $h_1, h_2, \ldots, h_t$ whose highest-index variable is $X_k$.

2. Backward: for $k = n$ to 2 do

   - If $X_p$ is observed ($X_k = a$), assign $X_k \leftarrow a$ in each $h_j$ and put the result in the highest-variable bucket of its scope (put constants in $\text{bucket}_1$).

   - Else for $h_1, h_2, \ldots, h_t$ in $\text{bucket}_k$ Generate an $(i, m)$-partitioning, $Q' = \{Q_1, \ldots, Q_r\}$. For each $Q_l \in Q'$, containing $h_1, \ldots, h_t$, do

     \[
     h_l \leftarrow \sum_{X_k} \Pi_{j=1}^t h_{1,j}, \quad \text{if } l = 1
     \]

     \[
     h_l \leftarrow \max_{X_k} \Pi_{j=1}^t h_{1,j}, \quad \text{if } k \neq 1
     \]

     Add $h_l$ to the bucket of the highest-index variable in its scope $\bigcup_{j=1}^l \text{scope}(h_{1,j}) - \{X_k\}$. (Put constant functions in $\text{bucket}_1$).

3. Return $P'(x_1, e) \leftarrow$ the product of functions in the bucket of $X_1$, which is an upper bound on $P(x_1, e)$.

$P'(e) \leftarrow \sum_{x_1} P'(\bar{x}_1, e)$, which is an upper bound on probability of evidence.

Figure 8.5: Algorithm MBE-bel-max(i,m).
Normalization

- MBE-bel computes upper/lower bound on the joint marginal distributions.

Alternatively, let $U_i$ and $L_i$ be the upper bound and lower bounding functions on $P(X_1 = x_i, \bar{e})$ obtained by $mbe-bel-max$ and $mbe-bel-min$, respectively. Then,

$$\frac{L_i}{P(\bar{e})} \leq P(x_i|\bar{e}) \leq \frac{U_i}{P(\bar{e})}$$

We sometime use normalization of the approximation, but then no guarantee. The problem is that we have to approximate also $P(e)$. 
Empirical Evaluation
(Dechter and Rish, 1997; Rish thesis, 1999)

- Randomly generated networks
  - Uniform random probabilities
  - Random noisy-OR
- CPCS networks
- Probabilistic decoding

Comparing MBE-mpe and anytime-mpe versus BE-mpe
Methodology for Empirical Evaluation (for mpe)

- U/L – accuracy
- Better (U/mpe) or mpe/L
- Benchmarks: Random networks
  - Given n,e,v generate a random DAG
  - For xi and parents generate table from uniform [0,1], or noisy-or
- Create k instances. For each, generate random evidence, likely evidence
- Measure averages
CPCS Networks – Medical Diagnosis (noisy-OR model)

Test case: no evidence

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

Time and parameter i

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>cpcs360</th>
<th>cpcs422</th>
</tr>
</thead>
<tbody>
<tr>
<td>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>
Agenda

• Mini-bucket elimination
• Weighted Mini-bucket
• Mini-clustering
• Iterative Belief propagation
• Iterative-join-graph propagation
The Power Sum and Holder Inequality

The power sum is defined as follows:

$$\sum_{x}^{w} f(x) = \left(\sum_{x} f(x)^{1/w}\right)^{w}$$  \hspace{1cm} (1.2)

where \( w \) is a non-negative weight. The power sum reduce to a standard summation when \( w = 1 \) and approaches max when \( w \rightarrow 0^{+} \).

[Holder inequality] Let \( f_i(x) \), \( i = 1...r \) be a set of functions and \( w_1, ..., w_r \) b a set of of non-zero weights, s.t., \( w = \sum_{i=1}^{r} w_i \) then,

$$\sum_{x}^{w} \prod_{i=1}^{r} f_i(x) \leq \prod_{i=1}^{r} \sum_{x}^{w_i} f_i(x)$$
Working Example

• Model:
  • Markov network

• Task:
  • Partition function

\[ Z = \sum_{A,B,C} f(A)f(B)f(C)f(A, B)f(A, C)f(B, C) \]

(Qiang Liu slides)
Mini-Bucket (Basic Principles)

- Upper bound

\[ \sum_{i} a_i b_i \leq \left( \sum_{i} a_i \right) \max(b_i) \]

- Lower bound

\[ \sum_{i} a_i b_i \geq \left( \sum_{i} a_i \right) \min(b_i) \]

(Qiang Liu slides)

I am using \( a_i \ b_i \) to represent the general constant.
Holder Inequality

\[ \sum_{i} a_i b_i \leq \left( \sum_{i} a_i^{1/w_1} \right)^{w_1} \left( \sum_{i} b_i^{1/w_2} \right)^{w_2} \]

- Where \( a_i > 0, b_i > 0 \) and \( \sum a_i^{1/w_1} = \sum b_i^{1/w_2} \)
- When \( \sum a_i^{1/w_1} = \sum b_i^{1/w_2} \) the equality is achieved.

(Qiang Liu slides)

Reverse Holder Inequality

- If $w_1 + w_2 = 1$, but $w_1 < 0, w_2 > 1$
  
  the direction of the inequality reverses.

$$\sum_i a_i b_i \geq \left(\sum_i a_i^{1/w_1}\right)^{w_1} \left(\sum_i b_i^{1/w_2}\right)^{w_2}$$

(Qiang Liu slides)

Mini-Bucket as Holder Inequality

\[ w_1 = 1; w_2 \rightarrow +0 \quad \text{(mbe-bel-max)} \]

\[ \sum_i a_i b_i \leq \left( \sum_i a_i \right) \max_i(b_i) \]

- \[ w_1 = 1, w_2 \rightarrow -0 \quad \text{(mbe-bel-min)} \]

\[ \sum_i a_i b_i \geq \left( \sum_i a_i \right) \min_i(b_i) \]
Algorithm Weighted WMBE(i,m), (w₁,...,wₙ)
Input: A belief network \( \mathcal{B} = (X,D,P_{\mathcal{G}},\Pi) \), an ordering \( d = (X_1, \ldots, X_n) \); evidence e
Output: an upper bound on \( \sum_x \prod_{i=1}^n P_i \)

1. Initialize: Partition \( P = \{P_1, \ldots, P_n\} \) into buckets bucket₁, ..., bucketₙ, where bucketₖ contains all CPTs \( h_1, h_2, \ldots, h_t \) whose highest-index variable is \( X_k \).

2. Backward: for \( k = n \) to 1 do
   - If \( X_p \) is observed \( (X_k = a) \), assign \( X_k \leftarrow a \) in each \( h_j \) and put the result in the highest-variable bucket of its scope (put constants in bucket₁).
   - Else for \( h_1, h_2, \ldots, h_t \) in bucketₖ Generate an \((i,m)\)-partitioning, \( Q' = \{Q_1, \ldots, Q_r\} \). Select a set of weights \( w_1, \ldots, w_r \) s.t. \( \sum_i w_i = w \).
     For each \( Q_t \in Q' \), containing \( h_{i_1}, \ldots, h_{i_t} \), do
       \[
       h_t \leftarrow \sum_{X_k} \prod_{j=1}^t h_{i_j} = \left( \sum_{X_k} \prod_{j=1}^t (h_{i_j})^{w_j} \right)^{\frac{1}{w_t}}
       \]
     Add \( h_t \) to the bucket of the highest-index variable in its scope (and put constant functions in bucket₁).

3. Return \( U \leftarrow \) the weighted product of functions in the bucket of \( X_1 \), which is an upper bound on \( P(x_1, e) \).
Weighted Mini-Bucket

(for summation)

Exact bucket elimination:

$$\lambda_B(a, c, d, e) = \sum_b [f(a, b) \cdot f(b, c) \cdot f(b, d) \cdot f(b, e)]$$

$$\leq \left[ \sum_b f(a, b) f(b, c) \right] \cdot \left[ \sum_b f(b, d) f(b, e) \right]$$

$$= \lambda_{B \rightarrow C}(a, c) \cdot \lambda_{B \rightarrow D}(d, e)$$

where

$$\sum_x f(x) = \left[ \sum_x f(x) \right]^{\frac{1}{w}}$$

is the weighted or "power" sum operator

$$\sum_x f_1(x) f_2(x) \leq \left[ \sum_x f_1(x) \right] \left[ \sum_x f_2(x) \right]$$

where

$$w_1 + w_2 = w \text{ and } w_1 > 0, w_2 > 0$$

(lower bound if $$w_1 > 0, w_2 < 0$$)

[where] $$w_1 + w_2 = w$$ and $$w_1 > 0, w_2 > 0$$ (lower bound if $$w_1 > 0, w_2 < 0$$)
Choosing the weights

\[ w_1 = 1/2, \ w_2 = 1/2 \ (\text{Cauchy–Schwarz inequality}) \]

\[ \sum_i a_i b_i \leq \left( \sum_i a_i^2 \right)^{1/2} \left( \sum_i b_i^2 \right)^{1/2} \]

What is the optimal weights?
Allocating the probabilities

\[
\sum a_i b_i \leq \left( \sum_i (a_i \xi_i)^{1/w_1} \right)^{w_1} \left( \sum_i \left( b_i \frac{1}{\xi_i} \right)^{1/w_2} \right)^{w_2}
\]

Notice that \( a_i b_i = (a_i \xi_i) \cdot \left( b_i \frac{1}{\xi_i} \right) \)

What is the optimal allocation?
Extention of Mini-Bucket

• Allocation of the probability:

\[ f_1(A) f_2(A) = f(A) \]

• weights: \( w_1 \) and \( w_2 \)

\[ w_1 + w_2 = 1 \]

\[
\sum_A f(A) f(A, B) f(A, C) \\
\leq \left( \sum_A (f_1(A) f(A, C)^{1/w_1}) \right)^{w_1} \left( \sum_A (f_2(A) f(B, C)^{1/w_2}) \right)^{w_2}
\]

(Qiang Liu slides)
Algorithm MBE-map(I,n)
Input: A Bayesian network \( \mathcal{D} = (X, D, P_{\mathcal{D}}, \mathbb{I}) \), \( P = \{P_1, \ldots, P_n\} \); a subset of hypothesis variables \( A = \{A_1, \ldots, A_k\} \); an ordering of the variables, \( d \), in which the \( A \)'s are first in the ordering; observations \( \psi \); \( \psi_i \) is the product of input function in the bucket of \( X_i \).
Output: An upper bound on the map and a suboptimal solution \( \rho = \sigma_0^* \).
1. Initialize: Generate an ordered partition of the conditional probability functions, \( \text{bucket}_0, \ldots, \text{bucket}_n \), where \( \text{bucket}_i \) contains all functions whose highest variable is \( X_i \).
2. Backwards For \( p = n \) downto 1, do
   for all the message functions \( \beta_1, \beta_2, \ldots, \beta_n \) in \( \text{bucket}_p \) do
   - If (observed variable) \( \text{bucket}_p \) contains the observation \( X_p = x_p \), assign \( X_p = x_p \) to each \( \beta_k \) and \( \psi_x \) and put each in appropriate bucket.
   - Else for \( h_1, h_2, \ldots, h_j \) in \( \text{bucket}_p \) generate an \((i,m)\)-partitioning, \( Q \) mini-buckets \( Q_1, \ldots, Q_r \).
   - If \( X_p \notin A \) (not a hypothesis variable)
     - for each \( Q_l \in Q \) containing \( h_1, \ldots, h_m \),
       - \( h_l \leftarrow \sum_{X_p} \Pi_{x_l=1} h_{i_x} \), if \( l = 1 \)
       - \( h_l \leftarrow \max_{X_p} \Pi_{x_l=1} h_{i_x} \), if \( l \neq 1 \)
   - Else, \( (x_p \in A) \) (a hypothesis variable)
     - for each \( Q_l \in Q \) containing \( h_1, \ldots, h_k \),
       - \( h_l \leftarrow \max_{X_p} \Pi_{x_l=1} h_{i_x} \)
   Add \( h_l \) to the bucket of the highest-index variable in \( U_l \leftarrow \bigcup_1^r \text{scope}(h_l) = \{X_1\} \) (put constants in \( \text{bucket}_1 \)).
3. Forward: for \( p = 1 \) to \( k \), given \( A_1 = \sigma_1^*, \ldots, A_{p-1} = \sigma_{p-1}^* \), assign a value \( \sigma_p^* \) to \( A_p \) that maximizes the product of all functions in \( \text{bucket}_p \).
4. Return An upper bound \( U = \max_{X_p} \Pi_{x\in \text{bucket}_0} h_0 \) on map, computed in the first bucket, and the assignment \( \sigma_k^* = (\sigma_1^*, \ldots, \sigma_k^*) \).

Figure 8.8: Algorithm MBE-map(I,n).
Algorithm MBE-map$(i,m)$

Input: A Bayesian network $B = (X, D, P, \Pi)$, $P = \{P_1, ..., P_n\}$; a subset of hypothesis variables $A = \{A_1, ..., A_k\}$; an ordering of the variables, $d_i$, in which the $A$'s are first in the ordering; observations $e$

Output: An upper bound on the map and a suboptimal solution $A = a_k^e$

1. Initialize: Partition $P = \{P_1, ..., P_n\}$ into $\text{bucket}_1, ..., \text{bucket}_m$, where $\text{bucket}_i$ contains all functions whose highest variable is $X_i$.
2. Backwards: For $p \leftarrow n$ downto $1$, do for all the functions $h_1, h_2, ..., h_j$ in $\text{bucket}_p$ do
   - If (observed variable) $\text{bucket}_p$ contains the observation $X_p = x_p$, assign $X_p = x_p$ to each $h_i$ and put each in appropriate bucket.
   - Else for $h_1, h_2, ..., h_j$ in $\text{bucket}_p$ generate an $(i,m)$-partitioning, $Q' = \{Q_1, ..., Q_r\}$.
   - If $X_P \not\in A$ assign $w_p = 1$, otherwise $w_p = 0$. Select weights for the mini-buckets in $X_p$ bucket: $w_{p_1}, ..., w_{p_r}$ s.t $\sum_r w_{p_i} = w_p$.
     foreach $Q_t \in Q'$, containing $h_{i_1}, ..., h_{i_t}$, do
       $$h_t \leftarrow \sum_{X_k} \prod_{j=1}^t h_{i_j} = \left(\sum_{X_k} (\prod_{j=1}^t h_{i_j})^{w_{p_t}}\right)\frac{1}{w_{p_t}}$$
     Add $h_t$ to the bucket of the highest-index variable in its scope.

3. Forward: for $p = 1$ to $k$, given $A_1 = a_1^a$, ..., $A_{p-1} = a_{p-1}^a$, assign a value $a_p^a$ to $A_p$ that maximizes the product of all functions in $\text{bucket}_p$. conditioned on earlier assignments.
4. Return: An upper bound $U = \max_a, \prod_{h_i \in \text{bucket}_p} h_i$ on the map value, computed in the first bucket, and the assignment $a_k^a = (a_1^a, ..., a_k^a)$.

Figure 8.7: Algorithm MBE-map$(i,m)$. 

MB and WMB for Marginal MAP

\[
\max_Y \sum_{X \setminus Y} \prod_j P_j
\]

\[
\lambda_{B \to C}(a, c) = \sum_b w_1 f(a, b) f(b, c)
\]

\[
\lambda_{B \to D}(d, e) = \sum_b w_2 f(b, d) f(b, e)
\]

\[
(w_1 + w_2 = 1)
\]

\[
\lambda_{E \to A}(a) = \max_e \lambda_{C \to E}(a, e) \lambda_{D \to E}(a, e)
\]

\[
U = \max_a f(a) \lambda_{E \to A}(a)
\]

Can optimize over cost-shifting and weights
(single pass “MM” or iterative message passing)

\[\text{[Liu and Ihler, 2011; 2013]}\]
\[\text{[Dechter and Rish, 2003]}\]
Probabilistic decoding

Error-correcting linear block code

State-of-the-art: approximate algorithm – iterative belief propagation (IBP) (Pearl’s poly-tree algorithm applied to loopy networks)
Example 7.3.1 We will next demonstrate the mini-bucket approximation for MAP on an example of probabilistic decoding (see Chapter 2). Consider a belief network which describes the decoding of a linear block code, shown in Figure 7.7. In this network, $U_i$ are information bits and $X_j$ are code bits, which are functionally dependent on $U_i$. The vector $(U, X)$, called the channel input, is transmitted through a noisy channel which adds Gaussian noise and results in the channel output vector $Y = (Y^u, Y^x)$. The decoding task is to assess the most likely values for the $U$'s given the observed values $Y = (y^u, y^x)$, which is the MAP task where $U$ is the set of hypothesis variables, and $Y = (y^u, y^x)$ is the evidence. After processing the observed buckets we get the following bucket configuration (lower case $y$'s are observed values):

\[
\begin{align*}
\text{bucket}(X_0) &= P(y_0^u|X_0), P(X_0|U_0, U_1, U_2), \\
\text{bucket}(X_1) &= P(y_1^u|X_1), P(X_1|U_1, U_2, U_3), \\
\text{bucket}(X_2) &= P(y_2^u|X_2), P(X_2|U_3, U_4, U_5), \\
\text{bucket}(X_3) &= P(y_3^u|X_3), P(X_3|U_4, U_5, U_6), \\
\text{bucket}(X_4) &= P(y_4^u|X_4), P(X_4|U_5, U_6, U_7), \\
\text{bucket}(U_0) &= P(U_0), P(y_0^u|U_0), \\
\text{bucket}(U_1) &= P(U_1), P(y_1^u|U_1), \\
\text{bucket}(U_2) &= P(U_2), P(y_2^u|U_2), \\
\text{bucket}(U_3) &= P(U_3), P(y_3^u|U_3), \\
\text{bucket}(U_4) &= P(U_4), P(y_4^u|U_4).
\end{align*}
\]

Processing by sube-map(4,1) of the first five buckets by summation and the rest by maximization, results in the following mini-bucket partitionings and function generation:
bucket($X_0$) = \{P(y^g_0 | X_0), P(X_0 | U_0, U_1, U_2)\},

bucket($X_1$) = \{P(y^g_1 | X_1), P(X_1 | U_1, U_2, U_3)\},

bucket($X_2$) = \{P(y^g_2 | X_2), P(X_2 | U_2, U_3, U_4)\},

bucket($X_3$) = \{P(y^g_3 | X_3), P(X_3 | U_3, U_4, U_0)\},

bucket($X_4$) = \{P(y^g_4 | X_4), P(X_4 | U_4, U_1, U_3)\},

bucket($U_0$) = \{P(U_0), P(y^g_0 | U_0), h^{X_0}(U_0, U_1, U_2)\},

bucket($U_1$) = \{P(U_1), P(y^g_1 | U_1), h^{X_1}(U_1, U_2, U_3), h^{X_0}(U_1, U_2)\},

bucket($U_2$) = \{P(U_2), P(y^g_2 | U_2), h^{X_2}(U_2, U_3, U_4), h^{X_1}(U_2, U_3)\},

bucket($U_3$) = \{P(U_3), P(y^g_3 | U_3), h^{X_3}(U_3, U_4), h^{X_2}(U_3, U_4)\},

bucket($U_4$) = \{P(U_4), P(y^g_4 | U_4), h^{X_4}(U_4)\}.

The first five buckets are not partitioned at all and are processed as full buckets, since in this case a full bucket is a (4,1)-partitioning. This processing generates five new functions, three are placed in bucket $U_0$, one in bucket $U_1$ and one in bucket $U_2$. Then bucket $U_0$ is partitioned into three mini-buckets processed by maximization, creating two functions placed in bucket $U_1$ and one function placed in bucket $U_3$. Bucket $U_1$ is partitioned into two mini-buckets, generating functions placed in bucket $U_2$ and bucket $U_3$. Subsequent buckets are processed as full buckets. Note that the scope of recorded functions is bounded by 3.

In the bucket of $U_4$ we get an upper bound $U$ satisfying $U \geq \text{MAP} = P(U, \bar{g}_n, \bar{g}_r)$ where $\bar{g}_n$ and $\bar{g}_r$ are the observed outputs for the $U$'s and the $X$'s bits transmitted. In order to bound $P(U | \bar{e})$, where $\bar{e} = (\bar{g}_n, \bar{g}_r)$, we need $P(\bar{e})$ which is not available. Yet, again, in most cases we are interested in the ratio $P(U = u_1 | \bar{e}) / P(U = u_2 | \bar{e})$ for competing hypotheses $U = u_1$ and $U = u_2$ rather than in the absolute values. Since $P(U | \bar{e}) = P(U | \bar{e}) / P(\bar{e})$ and the probability of the evidence is just a constant factor independent of $U$, the ratio is equal to $P(U_1, \bar{e}) / P(U_2, \bar{e})$. □
Complexity and Tractability of MBE(i,m)

**Theorem 7.6.1** Algorithm mbe(i,m) takes $O(r \cdot \exp(i))$ time and space, where $r$ is the number of input functions, and where $|F|$ is the maximum scope of any input function, $|F| \leq i \leq n$. For $m = 1$, the algorithm is time and space $O(r \cdot \exp(|F|))$. 
Iterative Belief Propagation

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

\[ \begin{align*}
\lambda_{X_1}(u_1) & \quad \pi_{U_2}(x_1) \\
\pi_{U_3}(x_1) & \quad \lambda_{X_2}(u_1)
\end{align*} \]

One step:
CTE - bel(U_1)

- No guarantees for convergence
- Works well for many coding networks
MBE-mpe vs. IBP

**mbmpe 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):

- **Structured (50,25) block code, P=7**
- **Random (100,50) block code, P=4**
Mini-Buckets: Summary

- Mini-buckets – local inference approximation

- Idea: bound size of recorded functions

- MBE-mpe(i) - mini-bucket algorithm for MPE
  - Better results for noisy-OR than for random problems
  - Accuracy increases with decreasing noise in coding
  - Accuracy increases for likely evidence
  - Sparser graphs -> higher accuracy
  - Coding networks: MBE-mpe outperforms IBP on low-induced width codes
Agenda

• Mini-bucket elimination
• Mini-clustering
• Iterative Belief propagation
• Iterative-join-graph propagation
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: $O(\text{deg} \times (n+N) \times d^{w^*+1})$

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

where 

- $\text{deg} =$ 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
Join-Tree Clustering (Cluster-Tree Elimination)

**EXACT algorithm**

*Time and space:*

\[ \exp(\text{cluster size}) = \exp(\text{treewidth}) \]
**Mini-Clustering**

Split a cluster into mini-clusters => bound complexity

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

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

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
Mini-Clustering - example

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

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

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

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

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

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

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

\[ H_{(4,3)}(e, f) := p(G = g_e \mid e, f) \]
Cluster Tree Elimination vs. Mini-Clustering

1. ABC
   - $h_{(1,2)}(b, c)$

2. BCDF
   - $h_{(2,1)}(b, c)$
   - $h_{(2,3)}(b, f)$

3. BEF
   - $h_{(3,2)}(b, f)$
   - $h_{(3,4)}(e, f)$

4. EFG
   - $h_{(4,3)}(e, f)$

1. ABC
   - $H_{(1,2)} = h_{(1,2)}(b, c)$

2. BCDF
   - $H_{(2,1)} = h_{(2,1)}(b)$
   - $h_{(2,3)}(b, c)$

3. BEF
   - $H_{(3,2)} = h_{(3,2)}(b, f)$
   - $h_{(3,4)}(e, f)$

4. EFG
   - $H_{(4,3)} = h_{(4,3)}(e, f)$
Semantic of node duplication for mini-clustering

• We can have a different duplication of nodes going up and down. Example: going down.

Figure 1.14: Node duplication semantics of MC: (a) trace of MC-BU(3); (b) trace of CTE-BU.
We can replace \( \max \) operator by

- \( \min \) \( \Rightarrow \) lower bound on the joint

- \( \text{mean} \) \( \Rightarrow \) approximation of the joint
Grid 15x15 - 10 evidence

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

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

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

Grid 15x15, evid=10, w*=22, 10 instances
CPCS422 - Absolute error

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

CPCS 422, evid=10, w*=23, 1 instance

evidence=0
evidence=10
Coding networks - Bit Error Rate

Coding networks, $N=100$, $P=4$, $\sigma=0.22$, $w^*=12$, 50 instances

Coding networks, $N=100$, $P=4$, $\sigma=0.51$, $w^*=12$, 50 instances

$\sigma=0.22$

$\sigma=0.51$
CPCS 422 - Absolute error

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

CPCS 422, evid=10, $w^* = 23$, 1 instance

Absolute error

证据=0

证据=10
Coding networks - Bit Error Rate

Coding networks, $N=100$, $P=4$, $\sigma=0.22$, $w^*=12$, 50 instances

Coding networks, $N=100$, $P=4$, $\sigma=0.51$, $w^*=12$, 50 instances

$\sigma=0.22$  

$\sigma=0.51$  

MC  

IBP
Heuristic for Partitioning

**Scope-based Partitioning Heuristic.** The *scope-based* partition heuristic (SCP) aims at minimizing the number of mini-buckets in the partition by including in each minibucket as many functions as possible as long as the $i$ bound is satisfied. First, single function mini-buckets are decreasingly ordered according to their arity. Then, each minibucket is absorbed into the left-most mini-bucket with whom it can be merged.

The time and space complexity of $\text{Partition}(B, i)$, where $B$ is the partitioned bucket, using the SCP heuristic is $O(|B| \log(|B|) + |B|^2)$ and $O(\exp(i))$, respectively.

The scope-based heuristic is quite fast, its shortcoming is that it does not consider the actual information in the functions.
Content-based heuristics
(Rollon and Dechter 2010)

Use greedy heuristic derived from a distance function to decide which functions go into a single mini-bucket.
Greedy Partition as a function of a distance function $h$

```
function GreedyPartition($B$, $i$, $h$)
1. Initialize $Q$ as the bottom partition of $B$;
2. While $\exists Q' \in ch(Q)$ which is a $i$-partition
   $Q \leftarrow \arg\min_{Q'} \{h(Q \rightarrow Q')\}$ among child $i$-partitions of $Q$;
3. Return $Q$;
```

Figure 8.13: Greedy partitioning

**Proposition 8.6.5** The time complexity of `GreedyPartition` is $O(|B| \times T')$ where $O(T')$ is the time complexity of selecting the min child partition according to $h$. 
Agenda

• Mini-bucket elimination
• Mini-clustering
• Reparameterization, cost-shifting
• Iterative Belief propagation
• Iterative-joint-graph propagation
Cost-Shifting

(Reparameterization)

<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>6 + 3</td>
</tr>
<tr>
<td>b</td>
<td>g</td>
<td>0 - 1</td>
</tr>
<tr>
<td>g</td>
<td>b</td>
<td>0 + 3</td>
</tr>
<tr>
<td>g</td>
<td>g</td>
<td>6 - 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B</th>
<th>λ(B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>3</td>
</tr>
<tr>
<td>g</td>
<td>-1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>f(A,B,C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>b</td>
<td>b</td>
<td>12</td>
</tr>
<tr>
<td>b</td>
<td>b</td>
<td>g</td>
<td>6</td>
</tr>
<tr>
<td>b</td>
<td>g</td>
<td>b</td>
<td>0</td>
</tr>
<tr>
<td>b</td>
<td>g</td>
<td>g</td>
<td>6</td>
</tr>
<tr>
<td>g</td>
<td>b</td>
<td>b</td>
<td>6</td>
</tr>
<tr>
<td>g</td>
<td>b</td>
<td>g</td>
<td>0</td>
</tr>
<tr>
<td>g</td>
<td>g</td>
<td>b</td>
<td>6</td>
</tr>
<tr>
<td>g</td>
<td>g</td>
<td>g</td>
<td>12</td>
</tr>
</tbody>
</table>

- Modify the individual functions
- but –

keep the sum of functions the same

+ λ(B)

<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>6 - 3</td>
</tr>
<tr>
<td>b</td>
<td>g</td>
<td>0 - 3</td>
</tr>
<tr>
<td>g</td>
<td>b</td>
<td>0 + 1</td>
</tr>
<tr>
<td>g</td>
<td>g</td>
<td>6 + 1</td>
</tr>
</tbody>
</table>

- λ(B)

= 0 + 6

Modify the individual functions
- but –

keep the sum of functions the same
Dual Decomposition

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

- Bound solution using decomposed optimization
- Solve independently: optimistic bound
**Dual Decomposition**

\[ F^* = \min_x \sum_{\alpha} f_{\alpha}(x) \geq \max_{\lambda_{i \rightarrow \alpha}} \sum_{\alpha} \min_x \left[ f_{\alpha}(x) + \sum_{i \in \alpha} \lambda_{i \rightarrow \alpha}(x_i) \right] \]

- Bound solution using decomposed optimization
- Solve independently: optimistic bound
- Tighten the bound by reparameterization
  - Enforce lost equality constraints via Lagrange multipliers
Dual Decomposition

\[
F^* = \min_x \sum_{\alpha} f_{\alpha}(x) \geq \max_{\lambda_i \to \alpha} \sum_{\alpha} \min_x \left[ f_{\alpha}(x) + \sum_{i \in \alpha} \lambda_{i \to \alpha}(x_i) \right]
\]

Many names for the same class of bounds:
- Dual decomposition \[\text{[Komodakis et al. 2007]}\]
- TRW, MPLP \[\text{[Wainwright et al. 2005; Globerson \\& Jaakkola, 2007]}\]
- Soft arc consistency \[\text{[Cooper \\& Schiex, 2004]}\]
- Max-sum diffusion \[\text{[Warner 2007]}\]
Dual Decomposition

\[
F^* = \min_x \sum_{\alpha} f_{\alpha}(x) \geq \max_{\lambda_{i\rightarrow\alpha}} \sum_{\alpha} \min_x \left[ f_{\alpha}(x) + \sum_{i \in \alpha} \lambda_{i\rightarrow\alpha}(x_i) \right]
\]

Many ways to optimize the bound:
- Sub-gradient descent [Komodakis et al. 2007; Jojic et al. 2010]
- Proximal optimization [Ravikumar et al. 2010]
- ADMM [Meshi & Globerson 2011; Martins et al. 2011; Forouzan & Ihler 2013]
Mini-Bucket as Dual Decomposition

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

bucket C: \( \lambda_{B \rightarrow C}(a, c) \quad f(a, c) \quad f(c, e) \)

bucket D: \( f(a, d) \quad \lambda_{B \rightarrow D}(d, e) \)

bucket E: \( \lambda_{C \rightarrow E}(a, e) \quad \lambda_{D \rightarrow E}(a, e) \)

bucket A: \( f(a) \quad \lambda_{E \rightarrow A}(a) \)

\( L = \text{lower bound} \)
Mini-Bucket as Dual Decomposition

\[
\min_{a,c,b}[f(a,b) + f(b,c) - \lambda_{B\to C}(a,c)] = 0
\]

\[
\min_{d,e,b}[f(b,d) + f(b,e) - \lambda_{B\to D}(d,e)] = 0
\]

\[
\begin{align*}
\text{bucket B:} & \quad f(a, b) \quad f(b, c) \\
\text{bucket C:} & \quad \lambda_{B\to C}(a, c) \quad f(a, c) \quad f(c, e) \\
\text{bucket D:} & \quad f(a, d) \quad \lambda_{B\to D}(d, e) \\
\text{bucket E:} & \quad \lambda_{C\to E}(a, e) \quad \lambda_{D\to E}(a, e) \\
\text{bucket A:} & \quad f(a) \quad \lambda_{E\to A}(a)
\end{align*}
\]

\[L = \text{lower bound}\]
Mini-Bucket as Dual Decomposition

\[
\min_{a,c,b}[f(a,b) + f(b,c) - \lambda_{B\rightarrow C}(a,c)] = 0
\]

\[
\min_{d,e,b}[f(b,d) + f(b,e) - \lambda_{B\rightarrow D}(d,e)] = 0
\]

\[
\min_{a,e,c}[\lambda_{B\rightarrow C}(a,c) + f(a,c) + f(c,e) - \lambda_{C\rightarrow E}(a,e)] = 0
\]
Mini-Bucket as Dual Decomposition

\[
\begin{align*}
\min_{a,c,b} [f(a, b) + f(b, c) - \lambda_{B \rightarrow C}(a, c)] &= 0 \\
\min_{d,e,b} [f(b, d) + f(b, e) - \lambda_{B \rightarrow D}(d, e)] &= 0 \\
\min_{a,e,c} [\lambda_{B \rightarrow C}(a, c) + f(a, c) + f(c, e) - \lambda_{C \rightarrow E}(a, e)] &= 0 \\
\min_{a,d} [f(a, d) + \lambda_{B \rightarrow D}(d, e) - \lambda_{D \rightarrow E}(a, e)] &= 0
\end{align*}
\]

bucket B: \[
\begin{array}{ccc}
& f(a, b) & f(b, c) \\
& f(b, d) & f(b, e)
\end{array}
\]

bucket C: \[
\begin{array}{ccc}
\lambda_{B \rightarrow C}(a, c) & f(a, c) & f(c, e)
\end{array}
\]

bucket D: \[
\begin{array}{ccc}
f(a, d) & \lambda_{B \rightarrow D}(d, e)
\end{array}
\]

bucket E: \[
\begin{array}{ccc}
\lambda_{C \rightarrow E}(a, e) & \lambda_{D \rightarrow E}(a, e)
\end{array}
\]

bucket A: \[
\begin{array}{ccc}
f(a) & \lambda_{E \rightarrow A}(a)
\end{array}
\]

\[L = \text{lower bound}\]
Mini-Bucket as Dual Decomposition

\[
\begin{align*}
\min_{a,c,b} & \left[ f(a,b) + f(b,c) - \lambda_{B\rightarrow C}(a,c) \right] = 0 \\
\min_{d,e,b} & \left[ f(b,d) + f(b,e) - \lambda_{B\rightarrow D}(d,e) \right] = 0 \\
\min_{a,e,c} & \left[ \lambda_{B\rightarrow C}(a,c) + f(a,c) + f(c,e) - \lambda_{C\rightarrow E}(a,e) \right] = 0 \\
\min_{a,d} & \left[ f(a,d) + \lambda_{B\rightarrow D}(d,e) - \lambda_{D\rightarrow E}(a,e) \right] = 0 \\
\min_{a,e} & \left[ \lambda_{C\rightarrow E}(a,e) + \lambda_{D\rightarrow E}(a,e) - \lambda_{E\rightarrow A}(a) \right] = 0
\end{align*}
\]

bucket B: \[ f(a, b) \quad f(b, c) \quad f(b, d) \quad f(b, e) \]

bucket C: \[ \lambda_{B\rightarrow C}(a, c) \quad f(a, c) \quad f(c, e) \]

bucket D: \[ f(a, d) \quad \lambda_{B\rightarrow D}(d, e) \]

bucket E: \[ \lambda_{C\rightarrow E}(a, e) \quad \lambda_{D\rightarrow E}(a, e) \]

bucket A: \[ f(a) \quad \lambda_{E\rightarrow A}(a) \]

\[ L = \text{lower bound} \]
Mini-Bucket as Dual Decomposition

\[
\begin{align*}
\min_{a,c,b} [f(a, b) + f(b, c) - \lambda_{B\rightarrow C}(a, c)] &= 0 \\
\min_{d,e,b} [f(b, d) + f(b, e) - \lambda_{B\rightarrow D}(d, e)] &= 0 \\
\min_{a,e,c} [\lambda_{B\rightarrow C}(a, c) + f(a, c) + f(c, e) - \lambda_{C\rightarrow E}(a, e)] &= 0 \\
\min_{a,d} [f(a, d) + \lambda_{B\rightarrow D}(d, e) - \lambda_{D\rightarrow E}(a, e)] &= 0 \\
\min_{a,e} [\lambda_{C\rightarrow E}(a, e) + \lambda_{D\rightarrow E}(a, e) - \lambda_{E\rightarrow A}(a)] &= 0 \\
\min_{a} [f(a) + \lambda_{E\rightarrow A}(a)] &= L
\end{align*}
\]

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

bucket C: \( \lambda_{B\rightarrow C}(a, c) \quad f(a, c) \quad f(c, e) \)

bucket D: \( f(a, d) \quad \lambda_{B\rightarrow D}(d, e) \)

bucket E: \( \lambda_{C\rightarrow E}(a, e) \quad \lambda_{D\rightarrow E}(a, e) \)

bucket A: \( f(a) \quad \lambda_{E\rightarrow A}(a) \)

\( L = \text{lower bound} \)
Various Update Schemes

• Can use any decomposition updates
  • (message passing, subgradient, augmented, etc.)

• **FGLP**: Update the original factors

• **JGLP**: Update clique function of the join graph

• **MBE-MM**: Mini-bucket with moment matching
  • Apply cost-shifting within each bucket only
Algorithm 1: Factor graph LP (FGLP)

[1] **Input:** Graphical Model \( \langle X, D, F, \sum \rangle \), where \( f_\alpha \) is a function defined on variables \( X_\alpha \).

**Output:** Re-parameterized factors \( F^\prime \), upper bound on optimum value.

1. Iterate until convergence:
2. For each variable \( X_i \), do:
3. Let \( F_i = \{ \alpha : i \in \alpha \} \) with \( X_i \) in their scope
4. \( \forall \alpha \in F_i, \) compute max-marginals: \( \lambda_\alpha(X_i) = \max_{X_\alpha, X_i} f_\alpha(X_\alpha) \)
5. \( \forall \alpha \in F_i, \) update parameterization: \( f_\alpha(X_\alpha) \leftarrow f_\alpha(X_\alpha) - \lambda_\alpha(X_i) + \frac{1}{|F_i|} \sum_{\beta \in F_i} \lambda_\beta(X_i) \)
6. **Return:** Reparameterized factors \( F^\prime \) and bound \( \sum_{\alpha \in \sum} \max_{X} f_\alpha(X_\alpha) \)

\[
C^* \leq \min_{\lambda \in \Lambda} \sum_{(ij) \in F} \max_X (f_{ij}(X_i, X_j) + \lambda_{ij}(X_i) + \lambda_{ji}(X_j)).
\]

The new functions \( \tilde{f}_{ij} = f_{ij}(X_i, X_j) + \lambda_{ij}(X_i) + \lambda_{ji}(X_j) \) define a re-parameterization or cost-shifting of the original distribution. The \( \lambda_{ij} \) can be interpreted in various ways \([81, 77]\) and they can be tightened by different methods \([38, 75]\). MPLP \([25]\), soft-arc consistency, and many other LP algorithms operate directly
Factor graph Linear Programming

- Update the original factors (FGLP)
- Tighten all factors over $x_i$ simultaneously
- Compute $\text{max-marginals}$ $\forall \alpha, \gamma_\alpha(x_i) = \max_{x_\alpha \setminus x_i} f_\alpha$
- & update:
  $$\forall \alpha, f_\alpha(x_\alpha) \leftarrow f_\alpha(x_\alpha) - \gamma_\alpha(x_i) + \frac{1}{|F_i|} \sum_\beta \gamma_\beta(x_i)$$

UTA 1/2015
Figure 5.2: FGLP example: local messages involving variable $X_i$. 

$\gamma_s(X_i)$

$\beta_s(X_i)$

$\gamma_S_p(X_i)$

$\beta_S_p(X_i)$

$f_s(X_i, X_k, X_m)$

$f_{S_p}(X_i, X_j)$

$f_s(X_i, X_n)$

$X_i$

$X_k$

$X_j$

$X_n$
**Algorithm 23: Factor Graph Linear Programming (FGLP, based on [40])**

**Input:** A graphical model $\mathcal{M} = (X, D, F, \sum)$, variable ordering $\sigma$

**Output:** Upper bound on the optimum value of MPE cost

1. while NOT converged do
2.   for each variable $X_i$ do
3.     Get factors $F_i = f_{S_k} : X_i \in S_k$ with $X_i$ in their scope;
4.     // for each function compute max-marginals $\gamma$ marginalizing out all variables except for $X_i$:
5.     $\forall f_{S_k} \gamma_{S_k}(X_i) = \max_{S_k \setminus X_i} f_{S_k};$
6.     // compute messages $\beta_{S_k}(X_i)$ from $X_i$ back to a function $f_{S_k}$ correcting for the function's own max-marginal $\gamma_{S_k}$:
7.     $\forall f_{S_k} \beta_{S_k} = \frac{1}{|V_i|} \sum_{f_{S_j} \in F_i} \gamma_{S_j}(X_i) - \gamma_{S_k}(X_i)$
8.     // update (re-parametrize) each function:
9.     $\forall f_{S_k}, f_{S_k} \leftarrow f_{S_k} + \beta_{S_k};$

The update messages from variable $X_i$ back to the functions are:

$$\beta_{S_q}(X_i) = \frac{1}{3} (\gamma_{S_q}(X_i) + \gamma_{S_p}(X_i) + \gamma_{S_i}(X_i)) - \gamma_{S_q}(X_i)$$

$$\beta_{S_p}(X_i) = \frac{1}{3} (\gamma_{S_q}(X_i) + \gamma_{S_p}(X_i) + \gamma_{S_i}(X_i)) - \gamma_{S_p}(X_i)$$

$$\beta_{S_i}(X_i) = \frac{1}{3} (\gamma_{S_q}(X_i) + \gamma_{S_p}(X_i) + \gamma_{S_i}(X_i)) - \gamma_{S_i}(X_i)$$
Complexity of FGLP

Theorem 5.1 (Complexity of FGLP). The total time complexity of a single iteration of FGLP is $O(n \cdot Q \cdot k^{Sc})$, where $n$ is the number of variables in the problem, $k$ is the largest domain size, $|F|$ is the number of functions, $Sc$ bounds the largest scope of the original functions, $Q$ is the largest number of functions having the same variable $X_j$ in their scopes. The space complexity is $O(|F| \cdot k^{Sc})$. 
Mini-Bucket as Dual Decomposition

- Downward pass as cost-shifting
- Can also do cost-shifting within mini-buckets
- “Join graph” message passing
- “Moment matching” version: one message update within each bucket during downward sweep

Join graph:

bucket B: \{a, b, c\} → \{b, d, e\}
bucket C: \{a, c, e\}
bucket D: \{a, d, e\}
bucket E: \{a, e\}
bucket A: \{a\}

\[ L = \text{lower bound} \]
MBE-MM: MBE with moment matching

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

\[m_{11}, m_{12} \text{- moment-matching messages}\]

\[\max_B \Pi \quad \max_B \Pi\]

MPE* is an upper bound on MPE --U
Generating a solution yields a lower bound--L
Algorithm 26: Algorithm MBE-MM

Input: A graphical model $\mathcal{M} = (\mathcal{X}, \mathcal{D}, \mathcal{F}, \Sigma)$, variable order $o = \{X_1, \ldots, X_n\}$, i-bound parameter $i$
Output: Upper bound on the optimum value of MPE cost

//Initialize:
1. Partition the functions in $\mathcal{F}$ into $\mathcal{B}_{X_1}, \ldots, \mathcal{B}_{X_n}$, where $\mathcal{B}_{X_k}$ contains all functions $f_j$ whose highest variable is $X_k$.

//processing bucket $\mathcal{B}_{X_k}$
2. for $k \leftarrow n$ down to 1 do
3. Partition functions $g$ (both original and messages generated in previous buckets) in $\mathcal{B}_{X_k}$ into the mini-buckets defined $Q_{X_k} = \{q_k^1, \ldots, q_k^{|Q_{X_k}|}\}$, where each $q_k^i$ has no more than $i + 1$ variables;
4. Find the set of variables common to all the mini-buckets of variable $X_k$:
   $S_k = \text{Scope}(q_k^1) \cap \cdots \cap \text{Scope}(q_k^{|Q_{X_k}|})$;
5. Find the function of each mini-bucket
   $f_k^i = \prod_{q_k^i \in Q_{X_k}} g$;
6. Find the max-marginals of each mini-bucket
   $\gamma_k^i = \max_{\text{Scope}(q_k^i)} \bar{S}_k(f_k^i)$;
7. Update functions of each mini-bucket
   $F_k^i = F_k^i - \bar{\gamma}_k^i + \sum_{j=1}^i \gamma_k^j$;
8. Generate messages $h_{X_{k-1},X_k} = \max_{X_k} F_k^i$ and place each in the bucket of highest in the ordering $o$ variable $X_m$ in $\text{Scope}(q_k^i)$;
9. return All the buckets and the cost bound from $B_i$;

Theorem 5.3 (Complexity of MBE-MM). Given a problem with $n$ variables having domain of size $k$ and an i-bound $i$, the worst-case time complexity of MBE-MM is $O(n \cdot Q \cdot k^{i+1})$ and its space complexity is $O(n \cdot k^i)$, where $Q$ bounds the number of functions having the same variable $X_i$ in their scopes.
Anytime Approximation

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

**Initialize**: \( i = i_0 \)

**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

\[
\text{if } 1 \leq \frac{U}{L} \leq 1 + \varepsilon, \text{ return solution}
\]

**end**

**return** the largest \( L \) and the smallest \( U \)

[Dechter and Rish, 2003]
Anytime Approximation

- Can tighten the bound in various ways
  - Cost-shifting (improve consistency between cliques)
  - Increase i-bound (higher order consistency)
- Simple moment-matching step improves bound significantly
Anytime Approximation

• Can tighten the bound in various ways
  • Cost-shifting (improve consistency between cliques)
  • Increase i-bound (higher order consistency)
• Simple moment-matching step improves bound significantly
Anytime Approximation

- Can tighten the bound in various ways
  - Cost-shifting (improve consistency between cliques)
  - Increase i-bound (higher order consistency)
- Simple moment-matching step improves bound significantly
Weighted Mini-Bucket

(for summation)

Exact bucket elimination:

$$\lambda_B(a, c, d, e) = \sum_b [f(a, b) \cdot f(b, c) \cdot f(b, d) \cdot f(b, e)]$$

$$\leq \left[ \sum_b f(a, b)f(b, c) \right] \cdot \left[ \sum_b f(b, d)f(b, e) \right]$$

$$= \lambda_{B\rightarrow C}(a, c) \cdot \lambda_{B\rightarrow D}(d, e)$$

(mini-buckets)

where $$\sum_x f(x) = \left[ \sum_x f(x)^{1/w} \right]^w$$ is the weighted or "power" sum operator

$$\sum_x f_1(x)f_2(x) \leq \left[ \sum_x f_1(x) \right] \left[ \sum_x f_2(x) \right]$$

where $$w_1 + w_2 = w$$ and $$w_1 > 0, w_2 > 0$$

(lower bound if $$w_1 > 0, w_2 < 0$$)

U = upper bound

[Source: Liu and Ihler, 2011]
Weighted Mini-Bucket

• Related to conditional entropy decomposition but, with an efficient “primal” bound form

• Can optimize the bound over:
  • Cost-shifting
  • Weights

• Again, involves message passing over JG

• Similar, one-pass “moment-matching” variant

Join graph:

bucket B: \(\{a, b, c\}\) \(\{b, d, e\}\)

bucket C: \(\{a, c, e\}\)

bucket D: \(\{a, d, e\}\)

bucket E: \(\{a, e\}\)

bucket A: \(\{a\}\)

\[U = \text{upper bound}\]

[Liu and Ihler, 2011]
MB and WMB for Marginal MAP

$$\max_Y \sum_{X \setminus Y} \prod_j P_j$$

\[
\begin{align*}
\lambda_{B \to C}(a, c) &= \sum_b f(a, b) f(b, c) \\
\lambda_{B \to D}(d, e) &= \sum_b f(b, d) f(b, e) \\
\lambda_{E \to A}(a) &= \max_e \lambda_{C \to E}(a, e) \lambda_{D \to E}(a, e) \\
U &= \max_a f(a) \lambda_{E \to A}(a)
\end{align*}
\]

Marginal MAP

- Bucket B:
  - $$\Sigma_B$$: $$f(a, b) f(b, c) f(b, d) f(b, e)$$
- Bucket C:
  - $$\Sigma_C$$: $$\lambda_{B \to C}(a, c) f(a, c) f(c, e)$$
- Bucket D:
  - $$\max_D$$: $$f(a, d) \lambda_{B \to D}(d, e)$$
- Bucket E:
  - $$\lambda_{C \to E}(a, e) \lambda_{D \to E}(a, e)$$
- Bucket A:
  - $$\lambda_{E \to A}(a)$$

$$U = \text{upper bound}$$

Can optimize over cost-shifting and weights (single pass “MM” or iterative message passing)

[Liu and Ihler, 2011; 2013]
[Dechter and Rish, 2003]

ECAI 2016