Bounded Inference Non-iteratively; Mini-Bucket Elimination

COMPSCI 276,Spring 2018

Class 6: Rina Dechter

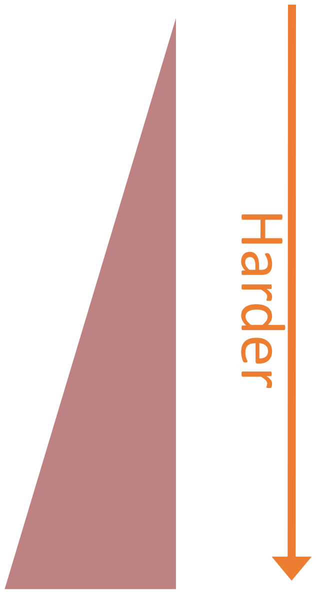
Reading: Class Notes (8), Darwiche chapters 14

Outline

- Mini-bucket elimination
- Weighted Mini-bucket
- Mini-clustering
- Re-parameterization, cost-shifting
- Iterative Belief propagation
- Iterative-join-graph propagation

Types of queries

Max-Inference	$f(\mathbf{x}^*) = \max_{\mathbf{x}} \prod_{\alpha} f_\alpha(\mathbf{x}_\alpha)$
Sum-Inference	$Z = \sum_{\mathbf{x}} \prod_{\alpha} f_\alpha(\mathbf{x}_\alpha)$
Mixed-Inference	$f(\mathbf{x}_M^*) = \max_{\mathbf{x}_M} \sum_{\mathbf{x}_S} \prod_{\alpha} f_\alpha(\mathbf{x}_\alpha)$



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

Queries

- **Probability of evidence (or partition function)**

$$P(e) = \sum_{X - \mathrm{var}(e)} \prod_{i=1}^{n} P(x_i \mid pa_i)|_e \qquad Z = \sum_X \prod_i \psi_i(C_i)$$

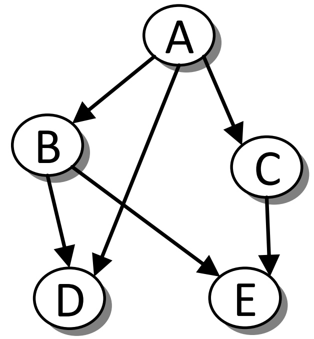
- **Posterior marginal (beliefs):**

$$P(x_i \mid e) = \frac{P(x_i, e)}{P(e)} = \frac{\sum_{X - \mathrm{var}(e) - X_i} \prod_{j=1}^{n} P(x_j \mid pa_j)|_e}{\sum_{X - \mathrm{var}(e)} \prod_{j=1}^{n} P(x_j \mid pa_j)|_e}$$

- **Most Probable Explanation**

$$\bar{x}^* = \arg\max_{\bar{x}} P(\bar{x}, e)$$

Bucket Elimination



Query: $P(a \mid e=0) \propto P(a, e=0)$ **Elimination Order**: d,e,b,c

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

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

Original Functions

D: $P(d \mid a,b)$

E: $P(e \mid b,c)$

B: $P(b \mid a)$

C: $P(c \mid a)$

A: $P(a)$

Messages

$\boxed{f_D(a,b)} = \sum_{d} P(d \mid a,b)$

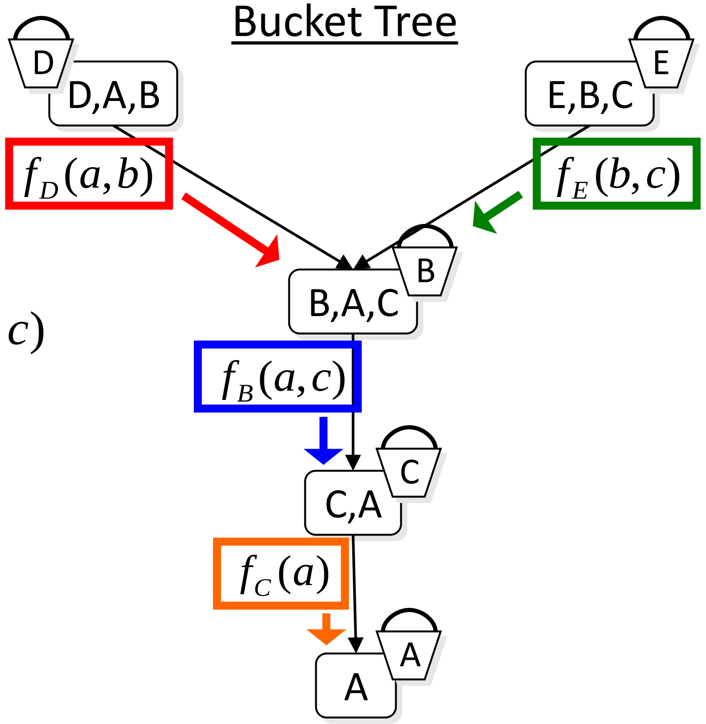
$\boxed{f_E(b,c)} = P(e=0 \mid b,c)$

$\boxed{f_B(a,c)} = \sum_{b} P(b \mid a)\, f_D(a,b)\, f_E(b,c)$

$\boxed{f_C(a)} = \sum_{c} P(c \mid a)\, f_B(a,c)$

$P(a, e=0) = p(A)\, f_C(a)$

Bucket Tree



Time and space exp(w*)

Finding $MPE = \max_{\bar{x}} P(\bar{x})$

Algorithm *elim-mpe* (Dechter 1996)

$\sum$ is replaced by ***max*** :

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

$$\max_b \prod$$ ← Elimination operator

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

bucket C: P(c|a)

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

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

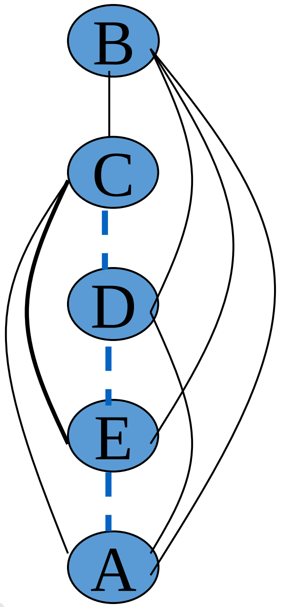
bucket A: P(a) $h^E(a)$

MPE

W*=4

"induced width"

(max clique size)



Generating the MPE-tuple

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

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

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

2. $e' = 0$

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

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

Outline

- Mini-bucket elimination
- Weighted Mini-bucket
- Mini-clustering
- Re-parameterization, cost-shifting
- Iterative Belief propagation
- Iterative-join-graph propagation

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

Decomposition Bounds

- Upper & lower bounds via approximate problem decomposition
- Example: MAP inference $F(x) = f_1(x) + f_2(x)$

X	F(X)
0	2.0
1	4.0
2	5.0
3	4.0

=

X	f_1(X)
0	1.0
1	2.0
2	3.0
3	4.0

+

X	f_2(X)
0	1.0
1	2.0
2	2.0
3	0.0

$$\max_x F(x) \quad = \quad \max_x \left[f_1(x) + f_2(x) \right]$$

$$5.0 \quad \leq \quad \left[\max_x f_1(x) \; + \; \max_x f_2(x) \right] \quad = \; 4.0 \; + \; 2.0 \; = \; 6.0$$

- Relaxation: two "copies" of x, no longer required to be equal
- Bound is tight (equality) if f_1, f_2 agree on maximizing value x

Mini-Bucket Approximation

Split a bucket into mini-buckets —> bound complexity

bucket (X) =

$$\underbrace{\left\{ f_1,\ f_2,\ \ldots\ f_r,\ f_{r+1},\ \ldots\ f_n \right\}}_{}$$

$$\lambda_X(\cdot) = \max_x \prod_{i=1}^{n} f_i(x, \ldots)$$

$$\left\{ f_1,\ \ldots\ f_r \right\} \qquad \left\{ f_{r+1},\ \ldots\ f_n \right\}$$

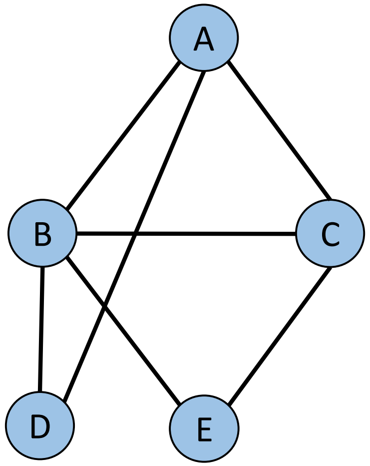
$$\lambda_{X,1}(\cdot) = \max_x \prod_{i=1}^{r} f_i(x, \ldots) \qquad \lambda_{X,2}(\cdot) = \max_x \prod_{i=r+1}^{n} f_i(x, \ldots)$$

$$\lambda_X(\cdot)\ \leq\ \lambda_{X,1}(\cdot)\ \lambda_{X,2}(\cdot)$$

Exponential complexity decrease: $O(e^n) \longrightarrow O(e^r) + O(e^{n-r})$

Mini-Bucket Elimination

[Dechter & Rish 2003]



mini-buckets

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

bucket C: $f(c,a)\ f(c,e)\ \lambda_{B\to C}(a,c)$

bucket 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: $f(a)\ \lambda_{E\to A}(a)$

U = upper bound

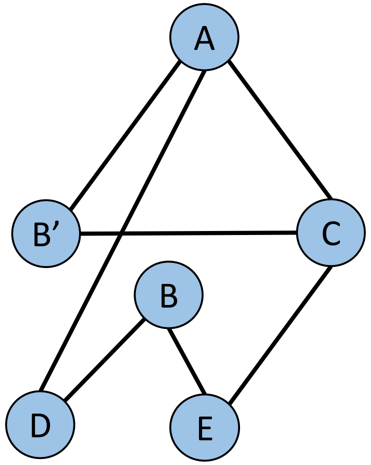
$$\lambda_{B\to C}(a,c) = \max_b f(a,b)\ f(b,c)$$

$$\lambda_{B\to D}(d,e) = \max_b f(b,d)\ f(b,e)$$

$$\lambda_{C\to E}(a,e) = \max_c \ldots$$

Mini-Bucket Elimination

[Dechter & Rish 2003]



mini-buckets

bucket B: $\underbrace{f(a,b') \ f(b',c)} \quad \underbrace{f(b,d) \ f(b,e)}$

bucket C: $f(c,a) \ f(c,e) \ \lambda_{B \to C}(a,c)$

bucket 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: $f(a) \ \lambda_{E \to A}(a)$

U = upper bound

$$\lambda_{B \to C}(a,c) = \max_b f(a,b) \ f(b,c)$$

$$\lambda_{B \to D}(d,e) = \max_b f(b,d) \ f(b,e)$$

$$\lambda_{C \to E}(a,e) = \max_c \ldots$$

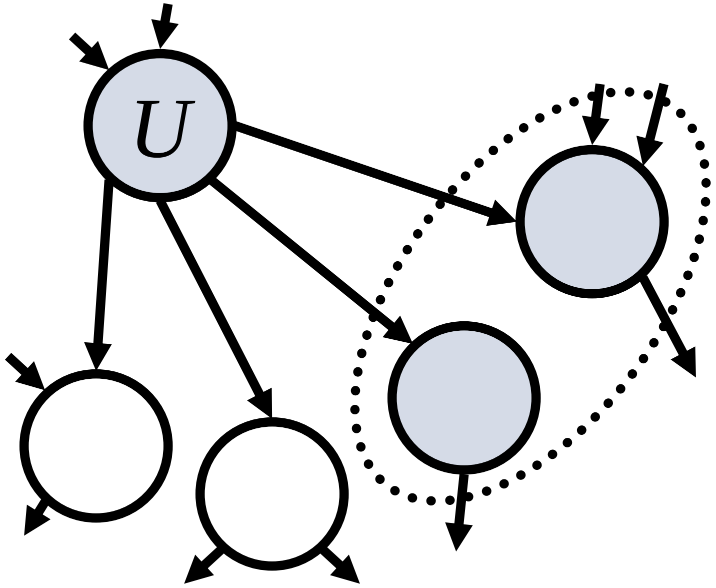
Can interpret process as "duplicating" B

[Kask et al. 2001, Geffner et al. 2007,
Choi et al. 2007, Johnson et al. 2007]

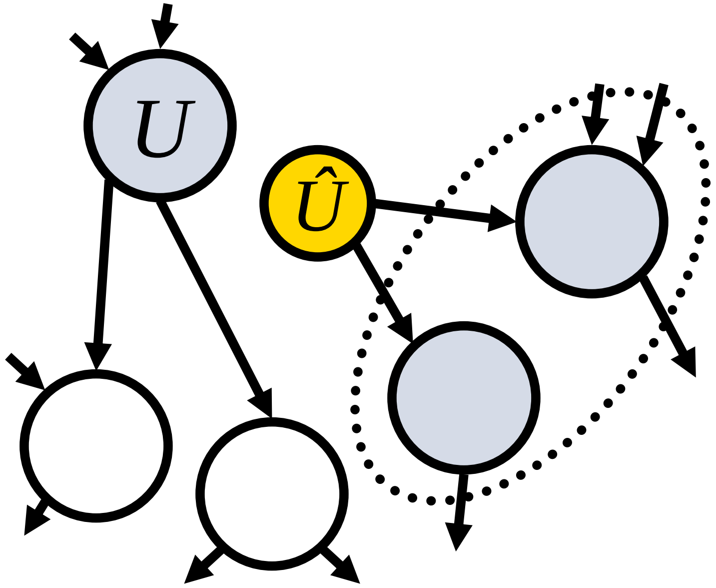
Semantics of Mini-Bucket: Splitting a Node

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

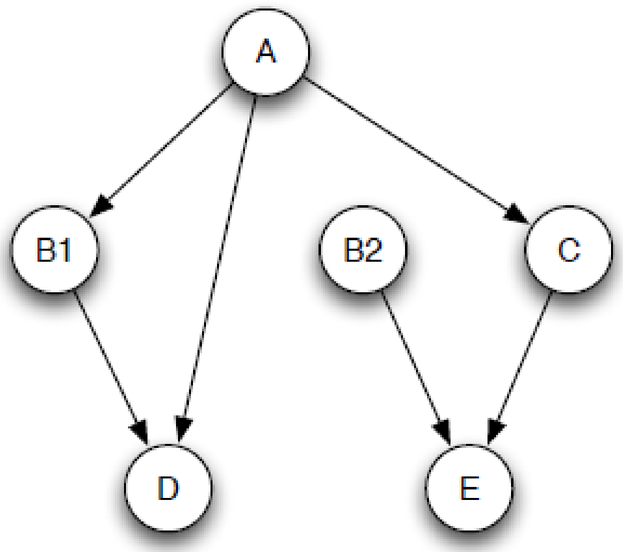
Before Splitting:
Network *N*



After Splitting:
Network *N*′



Relaxed Network Example



(a)

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

	Example: MBE-mpe(3)	max variables in a mini-bucket		BE-mpe
B:	$f(a,b)\ f(b,c)$ $f(b,d)\ f(b,e)$	3:	B:	$f(a,b)\ f(b,c)\ f(b,d)\ f(b,e)$
C:	$\lambda_{B\to C}(a,c)\ f(c,a)\ f(c,e)$	3:	C:	$\lambda_{B\to C}(a,c,d,e)\ f(c,a)\ f(c,e)$
D:	$f(a,d)\ \lambda_{B\to D}(d,e)$	3:	D:	$f(a,d)\ \lambda_{C\to D}(a,d,e)$
E:	$\lambda_{C\to E}(a,e)\ \lambda_{D\to E}(a,e)$	2:	E:	$\lambda_{D\to E}(a,e)$
A:	$f(a)\ \lambda_{E\to A}(a)$	1:	A:	$f(a)\ \lambda_{E\to A}(a)$
	U = Upper bound $w^* = 2$			**OPT** $w^* = 4$

[Dechter and Rish, 1997]

Mini-Bucket Decoding

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

$$\hat{c} = \arg\min_c \lambda_{B\to C}(\hat{a}, c) + f(c, \hat{a}) + f(c, \hat{e})$$

$$\hat{d} = \arg\min_d f(\hat{a}, d) + \lambda_{B\to D}(d, \hat{e})$$

$$\hat{e} = \arg\min_e \lambda_{C\to E}(\hat{a}, e) + \lambda_{D\to E}(\hat{a}, e)$$

$$\hat{a} = \arg\min_a f(a) + \lambda_{E\to A}(a)$$

mini-buckets

bucket B: $\underbrace{f(a,b)\ \ f(b,c)}_{\min_B \sum f(\cdot)}$ $\quad$ $\underbrace{f(b,d)\ \ f(b,e)}_{\min_B \sum f(\cdot)}$

bucket C: $\lambda_{B\to C}(a,c)\ \ f(c,a)\ \ f(c,e)$

bucket 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: $f(a)\ \ \lambda_{E\to A}(a)$

Greedy configuration = upper bound

L = lower bound

[Dechter and Rish, 2003]

(i,m)-Patitionings

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 $\mathcal{B} =< X, D, G, \mathcal{P} >$, where $\mathcal{P} = \{P_1, ..., P_n\}$; an ordering of the variables, $d = X_1, ..., X_n$; observations **e**.

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

1. **Initialize:** Generate an ordered partition of the conditional probability function, $bucket_1, \ldots, bucket_n$, where $bucket_i$ contains all functions whose highest variable is X_i. Put each observed variable in its bucket. Let ψ_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 $h_1, h_2, ..., h_j$ in $bucket_p$, do

- **If** (observed variable) $bucket_p$ 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, $Q' = \{Q_1, ..., Q_r\}$ of $h_1, h_2, ..., h_t$ in $bucket_p$.
- **for each** $Q_l \in Q'$ containing $h_{l_1}, ...h_{l_t}$, **do**

$$h_l \leftarrow max_{X_p} \prod_{j=1}^{t} h_{l_j} \tag{1.1}$$

Add h_l to the bucket of the largest-index in $scope(h_l)$.Put constants in $bucket_1$.

3. **Forward:**

- Generate an mpe upper bound cost by maximizing over X_1, the product in $bucket_1$. Namely $U \leftarrow max_{X_1} \psi_1 \prod_j h_{1_j}$.
- (Generate an approximate mpe tuple): Given $x_1^a, ..., x_{p-1}^a$, assign x_p^a to X_p that maximizes the product of all functions in $bucket_p$. $L \leftarrow P(x_1^a, ..., x_n^a)$

4. **Return** U and L and configuration: $\bar{x}^a = (x_1^a, ..., x_n^a)$

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 $h^p \leq g^p_{Q'} \leq g^p_{Q''}$.*

Properties of MBE(i)

- **Complexity**: O(r exp(i)) time and O(exp(i)) space
- Yields a lower bound and an upper bound
- **Accuracy**: determined by upper/lower (U/L) bound
- Possible use of mini-bucket approximations
  - As anytime algorithms
  - As heuristics in search
- Other tasks (similar mini-bucket approximations)
  - Belief updating, Marginal MAP, MEU, WCSP, Max-CSP

  [Dechter and Rish, 1997], [Liu and Ihler, 2011], [Liu and Ihler, 2013]

Anytime Approximation

Algorithm anytime-mpe(ϵ)
Input: Initial values of i and m, i_0 and m_0; increments i_{step} and m_{step}, and desired approximation error ϵ.
Output: U and L

1. **Initialize:** $i = i_0, m = m_0$.
2. **do**
3. run *mbe-mpe(i,m)*
4. $U \leftarrow$ upper bound of *mbe-mpe(i,m)*
5. $L \leftarrow$ lower bound of *mbe-mpe(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: $i \leftarrow i + i_{step}$ and $m \leftarrow m + m_{step}$
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:

$$\sum_X f(x) \bullet g(x) \leq \sum_X f(x) \bullet \sum_X g(x)$$
$$\sum_X f(x) \bullet g(x) \leq \sum_X f(x) \bullet \max_X g(X)$$

- 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 mini-buckets will be maximized, sum mini-buckets will be sum-max. Approximates BE-map.

Algorithm MBE-bel-max(i,m)

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

Output: an upper bound on $P(X_1, \bar{e})$ and an upper bound on $P(e)$.

1. **Initialize:** Partition $P = \{P_1, ..., P_n\}$ into buckets $bucket_1, \ldots, bucket_n$, where $bucket_k$ contains all CPTs $h_1, h_2, ..., h_t$ 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 $bucket_1$).
- **Else** for $h_1, h_2, ..., h_t$ in $bucket_k$ Generate an (i, m)-partitioning, $Q^{'} = \{Q_1, ..., Q_r\}$. **For each** $Q_l \in Q^{'}$, containing $h_{l_1}, ...h_{l_t}$, do

$$h_l \leftarrow \sum_{X_k} \Pi_{j=1}^{t} h_{1_j},, \; if \; l = 1$$

$$h_l \leftarrow max_{X_k} \Pi_{j=1}^{t} h_{l_j}, \; if \; k \neq 1$$

  Add h_l to the bucket of the highest-index variable in its scope $\bigcup_{j=1}^{t} scope(h_{l_j}) - \{X_k\}$. (put constant functions in $bucket_1$).

3. **Return** $P'(\bar{x_1}, e) \leftarrow$ the product of functions in the bucket of X_1, which is an upper bound on $P(x_1, \bar{e})$.
$P'(e) \leftarrow \sum_{x_1} P'(\bar{x_1}, e)$, which is an upper bound on probability of evidence.

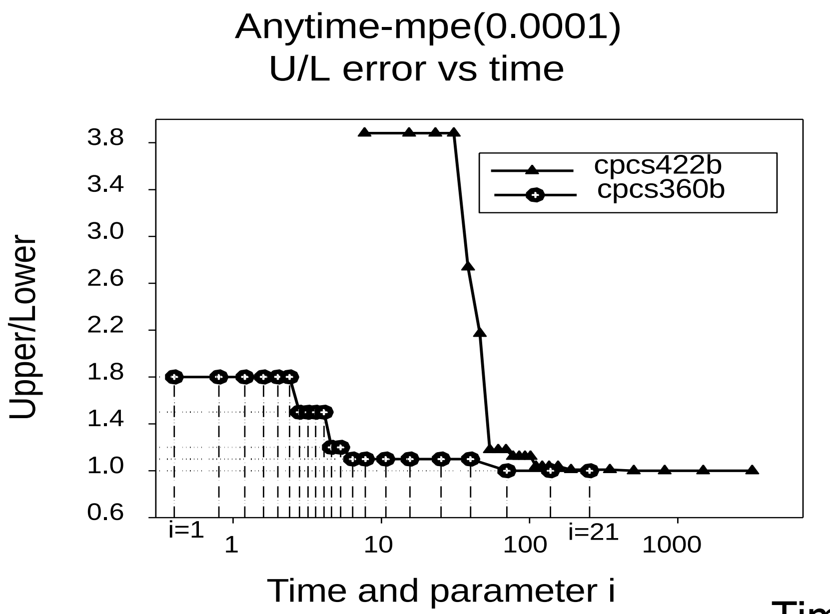
Figure 8.5: Algorithm *MBE-bel-max(i,m)*.

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



Algorithm	Time (sec) cpcs360	cpcs422
elim-mpe	115.8	1697.6
anytime-mpe(ε), $\varepsilon = 10^{-4}$	70.3	505.2
anytime-mpe(ε), $\varepsilon = 10^{-1}$	70.3	110.5

Outline

- Mini-bucket elimination
- Weighted Mini-bucket
- Mini-clustering
- Re-parameterization, cost-shifting
- Iterative Belief propagation
- Iterative-join-graph propagation

The Power Sum and Holder Inequality

The power sum is defined as follows:

$$\sum_x^w f(x) = (\sum_x f(x)^{\frac{1}{w}})^w \tag{1.2}$$

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

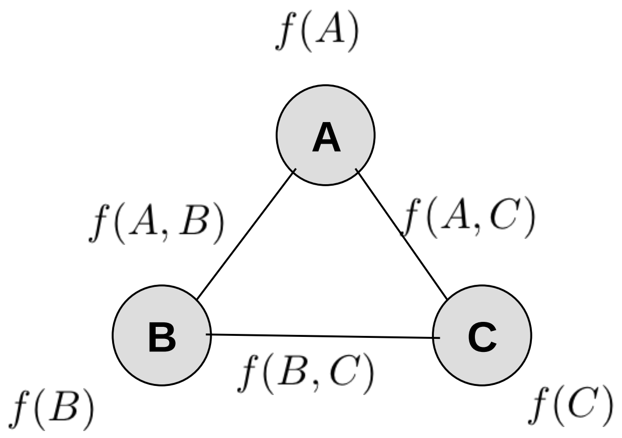
[Holder inequality] Let $f_i(x)$, $i = 1..r$ be a set of functions and $w_1, ..., w_r$ b a set of of non-zero weights, s.t., $w = \sum_{i=1}^r w_i$ then,

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

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

$$\sum_i a_i b_i \leq (\sum_i a_i) \max_i(b_i)$$

- Lower bound

$$\sum_i a_i b_i \geq (\sum_i a_i) \min_i(b_i)$$

(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 $w_1 + w_2 = 1$ $w_1 > 0, w_2 > 0$

- When $\frac{a_i^{1/w_1}}{\sum a_i^{1/w_1}} = \frac{b_i^{1/w_2}}{\sum b_i^{1/w_2}}$, the equality is achieved.

(Qiang Liu slides)

G. H. Hardy, J. E. Littlewood and G. Pólya, *Inequalities*, *Cambridge Univ. Press, London and New York, 1934.*

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)

G. H. Hardy, J. E. Littlewood and G. Pólya, *Inequalities, Cambridge Univ. Press, London and New York, 1934.*

Mini-Bucket as Holder Inequality

$w_1 = 1; w_2 \rightarrow +0$ (mbe-bel-max)

$$\sum_i a_i b_i \leq (\sum_i a_i) \max_i(b_i)$$

- $w_1 = 1, w_2 \rightarrow -0$ (mbe-bel-min)

$$\sum_i a_i b_i \geq (\sum_i a_i) \min_i(b_i)$$

(Qiang Liu slides)

Weighted Mini-Bucket

(for summation)

Exact bucket elimination:

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

$$\leq \left[\sum_b^{w_1} f(a,b)f(b,c)\right]\cdot\left[\sum_b^{w_2} f(b,d)f(b,e)\right]$$

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

(mini-buckets)

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

is the weighted or "power" sum operator

$$\sum_x^w f_1(x)f_2(x) \leq \left[\sum_x^{w_1} f_1(x)\right]\left[\sum_x^{w_2} f_2(x)\right]$$

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

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

mini-buckets $\sum_x^w \prod f(x)$

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

bucket C: $\lambda_{B\to C}(a,c)\ \ f(a,c)\ \ f(c,e)$

bucket 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: $f(a)\ \ \lambda_{E\to A}(a)$

U = upper bound

[Liu and Ihler, 2011]

Algorithm Weighted WMBE(i,m), $(w_1, ... w_n)$

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

Output: an upper bound on $\sum_{\mathbf{X}}^{w} \prod_{i=1}^{n} P_i$

1. **Initialize:** Partition $P = \{P_1, ..., P_n\}$ into buckets $bucket_1$, ..., $bucket_n$, where $bucket_k$ contains all CPTs $h_1, h_2, ..., h_t$ 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 $bucket_1$).
   - Else for $h_1, h_2, ..., h_t$ in $bucket_k$ Generate an (i, m)-partitioning, $Q' = \{Q_1, ..., Q_r\}$. Select a set of weights $w_1, ... w_r$ s.t $\sum_l w_l = w$.
     For each $Q_l \in Q'$, containing $h_{l_1}, ... h_{l_t}$, do

     $$h_l \leftarrow \sum_{X_k}^{w_l} \prod_{j=1}^{t} h_{l_j} = (\sum_{X_k} \prod_{j=1}^{t} (h_{l_j})^{w_l})^{\frac{1}{w_l}}$$

     Add h_l to the bucket of the highest-index variable in its scope (and put constant functions in $bucket_1$).

3. **Return** $U) \leftarrow$ the weighted product of functions in the bucket of X_1, which is an upper bound on $P(x_1, \mathrm{e})$.

Choosing the weights

- $w_1 = 1/2, w_2 = 1/2$ (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?

(Qiang Liu slides)

Weighted-mini-bucket for Marginal Map

MB and WMB for Marginal MAP

$\max_Y \Sigma_{X \backslash Y} \Pi_j P_j$

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

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

$$(w_1 + w_2 = 1)$$

$\vdots$

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

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

Marginal MAP

mini-buckets

Σ_B	bucket B:	$f(a,b)\ f(b,c)$ $\quad$ $f(b,d)\ f(b,e)$
Σ_C	bucket C:	$\lambda_{B\to C}(a,c)\ f(a,c)\ f(c,e)$
$\max_D$	bucket D:	$f(a,d)\ \lambda_{B\to D}(d,e)$
$\max_E$	bucket E:	$\lambda_{C\to E}(a,e)\ \lambda_{D\to E}(a,e)$
$\max_A$	bucket A:	$f(a)\ \lambda_{E\to A}(a)$

U = 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]

MBE-map

Process max buckets
With max mini-buckets
And sum buckets with sum
Mini-bucket and max
mini-buckets

Algorithm MBE-map(i,m)

Input: A Bayesian network $\mathcal{B} = \langle \mathbf{X}, \mathbf{D}, \mathbf{P}_G, \prod \rangle$, $P = \{P_1, ..., P_n\}$; a subset of hypothesis variables $A = \{A_1, ..., A_k\}$; an ordering of the variables, d, in which the A's are first in the ordering; observations $\mathbf{e}$.

Output: An upper bound on the map and a and a suboptimal solution $A = \bar{a}_k^a$.

1. **Initialize:** Partition $P = \{P_1, ..., P_n\}$ into $bucket_1, \ldots, bucket_n$, where $bucket_i$ 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 $bucket_p$ do

- **If** (observed variable) $bucket_p$ contains the observation $X_p = x_p$, assign $X_p = x_p$ to each h_i and and put each in appropriate bucket.
- **Else** for $h_1, h_2, ..., h_j$ in $bucket_p$ generate an (i,m)-partitioning, $Q' = \{Q_1, ..., Q_r\}$.
- **If** $X_P \notin A$ 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 $\sum_i w_{p_i} = w_p$.
  foreach $Q_l \in Q'$, containing $h_{l_1}, ...h_{l_t}$, do

$$h_l \leftarrow \sum_{X_k}^{w_{pl}} \prod_{j=1}^{t} h_{l_j} = (\sum_{X_k} (\prod_{j=1}^{t} h_{l_j})^{w_{pl}})^{\frac{1}{w_{pl}}}$$

  Add h_l 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 $bucket_p$. conditioned on earlier assignments.

4. **Return** An upper bound $U = max_{a_1} \prod_{h_i \in bucket_1} h_i$ on the map value, computed in the first bucket, and the assignment $\bar{a}_k^a = (a_1^a, ..., a_k^a)$.

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

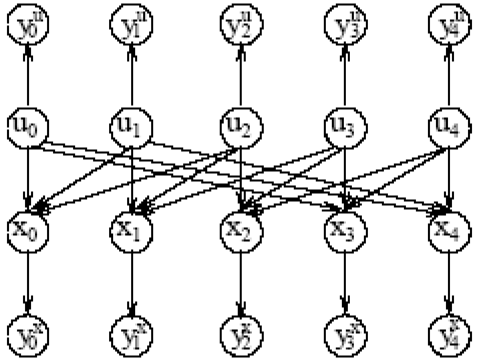


Figure 7.7: Belief network for a linear block code.

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 $Y = (\bar{y}^u, \bar{y}^x)$, which is the MAP task where U is the set of hypothesis variables, and $Y = (\bar{y}^u, \bar{y}^x)$ is the evidence. After processing the observed buckets we get the following bucket configuration (lower case y's are observed values):

$$\left.\begin{array}{l}
bucket(X_0) = P(y_0^x|X_0), P(X_0|U_0, U_1, U_2),\\
bucket(X_1) = P(y_1^x|X_1), P(X_1|U_1, U_2, U_3),\\
bucket(X_2) = P(y_2^x|X_2), P(X_2|U_2, U_3, U_4),\\
bucket(X_3) = P(y_3^x|X_3), P(X_3|U_3, U_4, U_0),\\
bucket(X_4) = P(y_4^x|X_4), P(X_4|U_4, U_0, U_1),\\
bucket(U_0) = P(U_0), P(y_0^u|U_0),\\
bucket(U_1) = P(U_1), P(y_1^u|U_1),\\
bucket(U_2) = P(U_2), P(y_2^u|U_2),\\
bucket(U_3) = P(U_3), P(y_3^u|U_3),\\
bucket(U_4) = P(U_4), P(y_4^u|U_4).
\end{array}\right\}\ \text{Initial partitioning}$$

Processing by *mbe-map(4,1)* of the first top five buckets by summation and the rest by maximization, results in the following mini-bucket partitionings and function generation:

$$
\begin{aligned}
&bucket(X_0) = \{P(y_0^x|X_0), P(X_0|U_0,U_1,U_2)\},\\
&bucket(X_1) = \{P(y_1^x|X_1), P(X_1|U_1,U_2,U_3)\},\\
&bucket(X_2) = \{P(y_2^x|X_2), P(X_2|U_2,U_3,U_4)\},\\
&bucket(X_3) = \{P(y_3^x|X_3), P(X_3|U_3,U_4,U_0)\},\\
&bucket(X_4) = \{P(y_4^x|X_4), P(X_4|U_4,U_0,U_1)\},\\
&bucket(U_0) = \boxed{\{P(U_0), P(y_0^u|U_0), h^{X_0}(U_0,U_1,U_2)\}}, \boxed{\{h^{X_3}(U_3,U_4,U_0)\}}, \boxed{\{h^{X_4}(U_4,U_0,U_1)\}},\\
&bucket(U_1) = \{P(U_1), P(y_1^u|U_1), h^{X_1}(U_1,U_2,U_3), h^{U_0}(U_1,U_2)\}, \{h^{U_0}(U_4,U_1)\},\\
&bucket(U_2) = \{P(U_2), P(y_2^u|U_2), h^{X_2}(U_2,U_3,U_4), h^{U_1}(U_2,U_3)\},\\
&bucket(U_3) = \{P(U_3), P(y_3^u|U_3), h^{U_0}(U_3,U_4), h^{U_1}(U_3,U_4), h^{U_2}(U_3,U_4)\},\\
&bucket(U_4) = \{P(U_4), P(y_4^u|U_4), h^{U_1}(U_4), h^{U_3}(U_4)\}.
\end{aligned}
$$

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 $U \geq MAP = P(U, \bar{y}^u, \bar{y}^x)$ where $\bar{y}^u$ and $,\bar{y}^x$ are the observed outputs for the U's and the X's bits transmitted. In order to bound $P(U|\bar{e})$, where $\bar{e} = (\bar{y}^u, \bar{y}^x)$, we need $P(\bar{e})$ which is not available. Yet, again, in most cases we are interested in the ratio $P(U = \bar{u}_1|\bar{e})/P(U = \bar{u}_2|\bar{e})$ for competing hypotheses $U = \bar{u}_1$ and $U = \bar{u}_2$ 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 $P(U_1,\bar{e})/P(U_2,\bar{e})$. $\square$

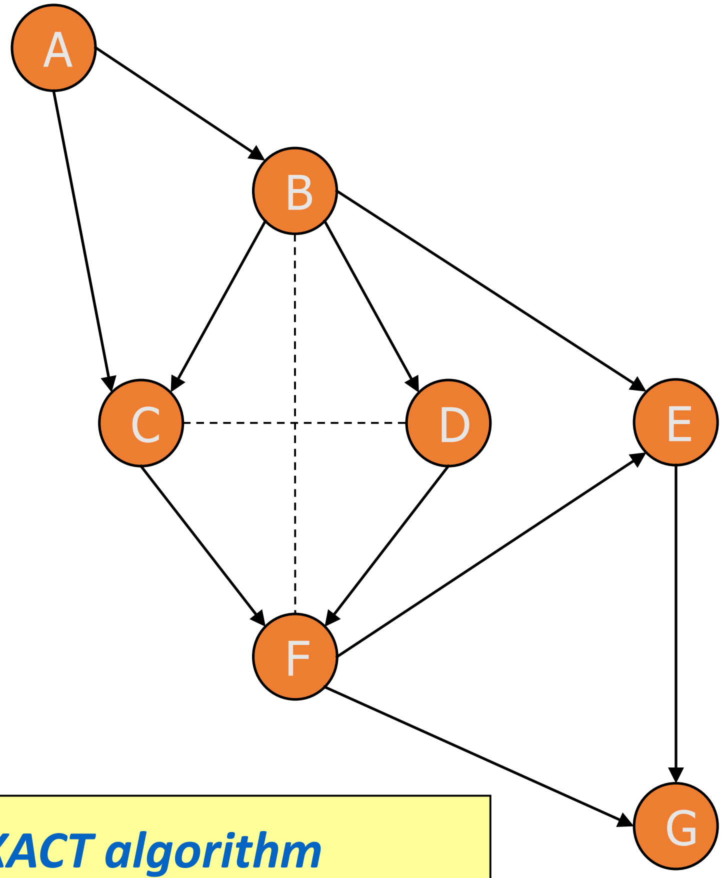
Complexity and Tractability of MBE(i,m)

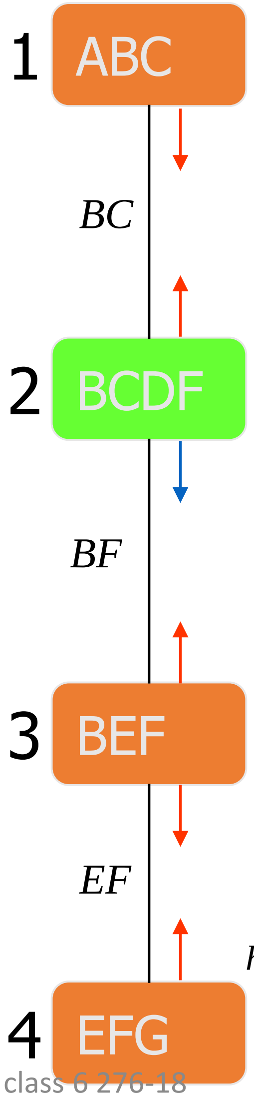
Theorem 7.6.1 *Algorithm* mbe(i,m) *takes $O(r \cdot exp(i))$ time and space, where r is the number of input functions*[2], *and where $|F|$ is the maximum scope of any input function, $|F| \leq i \leq n$. For $m = 1$, the algorithm is time and space $O(r \cdot exp(|F|))$.*

Outline

- Mini-bucket elimination
- Weighted Mini-bucket
- Mini-clustering
- Re-parameterization, cost-shifting
- Iterative Belief propagation
- Iterative-join-graph propagation

Join-Tree Clustering (Cluster-Tree Elimination)





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

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

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

$$h_{(3,2)}(b,f) = \sum_{e} p(e|b,f) \cdot h_{(4,3)}(e,f)$$

$$h_{(3,4)}(e,f) = \sum_{b} p(e|b,f) \cdot h_{(2,3)}(b,f)$$

$$h_{(4,3)}(e,f) = p(G = g_e | e,f)$$

EXACT algorithm

Time and space:

exp(cluster size)=
exp(treewidth)

Mini-Clustering

Split a cluster into mini-clusters => bound complexity

$$\{h_1,\dots,h_r,h_{r+1},\dots,h_n\}$$

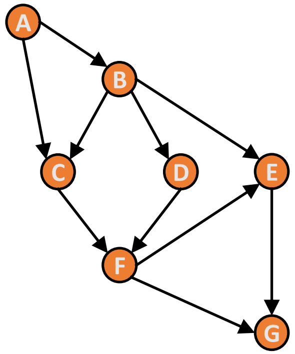
APPROXIMATE algorithm

$$\{h_1,\dots,h_r\} \qquad \{h_{r+1},\dots,h_n\}$$

$$\sum_{elim}\prod_{i=1}^{n} h_i \;\le\; \left(\sum_{elim}\prod_{i=1}^{r} h_i\right)\cdot\left(\sum_{elim}\prod_{i=r+1}^{n} h_i\right)$$

Exponential complexity decrease $O(e^n) \to O(e^{\mathrm{var}(r)}) + O(e^{\mathrm{var}(n-r)})$

Mini-Clustering, i-bound=3



1 **A B C**
p(a), p(b|a), p(c|a,b)

BC

$$h^1_{(1,2)}(b,c) = \sum_a p(a) \cdot p(b\,|\,a) \cdot p(c\,|\,a,b)$$

2 **B C D**
p(d|b), $h_{(1,2)}(b,c)$

C D F
p(f|c,d)

BF

$$h^1_{(2,3)}(b) = \sum_{c,d} p(d\,|\,b) \cdot h^1_{(1,2)}(b,c)$$

$$h^2_{(2,3)}(f) = \max_{c,d} p(f\,|\,c,d)$$

3 **B E F**
p(e|b,f),
$h^1_{(2,3)}(b)$, $h^2_{(2,3)}(f)$

EF

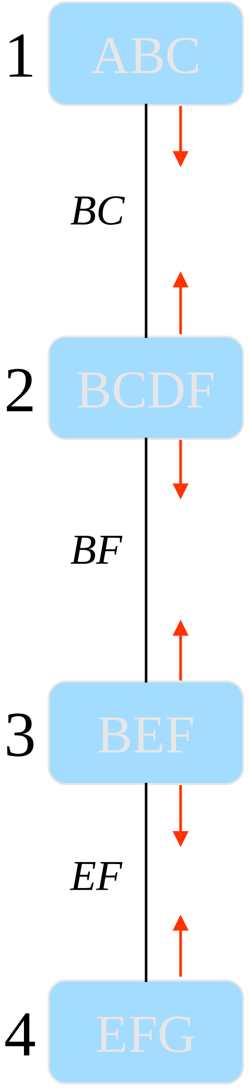
4 **E F G**
p(g|e,f)

APPROXIMATE algorithm

Time and space:
exp(i-bound)

Number of variables in a mini-cluster

Mini-Clustering - Example



$$H_{(1,2)} \quad \boxed{h^1_{(1,2)}(b,c) := \sum_a p(a)\cdot p(b\,|\,a)\cdot p(c\,|\,a,b)}$$

$$H_{(2,1)} \quad \boxed{\begin{array}{l} h^1_{(2,1)}(b) := \sum_{d,f} p(d\,|\,b)\cdot h^1_{(3,2)}(b,f) \\ h^2_{(2,1)}(c) := \max_{d,f} p(f\,|\,c,d) \end{array}}$$

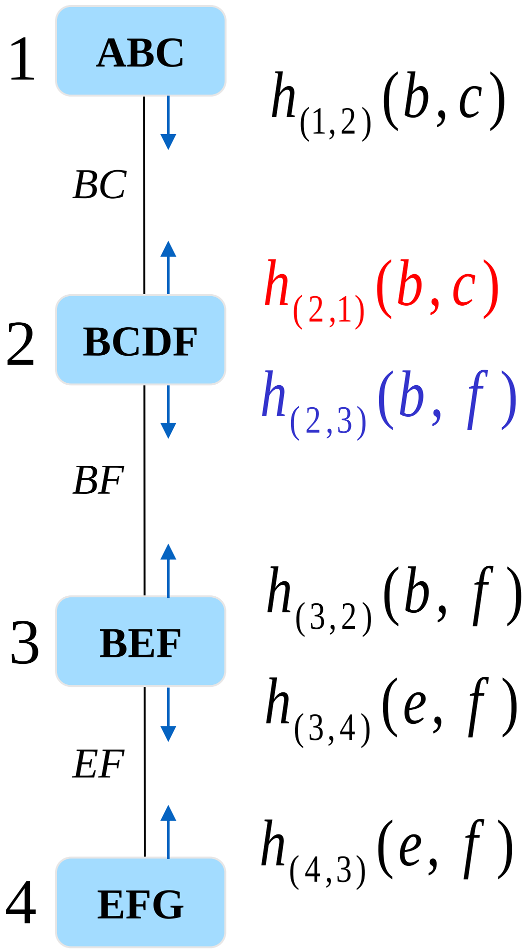
$$H_{(2,3)} \quad \boxed{\begin{array}{l} h^1_{(2,3)}(b) := \sum_{c,d} p(d\,|\,b)\cdot h^1_{(1,2)}(b,c) \\ h^2_{(2,3)}(f) := \max_{c,d} p(f\,|\,c,d) \end{array}}$$

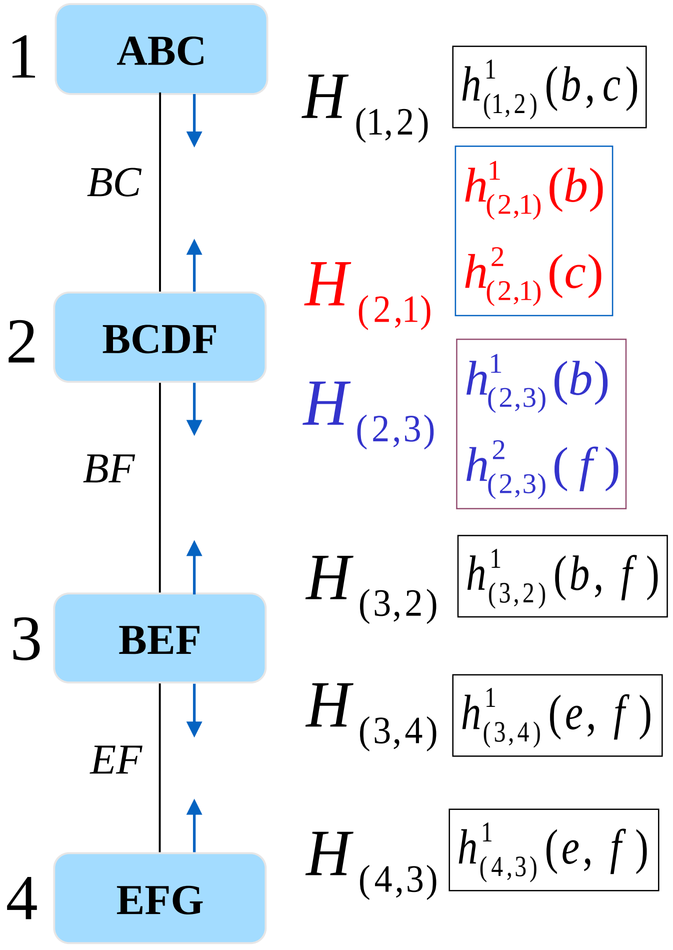
$$H_{(3,2)} \quad \boxed{h^1_{(3,2)}(b,f) := \sum_e p(e\,|\,b,f)\cdot h^1_{(4,3)}(e,f)}$$

$$H_{(3,4)} \quad \boxed{h^1_{(3,4)}(e,f) := \sum_b p(e\,|\,b,f)\cdot h^1_{(2,3)}(b)\cdot h^2_{(2,3)}(f)}$$

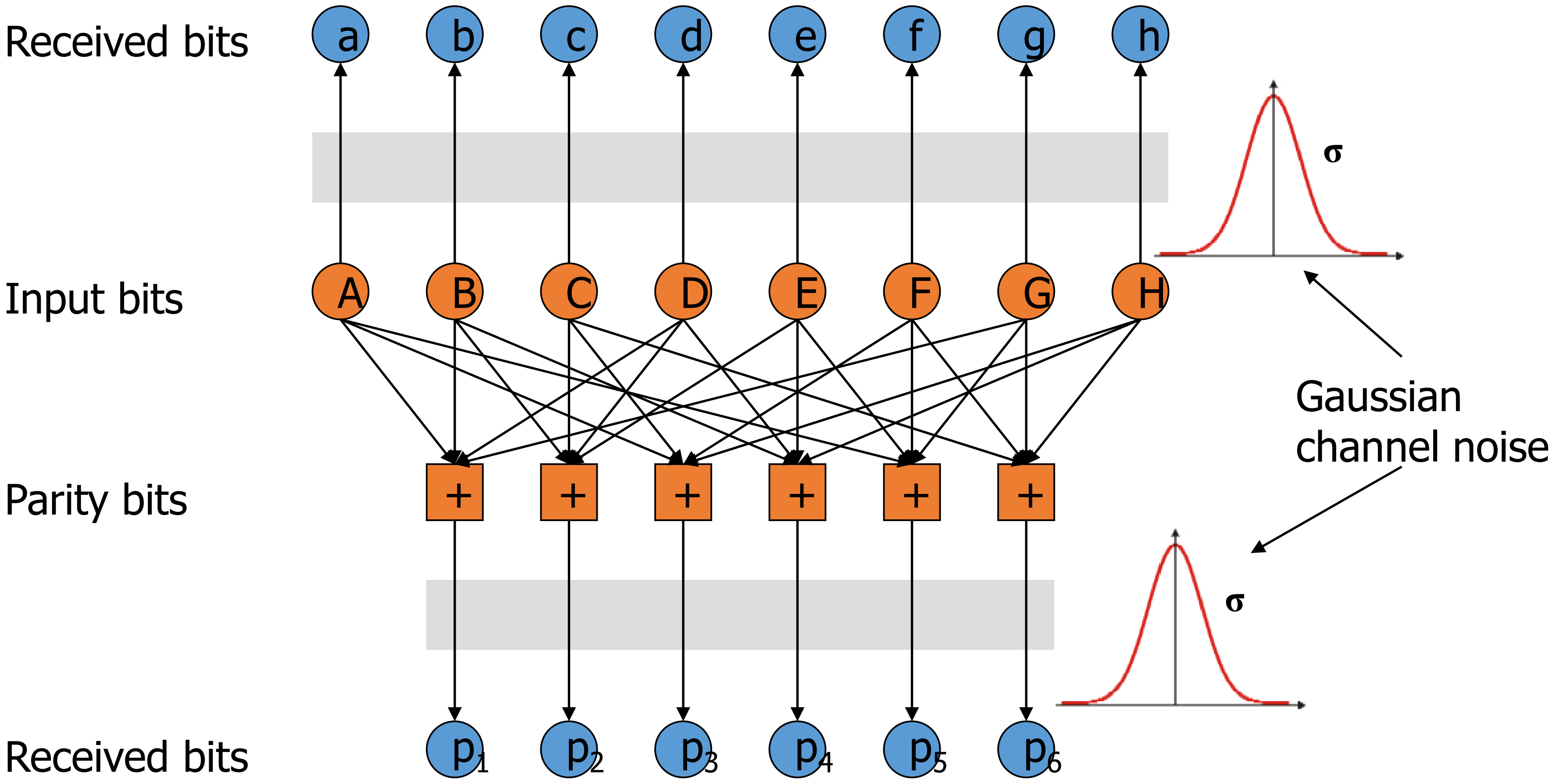
$$H_{(4,3)} \quad \boxed{h^1_{(4,3)}(e,f) := p(G = g_e\,|\,e,f)}$$

Cluster Tree Elimination vs. Mini-Clustering



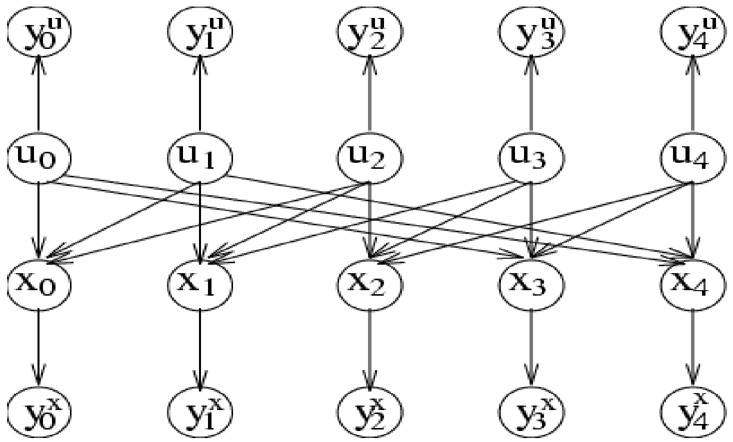


Linear Block Codes



Probabilistic decoding

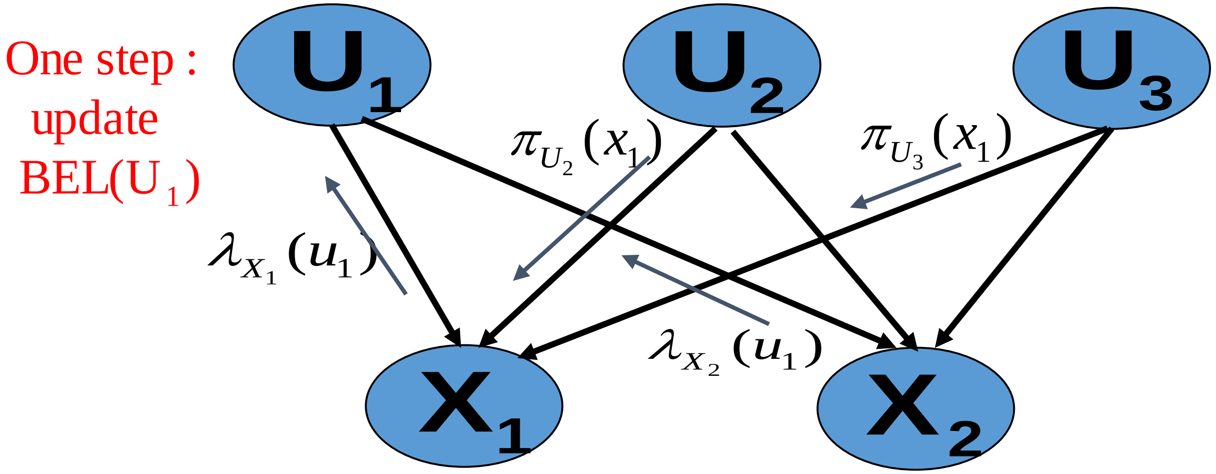
Error-correcting linear block code



State-of-the-art:
approximate algorithm – iterative belief propagation (IBP)
(Pearl's poly-tree algorithm applied to loopy networks)

Iterative Belief Proapagation

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

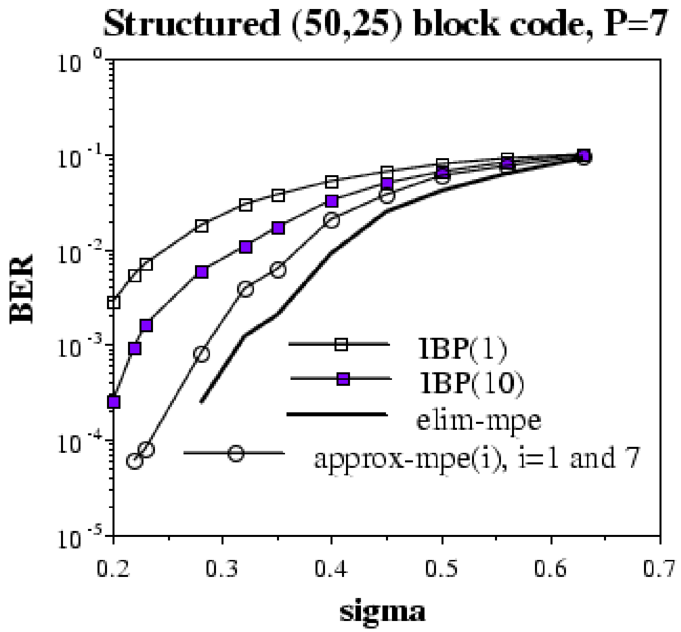


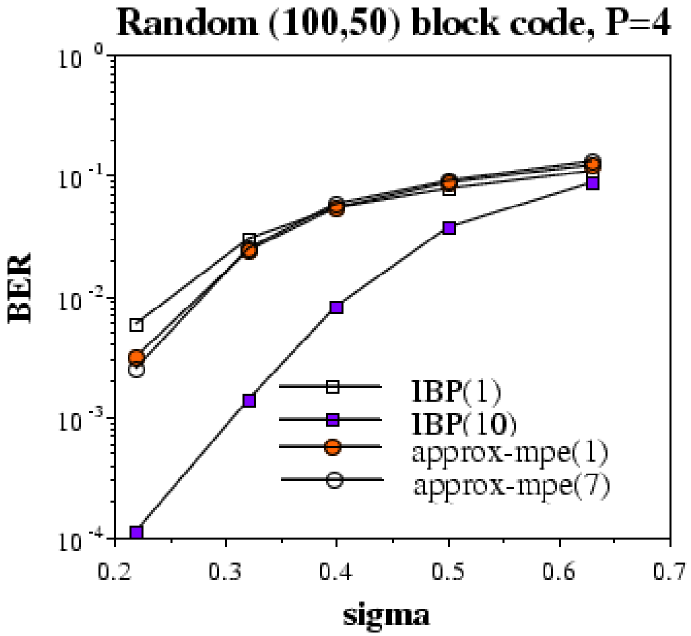
- No guarantees for convergence
- Works well for many coding networks

MBE-mpe vs. IBP

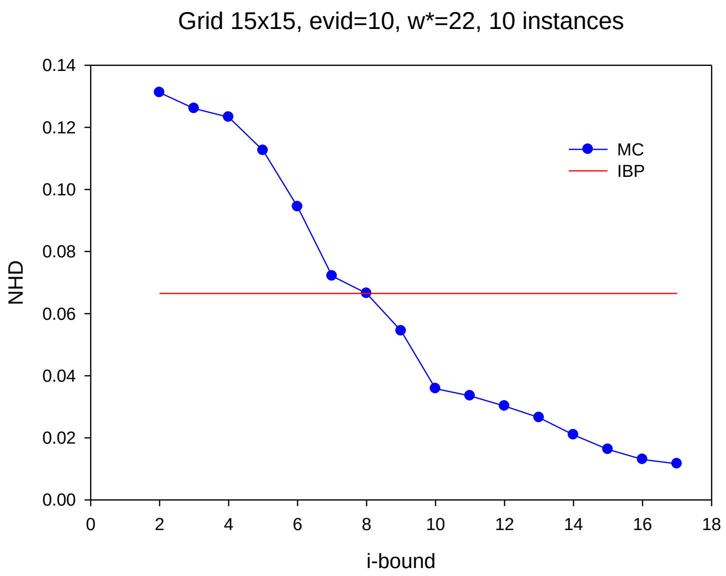
approx - mpe is better on low - w * codes
IBP is better on randomly generated (high - w *) codes

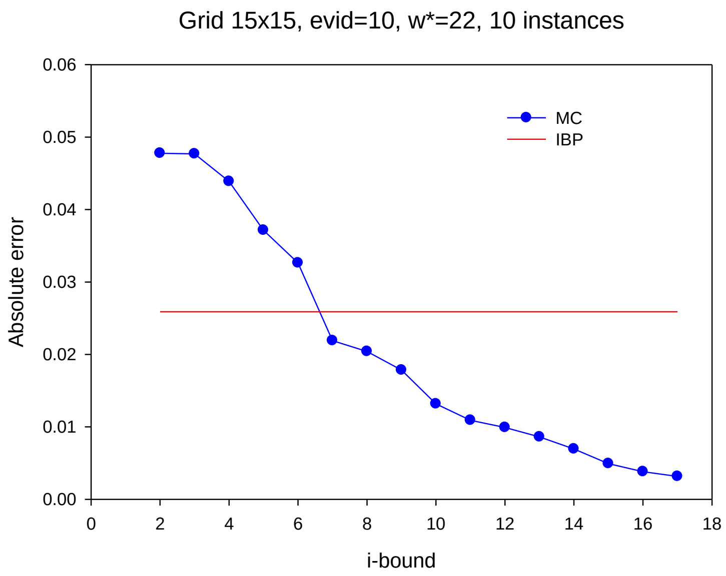
Bit error rate (BER) as a function of noise (sigma):

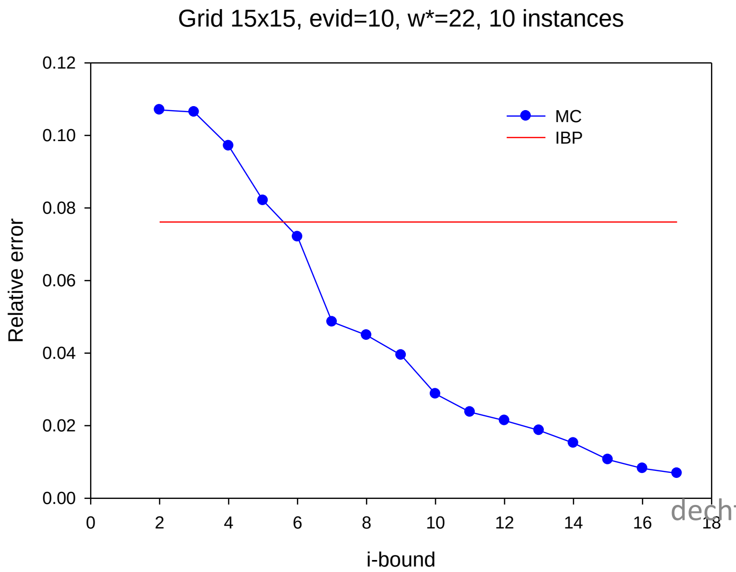


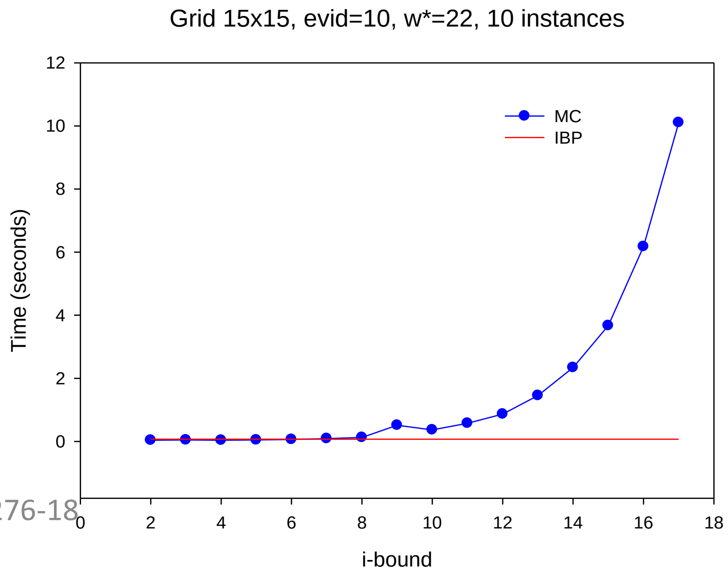


Grid 15x15 - 10 evidence





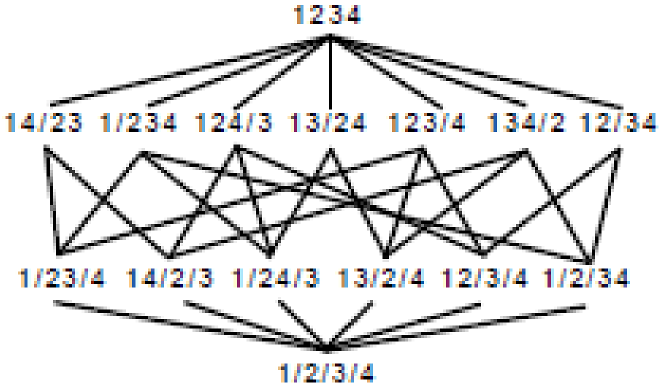




Heuristics for partitioning

(Dechter and Rish, 2003, Rollon and Dechter 2010)

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



Partitioning lattice of bucket $\{f_1, f_2, f_3, f_4\}$.

- *Log relative error*:

$$RE(f,h) = \sum_t (\log(f(t)) - \log(h(t)))$$

- *Max log relative error*:

$$MRE(f,h) = \max_t \{\log(f(t)) - \log(h(t))\}$$

Use greedy heuristic derived from a distance function to decide which functions go into a single mini-bucket

Greedy Scope-based Partitioning

Procedure **Greedy Partitioning**

Input: $\{h_1, \ldots, h_k\}$, *i-bound*;
Output: A partitioning $mb(1), \ldots, mb(p)$ such that every $mb(i)$ contains at most *i-bound* variables;
1. Sort functions by the size of their scopes. Let $\{h_1, \ldots, h_k\}$ be the sorted array of functions, with h_1 having the largest scope.
2. **for** $i = 1$ to k
   if h_i can be placed in existing mini-buckets without making the scope greater than the *i-bound*, place it in the one with the most functions.
   else create a new mini-bucket an place h_i in it.
 endfor

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 mini-bucket 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 from left to right. Then, each mini-bucket is absorbed into the left-most mini-bucket with whom it can be merged.

The time complexity of Partition(B, i) , where B is the bucket to be partitioned, and |B|,the number of functions in the bucket, using the SCP heuristic is $O(|B| \log (|B|) + |B|$^2) .

The scope-based
heuristic is is quite fast, its shortcoming is that it does not consider the actual information in the functions.

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 ∃Q' ∈ ch(Q) which is a i-partition
Q ← arg min_Q' {h(Q → Q')} among child i-partitions of Q;
3.Return Q;
```

Figure 8.13: Greedy partitioning

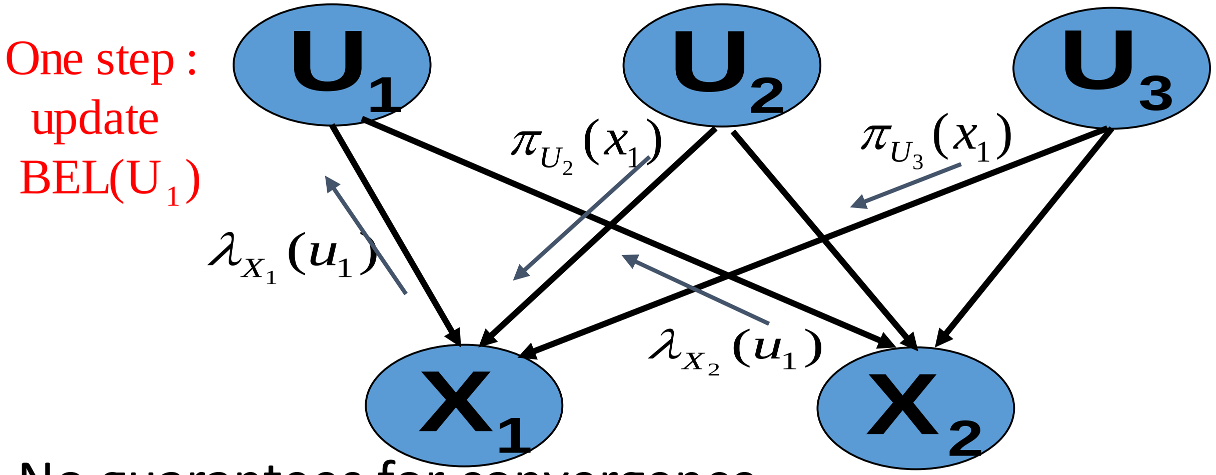
Proposition 8.6.5 *The time complexity of* `GreedyPartition` *is* $O(|B| \times T)$ *where* $O(T)$ *is the time complexity of selecting the min child partition according to* h.

Outline

- Mini-bucket elimination
- Weighted Mini-bucket
- Mini-clustering
- Iterative Belief propagation
- Iterative-join-graph propagation
- Re-parameterization, cost-shifting

Iterative Belief Proapagation

- 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

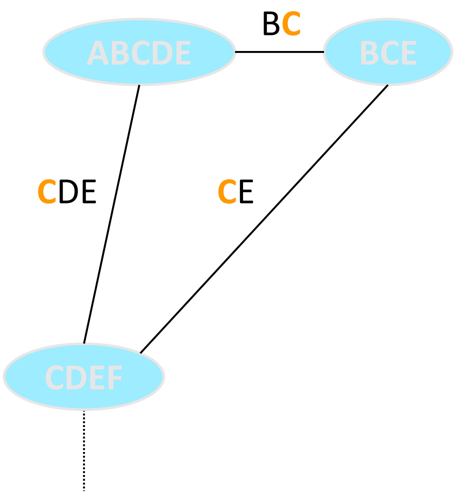
Iterative Join Graph Propagation

- Loopy Belief Propagation
  - Cyclic graphs
  - Iterative
  - Converges fast in practice (no guarantees though)
  - Very good approximations (e.g., turbo decoding, LDPC codes, SAT – survey propagation)
- Mini-Clustering(i)
  - Tree decompositions
  - Only two sets of messages (inward, outward)
  - Anytime behavior – can improve with more time by increasing the i-bound
- We want to combine:
  - Iterative virtues of Loopy BP
  - Anytime behavior of Mini-Clustering(i)

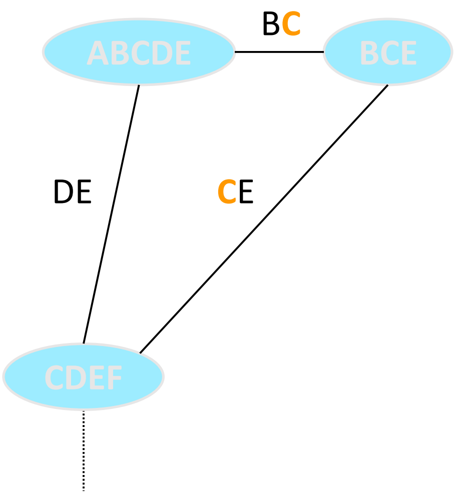
IJGP - The basic idea

- Apply Cluster Tree Elimination to any *join-graph*
- We commit to graphs that are *I-maps*
- Avoid cycles as long as I-mapness is not violated
- Result: use *minimal arc-labeled* join-graphs

Minimal Arc-Labeled Decomposition



a) Fragment of an
arc-labeled join-graph



a) Shrinking labels to make it a
minimal arc-labeled join-graph

- Use a DFS algorithm to eliminate cycles relative to each variable

Minimal arc-labeled join-graph

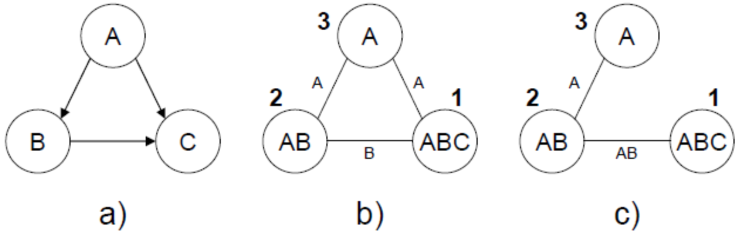


Figure 1.17: a) A belief network; b) A dual join-graph with singleton labels; c) A dual join-graph which is a join-tree

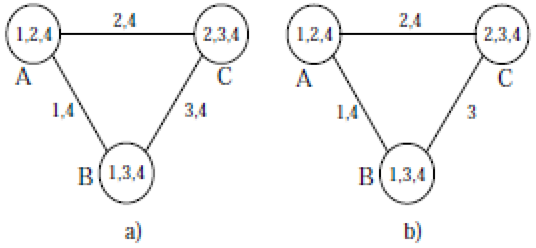
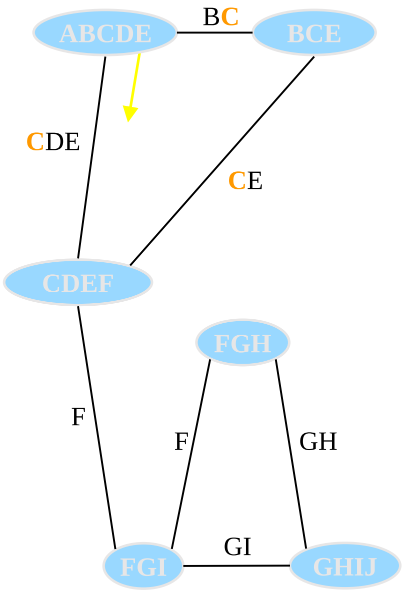
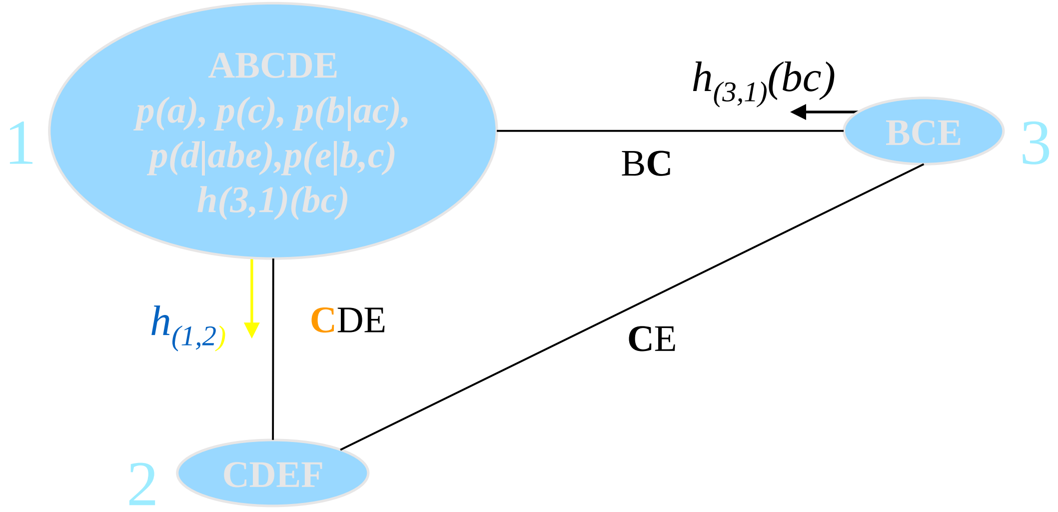


Figure 1.15: An arc-labeled decomposition

Message propagation





Minimal arc-labeled:

sep(1,2)={D,E}

elim(1,2)={A,B,C}

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

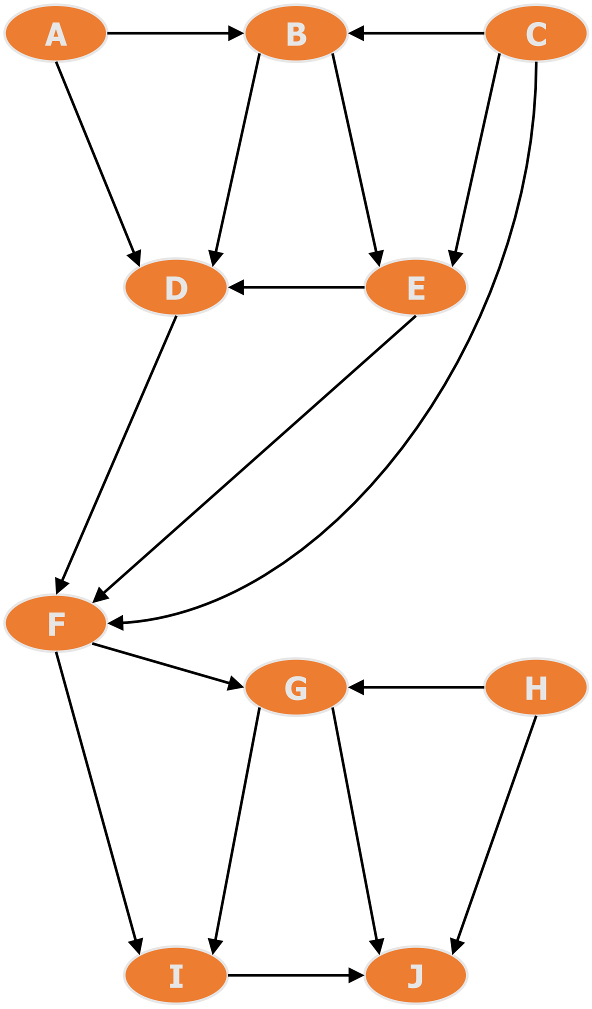
Non-minimal arc-labeled:

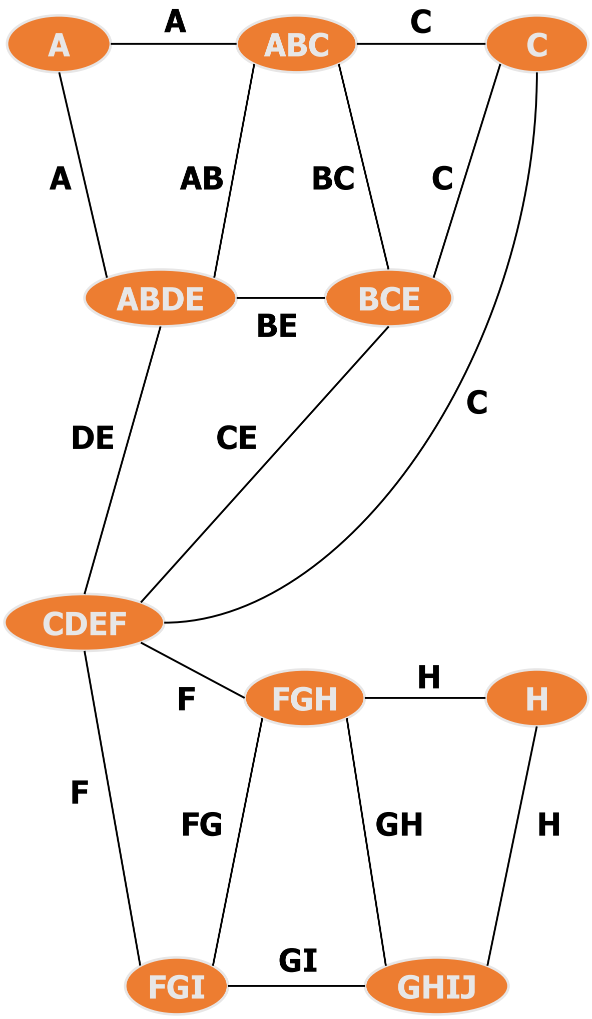
sep(1,2)={C,D,E}

elim(1,2)={A,B}

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

IJGP - Example

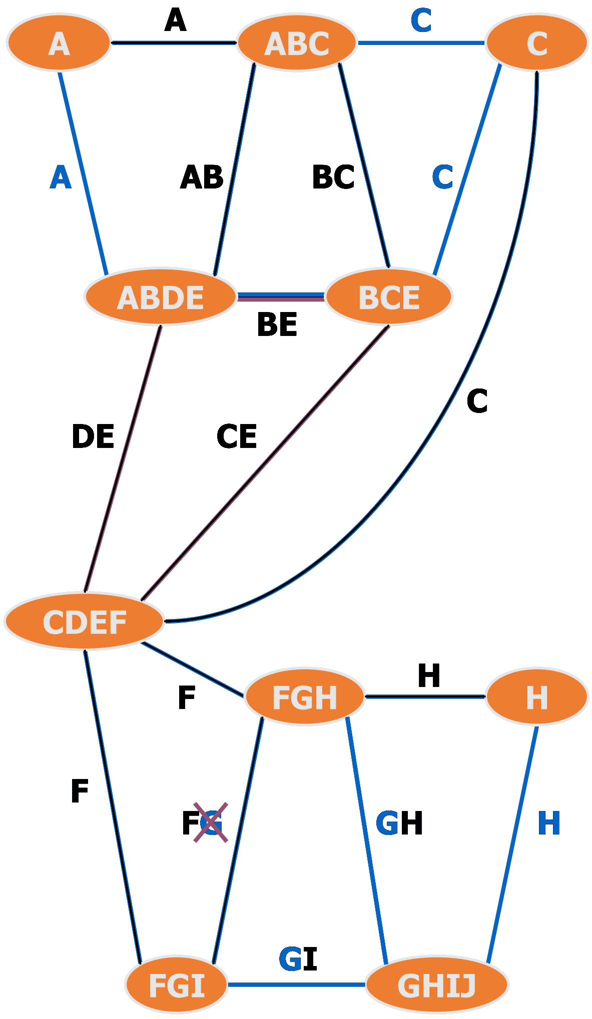




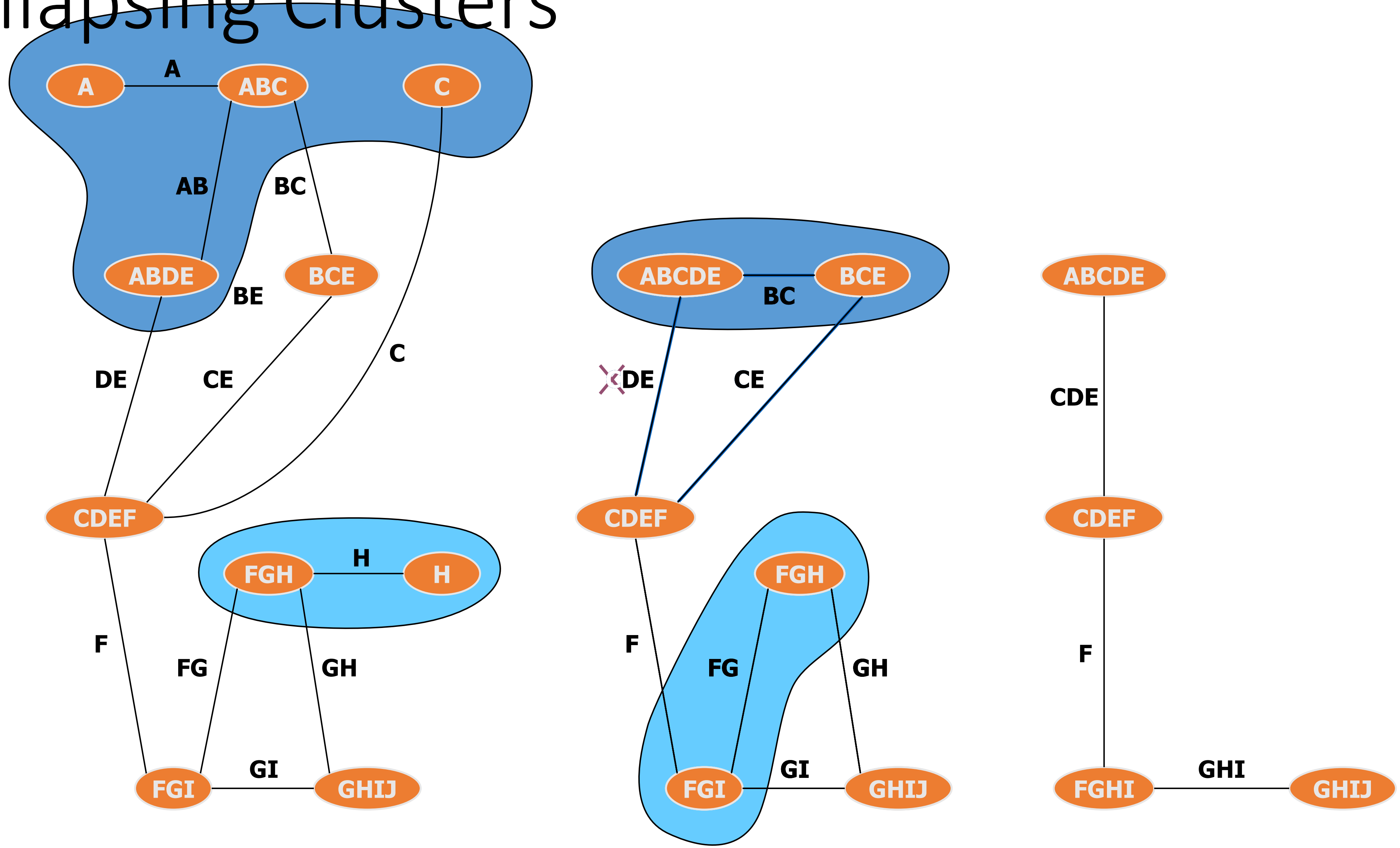
Belief network Loopy BP graph

Arc-Minimal Join-Graph

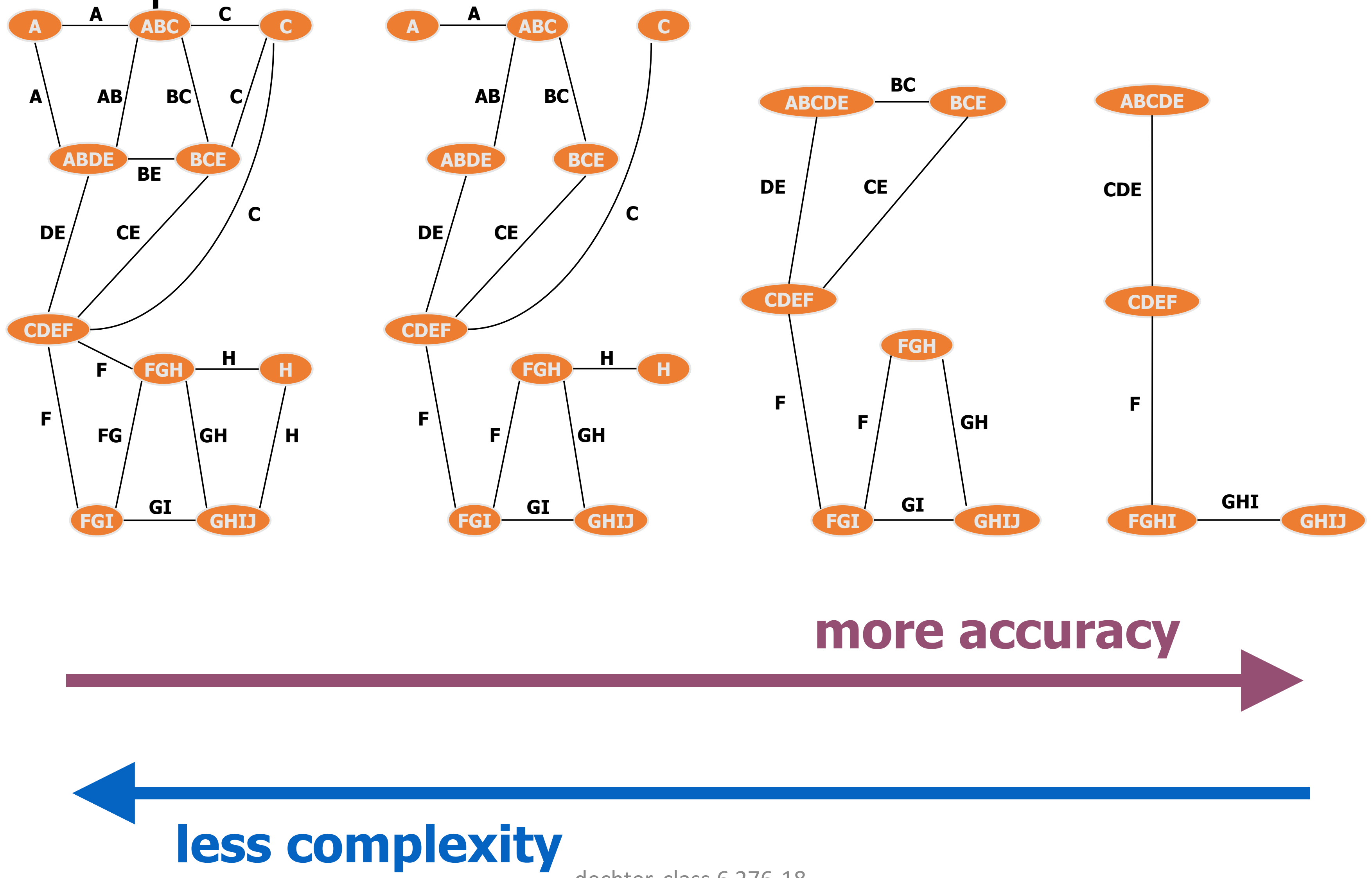
Arcs labeled with
any single variable
should form a TREE



Collapsing Clusters



Join-Graphs

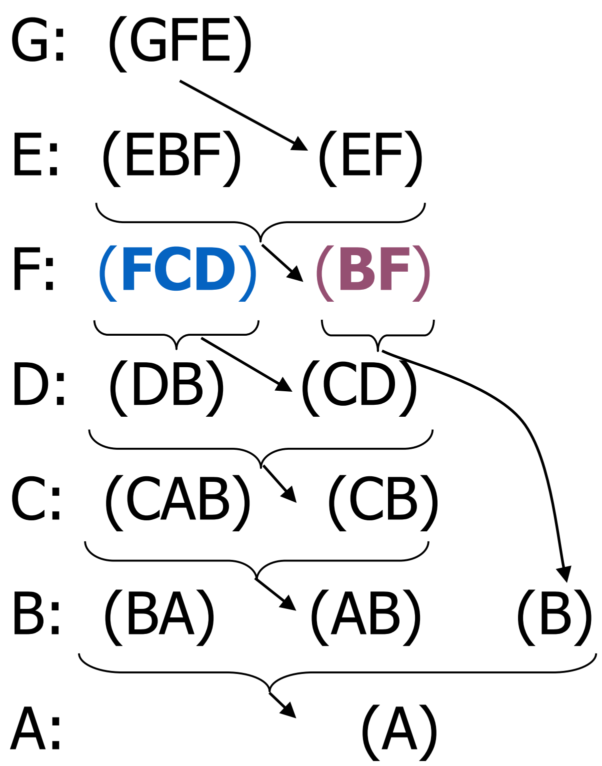

A
ABC
C
ABDE
BCE
CDEF
FGH
H
FGI
GHIJ
A
C
AB
BC
BE
DE
CE
F
FG
GH
GI
ABCDE
CDE
FGHI
GHI
more accuracy
less complexity

Bounded decompositions

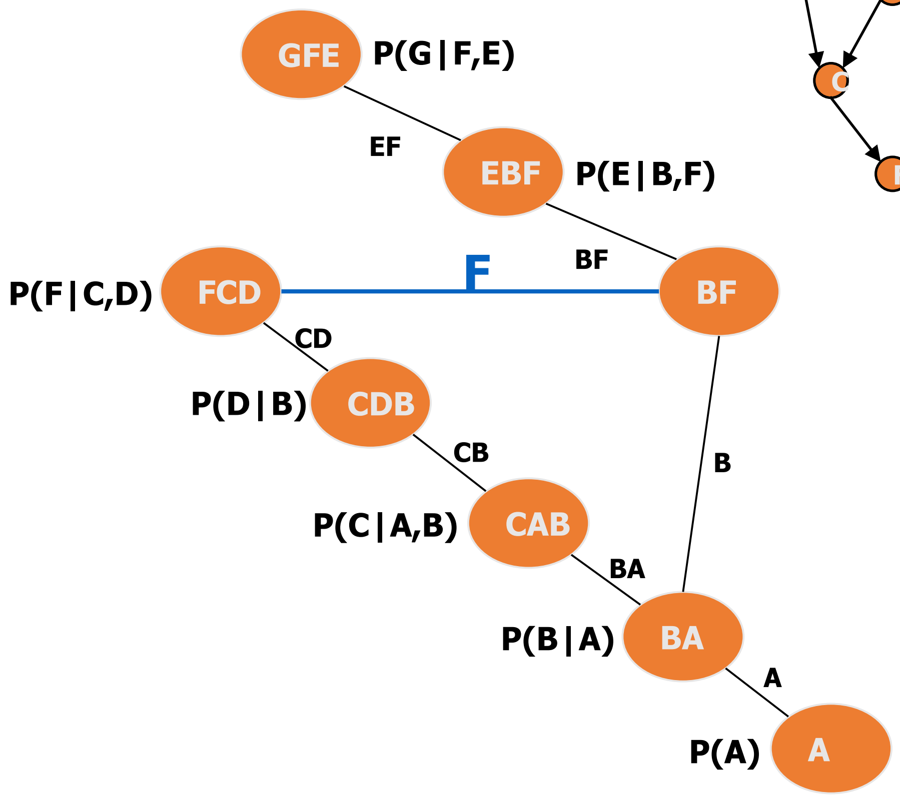
- We want arc-labeled decompositions such that:
  - the cluster size (internal width) is bounded by *i* (the accuracy parameter)
  - the width of the decomposition as a graph (external width) is as small as possible

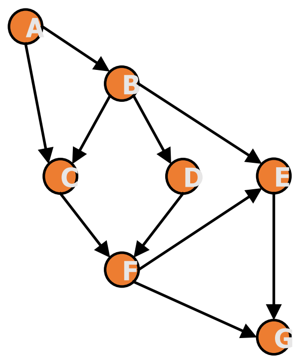
- Possible approaches to build decompositions:
  - partition-based algorithms - inspired by the mini-bucket decomposition
  - grouping-based algorithms

Constructing Join-Graphs



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





b) arc-labeled join-graph decomposition

IJGP properties

- IJGP(i) applies BP to min arc-labeled join-graph, whose cluster size is bounded by i
- On join-trees IJGP finds exact beliefs
- IJGP is a Generalized Belief Propagation algorithm (Yedidia, Freeman, Weiss 2001)
- Complexity of one iteration:
  - time: $O(deg \bullet (n+N) \bullet d^{i+1})$
  - space: $O(N \bullet d^{\theta})$

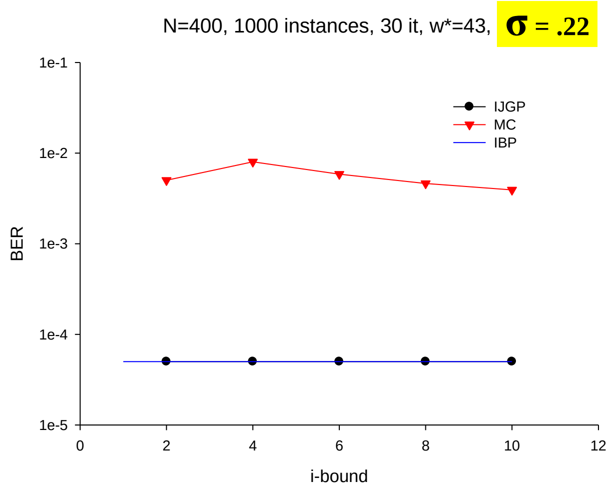
Empirical evaluation

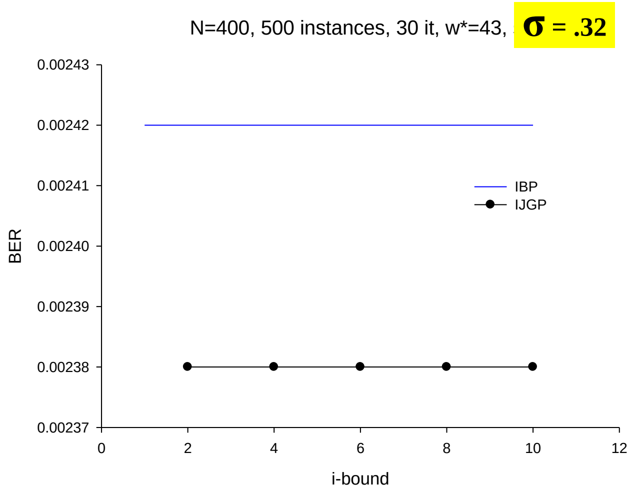
- Algorithms:
  - Exact
  - IBP
  - MC
  - IJGP

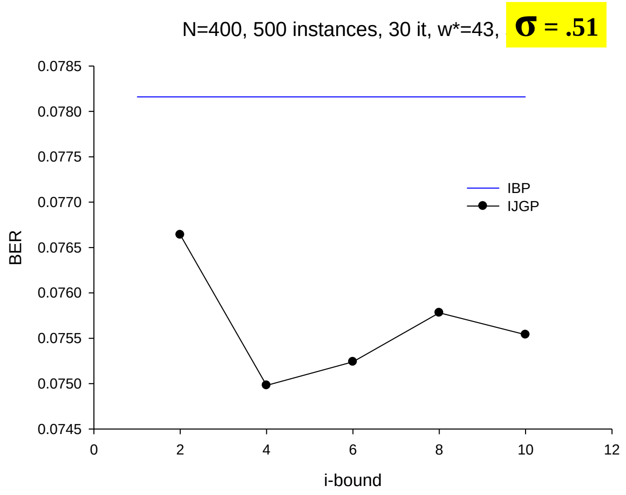
- Measures:
  - Absolute error
  - Relative error
  - Kulbach-Leibler (KL) distance
  - Bit Error Rate
  - Time

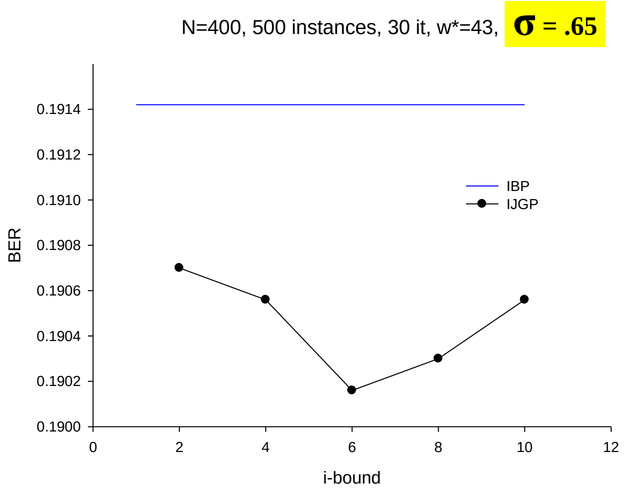
- Networks (all variables are binary):
  - Random networks
  - Grid networks (MxM)
  - CPCS 54, 360, 422
  - Coding networks

Coding Networks – Bit Error Rate

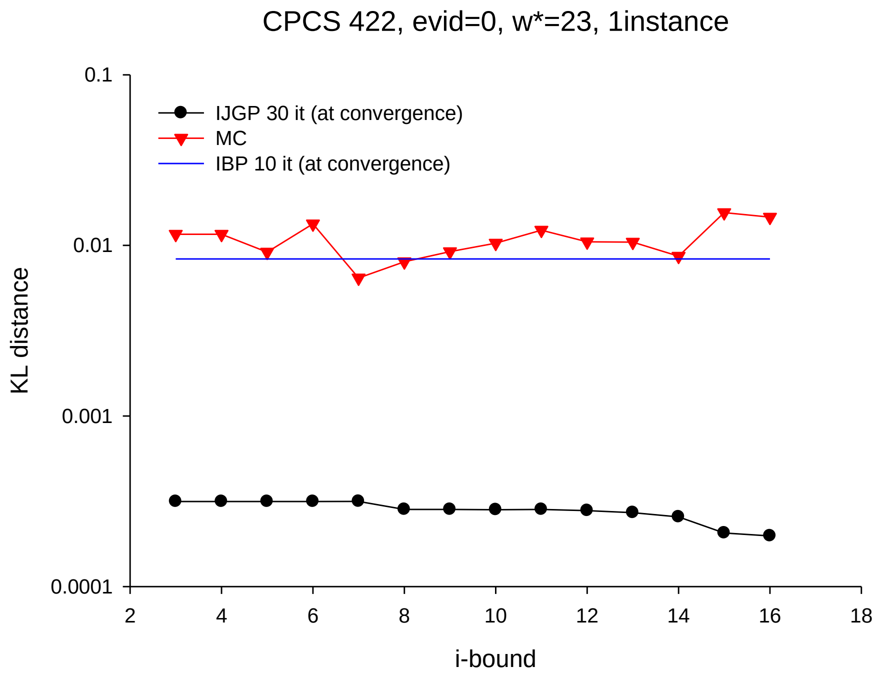




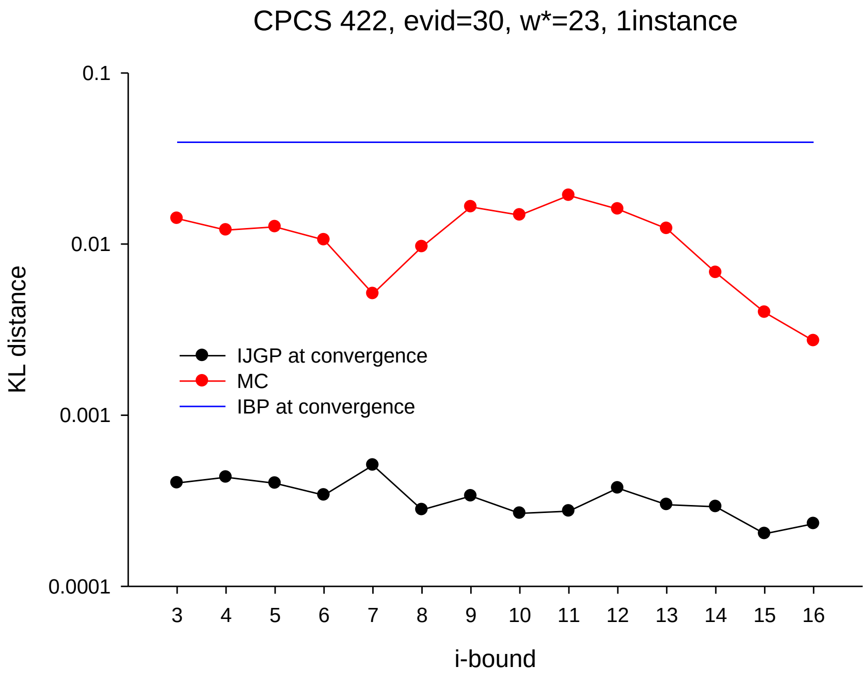




CPCS 422 – KL Distance

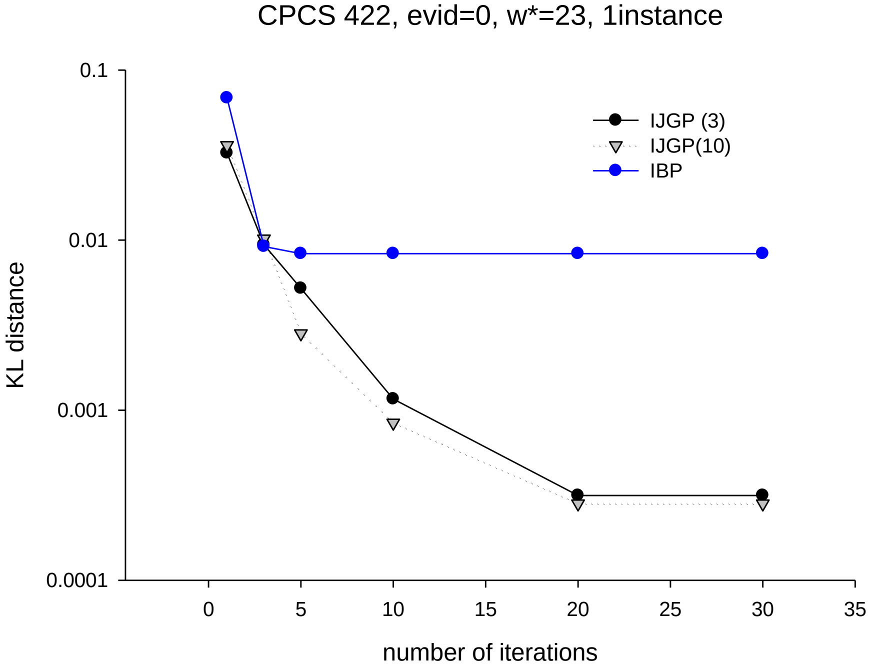


evidence=0

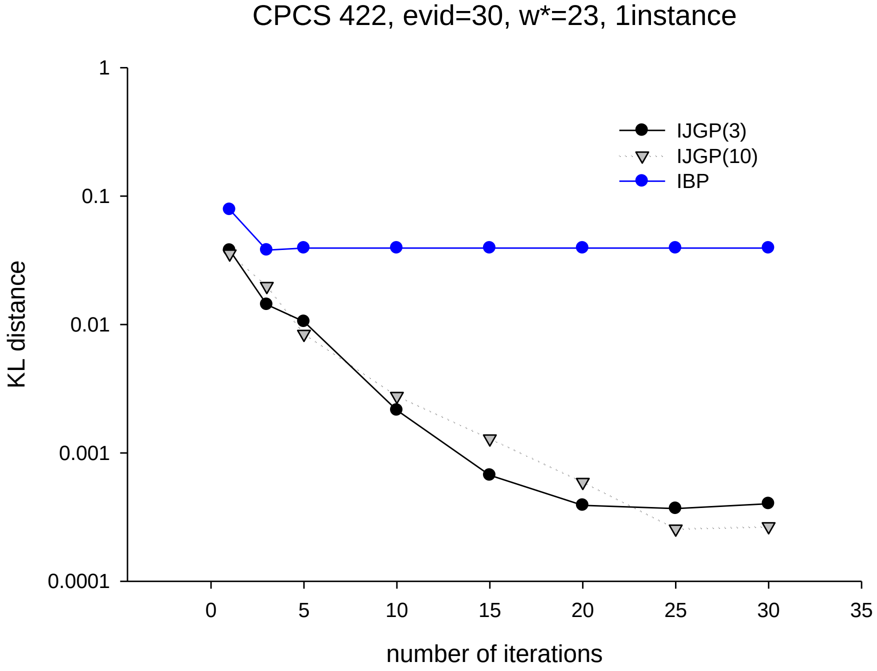


evidence=30

CPCS 422 – KL vs. Iterations

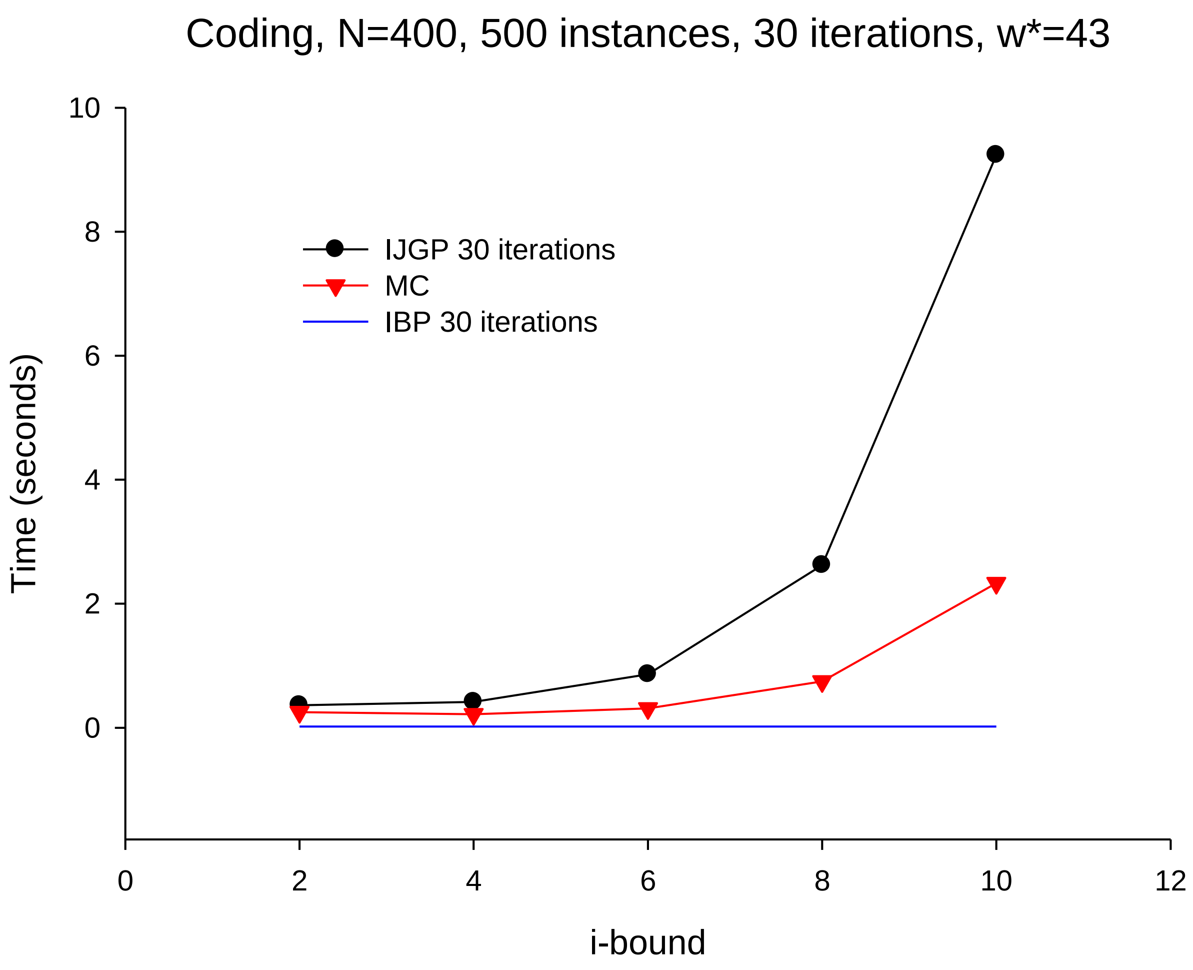


evidence=0



evidence=30

Coding networks - Time



Outline

- Mini-bucket elimination
- Weighted Mini-bucket
- Mini-clustering
- Iterative Belief propagation
- Iterative-join-graph propagation
- Re-parameterization, cost-shifting

Cost-Shifting

(Reparameterization)

$+\lambda(B)$

A	B	f(A,B)
b	b	6 + 3
b	g	0 – 1
g	b	0 + 3
g	g	6 – 1

A	B	C	f(A,B,C)
b	b	b	12
b	b	g	6
b	g	b	0
b	g	g	6
g	b	b	6
g	b	g	0
g	g	b	6
g	g	g	12

= 0 + 6

$-\lambda(B)$

B	C	f(B,C)
b	b	6 – 3
b	g	0 – 3
g	b	0 + 1
g	g	6 + 1

B	λ(B)
b	3
g	-1

Modify the individual functions

- but –

keep the sum of functions the same

Tightening the bound

- Reparameterization (or, “cost shifting”)
  - Decrease bound without changing overall function

A	B	f$_1$(A,B)
0	0	2.0
1	0	3.5
0	1	1.0
1	1	3.0

+

B	C	f$_2$(B,C)
0	0	1.0
0	1	0.0
1	0	1.0
1	1	3.0

=

$$f_{AB}(a,b) + f_{BC}(b,c)$$

A	B	C	F(A,B,C)
0	0	0	3.0
0	0	1	2.0
0	1	0	2.0
0	1	1	4.0
1	0	0	4.5
1	0	1	3.5
1	1	0	4.0
1	1	1	6.0

$$\max_{a,b} f_1(a,b) \quad + \quad \max_{b,c} f_2(b,c)$$

$$+\lambda_{B\to AB}(b) \qquad + \lambda_{B\to BC}(b)$$

$$\lambda_{B\to AB}(b) + \lambda_{B\to BC}(b) = 0$$

=

A	B	f$_1$(A,B)	.(B)
0	0	2.0	0
1	0	3.5	
0	1	1.0	+1
1	1	3.0	

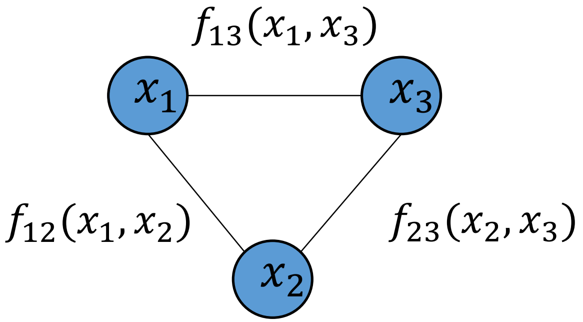
+

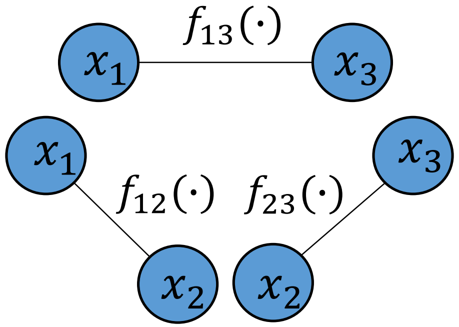
B	C	f$_2$(B,C)	-.(B)
0	0	1.0	0
0	1	0.0	
1	0	1.0	-1
1	1	3.0	

(Adjusting functions cancel each other)

(Decomposition bound is exact)

Dual Decomposition

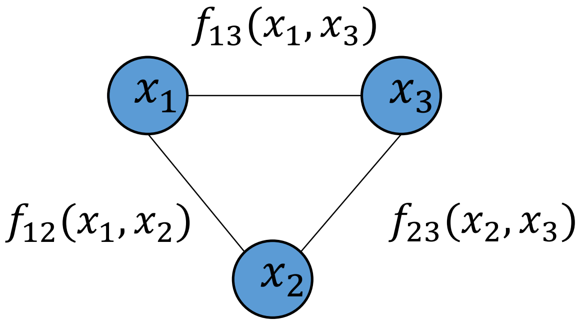


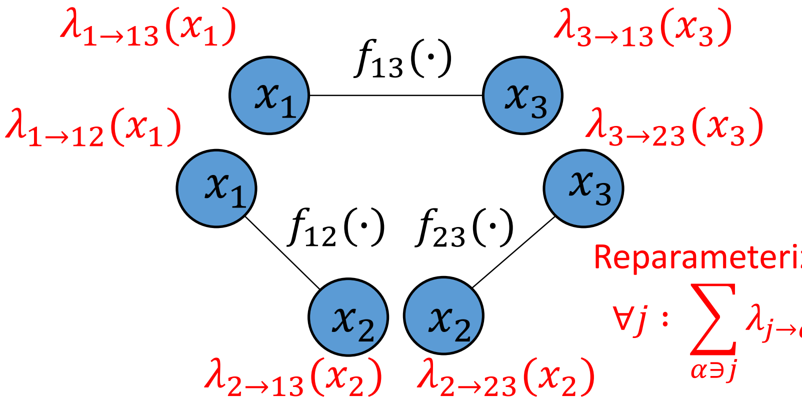


$$F^* = \min_x \sum_\alpha f_\alpha(x) \qquad \geq \qquad \sum_\alpha \min_x \; f_\alpha(x)$$

- Bound solution using decomposed optimization
- Solve independently: optimistic bound

Dual Decomposition





Reparameterization:

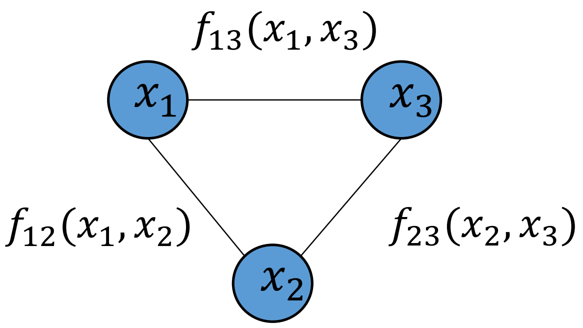
$$\forall j : \sum_{\alpha \ni j} \lambda_{j\to\alpha}(x_j) = 0$$

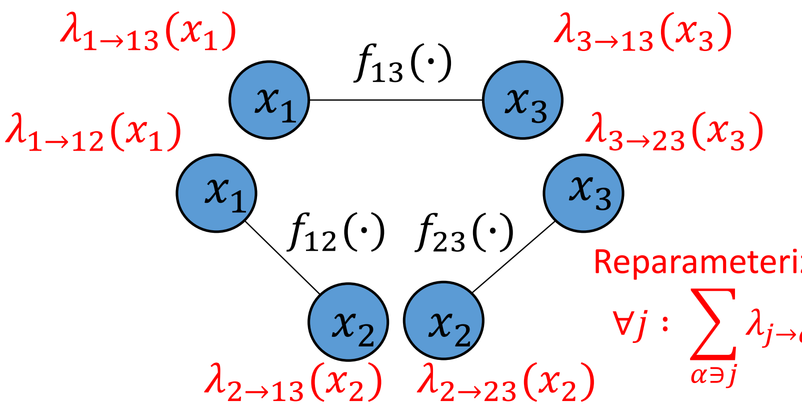
$$F^* = \min_x \sum_\alpha f_\alpha(x) \qquad \geq \qquad \max_{\lambda_{i\to\alpha}} \sum_\alpha \min_x \left[f_\alpha(x) + \sum_{i\in\alpha} \lambda_{i\to\alpha}(x_i) \right]$$

- Bound solution using decomposed optimization
- Solve independently: optimistic bound

- Tighten the bound by reparameterization
  - Enforce lost equality constraints via Lagrange multipliers

Dual Decomposition





Reparameterization:

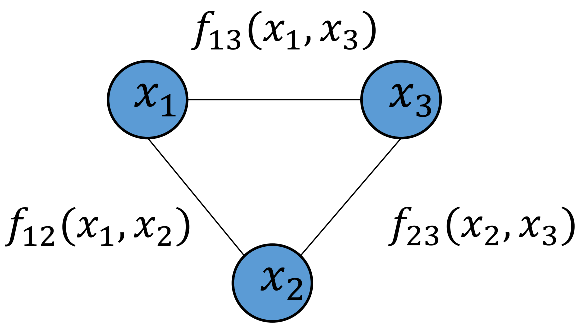
$$\forall j : \sum_{\alpha \ni j} \lambda_{j\to\alpha}(x_j) = 0$$

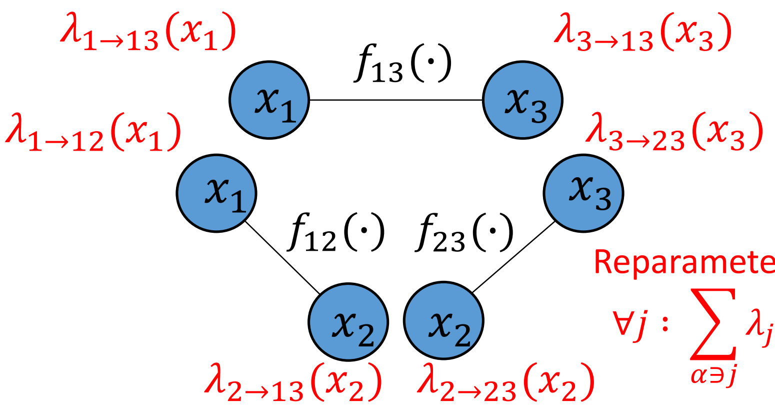
$$F^* = \min_x \sum_\alpha f_\alpha(x) \qquad \geq \qquad \max_{\lambda_{i\to\alpha}} \sum_\alpha \min_x \left[f_\alpha(x) + \sum_{i\in\alpha} \lambda_{i\to\alpha}(x_i) \right]$$

Many names for the same class of bounds:

- Dual decomposition [Komodakis et al. 2007]
- TRW, MPLP [Wainwright et al. 2005; Globerson & Jaakkola, 2007]
- Soft arc consistency [Cooper & Schiex, 2004]
- Max-sum diffusion [Warner 2007]

Dual Decomposition





Reparameterization:

$$\forall j : \sum_{\alpha \ni j} \lambda_{j\to\alpha}(x_j) = 0$$

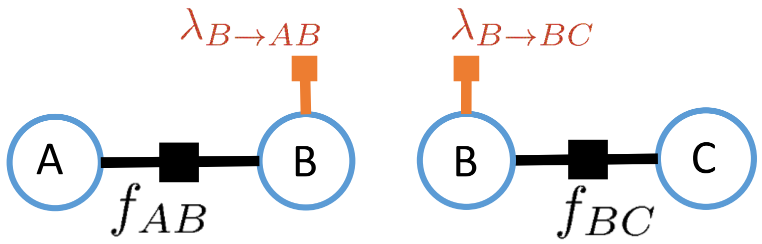
$$F^* = \min_x \sum_\alpha f_\alpha(x) \qquad \geq \qquad \max_{\lambda_{i\to\alpha}} \sum_\alpha \min_x \left[f_\alpha(x) + \sum_{i\in\alpha} \lambda_{i\to\alpha}(x_i) \right]$$

Many ways to optimize the bound:

- Sub-gradient descent [Komodakis et al. 2007; Jojic et al. 2010]
- Coordinate descent [Warner 2007; Globerson & Jaakkola 2007; Sontag et al. 2009; Ihler et al. 2012]
- Proximal optimization [Ravikumar et al, 2010]
- ADMM [Meshi & Globerson 2011; Martins et al. 2011; Forouzan & Ihler 2013]

Optimizing the bound

- Can optimize the bound in various ways:
  - (Sub-)gradient descent



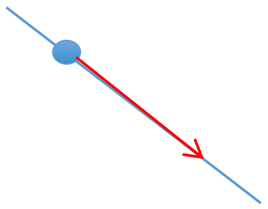
=

A	B	f$_1$(A,B)	¸(B)
0	0	1.0	0
1	0	0.0	
0	1	0.0	0
1	1	2.5	
0	2	1.0	0
1	2	3.0	

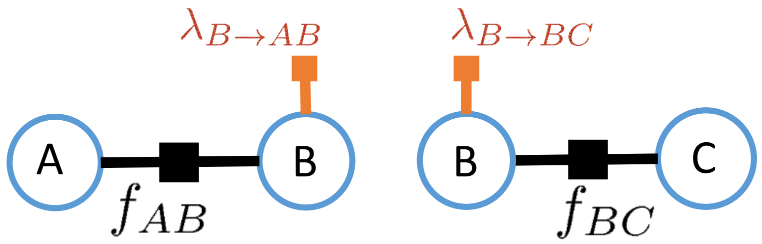
+

B	C	f$_2$(B,C)	-¸(B)
0	0	5.0	0
0	1	2.0	
1	0	1.0	0
1	1	1.5	
2	0	0.2	0
2	1	0.0	

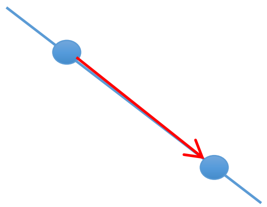
$$\max_x f_1(a,b) + \lambda_{B\to AB}(b) \quad + \quad \max_x f_2(b,c) + \lambda_{B\to BC}(b)$$



Optimizing the bound



- Can optimize the bound in various ways:
  - (Sub-)gradient descent



=

A	B	f_1(A,B)	¸(B)
0	0	1.0	+1
1	0	0.0	
0	1	0.0	0
1	1	2.5	
0	2	1.0	-1
1	2	3.0	

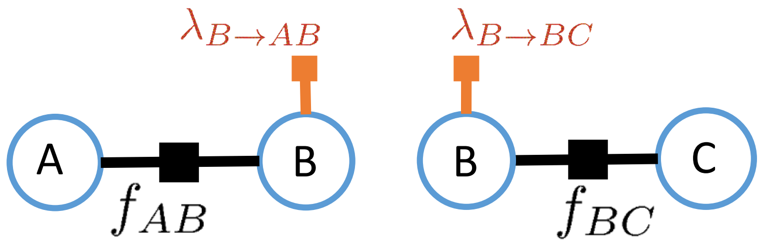
+

B	C	f_2(B,C)	-¸(B)
0	0	5.0	-1
0	1	2.0	
1	0	1.0	0
1	1	1.5	
2	0	0.2	+1
2	1	0.0	

$$\max_x f_1(a,b) + \lambda_{B\to AB}(b) \quad + \quad \max_x f_2(b,c) + \lambda_{B\to BC}(b)$$

Optimizing the bound

- Can optimize the bound in various ways:
  - (Sub-)gradient descent



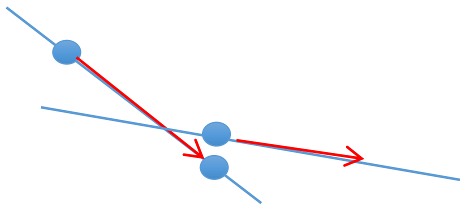
=

A	B	f₁(A,B)	¸(B)
0	0	1.0	+1
1	0	0.0	
0	1	0.0	0
1	1	2.5	
0	2	1.0	-1
1	2	3.0	

+

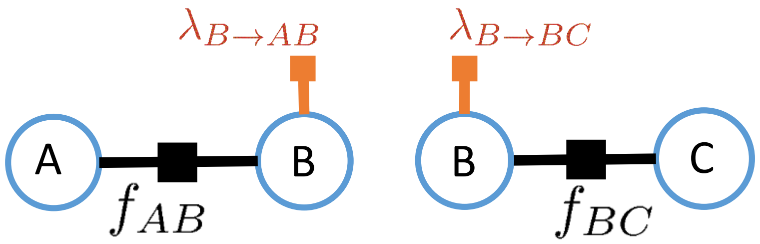
B	C	f₂(B,C)	-¸(B)
0	0	5.0	-1
0	1	2.0	
1	0	1.0	0
1	1	1.5	
2	0	0.2	+1
2	1	0.0	

$$\max_x f_1(a,b) + \lambda_{B\to AB}(b) \quad + \quad \max_x f_2(b,c) + \lambda_{B\to BC}(b)$$



Optimizing the bound

- Can optimize the bound in various ways:
  - (Sub-)gradient descent



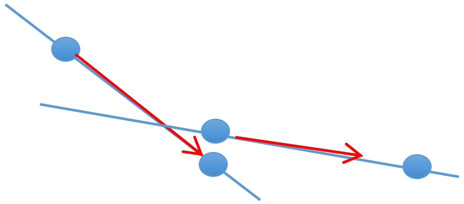
=

A	B	f₁(A,B)	¸(B)
0	0	1.0	+2
1	0	0.0	
0	1	0.0	-1
1	1	2.5	
0	2	1.0	-1
1	2	3.0	

+

B	C	f₂(B,C)	-¸(B)
0	0	5.0	-2
0	1	2.0	
1	0	1.0	+1
1	1	1.5	
2	0	0.2	+1
2	1	0.0	

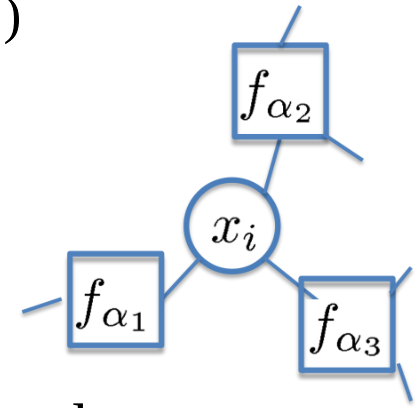
$$\max_x f_1(a,b) + \lambda_{B\to AB}(b) \quad + \quad \max_x f_2(b,c) + \lambda_{B\to BC}(b)$$



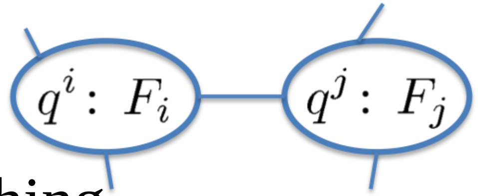
Various Update Schemes

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

- FGLP: Update the original factors



- JGLP: Update clique function of the join graph



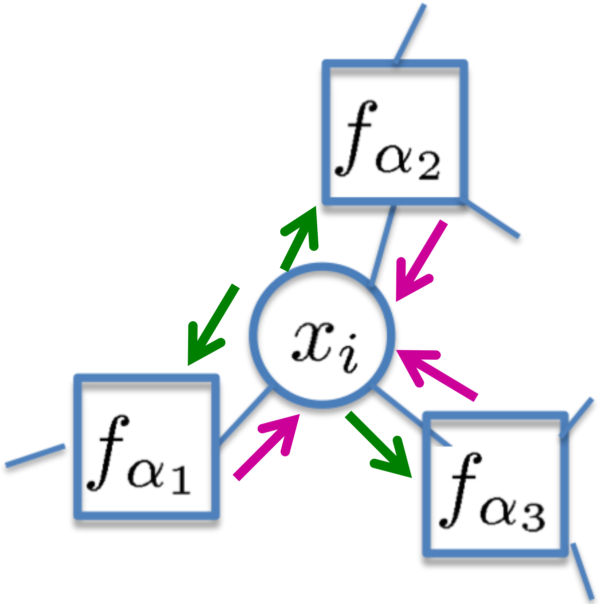
- **MBE-MM:** Mini-bucket with moment matching
  - Apply cost-shifting within each bucket only

Factor graph Linear Programming

- Update the original factors (FGLP)
  - Tighten all factors over over x_i simultaneously
  - Compute max-marginals
  - & update:

$$\forall \alpha, \ \gamma_\alpha(x_i) = \max_{x_\alpha \setminus x_i} f_\alpha$$

$$\forall \alpha, \ f_\alpha(x_\alpha) \leftarrow f_\alpha(x_\alpha) - \gamma_\alpha(x_i) + \frac{1}{|F_i|} \sum_\beta \gamma_\beta(x_i)$$



FGLP (Factor Graph Linear Programming)

Algorithm 1: Factor graph LP (FGLP)

---

[1] **Input:** Graphical Model $\langle X, D, F, \sum \rangle$, where f_α is a function defined on variables X_α
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 $\sum_{\alpha \in F} \max_X f_\alpha(X_\alpha)$

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

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 λ_{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

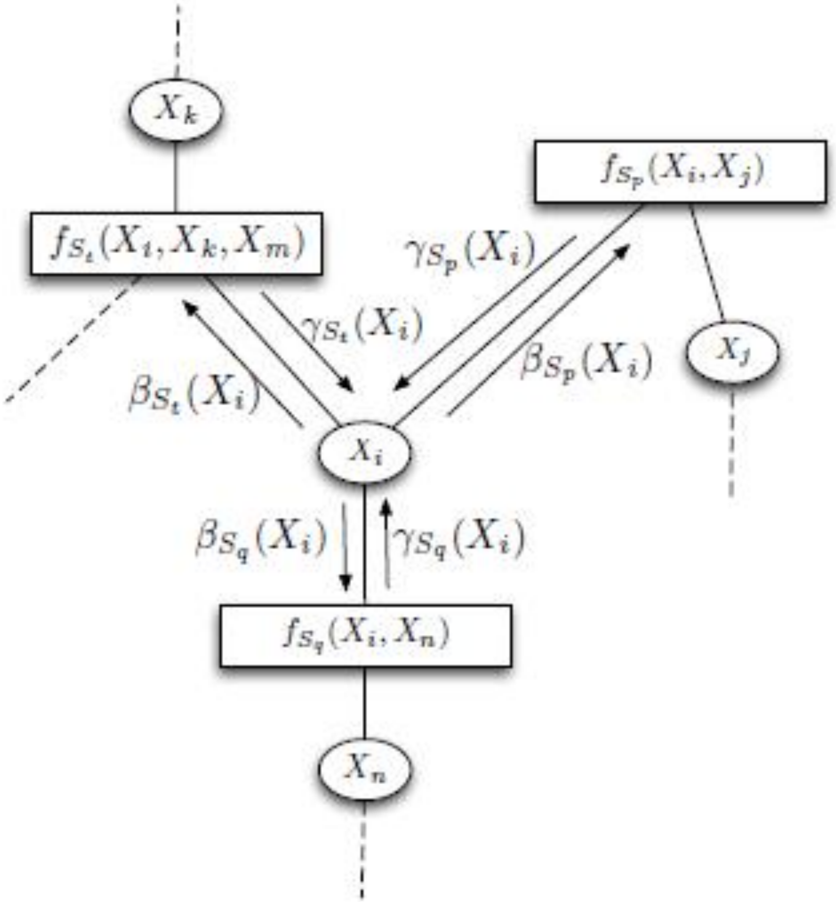


Figure 5.2: FGLP example: local messages involving variable X_i.

Factor Graph Linear Programming

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

Input: A graphical model $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F}, \sum \rangle$, variable ordering o

Output: Upper bound on the optimum value of MPE cost

while *NOT converged* **do**
- **for** *each variable* X_i **do**
  - Get factors $F_i = f_{\mathbf{S}_k} : X_i \in \mathbf{S}_k$ with X_i in their scope;
  - //for each function compute max-marginals γ marginalizing out all variables except for X_i:
  - $\forall f_{\mathbf{S}_k}\ \gamma_{\mathbf{S}_k}(X_i) = \max_{\mathbf{S}_k \setminus X_i} f_{\mathbf{S}_k}$;
  - // compute messages $\beta_{\mathbf{S}_k}(X_i)$ from X_i back to a function $f_{\mathbf{S}_k}$ correcting for the function's own max-marginal $\gamma_{\mathbf{S}_k}$:
  - $\forall f_{\mathbf{S}_k}\ \beta_{\mathbf{S}_k} = \frac{1}{|F_i|} \sum_{\{\mathbf{S}_j | f_{\mathbf{S}_j} \in F_i\}} \gamma_{\mathbf{S}_j}(X_i) - \gamma_{\mathbf{S}_k}(X_i)$
  - // update (re-parametrize) each function:
  - $\forall f_{\mathbf{S}_k},\ f_{\mathbf{S}_k} \leftarrow f_{\mathbf{S}_k} + \beta_{\mathbf{S}_k}$;

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

$$\beta_{\mathbf{S}_q}(X_i) = \frac{1}{3}(\gamma_{\mathbf{S}_q}(X_i) + \gamma_{\mathbf{S}_p}(X_i) + \gamma_{\mathbf{S}_t}(X_i)) - \gamma_{\mathbf{S}_q}(X_i)$$

$$\beta_{\mathbf{S}_p}(X_i) = \frac{1}{3}(\gamma_{\mathbf{S}_q}(X_i) + \gamma_{\mathbf{S}_p}(X_i) + \gamma_{\mathbf{S}_t}(X_i)) - \gamma_{\mathbf{S}_p}(X_i)$$

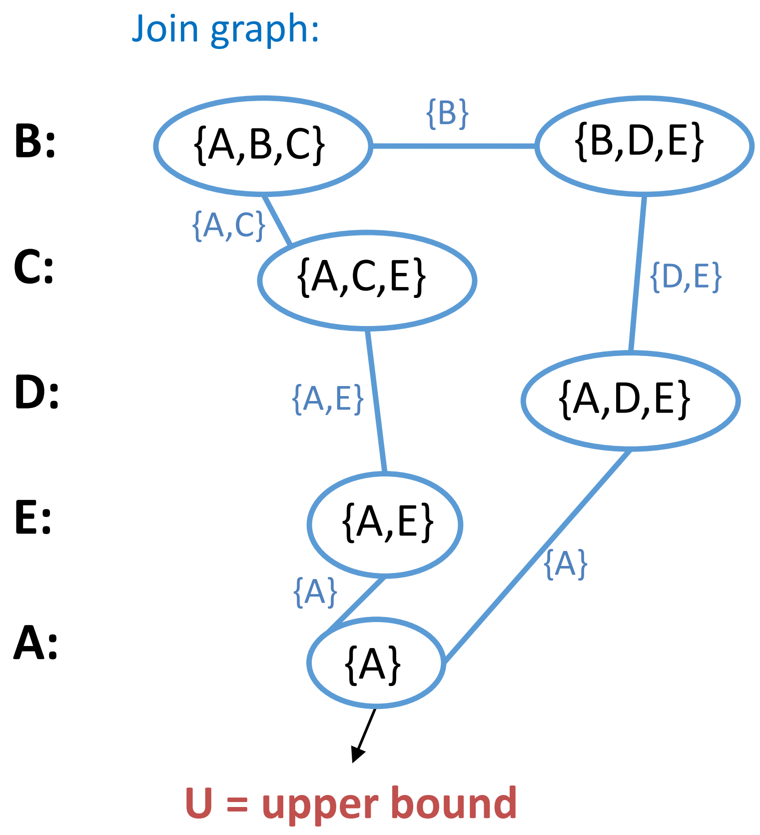
$$\beta_{\mathbf{S}_t}(X_i) = \frac{1}{3}(\gamma_{\mathbf{S}_q}(X_i) + \gamma_{\mathbf{S}_p}(X_i) + \gamma_{\mathbf{S}_t}(X_i)) - \gamma_{\mathbf{S}_t}(X_i)$$

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,* $|\boldsymbol{F}|$ *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* $O(|\boldsymbol{F}| \cdot k^{Sc})$*.*

Mini-Bucket as Decomposition [Ihler et al. 2012]

- Downward pass as cost shifting
- Can also do cost shifting within mini-buckets:
  "Join graph" message passing
- "Moment-matching" version:
  One message exchange within each bucket, during downward sweep
- Optimal bound defined by cliques ("regions") and cost-shifting f'n scopes ("coordinates")



MBE-MM: MBE with moment matching

m_{11}, m_{12}- moment-matching messages

$\max_B \prod$ $\qquad$ $\max_B \prod$

Bucket B $\quad$ P(E|B,C) $\;\underset{m_{12}}{\overset{m_{11}}{\rightleftarrows}}\;$ P(B|A) P(D|A,B)

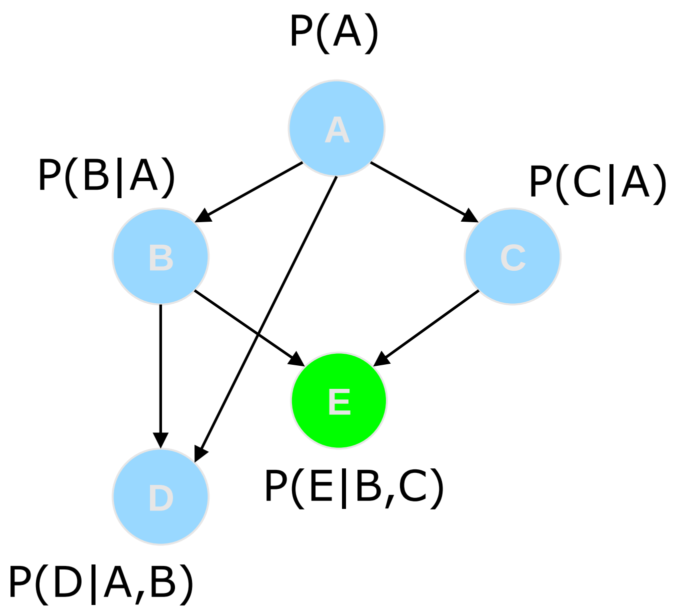
Bucket C $\quad$ P(C|A) $\quad h^B (C,E)$

Bucket D $\quad h^B (A,D)$

Bucket E $\quad$ E = 0 $\quad h^C (A,E)$

Bucket A $\quad$ P(A) $\quad h^E (A) \quad h^D (A)$

W=2



MPE* is an upper bound on MPE --U
Generating a solution yields a lower bound--L

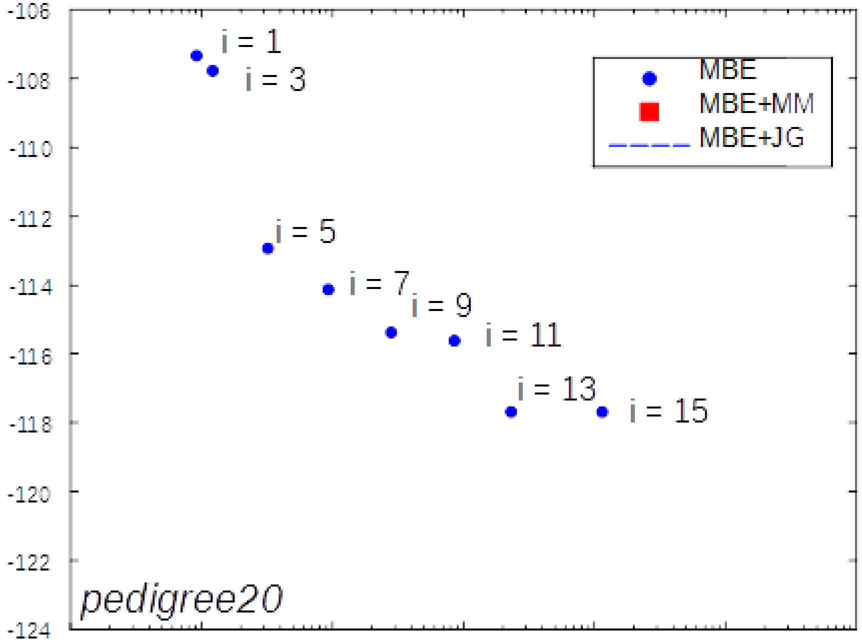
MBE-MM (MBE with Moment-Matching)

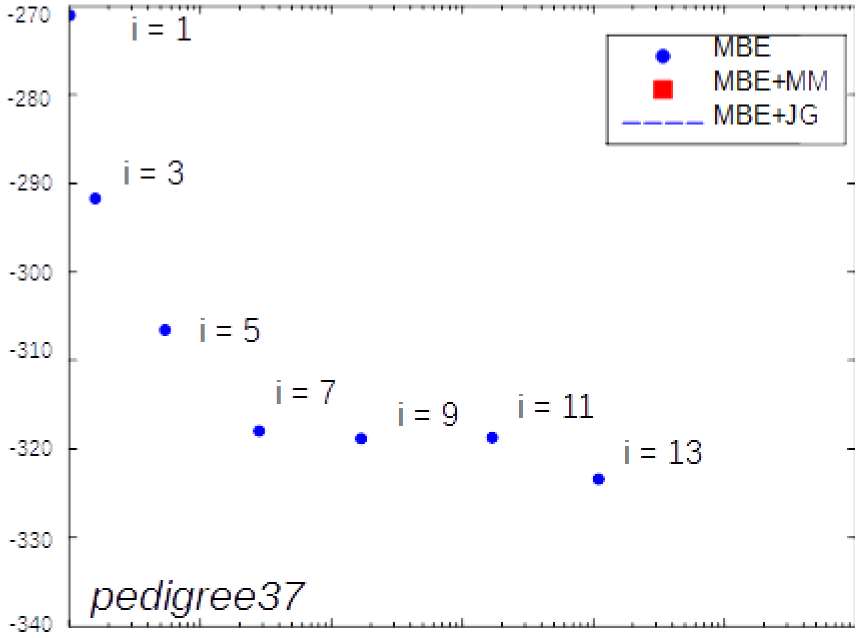
Algorithm 26: Algorithm MBE-MM

Input: A graphical model $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F}, \sum \rangle$, variable order $o = \{X_1, \ldots, X_n\}$, i-bound parameter i
Output: Upper bound on the optimum value of MPE cost
//Initialize:
Partition the functions in $\mathbf{F}$ into $\mathbf{B}_{X_1}, \ldots, \mathbf{B}_{X_n}$, where $\mathbf{B}_{X_k}$ contains all functions f_j whose highest variable is X_k;
//processing bucket $\mathbf{B}_{X_k}$
for $k \leftarrow n$ *down to 1* **do**
  Partition functions g (both original and messages generated in previous buckets) in $\mathbf{B}_{X_k}$ into the mini-buckets defined $Q_{X_k} = \{q_k^1, \ldots, q_k^t\}$, where each q_k^p has no more than $i+1$ variables;
  Find the set of variables common to all the mini-buckets of variable X_k:
  $\mathbf{S}_k = Scope(q_k^1) \cap \cdots \cap Scope(q_k^t)$;
  Find the function of each mini-bucket
  q_k^p: $F_k^p \leftarrow \prod_{g \in q_k^p} g$;
  Find the max-marginals of each mini-bucket
  q_k^p: $\gamma_{X_k}^p = \max_{Scope(q_k^p) \setminus \mathbf{S}_k}(F_k^p)$;
  Update functions of each mini-bucket
  $F_k^p \leftarrow F_k^p - \gamma_{X_k}^p + \frac{1}{t}\sum_{j=1}^{t} \gamma_{X_k}^j$;
  Generate messages $h_{X_k \rightarrow X_m}^p = \max_{X_k} F_k^p$ and place each in the bucket of highest in the ordering o variable X_m in $Scope(q_k^p)$;
return *All the buckets and the cost bound from* $\boldsymbol{B}_1$;

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.*

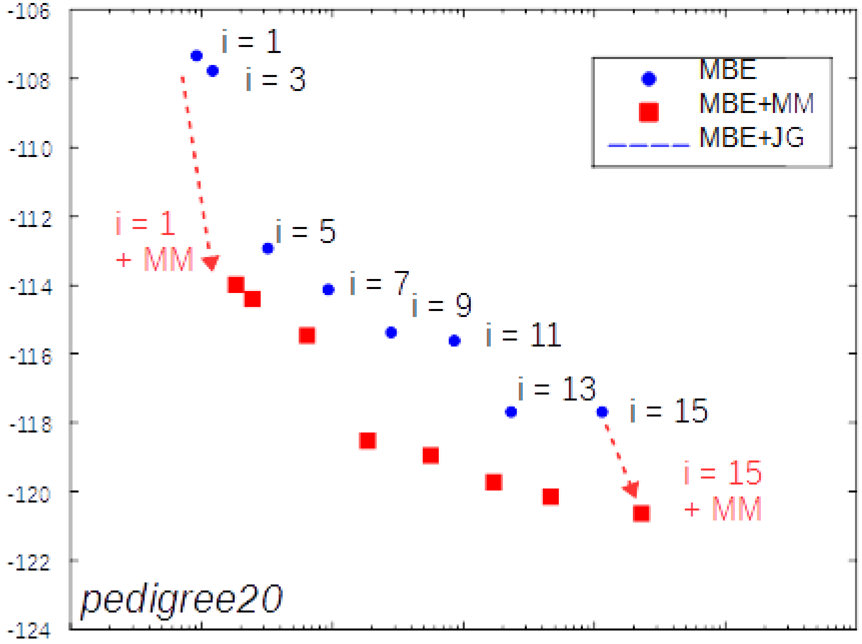
Anytime Approximation

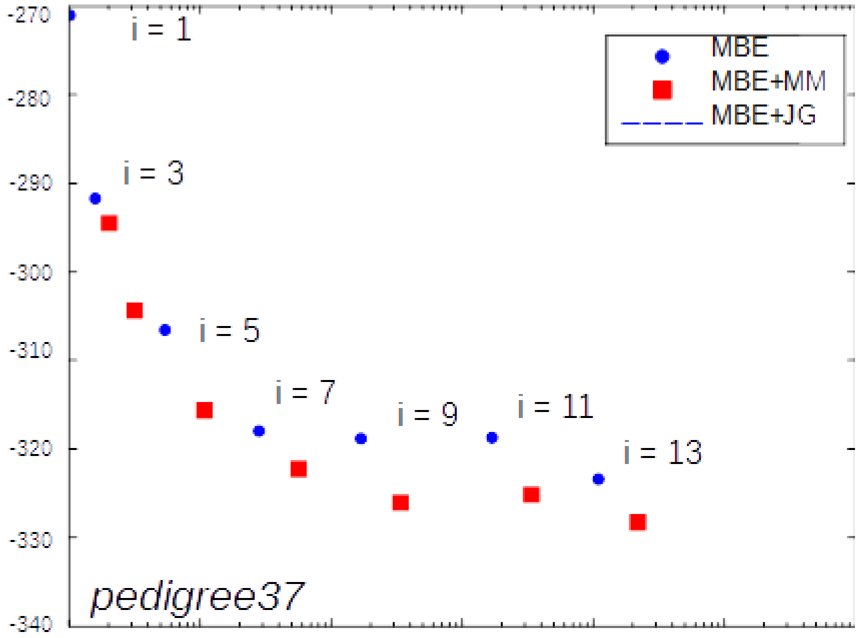




- Can tighten the bound in various ways
  - Cost-shifting (improve consistency between cliques)
  - Increase i-bound (higher order consistency)
- Simple moment-matching step improves bound significantly

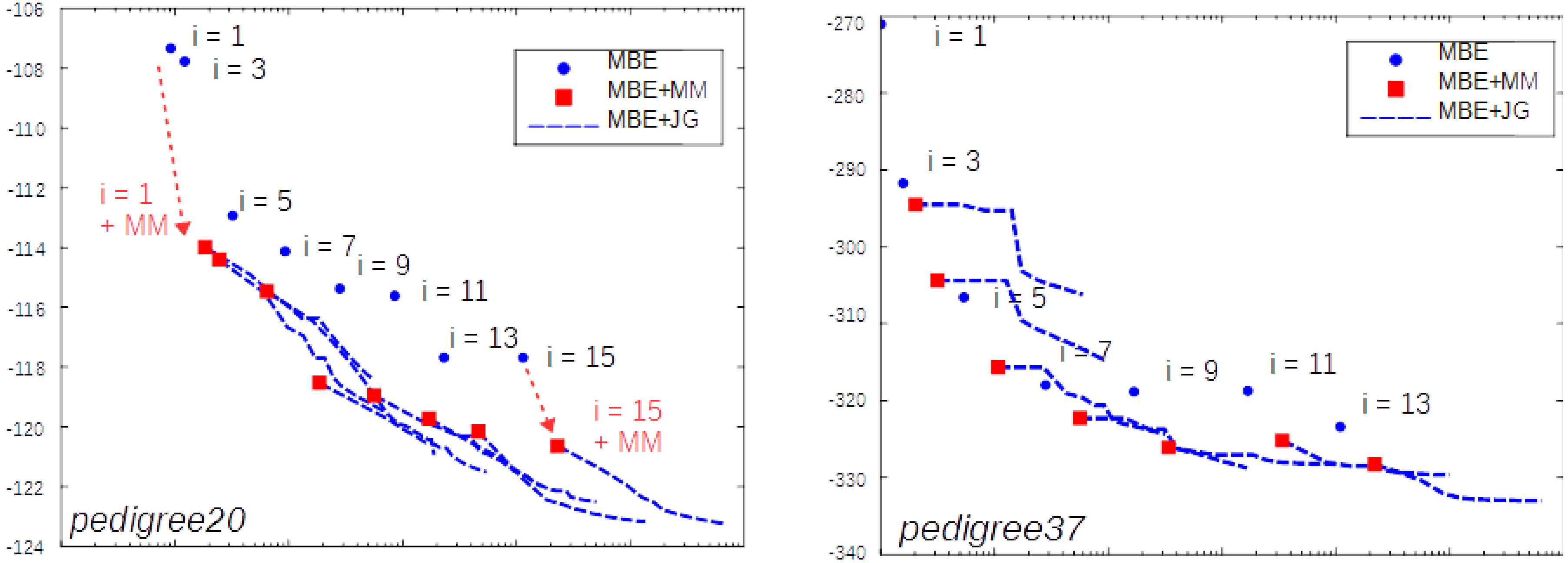
Anytime Approximation





- 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

Outline

- Mini-bucket elimination
- Weighted Mini-bucket
- Mini-clustering
- Iterative Belief propagation
- Iterative-join-graph propagation
- Re-parameterization, cost-shifting