Advances in Combinatorial Optimization for Graphical Models

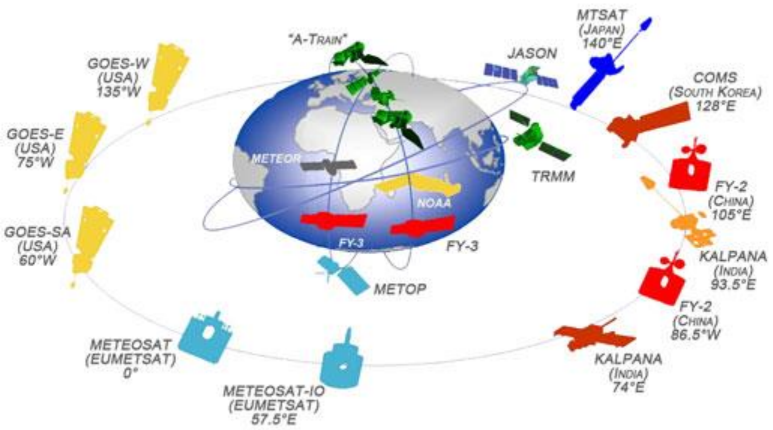
Rina Dechter
University of California, Irvine

Radu Marinescu
IBM Research - Ireland

Alexander Ihler
University of California, Irvine

Combinatorial Optimization

Planning & Scheduling



Find an optimal schedule for the satellite that maximizes the number of photographs taken, subject to on-board recording capacity

Computer Vision

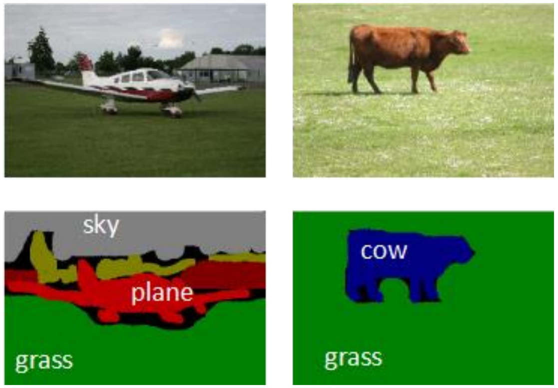
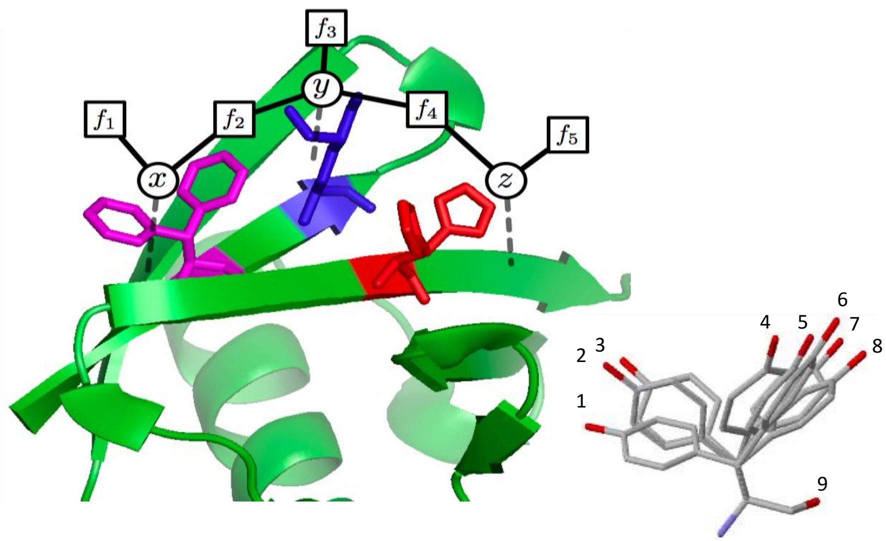


Image classification: label pixels in an image by their associated object class

[He et al. 2004; Winn et al. 2005]

Combinatorial Optimization

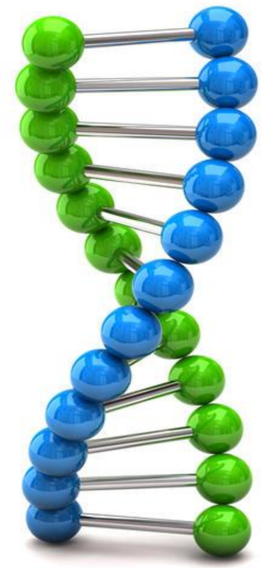
Protein folding



Find the most likely (minimum energy) configuration of the amino acids in a protein

[Yanover & Weiss 2002]

Bioinformatics



Find a joint haplotype configuration for all members of the pedigree which maximizes the probability of data

[Lauritzen & Sheehan 2003]

Outline

- **Introduction**
  - Graphical models
  - Optimization tasks for graphical models
- **Inference**
  - Variable elimination, bucket elimination
- **Bounds and heuristics**
  - Basics of search
  - Bounded variable elimination and iterative cost shifting
- **AND/OR search**
  - AND/OR search spaces
  - Depth-first branch and bound, best-first search
- **Exploiting parallelism**
  - Distributed and parallel search
- **Software**

Constraint Optimization Problems

A *finite COP* is a triple $R = \langle X, D, F \rangle$ where:

$X = \{X_1, ..., X_n\}$ - variables

$D = \{D_1, ..., D_n\}$ - domains

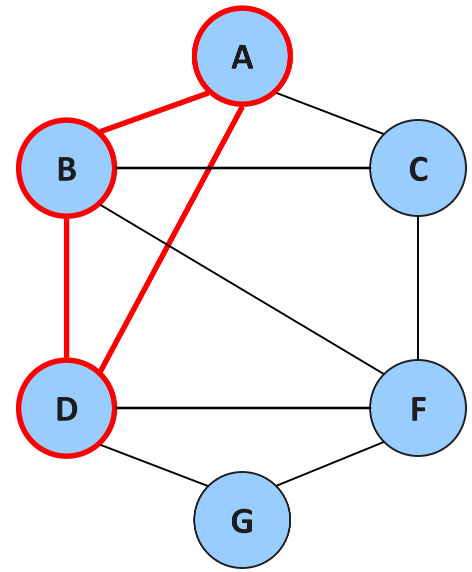
$F = \{f_1, ..., f_m\}$ - cost functions

f(A,B,D) has scope {A,B,D}

A	B	D	Cost
1	2	3	3
1	3	2	2
2	1	3	∞
2	3	1	0
3	1	2	5
3	2	1	0

Primal graph:
- Variables - nodes
- Functions - arcs / cliques

$$F(a,b,c,d,f,g) = f_1(a,b,d) + f_2(d,f,g) + f_3(b,c,f) + f_4(a,c)$$



Global Cost Function

$$F(X) = \sum_{\alpha} f_\alpha(x_\alpha)$$

Constraint Networks

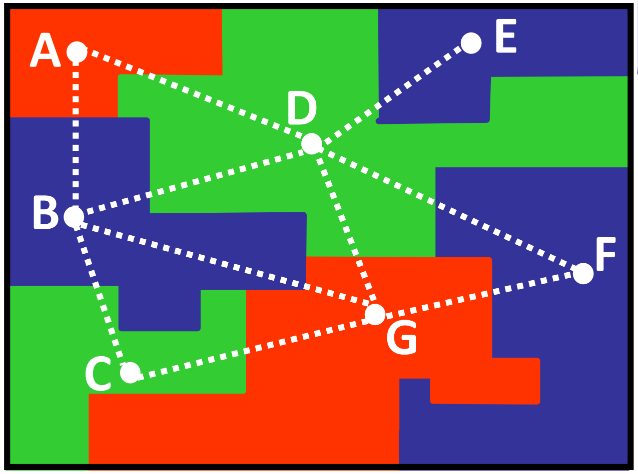
Map coloring

Variables: countries (A B C etc.)

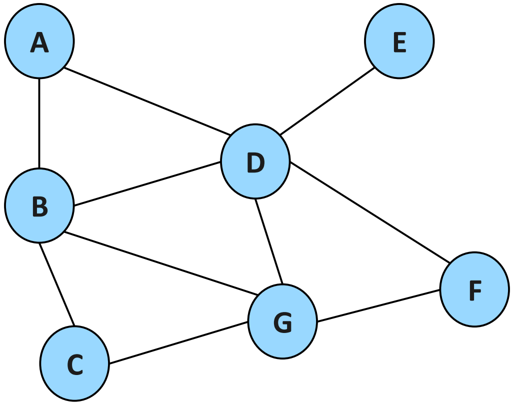
Values: colors (red green blue)

Constraints: A ≠ B; A ≠ D; B ≠ D; etc.

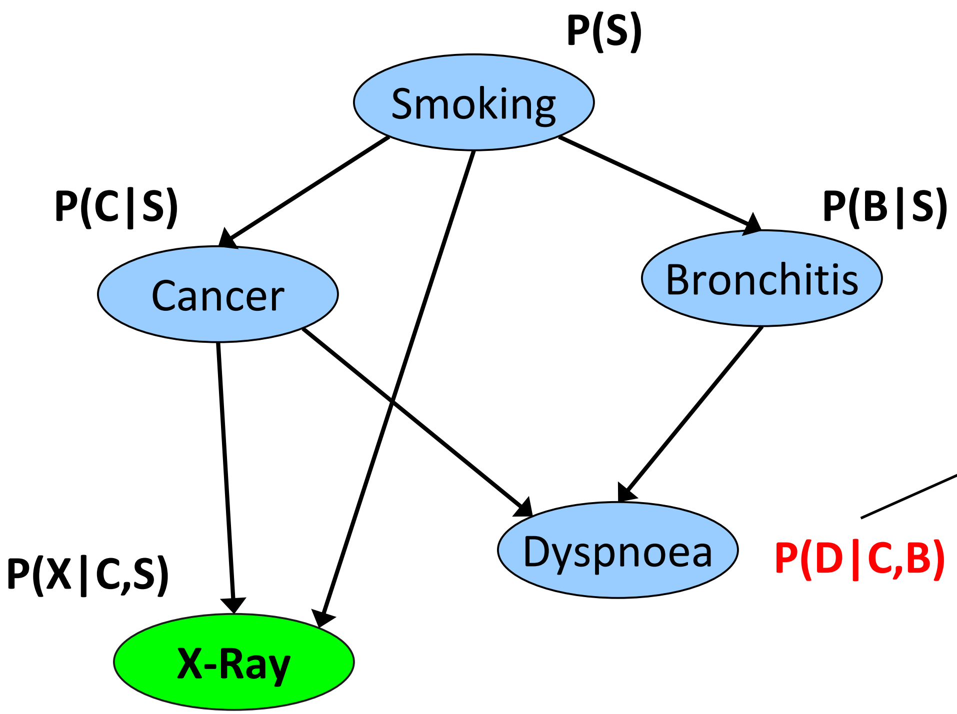
A	B
red	green
red	blue
green	red
green	Blue
blue	red
blue	green



Constraint graph



Probabilistic Networks



P(D|C,B)

C	B	D=0	D=1
0	0	0.1	0.9
0	1	0.7	0.3
1	0	0.8	0.2
1	1	0.9	0.1

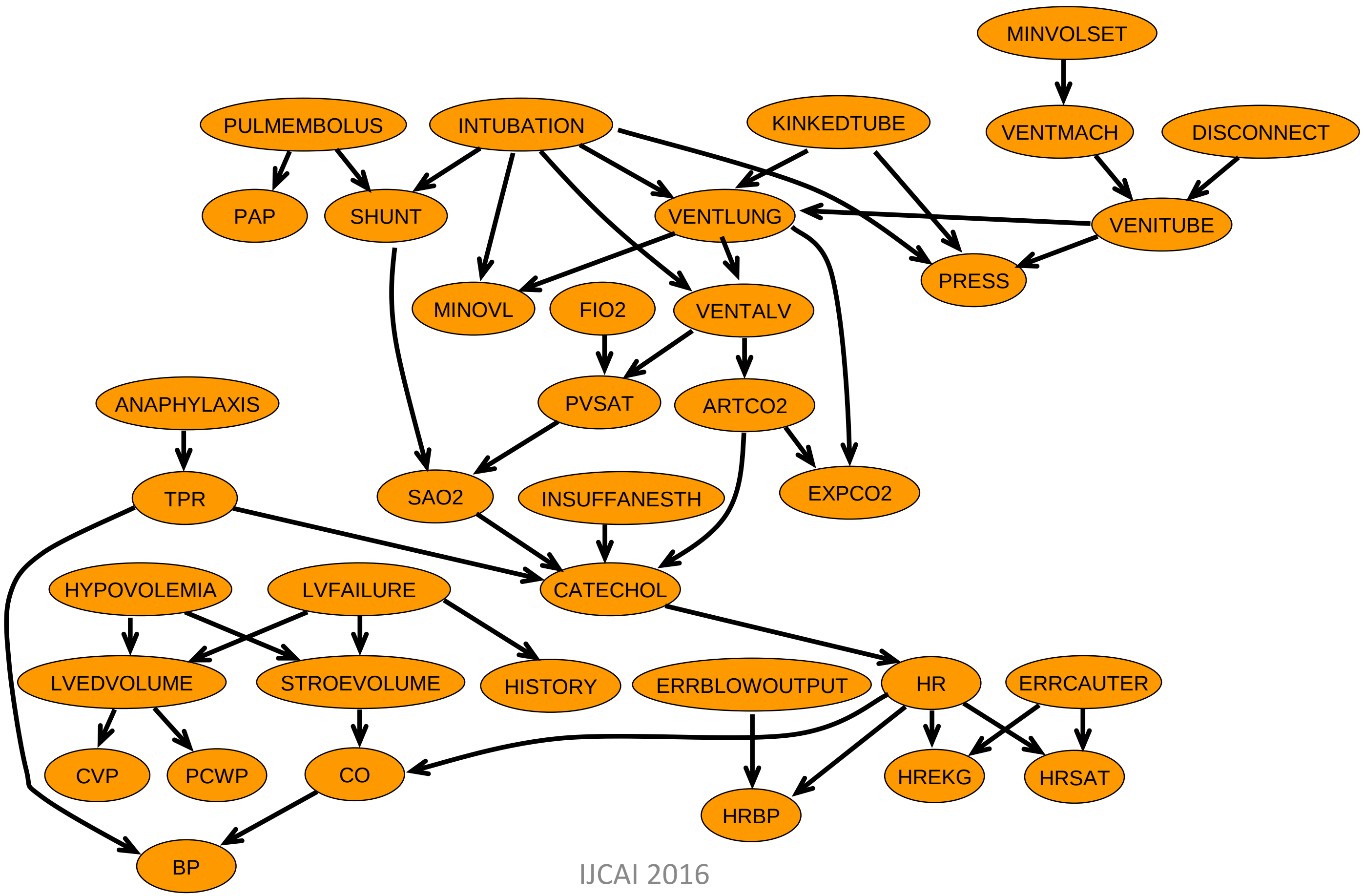
P(S,C,B,X,D) = P(S)· P(C|S)· P(B|S)· P(X|C,S)· P(D|C,B)

MPE= Find a maximum probability assignment, given evidence

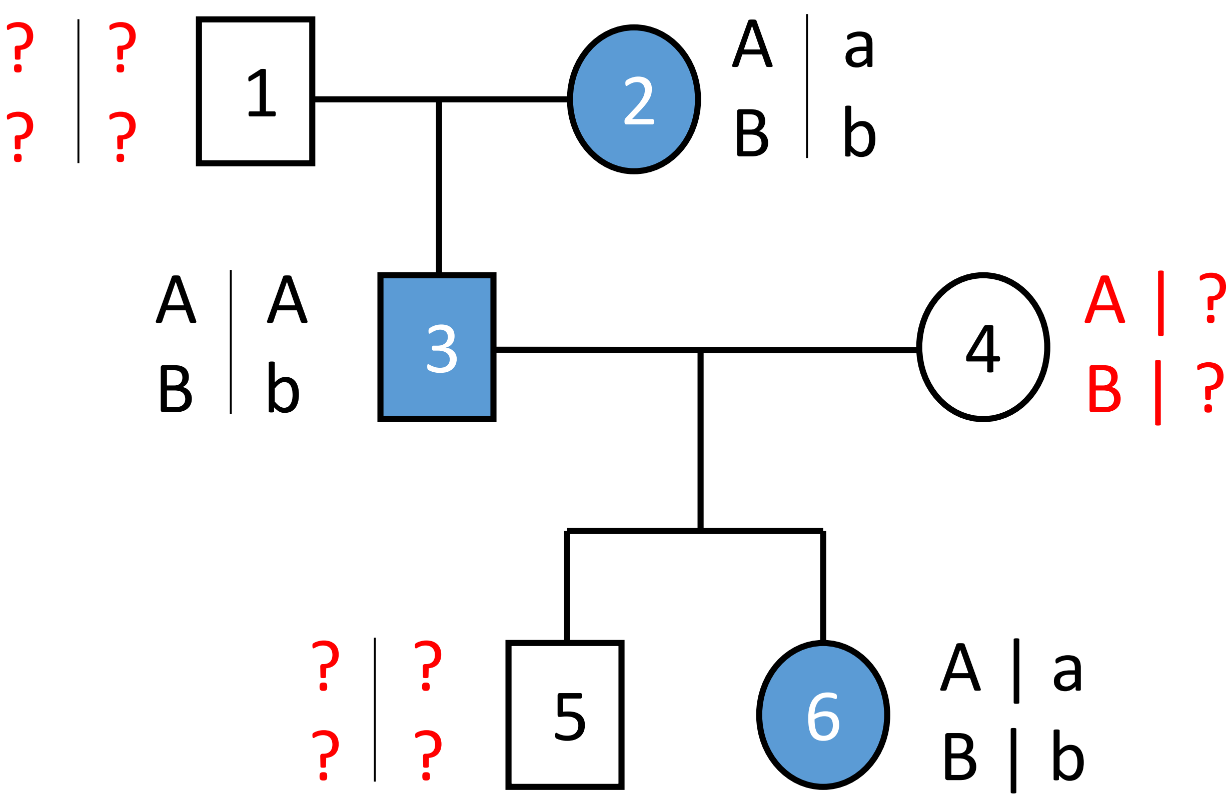
= **Find argmax** P(S)· P(C|S)· P(B|S)· P(X|C,S)· P(D|C,B)

Monitoring Intensive-Care Patients

The “alarm” network - 37 variables, 509 parameters (instead of 2^{37})

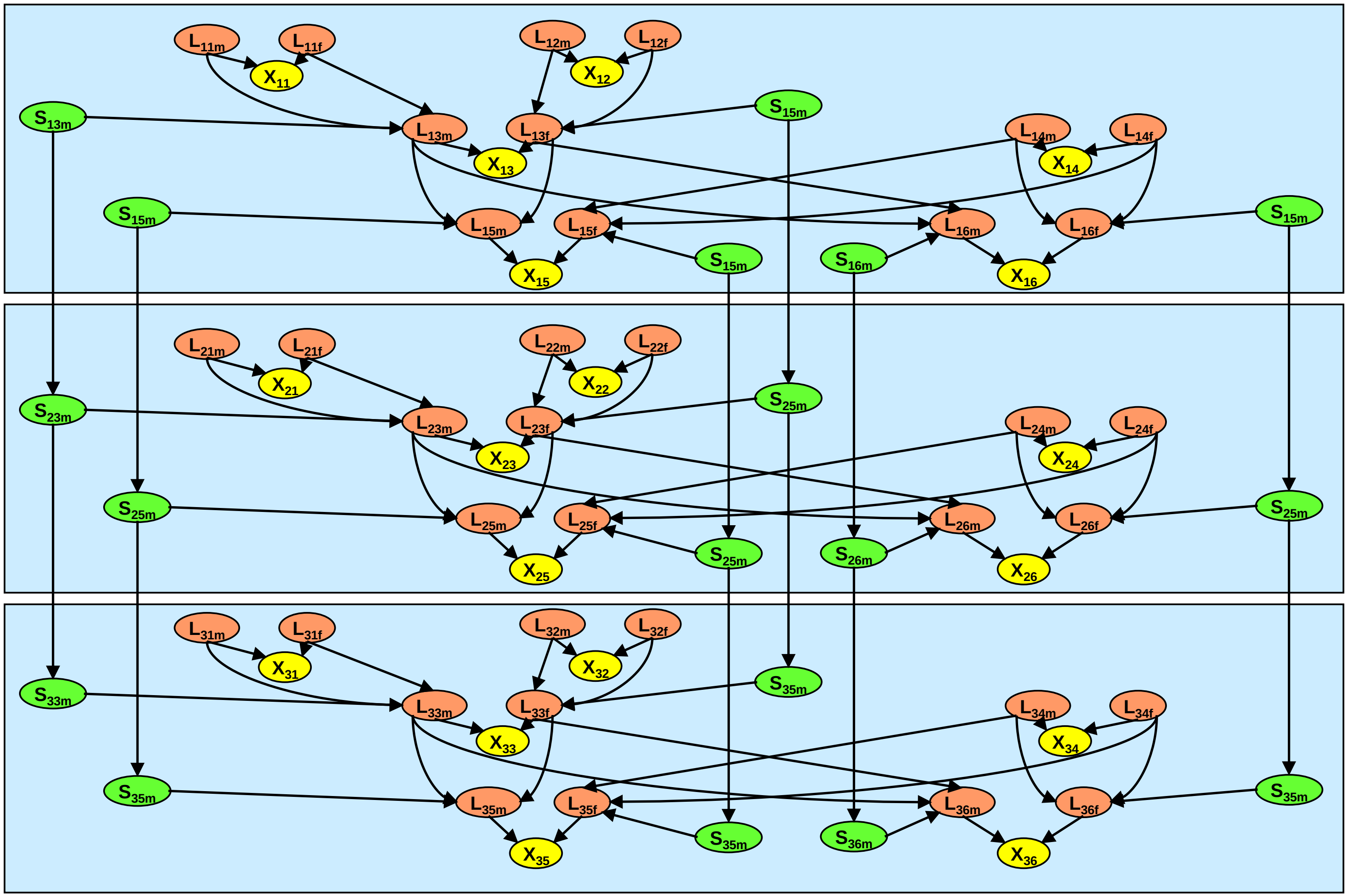


Genetic Linkage Analysis

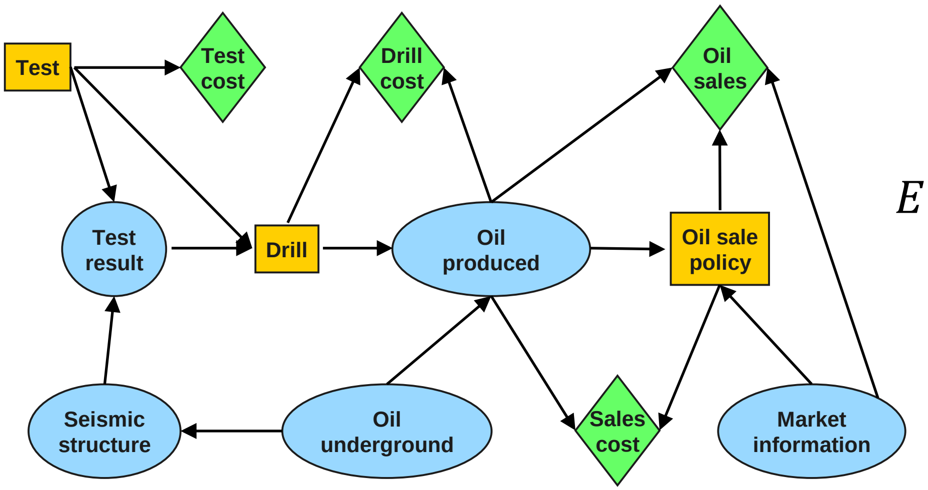


- 6 individuals
- Haplotype: *{2, 3}*
- Genotype: *{6}*
- Unknown

Pedigree: 6 people, 3 markers



Influence Diagrams



Task: find optimal policy

$$E = \max_{\Delta=(\delta_1,\ldots,\delta_m)} \sum_{x=(x_1,\ldots,x_n)} \prod_i P_i(x) \cdot u(x)$$

Chance variables: $X = x_1, \ldots, x_n$

Decision variables: $D = d_1, \ldots, d_m$

CPDs for chance variables: $P_i = P(x_i | x_{pa_i}), i = 1, \ldots, n$

Reward components: $r = \{r_1, \ldots, r_j\}$

Utility function: $u(x) = \sum_i r_i(x)$

Graphical Models

- A graphical model **(X,D,F)**:
  - **X** = $\{X_1,\ldots X_n\}$ variables
  - **D** = $\{D_1, \ldots D_n\}$ domains
  - **F** = $\{f_1,\ldots,f_m\}$ functions

    (constraints, CPTS, CNFs …)

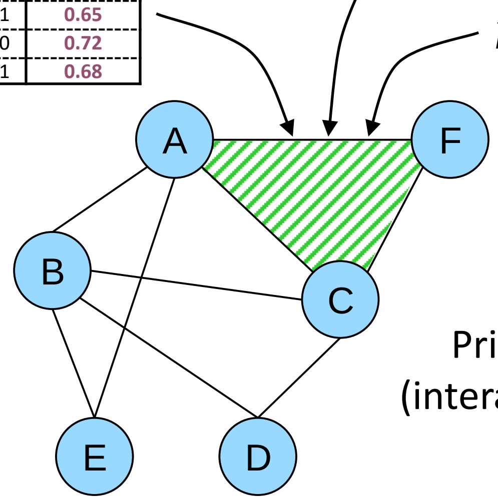
CPT

A	C	F	P(F|A,C)
0	0	0	0.14
0	0	1	0.96
0	1	0	0.40
0	1	1	0.60
1	0	0	0.35
1	0	1	0.65
1	1	0	0.72
1	1	1	0.68

Relation

A	C	F
red	green	blue
blue	red	red
blue	blue	green
green	red	blue

$f_i \;:=\; (F = A + C)$



Primal graph (interaction graph)

- Operators:
  - combination
  - elimination (projection)

- Tasks:
  - **Belief updating**: $\Sigma_{X\backslash y} \prod_j P_i$
  - **MPE/MAP**: $\max_X \prod_j P_j$
  - **Marginal MAP:** $\max_Y \Sigma_{X\backslash Y} \Pi_j P_j$
  - **CSP**: $\prod_X \times_j C_j$
  - **WCSP**: $\min_X \Sigma_j f_j$

All these tasks are NP-hard

- exploit problem structure
- identify special cases
- approximate

Example Domains for Graphical Models

- Natural Language Processing
  - Information extraction, semantic parsing, translation, topic models, …
- Computer Vision
  - Object recognition, scene analysis, segmentation, tracking, …
- Computational Biology
  - Pedigree analysis, protein folding / binding / design, sequence matching, …
- Networks
  - Webpage link analysis, social networks, communications, citations, …
- Robotics
  - Planning and decision making, …
- …

Combinatorial Optimization Tasks

- Most Probable Explanation (MPE)
  or Maximum A Posteriori (MAP)
- M Best MPE/MAP
- Marginal MAP (MMAP)
- Weighted CSPs (WCSP), Max-CSP, Max-SAT
- Integer Linear Programming (ILP)
- Maximum Expected Utility (MEU)

Outline

- **Introduction**
  - Graphical models
  - Optimization tasks for graphical models
  - Solving optimization problems by inference and search
- **Inference**
- **Bounds and heuristics**
- **AND/OR search**
- **Exploiting parallelism**
- **Software**

Solution Techniques

AND/OR search

Time: exp(treewidth*log n)

Space: linear

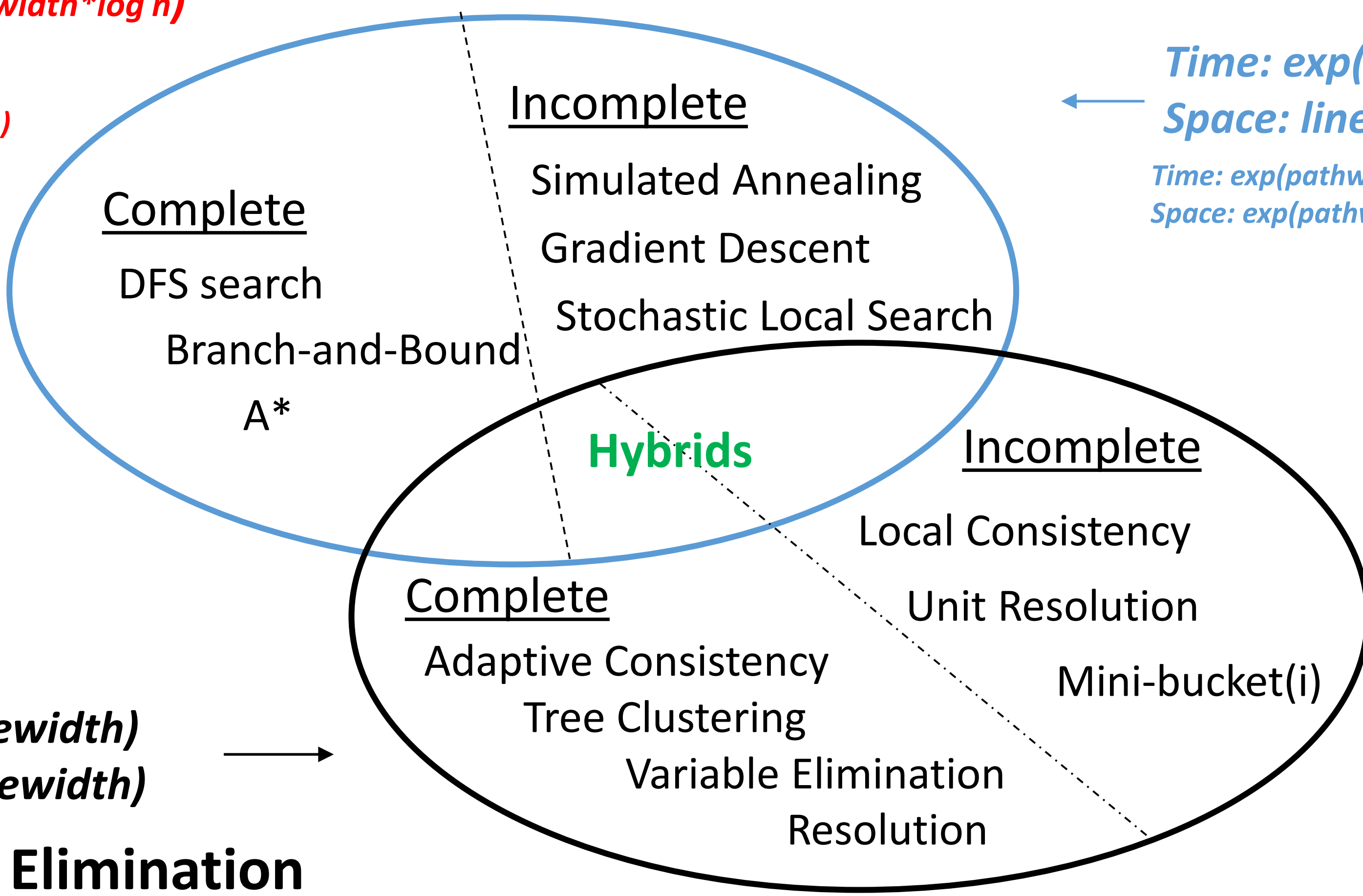
Space: exp(treewidth)
Time: exp(treewidth)

Search: Conditioning

Time: exp(n)

Space: linear

Time: exp(pathwidth)
Space: exp(pathwidth)



Time: exp(treewidth)

Space:exp(treewidth)

Inference: Elimination

Combination of Cost Functions

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

+

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

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

Elimination in a Cost Function

A	B	f(A,B)
b	b	4
b	g	6
b	r	1
g	b	2
g	g	6
g	r	3
r	b	1
r	g	1
r	r	6

Min

Elim(**f**,**B**)

A	g(A)
b	1
g	2
r	1

Elim(**g**,**A**)

$\mathbf{h}_{\varnothing}$
1

Conditioning in a Cost Function

A	B	f(A,B)
b	b	4
b	g	6
b	r	1
g	b	2
g	g	6
g	r	3
r	b	1
r	g	1
r	r	6

Assign **A=b**

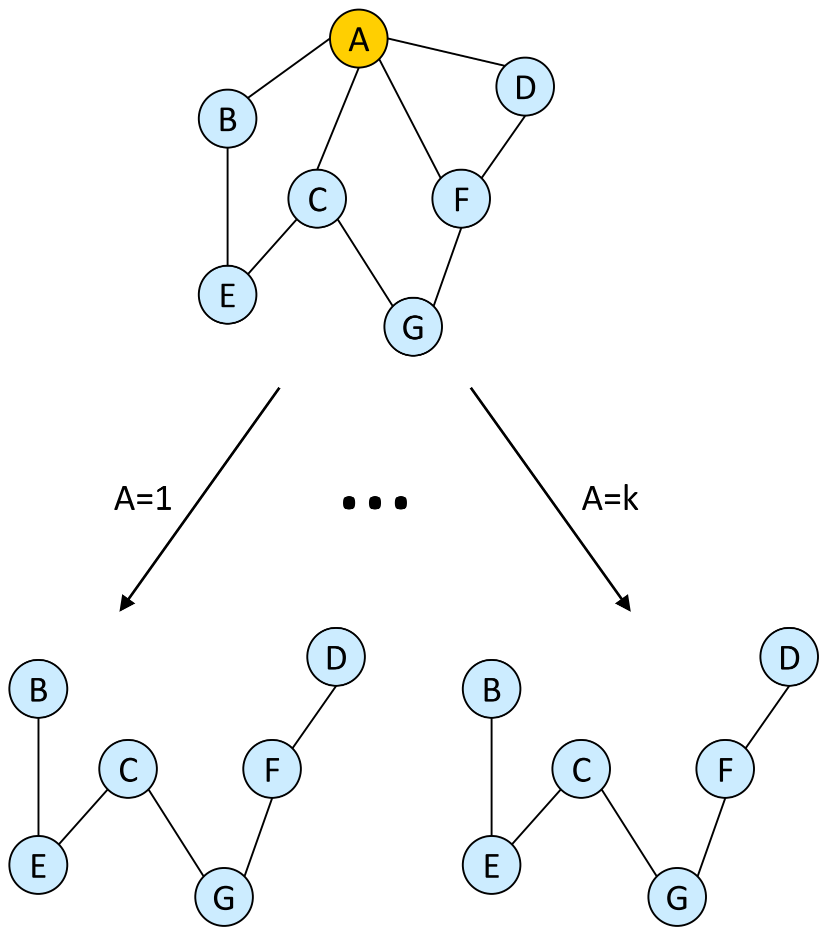
B	g(B)
b	4
g	6
r	1

Assign **B=r**

$h_{\varnothing}$
1

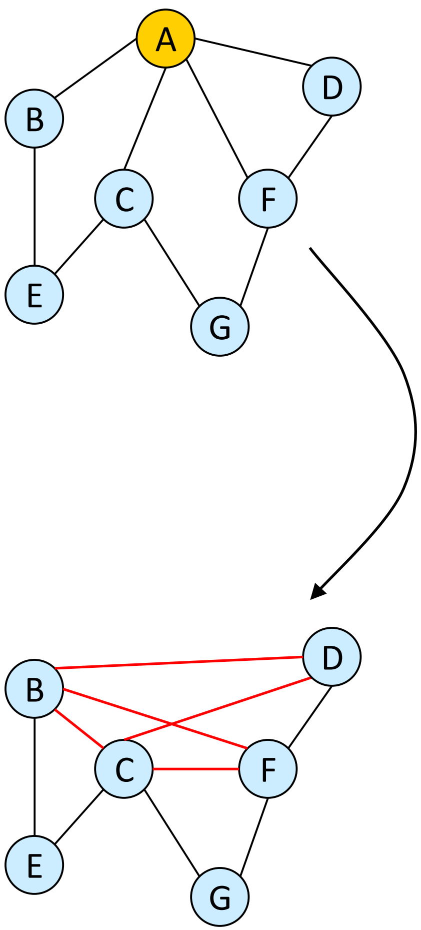
Conditioning versus Elimination

Conditioning (search)



k “sparser” problems

Elimination (inference)

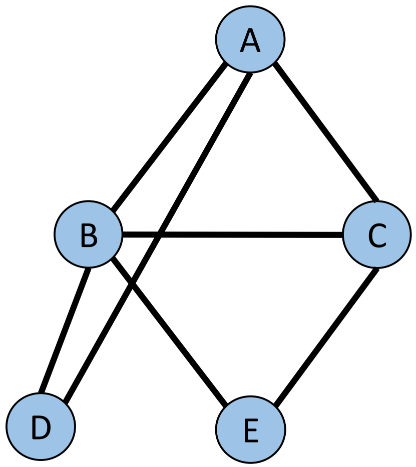


1 “denser” problem

Outline

- Introduction
- **Inference**
  - Variable Elimination, Bucket Elimination
- Bounds and heuristics
- AND/OR search
- Exploiting parallelism
- Software

Computing the Optimal Solution



Constraint graph

$$\textbf{OPT} = \min_{a,e,d,c,b} f(a) + f(a,b) + f(a,c) + f(a,d) + f(b,c) + f(b,d) + f(b,e) + f(c,e)$$

Combination

$$\min_{a} f(a) + \min_{e,d} f(a,d) + \min_{c} f(a,c) + f(c,e) + \min_{b} f(a,b) + f(b,c) + f(b,d) + f(b,e)$$

$$\lambda_B(a,d,c,e)$$

Variable Elimination

Bucket Elimination

Algorithm **elim-opt** [Dechter, 1996]
Non-serial Dynamic Programming [Bertele & Briochi, 1973]

$$OPT = \min_{a,e,d,c,b} f(a) + f(a,b) + f(a,c) + f(a,d) + f(b,d) + f(b,e) + f(c,e)$$

$\min_b \sum$ ← Elimination / Combination operators

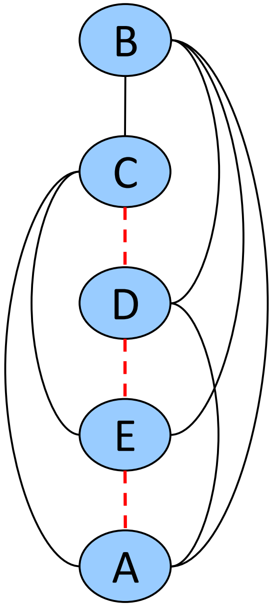
Bucket B: $f(a,b)$ $f(b,c)$ $f(b,d)$ $f(b,e)$

Bucket C: $f(c,a)$ $f(c,e)$

Bucket D: $f(a,d)$

Bucket E:

Bucket A: $f(a)$



Bucket Elimination

Algorithm **elim-opt** [Dechter, 1996]
Non-serial Dynamic Programming [Bertele & Briochi, 1973]

$$OPT = \min_{a,e,d,c,b} f(a) + f(a,b) + f(a,c) + f(a,d) + f(b,d) + f(b,e) + f(c,e)$$

$\min_b \sum$ ← Elimination / Combination operators

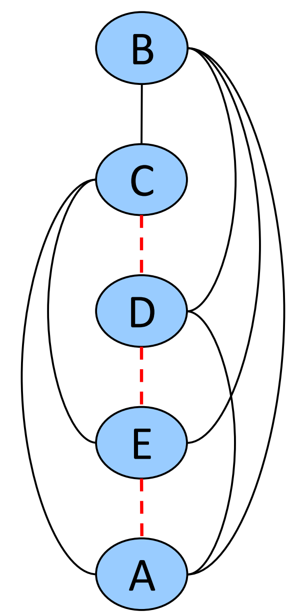
Bucket B: $f(a,b)$ $f(b,c)$ $f(b,d)$ $f(b,e)$

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

Bucket D: $f(a,d)$

Bucket E:

Bucket A: $f(a)$



Bucket Elimination

Algorithm **elim-opt** [Dechter, 1996]
Non-serial Dynamic Programming [Bertele & Briochi, 1973]

$$OPT = \min_{a,e,d,c,b} f(a) + f(a,b) + f(a,c) + f(a,d) + f(b,d) + f(b,e) + f(c,e)$$

$\min_b \sum$ ← Elimination / Combination operators

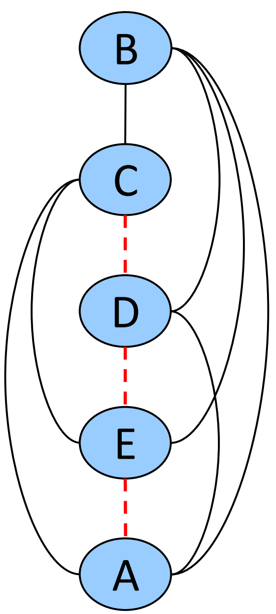
Bucket B: $f(a,b)$ $f(b,c)$ $f(b,d)$ $f(b,e)$

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

Bucket D: $f(a,d)$ $\lambda_{C \to D}(a,d,e)$

Bucket E:

Bucket A: $f(a)$



Bucket Elimination

Algorithm **elim-opt** [Dechter, 1996]
Non-serial Dynamic Programming [Bertele & Briochi, 1973]

$$OPT = \min_{a,e,d,c,b} f(a) + f(a,b) + f(a,c) + f(a,d) + f(b,d) + f(b,e) + f(c,e)$$

$\min_b \sum$ ← Elimination / Combination operators

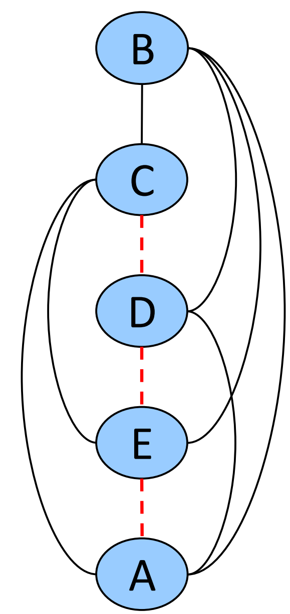
Bucket B: $f(a,b)$ $f(b,c)$ $f(b,d)$ $f(b,e)$

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

Bucket D: $f(a,d)$ $\lambda_{C\to D}(a,d,e)$

Bucket E: $\lambda_{D\to E}(a,e)$

Bucket A: $f(a)$



Bucket Elimination

Algorithm **elim-opt** [Dechter, 1996]
Non-serial Dynamic Programming [Bertele & Briochi, 1973]

$$OPT = \min_{a,e,d,c,b} f(a) + f(a,b) + f(a,c) + f(a,d) + f(b,d) + f(b,e) + f(c,e)$$

$\min_b \sum$ ← Elimination / Combination operators

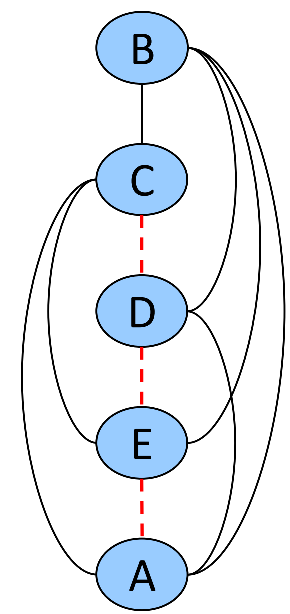
Bucket B: $f(a,b)$ $f(b,c)$ $f(b,d)$ $f(b,e)$

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

Bucket D: $f(a,d)$ $\lambda_{C\to D}(a,d,e)$

Bucket E: $\lambda_{D\to E}(a,e)$

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



Bucket Elimination

Algorithm **elim-opt** [Dechter, 1996]
Non-serial Dynamic Programming [Bertele & Briochi, 1973]

$$OPT = \min_{a,e,d,c,b} f(a) + f(a,b) + f(a,c) + f(a,d) + f(b,d) + f(b,e) + f(c,e)$$

$\min_b \sum$ ← Elimination / Combination operators

Bucket B: $f(a,b)$ $f(b,c)$ $f(b,d)$ $f(b,e)$

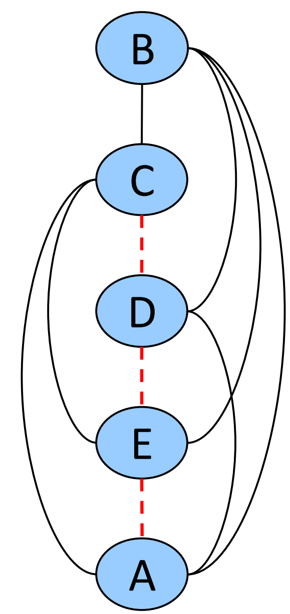
Bucket C: $f(c,a)$ $f(c,e)$ $\lambda_{B\rightarrow C}(a,d,c,e)$

Bucket D: $f(a,d)$ $\lambda_{C\rightarrow D}(a,d,e)$

Bucket E: $\lambda_{D\rightarrow E}(a,e)$

Bucket A: $f(a)$ $\lambda_{E\rightarrow A}(a)$

OPT



Generating the Optimal Assignment

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

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

D: $f(a,d)$ $\lambda_{C \to D}(a,d,e)$

E: $\lambda_{D \to E}(a,e)$

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

Return: $(a^*, b^*, c^*, d^*, e^*)$

Generating the Optimal Assignment

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

C: $f(c,a)$ $f(c,e)$ $\lambda_{B\rightarrow C}(a,d,c,e)$

D: $f(a,d)$ $\lambda_{C\rightarrow D}(a,d,e)$

E: $\lambda_{D\rightarrow E}(a,e)$

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

$a^* = argmin_a\, f(a) + \lambda_{E\rightarrow A}(a)$

Return: $(a^*, b^*, c^*, d^*, e^*)$

Generating the Optimal Assignment

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

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

D: $f(a,d)$ $\lambda_{C\to D}(a,d,e)$

E: $\lambda_{D\to E}(a,e)$

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

$$e^* = argmin_e\, \lambda_{D\to E}(a^*, e)$$

$$a^* = argmin_a\, f(a) + \lambda_{E\to A}(a)$$

Return: $(a^*, b^*, c^*, d^*, e^*)$

Generating the Optimal Assignment

$$b^* = argmin_b\ f(a^*, b) + f(b, c^*) + f(b, d^*) + f(b, e^*)$$

$$c^* = argmin_c\ f(c, a^*) + f(c, e^*) + \lambda_{B\to C}(a^*, d^*, c, e^*)$$

$$d^* = argmin_d\ f(a^*, d) + \lambda_{C\to D}(a^*, d, e^*)$$

$$e^* = argmin_e\ \lambda_{D\to E}(a^*, e)$$

$$a^* = argmin_a\ f(a) + \lambda_{E\to A}(a)$$

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

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

D: $f(a,d)$ $\lambda_{C\to D}(a,d,e)$

E: $\lambda_{D\to E}(a,e)$

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

Return: $(a^*, b^*, c^*, d^*, e^*)$

Complexity of Bucket Elimination

Algorithm **elim-opt** [Dechter, 1996]
Non-serial Dynamic Programming [Bertele & Briochi, 1973]

$$OPT = \min_{a,e,d,c,b} f(a) + f(a,b) + f(a,c) + f(a,d) + f(b,d) + f(b,e) + f(c,e)$$

$\min_b \sum$ ← Elimination / Combination operators

Bucket B: $f(a,b)$ $f(b,c)$ $f(b,d)$ $f(b,e)$

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

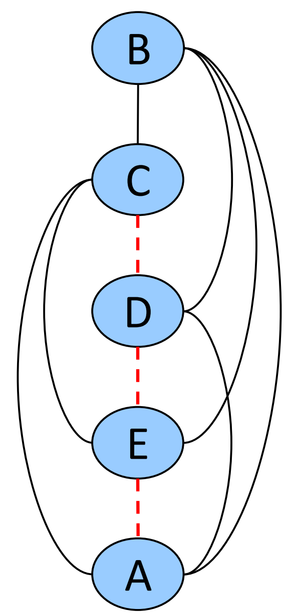
Bucket D: $f(a,d)$ $\lambda_{C\to D}(a,d,e)$

Bucket E: $\lambda_{D\to E}(a,e)$

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

OPT

$\exp(w^* = 4)$
"induced width"
(max clique size)



Complexity of Bucket Elimination

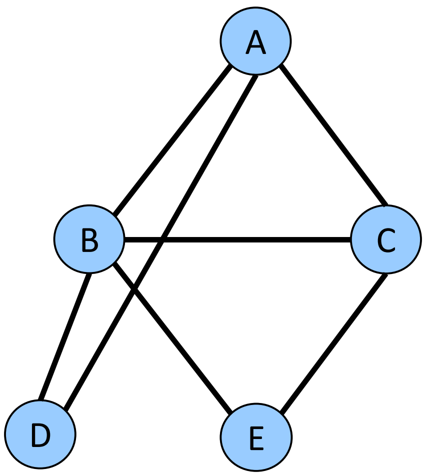
Bucket-Elimination is **time** and **space**

$O(r \cdot \exp(w_d^*))$

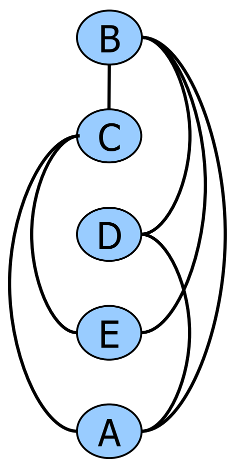
$w_d^* - \textit{the induced width of the primal graph along ordering } d$

r = number of functions

The effect of the ordering:



primal graph



$w_{d_1}^* = 4$

Complexity of Bucket Elimination

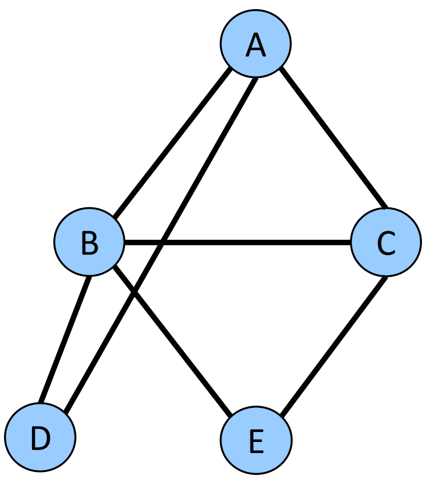
Bucket-Elimination is **time** and **space**

$$O(r \cdot \exp(w_d^*))$$

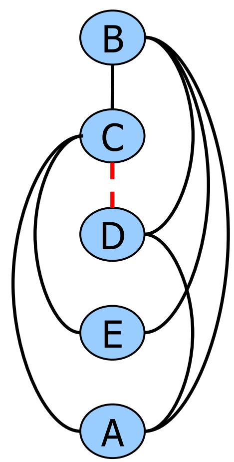
$w_d^* - \textit{the induced width of the primal graph along ordering } d$

r = number of functions

The effect of the ordering:



primal graph



$w_{d_1}^* = 4$

Complexity of Bucket Elimination

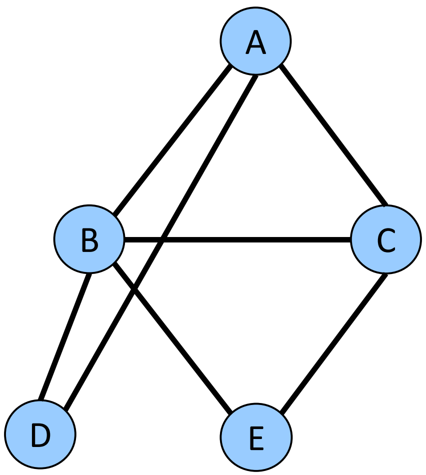
Bucket-Elimination is time and space

$O(r \cdot \exp(w_d^*))$

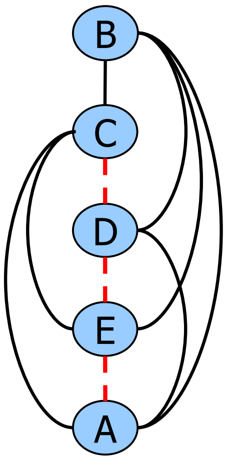
w_d^* – *the induced width of the primal graph along ordering d*

r = number of functions

The effect of the ordering:



primal graph



$w_{d_1}^* = 4$

Complexity of Bucket Elimination

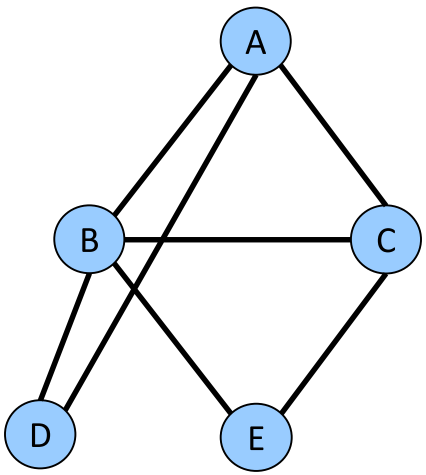
Bucket-Elimination is **time** and **space**

$O(r \cdot \exp(w_d^*))$

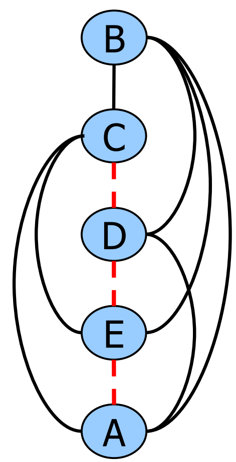
w_d^* – *the induced width of the primal graph along ordering d*

r = number of functions

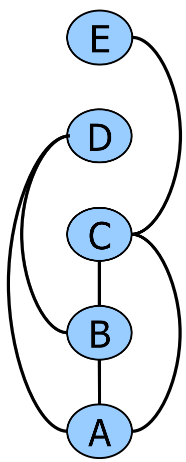
The effect of the ordering:



primal graph



$w^*_{d_1} = 4$



$w^*_{d_2} = 2$

Complexity of Bucket Elimination

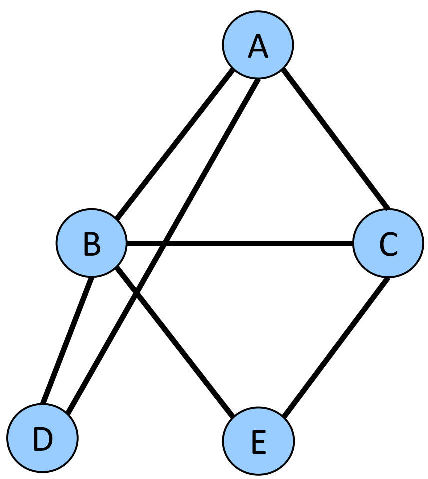
Bucket-Elimination is **time** and **space**

$O(r \cdot \exp(w_d^*))$

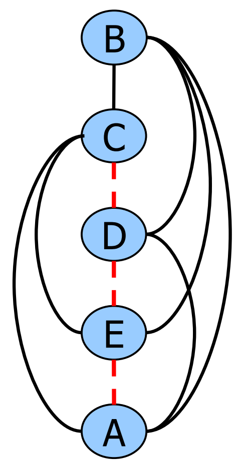
w_d^* $- \, the\ induced\ width\ of\ the\ primal\ graph\ along\ ordering\ d$

r = number of functions

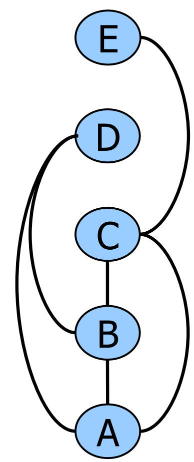
The effect of the ordering:



primal graph



$w_{d_1}^* = 4$



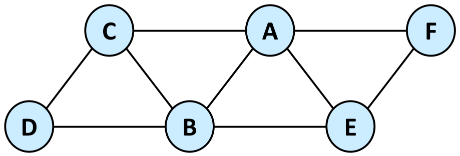
$w_{d_2}^* = 2$

Finding smallest induced-width is hard!

Outline

- Introduction
- Inference
- **Bounds and heuristics**
  - Basics of search: DFS versus BFS
  - Mini-bucket elimination
  - Weighted mini-buckets and iterative cost-shifting
  - Generating heuristics using mini-bucket elimination
- AND/OR search
- Exploiting parallelism
- Software

OR Search Spaces



A	B	f_1
0	0	**2**
0	1	**0**
1	0	**1**
1	1	**4**

A	C	f_2
0	0	**3**
0	1	**0**
1	0	**0**
1	1	**1**

A	E	f_3
0	0	**0**
0	1	**3**
1	0	**2**
1	1	**0**

A	F	f_4
0	0	**2**
0	1	**0**
1	0	**0**
1	1	**2**

B	C	f_5
0	0	**0**
0	1	**1**
1	0	**2**
1	1	**4**

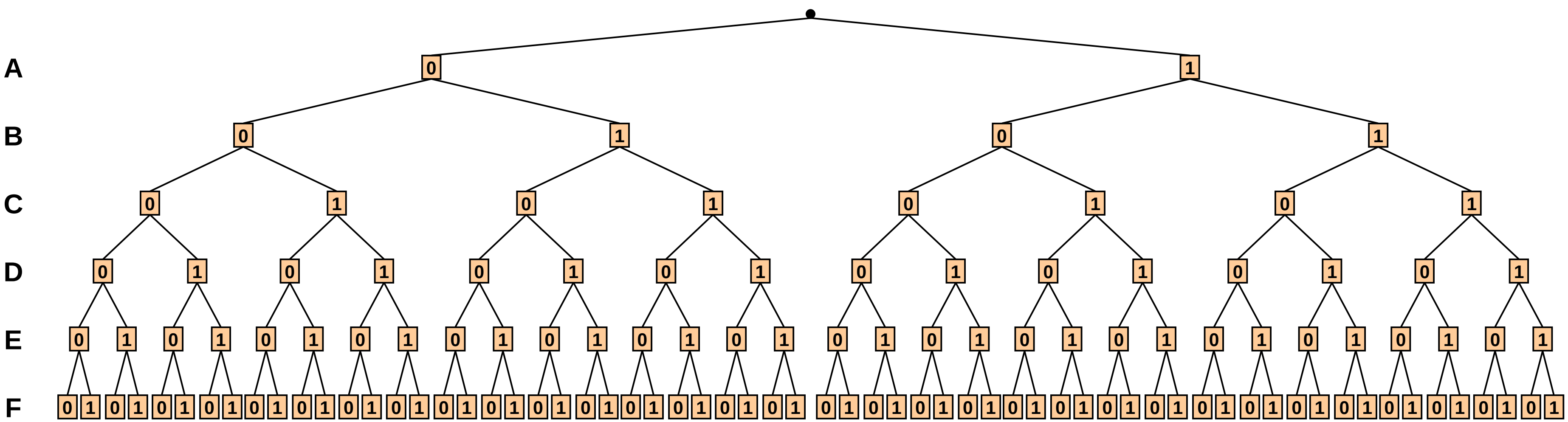
B	D	f_6
0	0	**4**
0	1	**2**
1	0	**1**
1	1	**0**

B	E	f_7
0	0	**3**
0	1	**2**
1	0	**1**
1	1	**0**

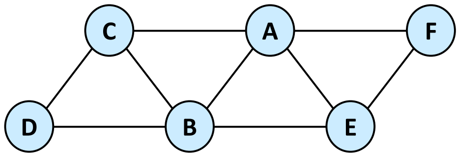
C	D	f_8
0	0	**1**
0	1	**4**
1	0	**0**
1	1	**0**

E	F	f_9
0	0	**1**
0	1	**0**
1	0	**0**
1	1	**2**

Objective function: $F^* = \min_x \sum_\alpha f_\alpha(x_\alpha)$



OR Search Spaces



A	B	f_1
0	0	**2**
0	1	**0**
1	0	**1**
1	1	**4**

A	C	f_2
0	0	**3**
0	1	**0**
1	0	**0**
1	1	**1**

A	E	f_3
0	0	**0**
0	1	**3**
1	0	**2**
1	1	**0**

A	F	f_4
0	0	**2**
0	1	**0**
1	0	**0**
1	1	**2**

B	C	f_5
0	0	**0**
0	1	**1**
1	0	**2**
1	1	**4**

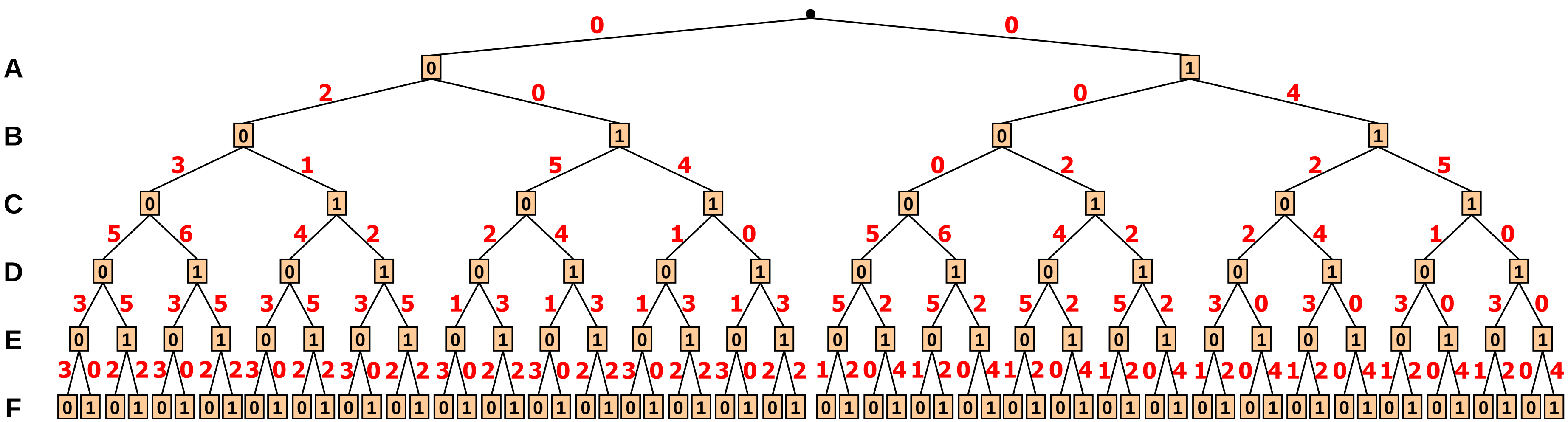
B	D	f_6
0	0	**4**
0	1	**2**
1	0	**1**
1	1	**0**

B	E	f_7
0	0	**3**
0	1	**2**
1	0	**1**
1	1	**0**

C	D	f_8
0	0	**1**
0	1	**4**
1	0	**0**
1	1	**0**

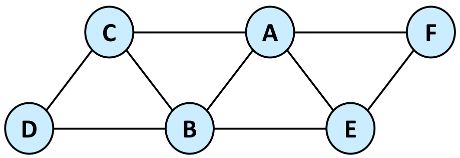
E	F	f_9
0	0	**1**
0	1	**0**
1	0	**0**
1	1	**2**

Objective function: $$F^* = \min_x \sum_\alpha f_\alpha(x_\alpha)$$



Arc cost is calculated based on cost functions with empty scope (conditioning)

The Value Function



A	B	f_1
0	0	**2**
0	1	**0**
1	0	**1**
1	1	**4**

A	C	f_2
0	0	**3**
0	1	**0**
1	0	**0**
1	1	**1**

A	E	f_3
0	0	**0**
0	1	**3**
1	0	**2**
1	1	**0**

A	F	f_4
0	0	**2**
0	1	**0**
1	0	**0**
1	1	**2**

B	C	f_5
0	0	**0**
0	1	**1**
1	0	**2**
1	1	**4**

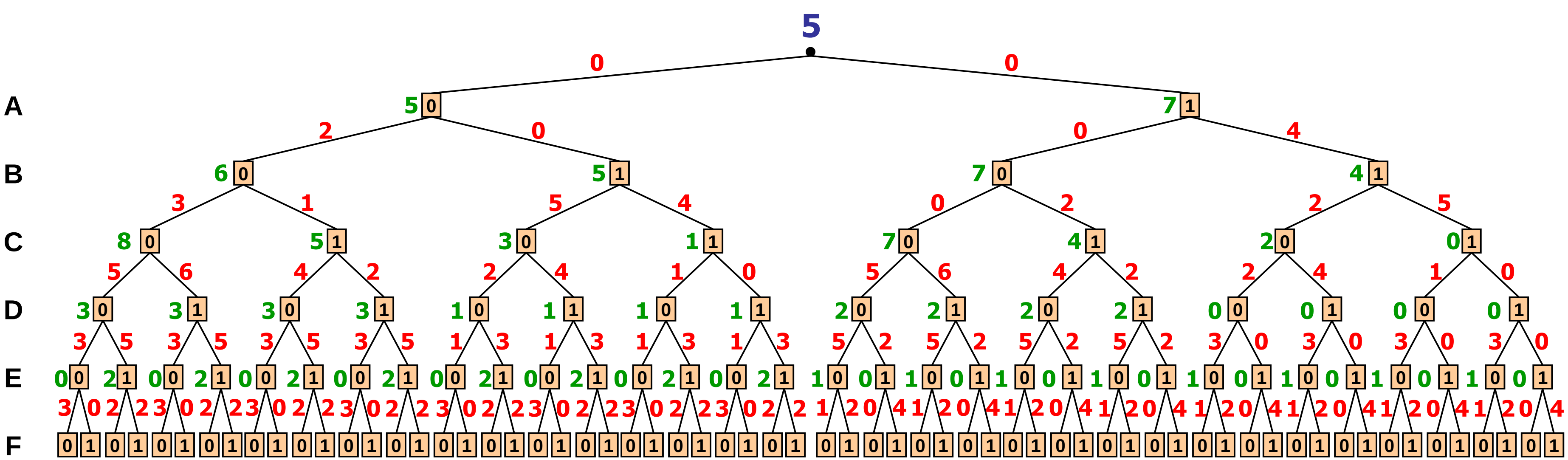
B	D	f_6
0	0	**4**
0	1	**2**
1	0	**1**
1	1	**0**

B	E	f_7
0	0	**3**
0	1	**2**
1	0	**1**
1	1	**0**

C	D	f_8
0	0	**1**
0	1	**4**
1	0	**0**
1	1	**0**

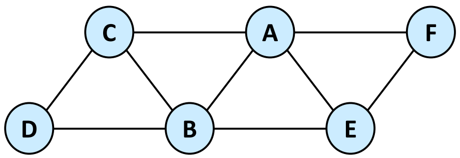
E	F	f_9
0	0	**1**
0	1	**0**
1	0	**0**
1	1	**2**

Objective function: $F^* = \min_x \sum_\alpha f_\alpha(x_\alpha)$



Value of node = minimal cost solution below it

The Optimal Solution



A	B	f_1
0	0	**2**
0	1	**0**
1	0	**1**
1	1	**4**

A	C	f_2
0	0	**3**
0	1	**0**
1	0	**0**
1	1	**1**

A	E	f_3
0	0	**0**
0	1	**3**
1	0	**2**
1	1	**0**

A	F	f_4
0	0	**2**
0	1	**0**
1	0	**0**
1	1	**2**

B	C	f_5
0	0	**0**
0	1	**1**
1	0	**2**
1	1	**4**

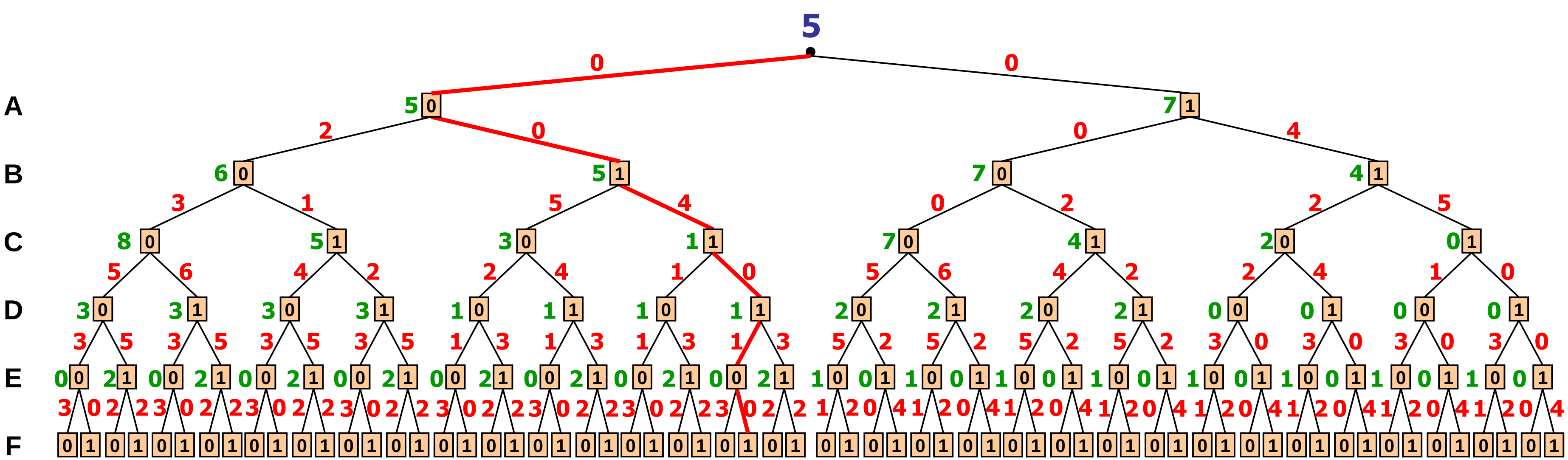
B	D	f_6
0	0	**4**
0	1	**2**
1	0	**1**
1	1	**0**

B	E	f_7
0	0	**3**
0	1	**2**
1	0	**1**
1	1	**0**

C	D	f_8
0	0	**1**
0	1	**4**
1	0	**0**
1	1	**0**

E	F	f_9
0	0	**1**
0	1	**0**
1	0	**0**
1	1	**2**

Objective function: $$F^* = \min_x \sum_\alpha f_\alpha(x_\alpha)$$



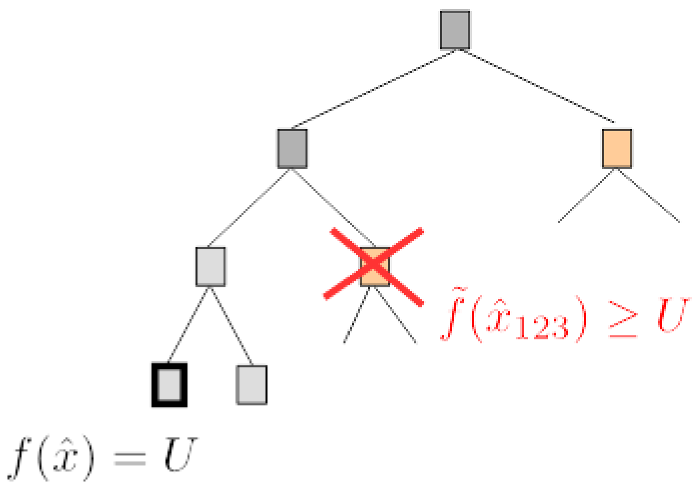
Value of node = minimal cost solution below it

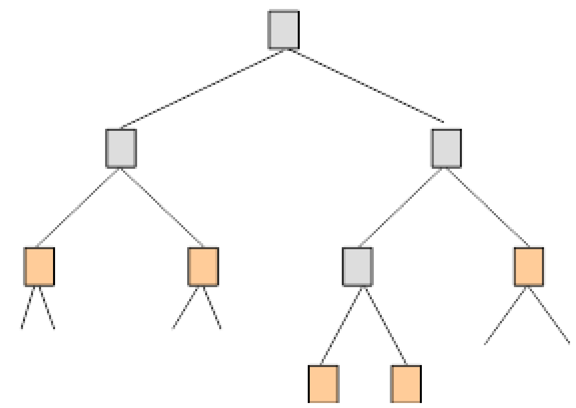
Basic Heuristic Search Schemes

Heuristic function $\tilde{f}(\hat{x}_p)$ computes a lower bound on the best extension of partial configuration $\hat{x}_p$ and can be used to guide heuristic search.
We focus on:

1. Branch-and-Bound
Use heuristic function $\tilde{f}(\hat{x}_p)$ to prune the depth-first search tree
Linear space

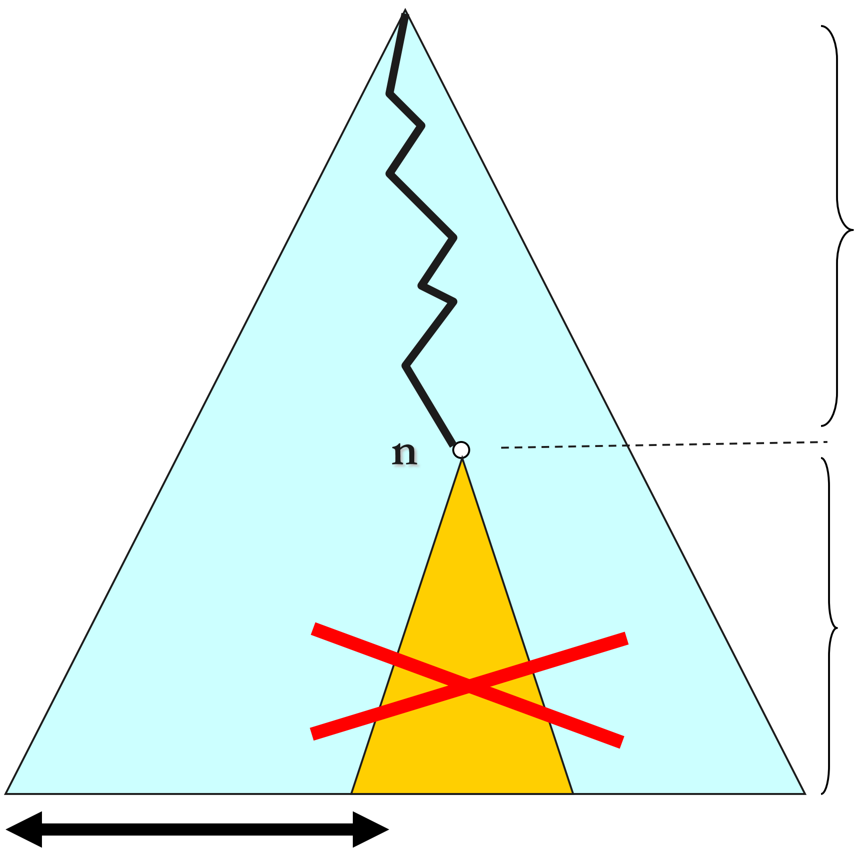
2. Best-First Search
Always expand the node with the lowest heuristic value $\tilde{f}(\hat{x}_p)$
Needs lots of memory





Depth-First Branch and Bound

Each node is a COP sub-problem
(defined by current conditioning)



$g(n)$: cost of the path from root to n

$$\tilde{f}(n) = g(n) + \tilde{h}(n)$$

(lower bound)

Prune if $\tilde{f}(n) \geq UB$

$\tilde{h}(n)$: under-estimates optimal cost below n

(UB) Upper Bound = best solution so far

Best-First vs Depth-First Branch and Bound

- **Best-First (A*)**:
  - Expands least number of nodes given h
  - Requires storing full search tree in memory
- **Depth-First BnB**:
  - Can use linear space
  - If finds an optimal solution early, will expand the same search space as Best-First (if search space is a tree)
  - BnB can improve the heuristic function dynamically

How to Generate Heuristics

- The principle of relaxed models
  - Mini-Bucket Elimination
  - Bounded directional consistency ideas
  - Linear relaxations for integer linear programs

Outline

- Introduction
- Inference
- **Bounds and heuristics**
  - Basics of search: DFS versus BFS
  - Mini-bucket elimination
  - Weighted mini-buckets and iterative cost-shifting
  - Generating heuristics using mini-bucket elimination
- AND/OR search
- Exploiting parallelism
- Software

Mini-Bucket Approximation

Split a bucket into mini-buckets —> bound complexity

bucket (X) =

$$\{\, f_1, \dots, f_r, f_{r+1}, \dots, f_n \,\}$$

$$\lambda_X(\cdot) = \min_x \sum_{i=1}^{n} f_i(x, \dots)$$

$$\{\, f_1, \dots, f_r \}$$

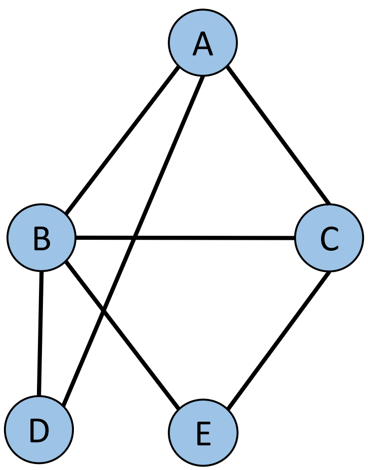
$$\{\, f_{r+1}, \dots, f_n \,\}$$

$$\lambda'_X(\cdot) = \left(\min_x \sum_{i=1}^{r} f_i(\cdot)\right) + \left(\min_x \sum_{i=r+1}^{n} f_i(\cdot)\right)$$

$$\lambda'_X(\cdot) \leq \lambda_X(\cdot)$$

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

Mini-Bucket Elimination



mini-buckets

$\min_B \sum f(\cdot)$ $\quad$ $\min_B \sum f(\cdot)$

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

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

bucket D: $f(a,d)\ \lambda_{B\to D}(d,e)$

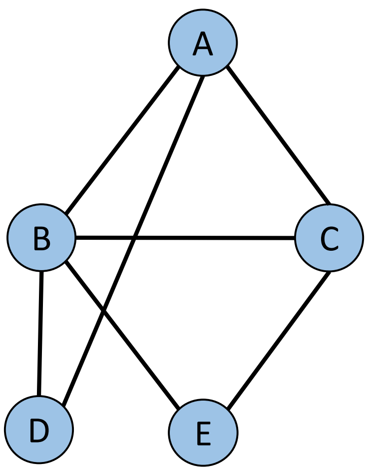
bucket E: $\lambda_{C\to E}(a,e)\ \lambda_{D\to E}(a,e)$

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

L = lower bound

[Dechter and Rish, 2003]

Mini-Bucket Elimination



mini-buckets: $\min_{B'} \sum f(\cdot)$ and $\min_{B} \sum f(\cdot)$

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

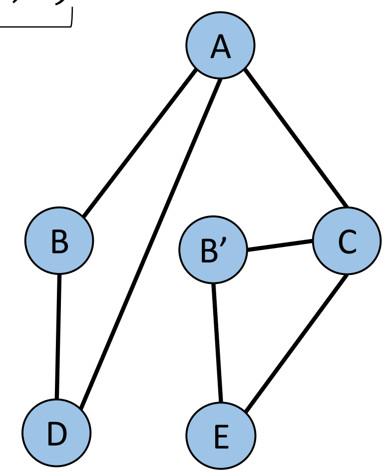
bucket C: $\lambda_{B\to C}(a,c)\ f(c,a)\ f(c,e)$

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

L = lower bound

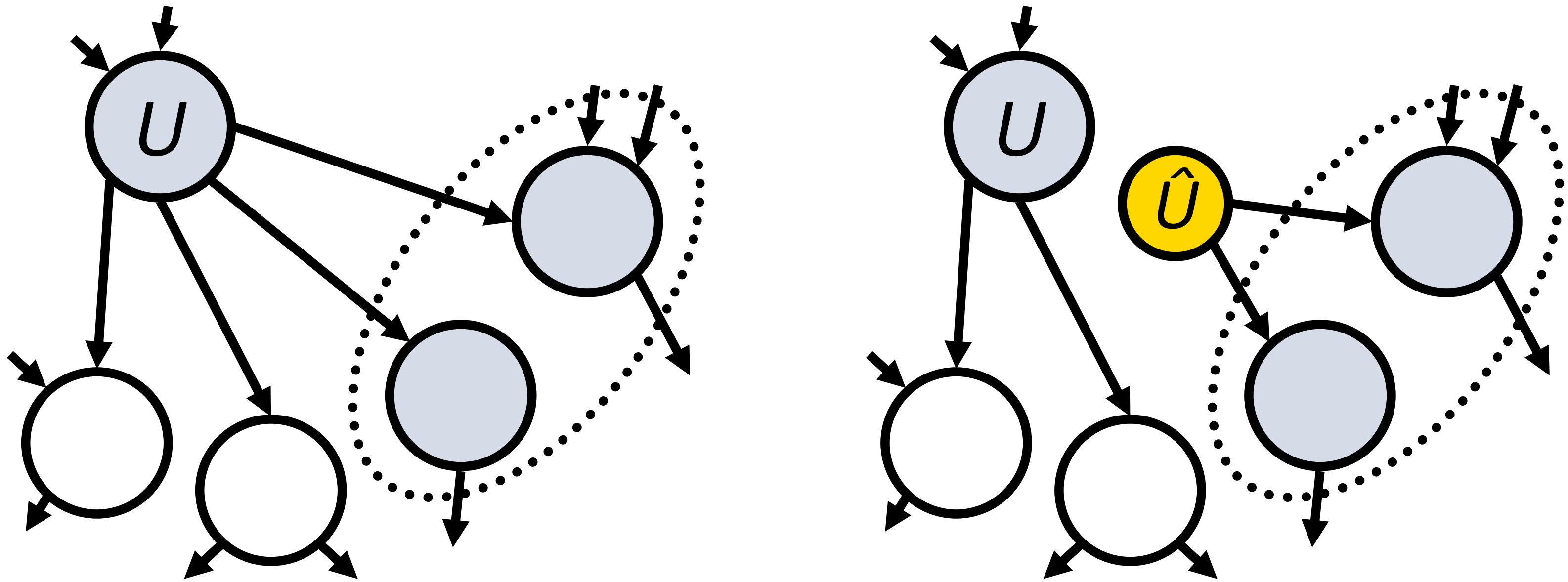


Semantics of Mini-Buckets: Splitting a Node

Variables in different buckets are renamed and duplicated
[Kask *et al.*, 2001], [Geffner *et al.*, 2007], [Choi *et al.*, 2007], [Johnson *et al.*, 2007]

Before Splitting:
Network N

After Splitting:
Network N'



MBE-MPE(i): Algorithm Approx-MPE

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

Example: approx-mpe(3) versus **elim-mpe**

approx-mpe(3)		max variables in a mini-bucket	elim-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:	$f(a)\ \lambda_{E\to A}(a)$
	L = lower bound $w^* = 2$			**OPT** $w^* = 4$

[Dechter and Rish, 1997]

Mini-Bucket Decoding

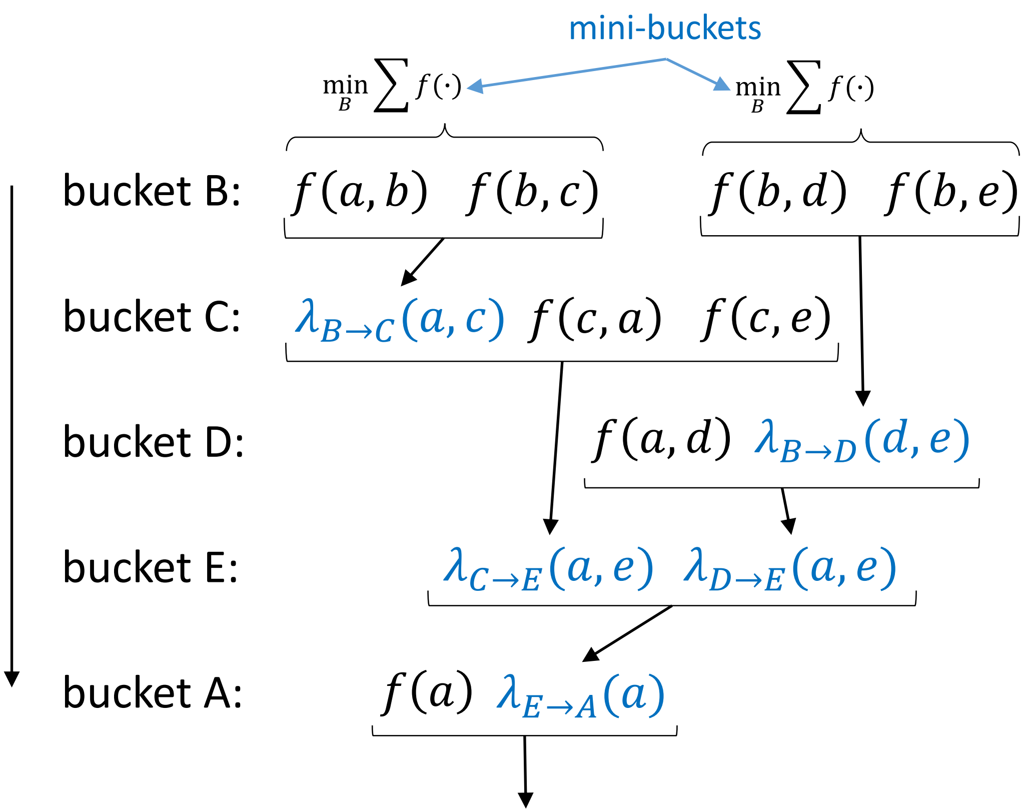
$$\hat{b} = \arg\min_b f(\hat{a}, b) + f(b, \hat{c}) + f(b, \hat{d}) + f(b, \hat{e})$$

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

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

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

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



Greedy configuration = upper bound

L = lower bound

[Dechter and Rish, 2003]

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]

Outline

- Introduction
- Inference
- **Bounds and heuristics**
  - Basics of search: DFS versus BFS
  - Mini-bucket elimination
  - Weighted mini-buckets and iterative cost-shifting
  - Generating heuristics using mini-bucket elimination
- AND/OR search
- Exploiting parallelism
- Software

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

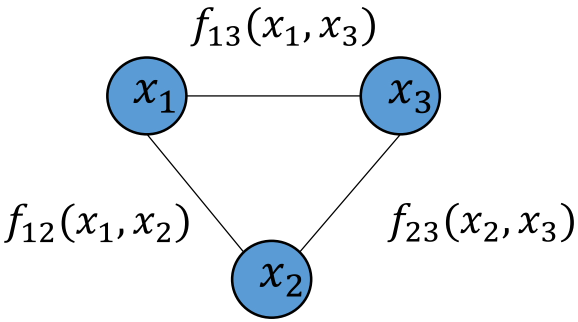
B	λ(B)
b	3
g	-1

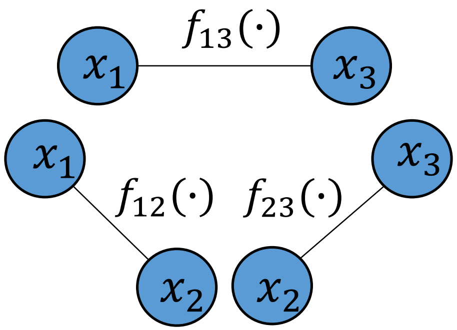
Modify the individual functions

- but –

keep the sum of functions the same

Dual Decomposition

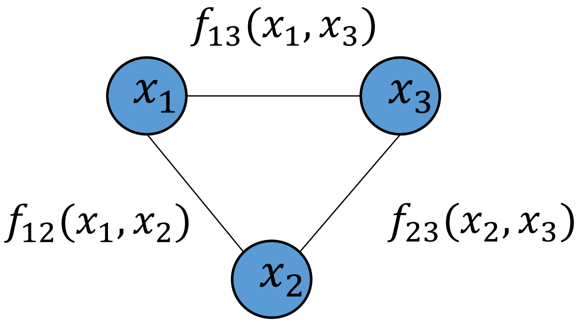


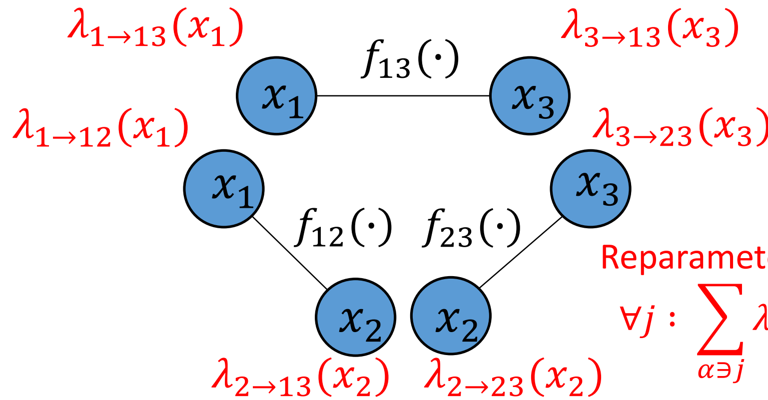


$$F^* = \min_x \sum_\alpha f_\alpha(x) \qquad \geq \qquad \sum_\alpha \min_x \; f_\alpha(x)$$

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

Dual Decomposition





Reparameterization:

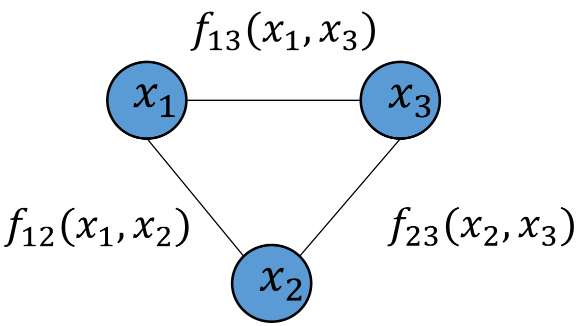
$$\forall j : \sum_{\alpha \ni j} \lambda_{j\to\alpha}(x_j) = 0$$

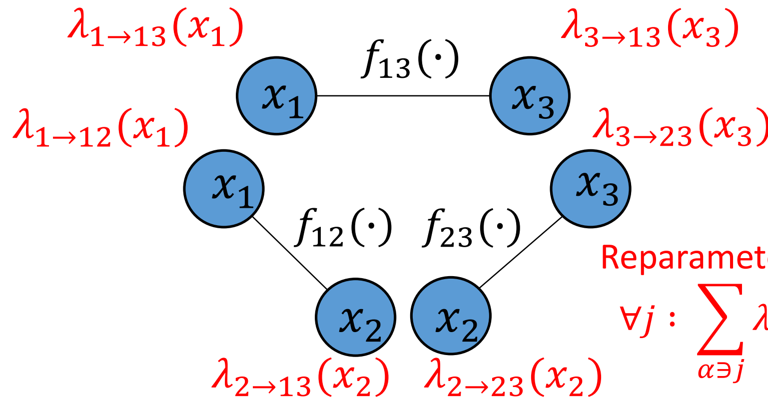
$$F^* = \min_x \sum_\alpha f_\alpha(x) \quad \geq \quad \max_{\lambda_{i\to\alpha}} \sum_\alpha \min_x \left[f_\alpha(x) + \sum_{i \in \alpha} \lambda_{i\to\alpha}(x_i) \right]$$

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

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

Dual Decomposition





Reparameterization:

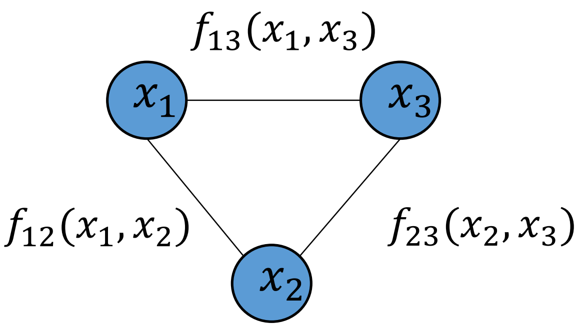
$$\forall j: \sum_{\alpha \ni j} \lambda_{j\to\alpha}(x_j) = 0$$

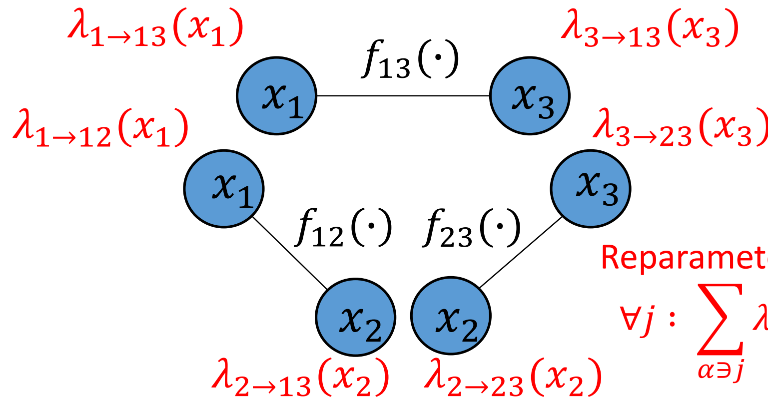
$$F^* = \min_x \sum_\alpha f_\alpha(x) \quad \geq \quad \max_{\lambda_{i\to\alpha}} \sum_\alpha \min_x \left[f_\alpha(x) + \sum_{i\in\alpha} \lambda_{i\to\alpha}(x_i) \right]$$

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:

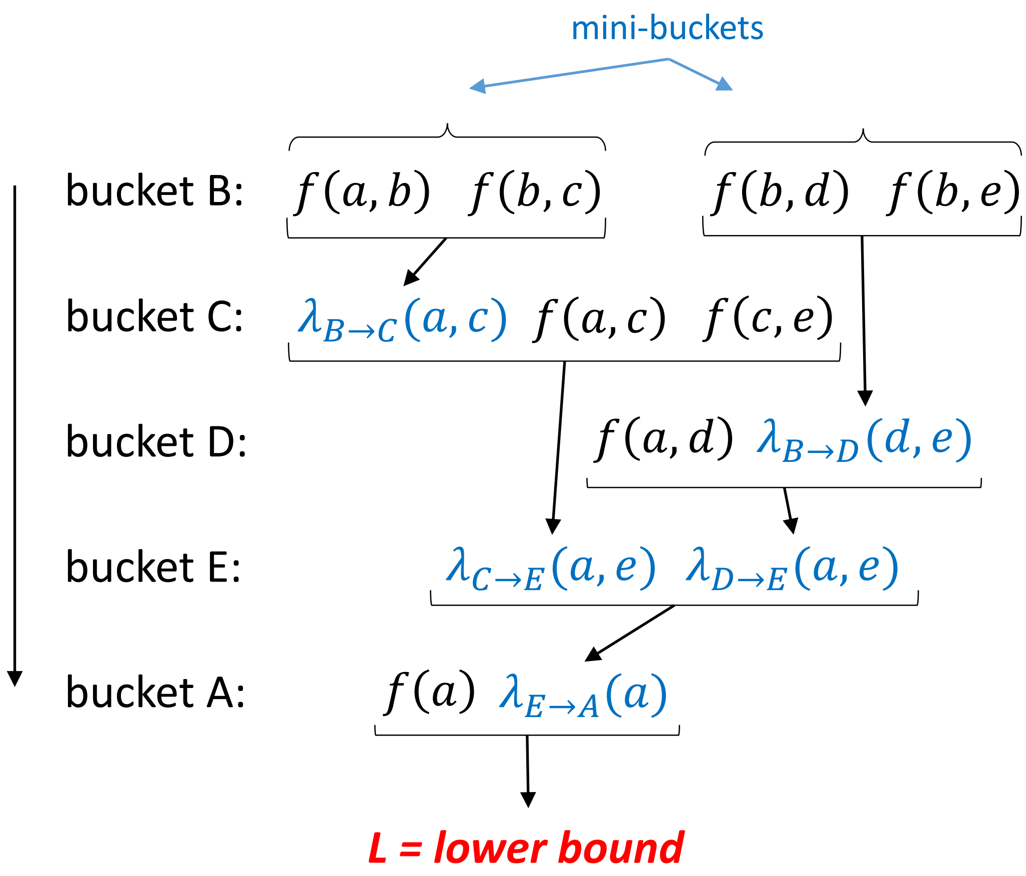
$$\forall j : \sum_{\alpha \ni j} \lambda_{j\to\alpha}(x_j) = 0$$

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

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]

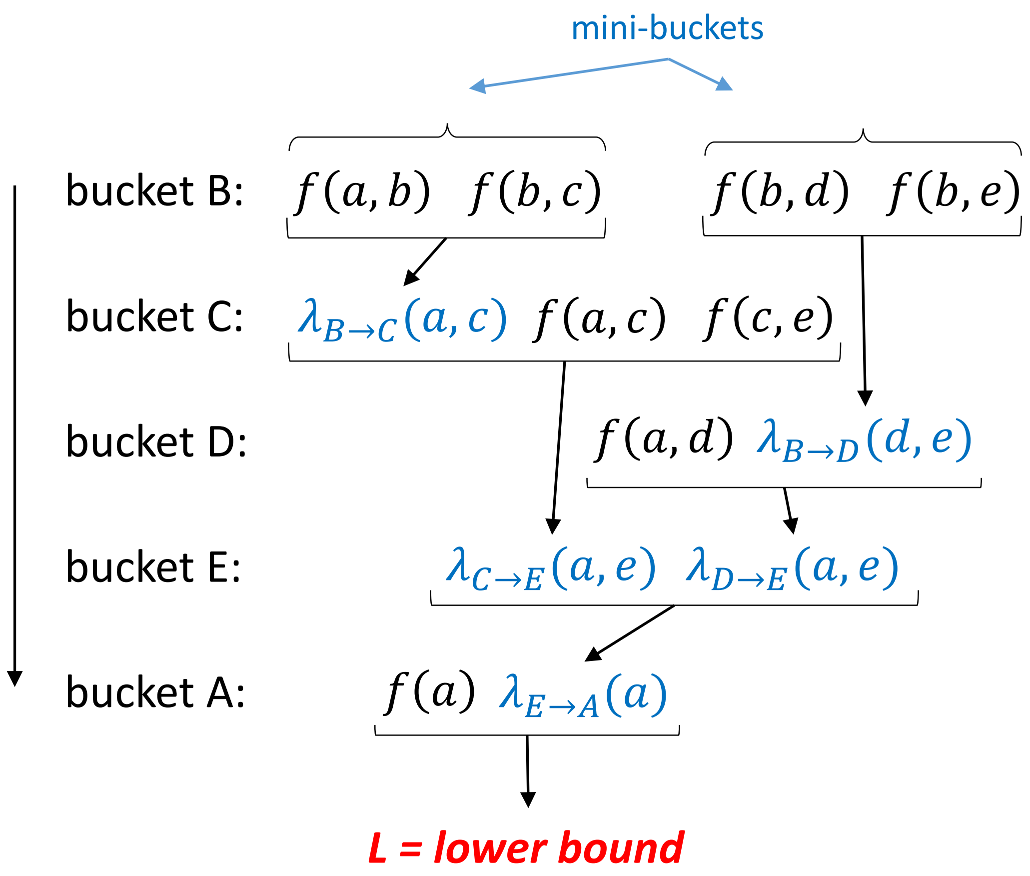
Mini-Bucket as Dual Decomposition



Mini-Bucket as Dual Decomposition

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

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

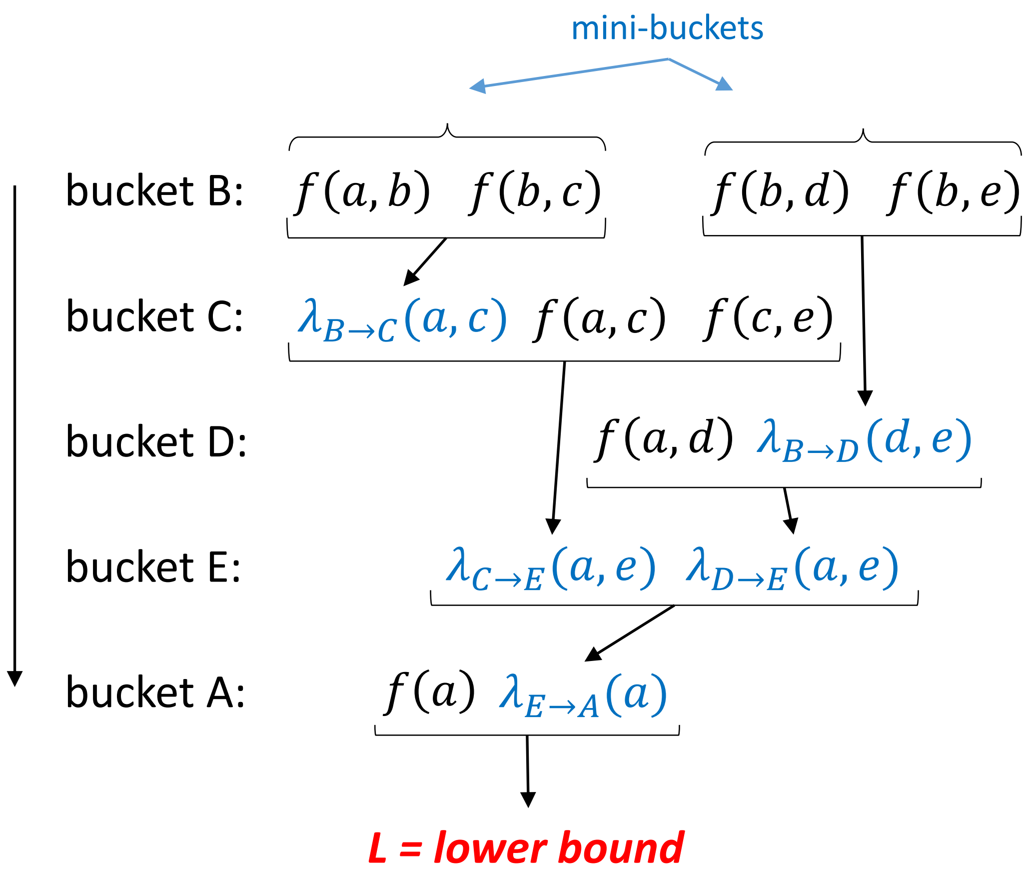


Mini-Bucket as Dual Decomposition

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

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

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



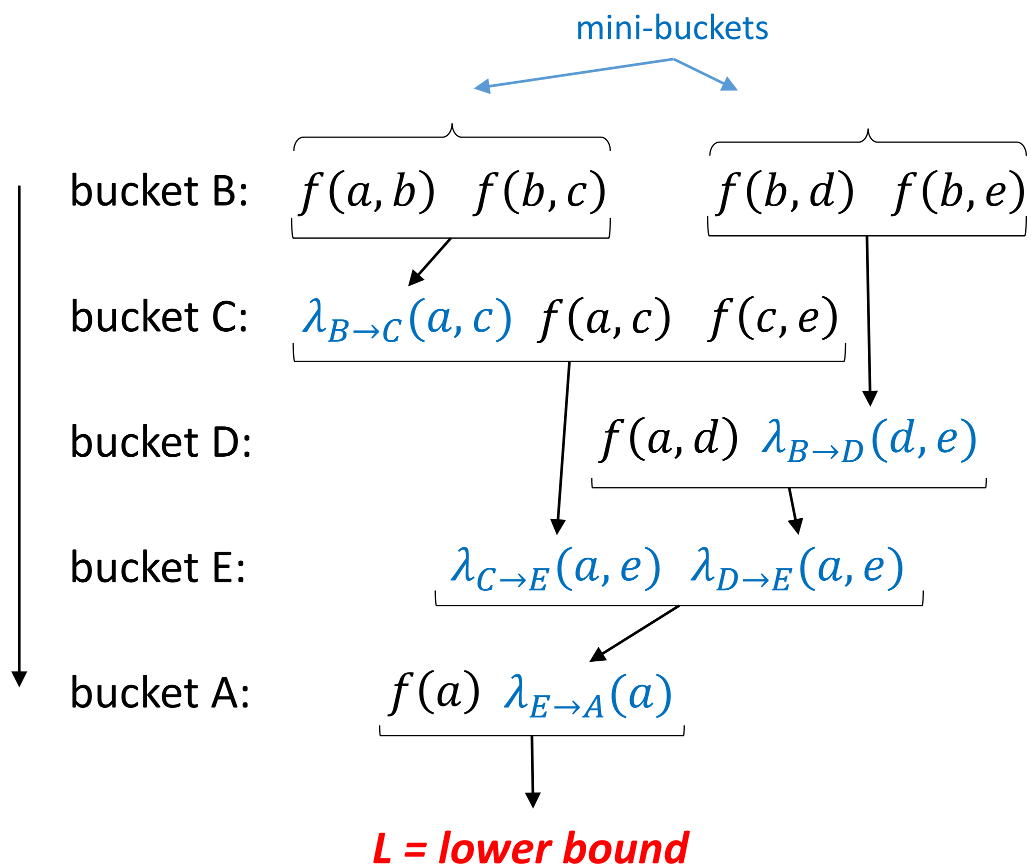
Mini-Bucket as Dual Decomposition

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

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

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

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



Mini-Bucket as Dual Decomposition

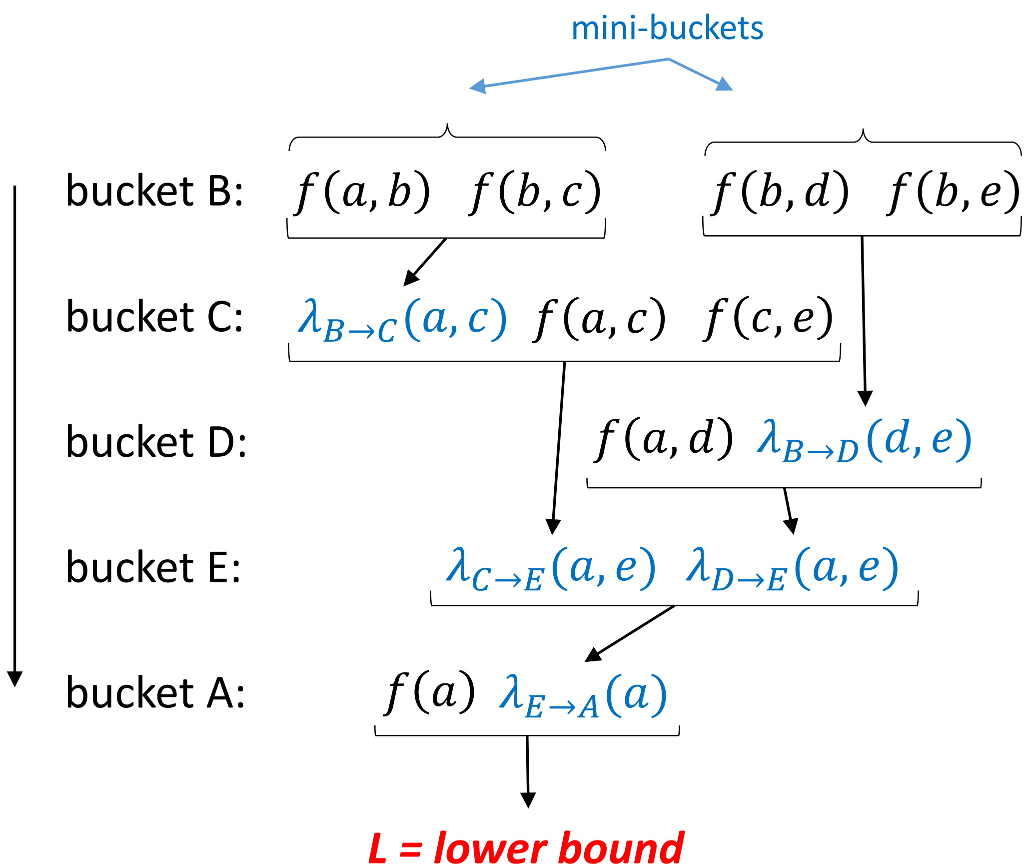
$$\min_{a,c,b}[f(a,b) + f(b,c) - \lambda_{B\to C}(a,c)] = 0$$

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

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

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

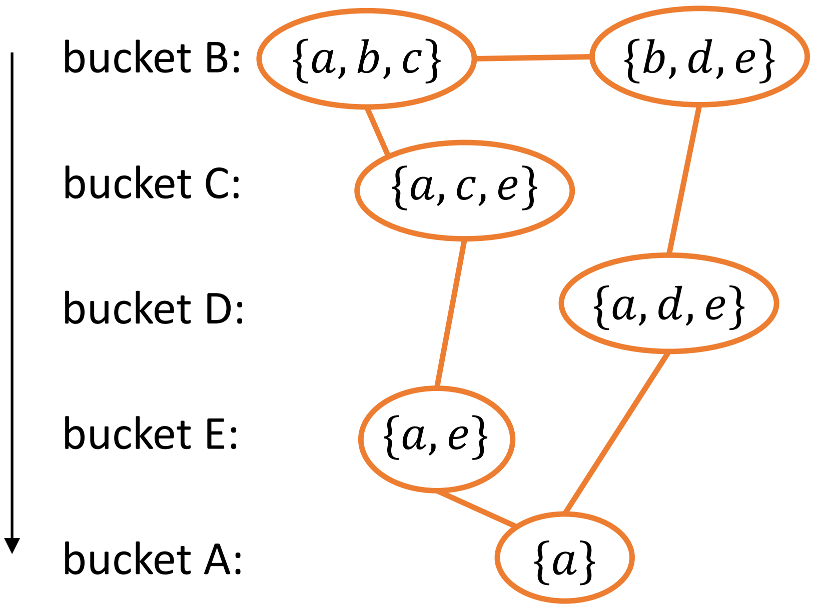
$$\min_{a,e}[\lambda_{C\to E}(a,e) + \lambda_{D\to E}(a,e) - \lambda_{E\to A}(a)] = 0$$



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:



L = lower bound

Anytime Approximation

$\textbf{anytime-mpe}(\varepsilon)$

Initialize : $i = i_0$

While time and space resources are available

$i \leftarrow i + i_{step}$

$U \leftarrow$ upper bound computed by *approx - mpe(i)*

$L \leftarrow$ lower bound computed by *approx - mpe(i)*

keep the best solution found so far

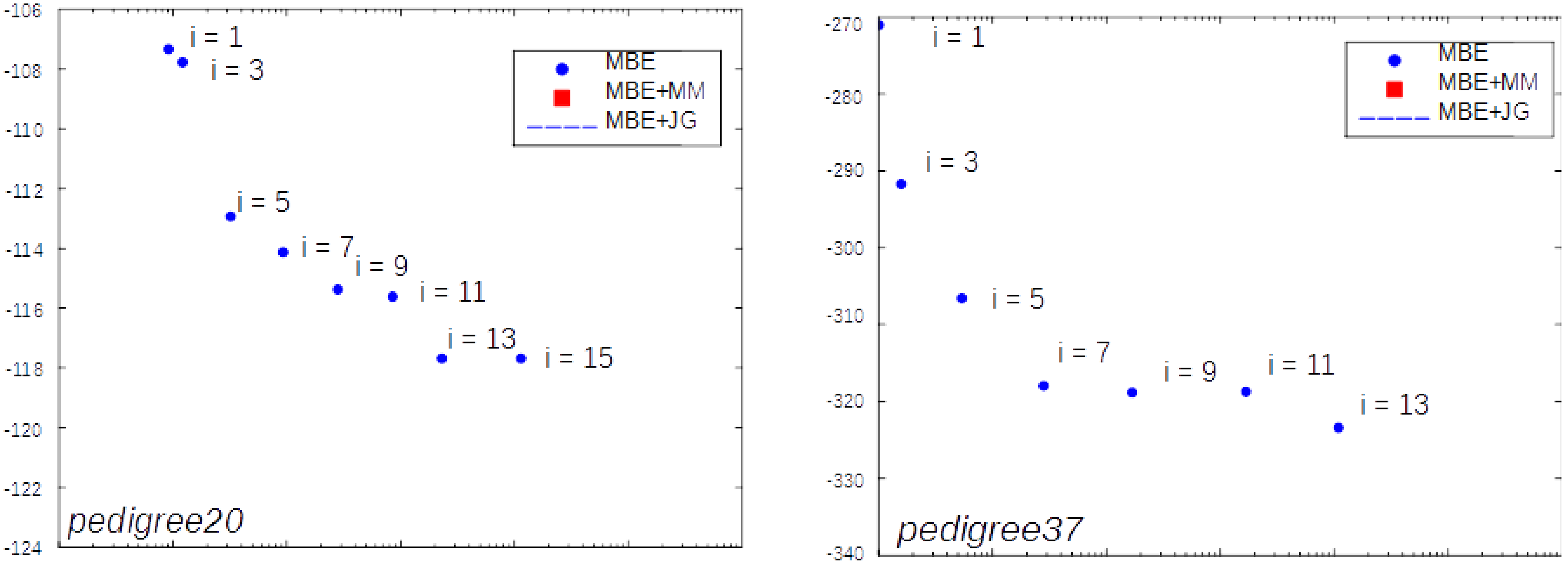
if $1 \le \frac{U}{L} \le 1 + \varepsilon$, return solution

end

return the largest L and the smallest U

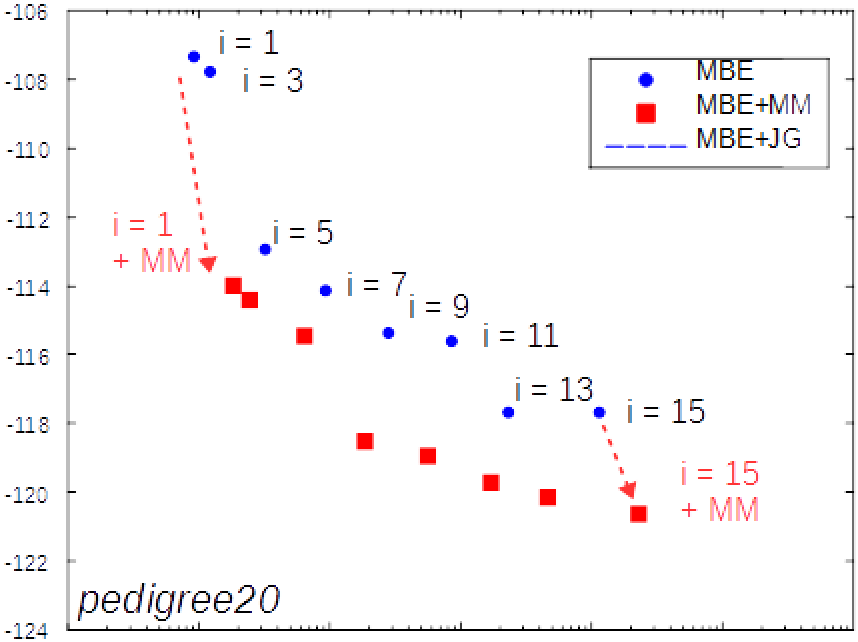
[Dechter and Rish, 2003]

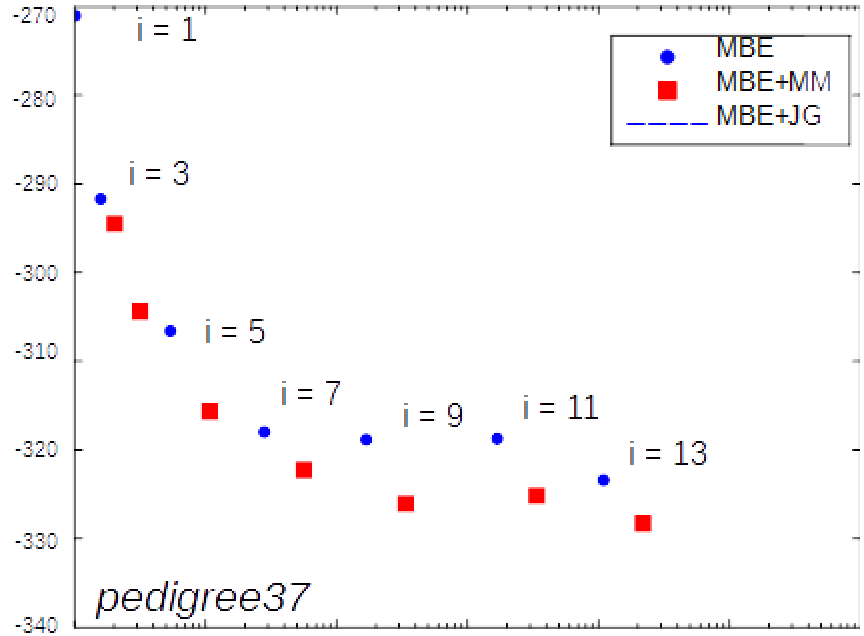
Anytime Approximation



- 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

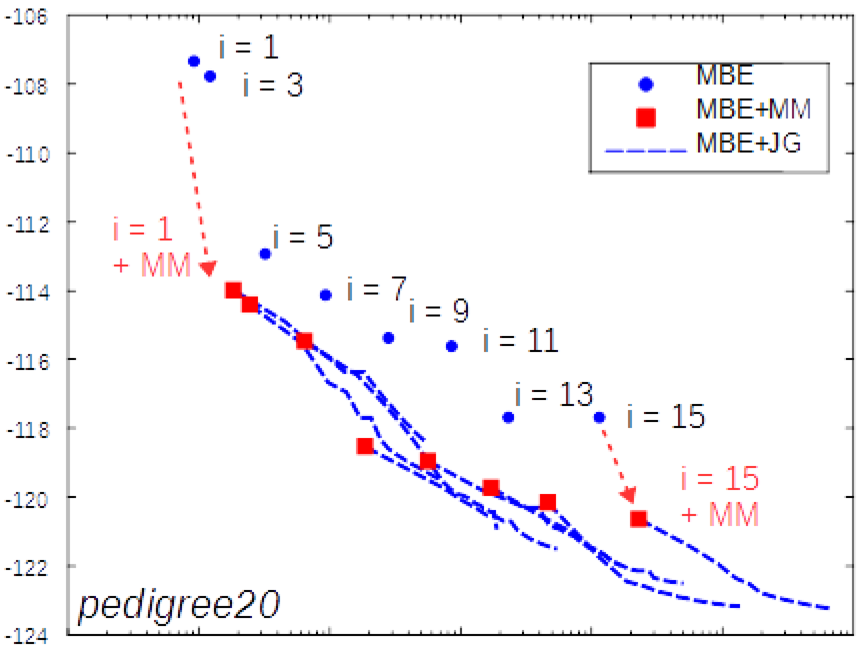
Anytime Approximation

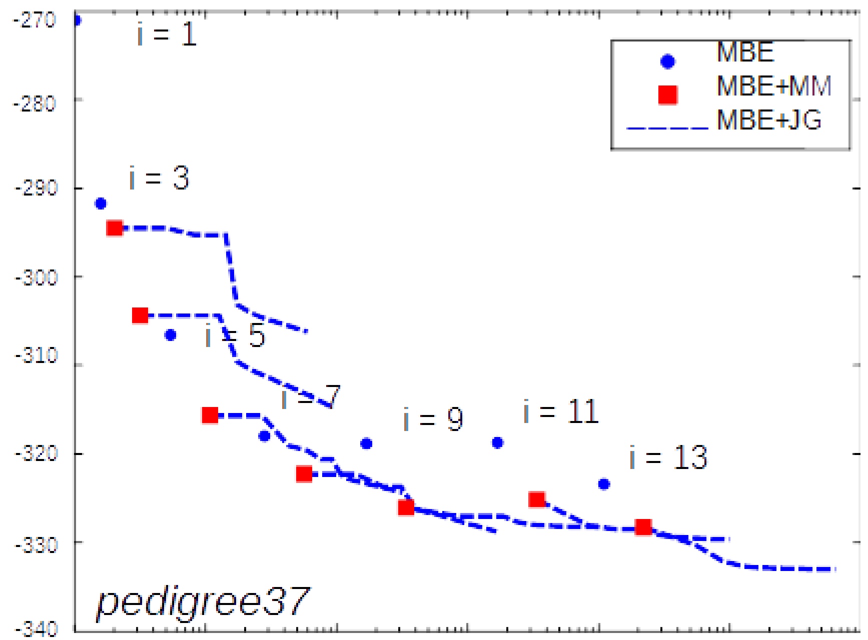




- Can tighten the bound in various ways
  - Cost-shifting (improve consistency between cliques)
  - Increase i-bound (higher order consistency)
- Simple moment-matching step improves bound significantly

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:

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

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

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

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

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: $\lambda_{C \to E}(a,e)\ \lambda_{D \to E}(a,e)$

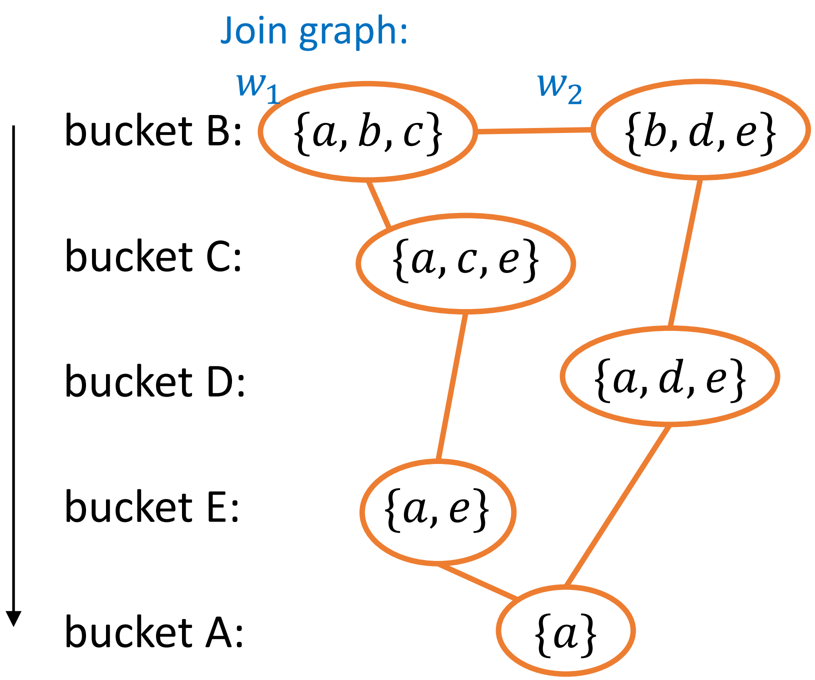
bucket A: $f(a)\ \lambda_{E \to A}(a)$

U = upper bound

[Liu and Ihler, 2011]

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



U = upper bound

[Liu and Ihler, 2011]

Outline

- Introduction
- Inference
- **Bounds and heuristics**
  - Basics of search: DFS versus BFS
  - Mini-bucket elimination
  - Weighted mini-buckets and iterative cost-shifting
  - Generating heuristics using mini-bucket elimination
- AND/OR search
- Exploiting parallelism
- Software

Generating Heuristics for Graphical Models

Given a cost function:

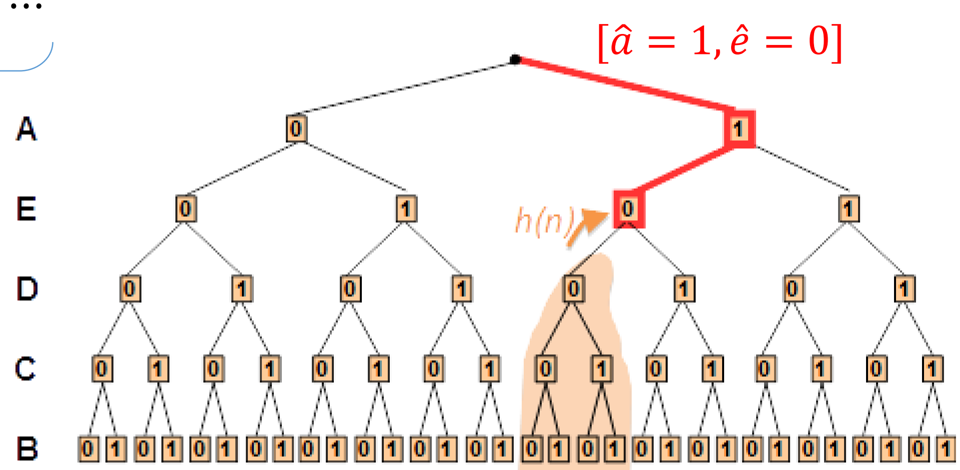
$$f(a, \ldots, e) = f(a) + f(a,b) + f(a,c) + f(a,d) + f(b,c) + f(b,d) + f(b,e) + f(c,e)$$

define an evaluation function over a partial assignment as the cost of its best extension:

$$f^*(\hat{a}, \hat{e}, D) = \min_{b,c} F(\hat{a}, b, c, D, \hat{e})$$

$$= f(\hat{a}) + \min_{b,c} f(\hat{a}, b) + f(\hat{a}, c) + \cdots$$

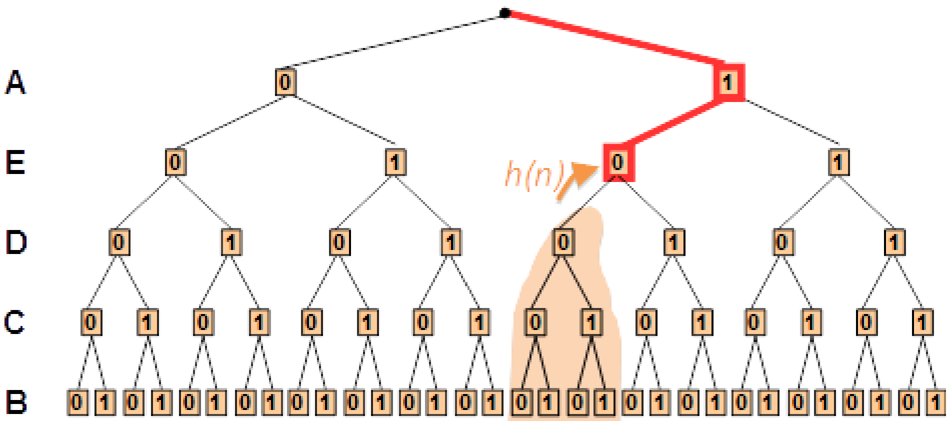
$$= g(\hat{a}, \hat{e}, D) + h^*(\hat{a}, \hat{e}, D)$$

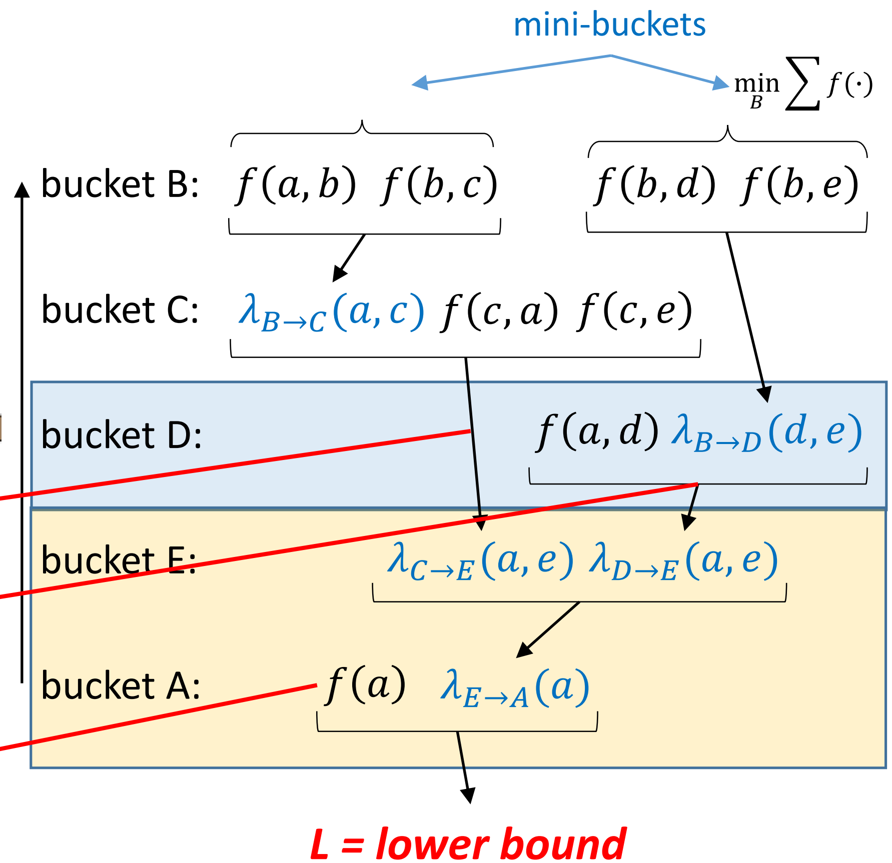


[Kask and Dechter, 2001]

Static Mini-Bucket Heuristics

Given a partial assignment, $[\hat{a} = 1, \hat{e} = 0]$
(weighted) mini-bucket gives an admissible heuristic:





cost to go:

$$\tilde{h}(\hat{a},\hat{e},D) = \lambda_{C\to E}(\hat{a},\hat{e}) + f(\hat{a},D) + \lambda_{B\to D}(D,\hat{e})$$

(admissible: $\tilde{h}(\hat{a},\hat{e},D) \leq h^*(\hat{a},\hat{e},D)$)

cost so far:

$$g(\hat{a},\hat{e},D) = f(A = \hat{a})$$

Properties of the Heuristics

- MB heuristic is monotone, admissible
- Computed in linear time (during search)
- **IMPORTANT**
  - Heuristic strength can vary by MB(i)
  - Higher i-bound → more pre-processing → more accurate heuristic → less search
- Allows controlled trade-off between pre-processing and search

Dynamic Mini-Bucket Heuristics

- Rather than pre-compile, compute the heuristics, dynamically, during search
- **Dynamic MB**: use the MB(i) algorithm to produce a bound for any node during search
- **Dynamic MBTE**: compute heuristics simultaneously for all un-instantiated variables using Mini-Bucket-Tree Elimination (MBTE)
  - MBTE is an approximation scheme defined over cluster trees; it outputs multiple bounds for each variable and value extension at once

[Marinescu, Kask and Dechter, 2003]