Advances in Search and Inference for Combinatorial Optimization

Rina Dechter

Radu Marinescu

Robert Mateescu

University of California, Irvine

Outline

- **Introduction**
  - Optimization tasks for graphical models
  - Planning as optimization
  - Solving optimization problems with inference and search
- **Inference**
  - Bucket Elimination, Dynamic Programming
  - Mini-Bucket Elimination
- **Search (OR)**
  - Branch-and-Bound and Best-First Search
  - Lower-bounding heuristics
- **AND/OR search spaces**
- **Hybrids of search and inference**
  - Cutset decomposition
  - Super-Bucket scheme
- **Software**

Outline

- **Introduction**
  - Optimization tasks for graphical models
  - Planning as optimization
  - Solving optimization problems with inference and search
- **Inference**
  - Bucket Elimination, Dynamic Programming
  - Mini-Bucket Elimination
- **Search (OR)**
  - Branch-and-Bound and Best-First Search
  - Lower-bounding heuristics
- **AND/OR search spaces**
- **Hybrids of search and inference**
  - Cutset decomposition
  - Super-Bucket scheme
- **Software**

Constraint Optimization Problems
for Graphical Models

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

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

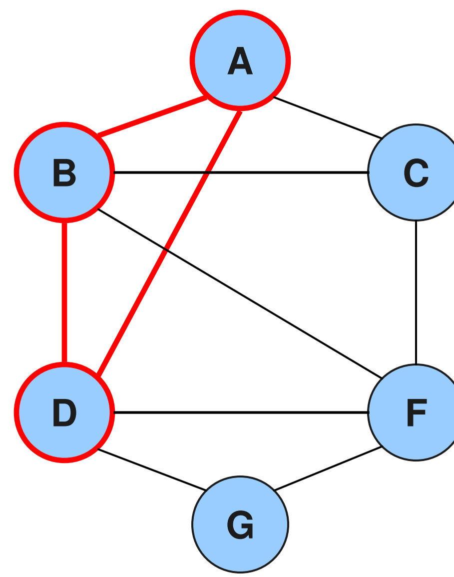
$D = \{D_1, ..., D_n\}$ - domains

$F = \{f_1, ..., f_m\}$ - 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	0
2	3	1	0
3	1	2	5
3	2	1	0

Primal graph =
Variables --> nodes
Functions, Constraints -→ arcs



F(a,b,c,d,f,g)= f1(a,b,d)+f2(d,f,g)+f3(b,c,f)

Global Cost Function

$$F(X) = \sum_{i=1}^{m} f_i(X)$$

Constraint Networks

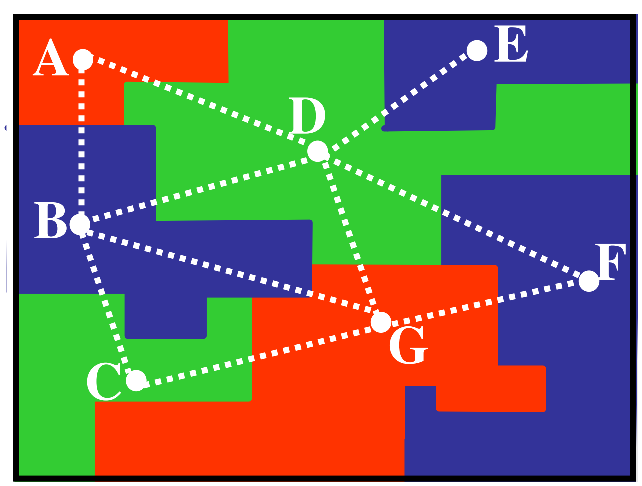
Map coloring

Variables: countries (A B C etc.)

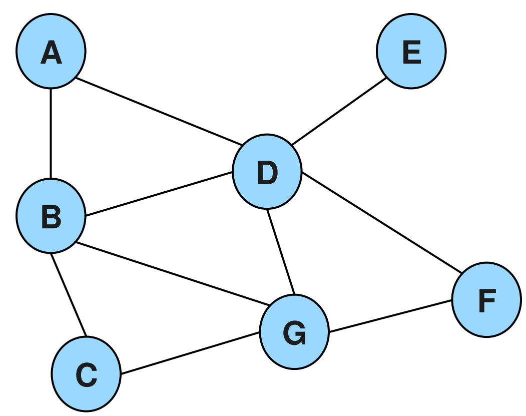
Values: colors (red green blue)

Constraints: **A ≠ B, A ≠ D, D ≠ E**, *etc.*

A	B
red	green
red	yellow
green	red
green	yellow
yellow	green
yellow	red

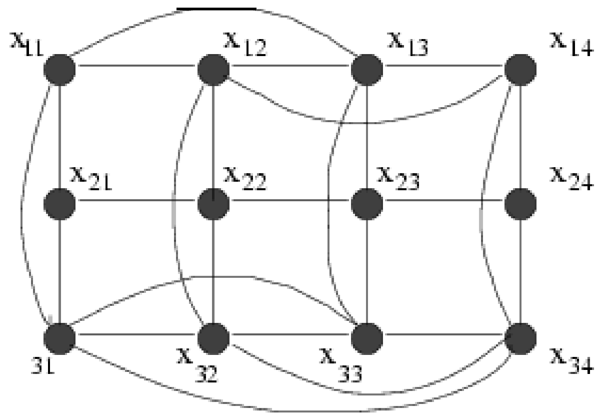


Constraint graph



Constrained Optimization

Example: power plant scheduling



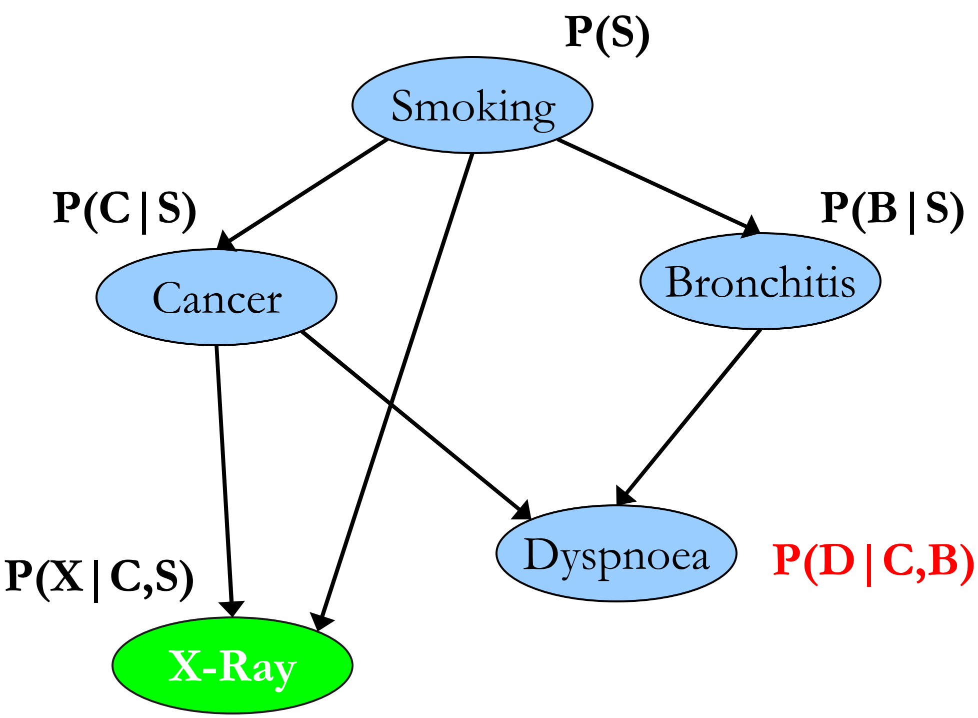
Unit #	Min Up Time	Min Down Time
1	3	2
2	2	1
3	4	1

Variables $= \{X_1,\ldots,X_n\}$, domain $= \{ON, OFF\}$.
Constraints : $X_1 \vee X_2, \neg X_3 \vee X_4$, min - up and min - down time,
power demand : $\sum \text{Power}(X_i) \geq Demand$

Objective : minimize $\text{TotalFuelCost}(X_1,\ldots,X_N)$

Probabilistic Networks

BN = (X,D,G,P)



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

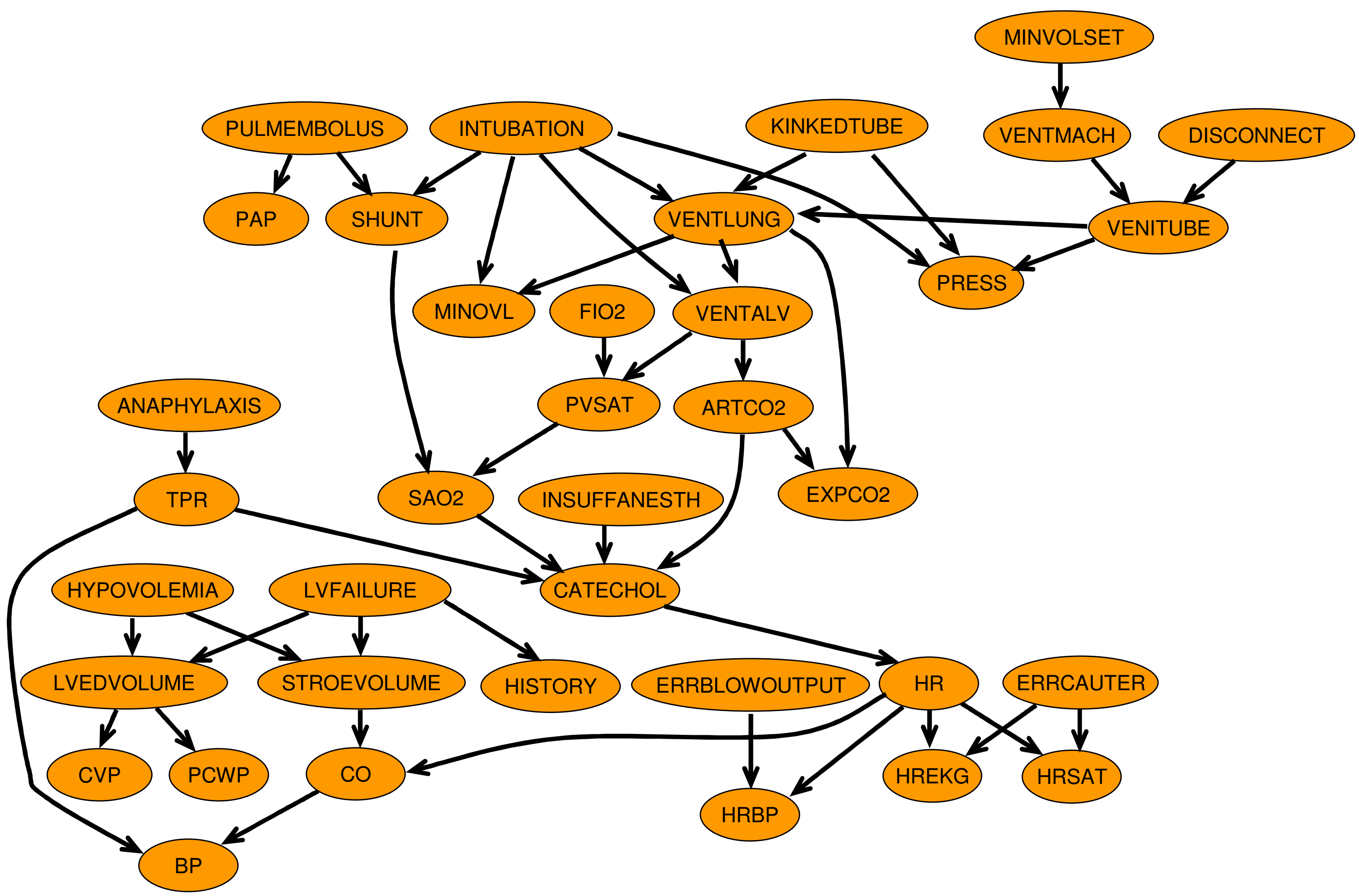
$P(S,C,B,X,D) = P(S) \cdot P(C|S) \cdot P(B|S) \cdot P(X|C,S) \cdot P(D|C,B)$

MPE= Find a maximum probability assignment, given evidence

MPE= find argmax $P(S) \cdot P(C|S) \cdot P(B|S) \cdot P(X|C,S) \cdot P(D|C,B)$

Monitoring Intensive-Care Patients

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

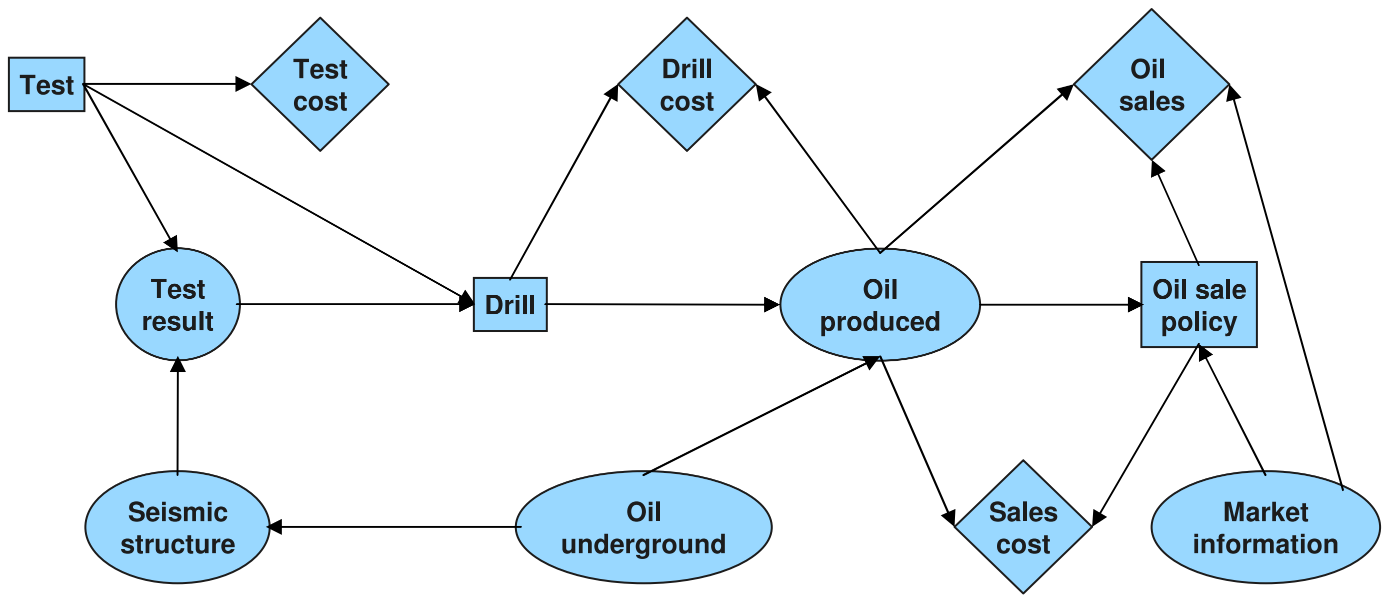


Influence Diagrams

Task: find optimal policy:

Influence diagram $ID = (X,D,P,R)$.

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



Chance variables: $X = X_1,\ldots, X_n$ **over domains.**

Decision variables: $D = D_1,\ldots, D_m$

CPT's for chance variables: $P_i = P(X_i \mid pa_i), i = 1..n$

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

Utility function: $u = \sum_i r_i$

Graphical Models

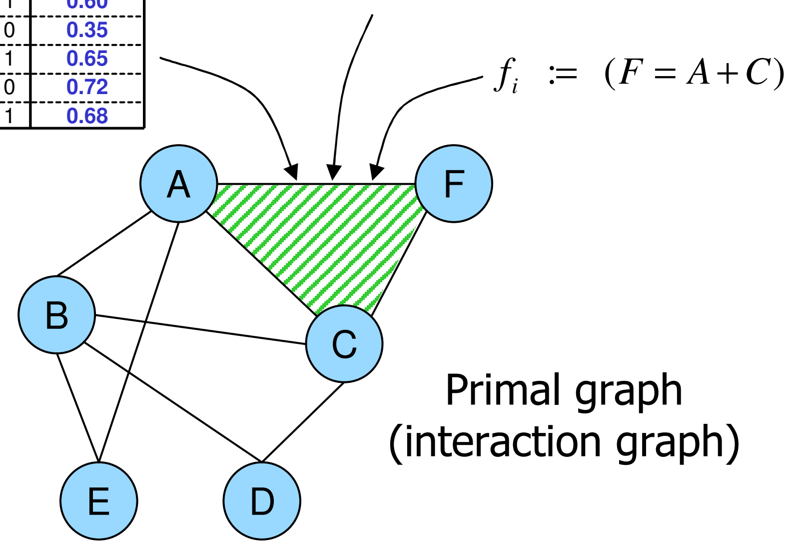
- A graphical model **(X,D,F)**:
  - **X** = $\{X_1,\ldots X_n\}$ variables
  - **D** = $\{D_1, \ldots D_n\}$ domains
  - **F** = $\{f_1,\ldots,f_r\}$ functions
    (constraints, CPTS, CNFs …)
- Operators:
  - combination
  - elimination (projection)
- Tasks:
  - **Belief updating**: $\Sigma_{X-y} \prod_j P_i$
  - **MPE**: $\max_X \prod_j P_j$
  - **CSP**: $\prod_X \times_j C_j$
  - **Max-CSP**: $\min_X \Sigma_j F_j$

Conditional Probability Table (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



Primal graph
(interaction graph)

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

Sample Domains for GM

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

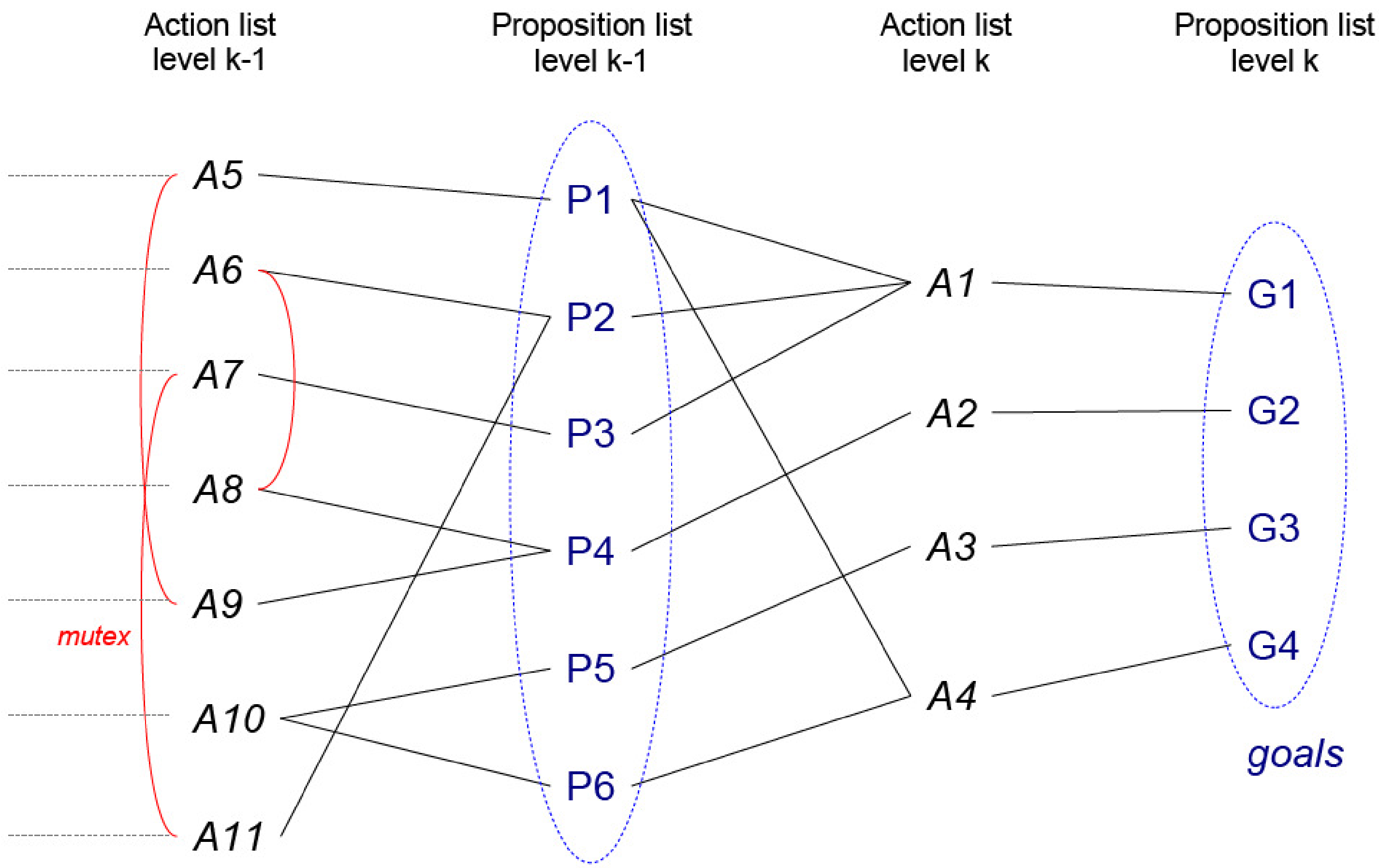
…

Outline

- **Introduction**
  - Optimization tasks for graphical models
  - Planning as optimization
  - Solving optimization problems with inference and search
- **Inference**
  - Bucket Elimination, Dynamic Programming
  - Mini-Bucket Elimination
- **Search (OR)**
  - Branch-and-Bound and Best-First
  - Lower-bounding heuristics
- **AND/OR search spaces**
- **Hybrids of search and inference**
  - Cutset decomposition
  - Super-Bucket scheme
- **Software**

Planning as Graphical Models

- MDPs and POMDPs
- Using Graph plans to convert to Weighted CSPs



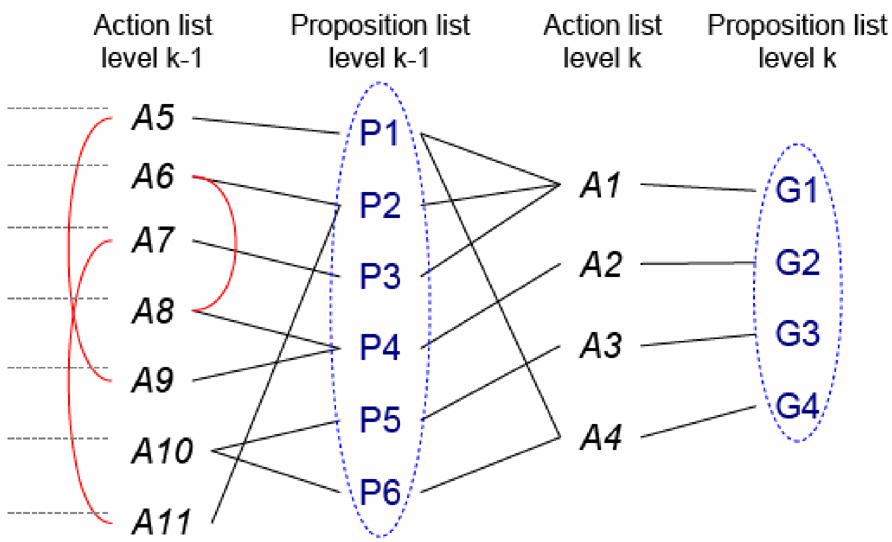
DCSP Translation Example

Variables: $G_1, \ldots, G_4, P_1, \ldots, P_6, \ldots, [\ldots]$
Goals: $G_1, \ldots, G_4$

Domains:
$G_1: \{A_1\}, G_2: \{A_2\}, G_3: \{A_3\}, G_4: \{A_4\}, P_1: \{A_5\}, P_2: \{A_6, A_{11}\}, P_3: \{A_7\},$
$P_4: \{A_8, A_9\}, P_5: \{A_{10}\}, P_6: \{A_{10}\}$

Mutex constraints:
$P_1 \neq A_5 \vee P_4 \neq A_9,$
$P_2 \neq A_6 \vee P_4 \neq A_8,$
$P_2 \neq A_{11} \vee P_3 \neq A_7$

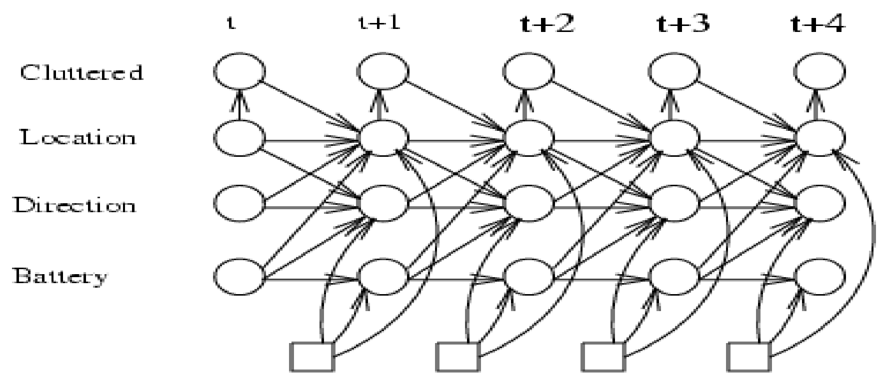
Activation constraints:
$G_1 = A_1 \Rightarrow Active\{P_1, P_2, P_3\},$
$G_2 = A_2 \Rightarrow Active\{P_4\},$
$G_3 = A_3 \Rightarrow Active\{P_5\},$
$G_4 = A_4 \Rightarrow Active\{P_1, P_6\}$

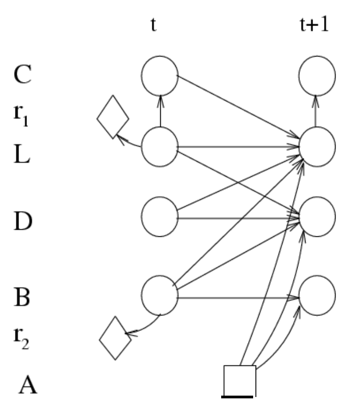


Initial state:
$Active\{G_1, G_2, G_3, G_4\}$

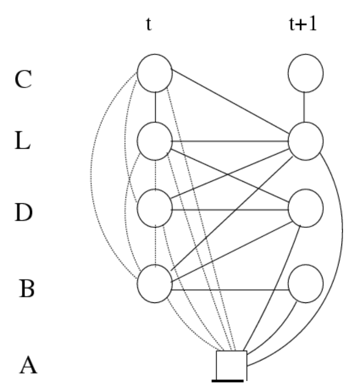
Dynamic Belief Networks (DBN)

MDPs and POMDPs





Two-stage influence diagram

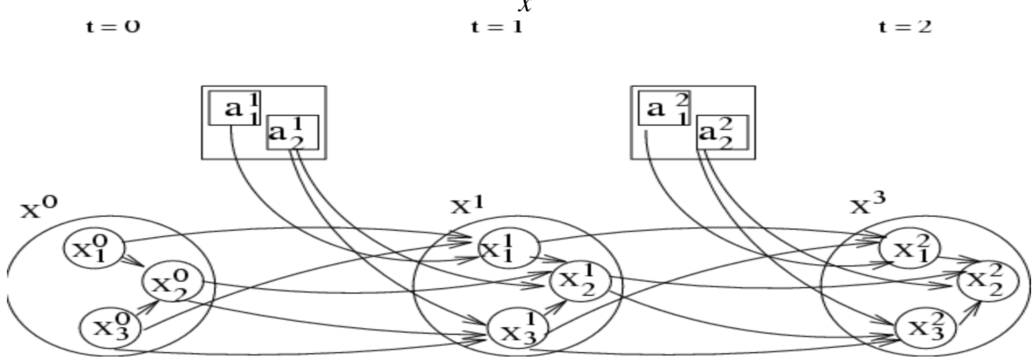


Interaction graph

Dynamic Programming: Elimination

Optimality Equation :

$$V(x^t) = \max_{a^t}\{r(x^t, a^t) + \sum_{x^{t+1}} P(x^{t+1} \mid x^t, a^t) V^{t+1}\}, \quad V^N = r^N(x^N)$$



Complexity of dynamic programming :

$$O(N \mid \Omega_A \mid\mid \Omega_X \mid^2) = O(N \mid D_A \mid^m \mid D_X \mid^{2n})$$

Decomposable utilities and probabilities :

$$r(x^t, a^t) = \sum_{i=1}^{n} r_i(x_i^t, a_i^t), \quad P(x^t \mid x^{t-1}, a^{t-1}) = \prod_{i=1}^{n} P(x_i^t \mid pa(x_i^t))$$

Types of Constraint Optimization

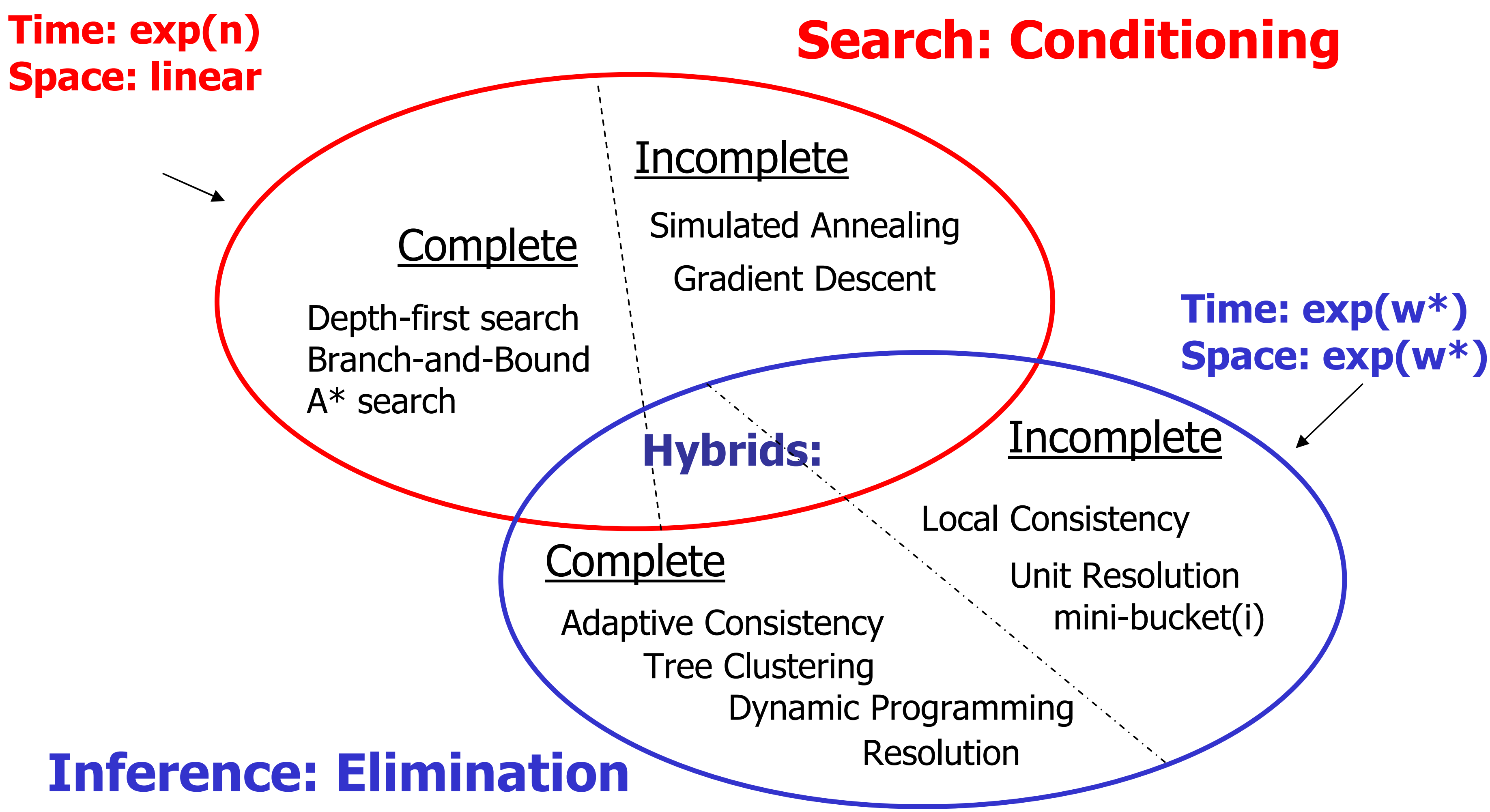
- Valued CSPs, Weighted CSPs, Max-CSPs, Max-SAT
- Most Probable Explanation (MPE)
- Linear Integer Programs

- **Examples**:
  - Problems translated from planning
  - Unit scheduling maintenance
  - Combinatorial auctions
  - Maximum-likelihood haplotypes in linkage

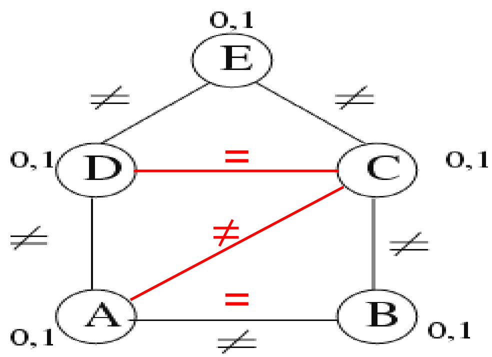
Outline

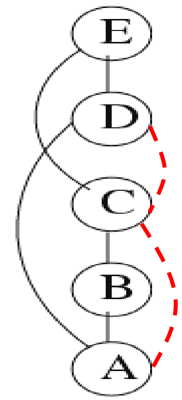
- **Introduction**
  - Optimization tasks for graphical models
  - Planning as optimization
  - Solving optimization problems with inference and search
- **Inference**
  - Bucket Elimination, Dynamic Programming
  - Mini-Bucket Elimination
- **Search (OR)**
  - Branch-and-Bound and Best-First
  - Lower-bounding heuristics
- **AND/OR search spaces**
- **Hybrids of search and inference**
  - Cutset decomposition
  - Super-Bucket scheme
- **Software**

Graphical Models Reasoning



Bucket Elimination
(Variable Elimination)





Bucket E: $E \neq D,\ E \neq C$

Bucket D: $D \neq A$ → **$D = C$**

Bucket C: $C \neq B$ → **$A \neq C$**

Bucket B: $B \neq A$ → **$B = A$**

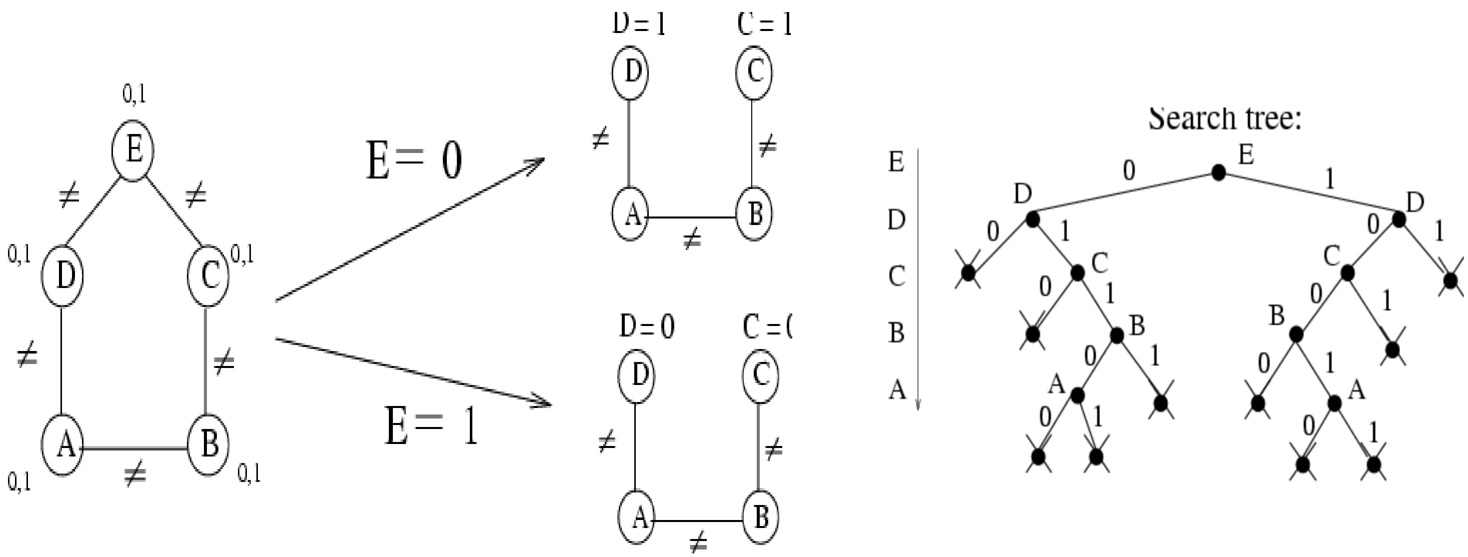
Bucket A: → **contradiction**

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

w^* - ***induced width, tree - width***

trees are easy : $w^* = 1$

The Idea of Conditioning



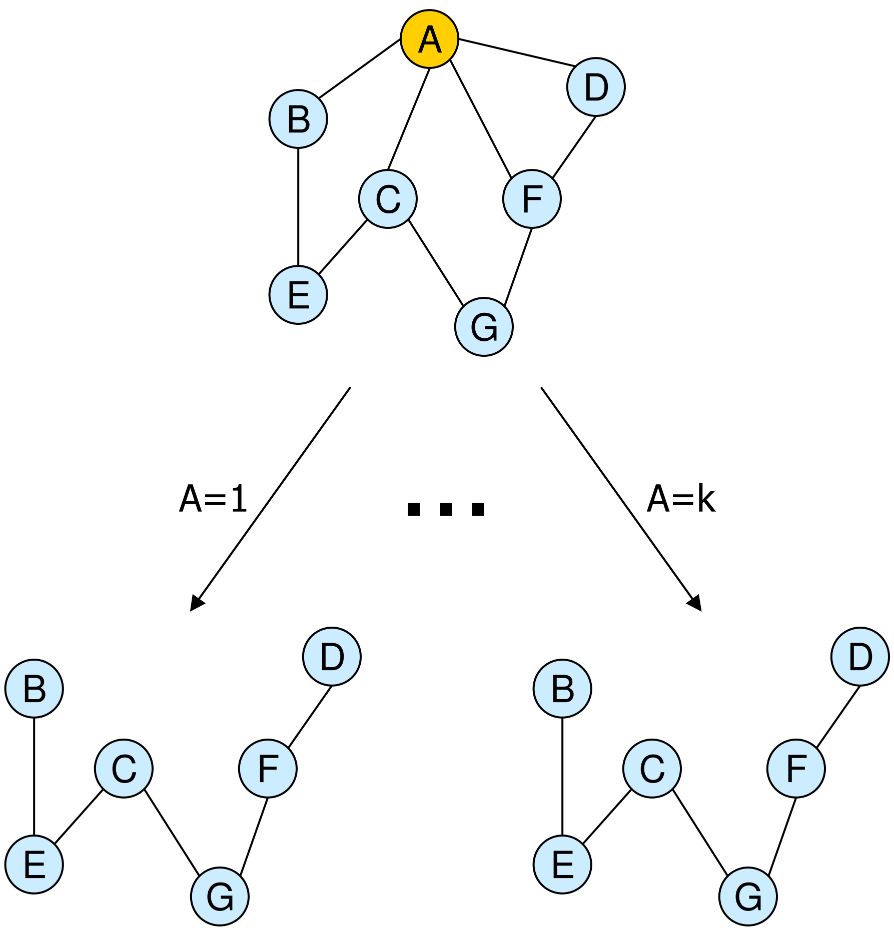
Complexity : ***exponential time, linear space***

Refined complexity : a) exponential in cycle - cutset size

b) in depth of dfs tree

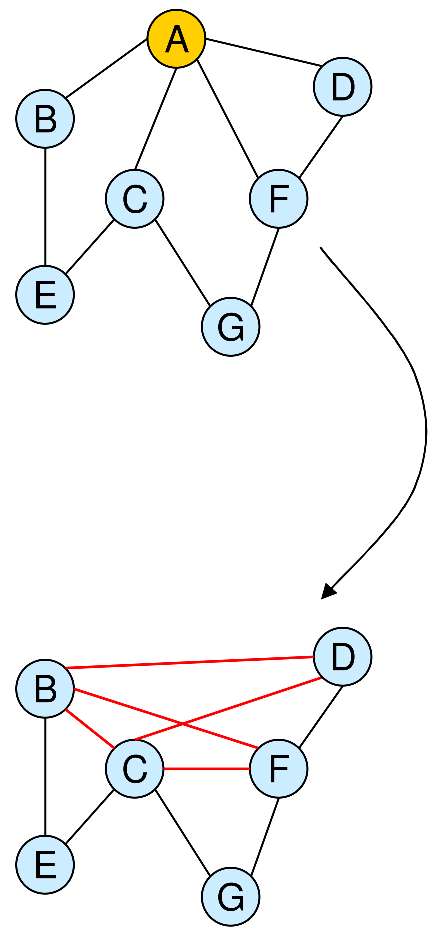
Conditioning vs. Elimination

Conditioning (search)



k "sparser" problems

Elimination (inference)

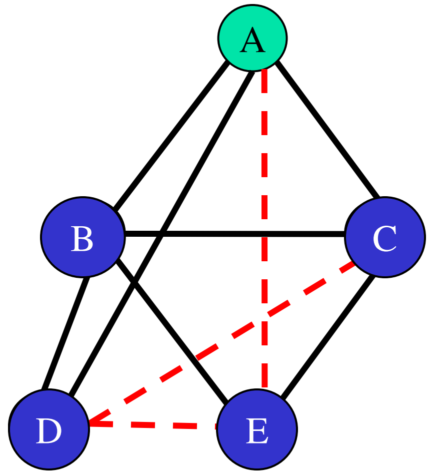


1 "denser" problem

Outline

- **Introduction**
  - Optimization tasks for graphical models
  - Solving optimization problems by inference and search
- **Inference**
  - **Bucket Elimination, Dynamic Programming**
  - Mini-Bucket Elimination
- **Search (OR)**
  - Branch-and-Bound and Best-First search
  - Lower-bounding heuristics
- **AND/OR search spaces**
- **Hybrids of search and inference**
  - Cutset decomposition
  - Super-Bucket scheme
- **Software**

Computing the Optimal Cost Solution



Constraint graph

$$\textbf{OPT} = \min_{e=0,d,c,b} F(a,b)+F(a,c)+F(a,d)+F(b,c)+F(b,d)+F(b,e)+F(c,e)$$

$$\min_{e=0}\min_{d} F(a,d) + \min_{c} F(a,c)+F(c,e) + \min_{b} \underbrace{F(a,b)+F(b,c)+F(b,d)+F(b,e)}_{h^B(a,d,c,e)}$$

Variable Elimination

$h^B(a,d,c,e)$

Finding $OPT = \min_{X_1,\dots,X_n} \sum_{j=1}^{r} f_j(X)$

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

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

$\min_b \sum$ ← Elimination operator

bucket B: F(a,b) F(b,c) F(b,d) F(b,e)

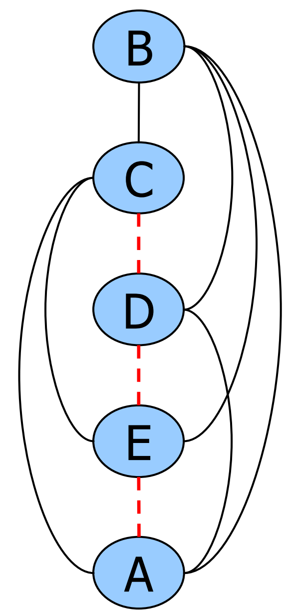
bucket C: F(c,a) F(c,e) ***h^B (a, d, c, e)***

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

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

bucket A: ***h^E (a)***

OPT



Generating the Optimal Assignment

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: F(a,b) F(b,c) F(b,d) F(b,e)

C: F(c,a) F(c,e) $h^B(a, d, c, e)$

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

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

A: $h^E(a)$

Return (a', b', c', d', e')

Complexity

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

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

$\min_b \sum$ ← Elimination operator

bucket B: F(a,b) F(b,c) F(b,d) F(b,e)

bucket C: F(c,a) F(c,e) $h^B(a, d, c, e)$

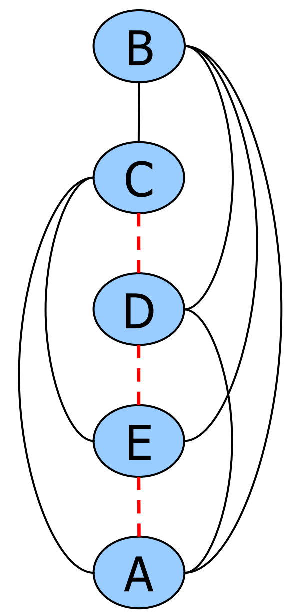
bucket D: F(a,d) $h^C(a, d, e)$

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

bucket A: $h^E(a)$

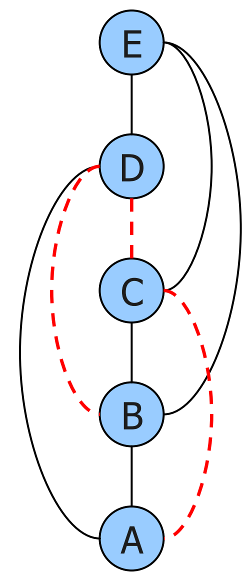
OPT

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



Induced-width

- Width along ordering *d*, w(d):
  - max # of previous neighbors (parents)
- Induced width along ordering d, w*(d):
  - The width in the ordered induced graph, obtained by connecting "parents" of each node X, recursively from top to bottom



Complexity of Bucket Elimination

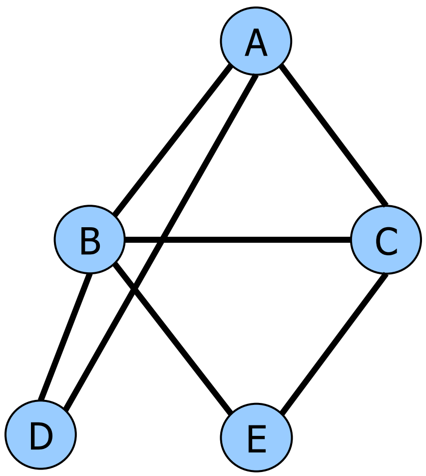
Bucket-Elimination is **time** and **space**

$$O(r \exp(w^*(d)))$$

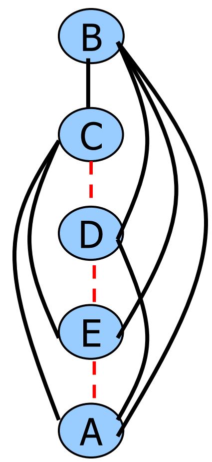
$w^*(d)$ – the induced width of the primal graph along ordering d

r = number of functions

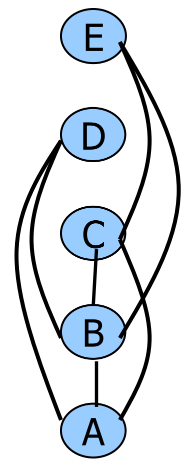
The effect of the ordering:



constraint graph



$w^*(d_1) = 4$



$w^*(d_2) = 2$

Finding smallest induced-width is hard!

DPOP

(Petcu & Faltings, 2005)

X_1

	d	e	f
g	2/6	3/7	0/4
h	3/8	1/6	8
i	9	8	1/7

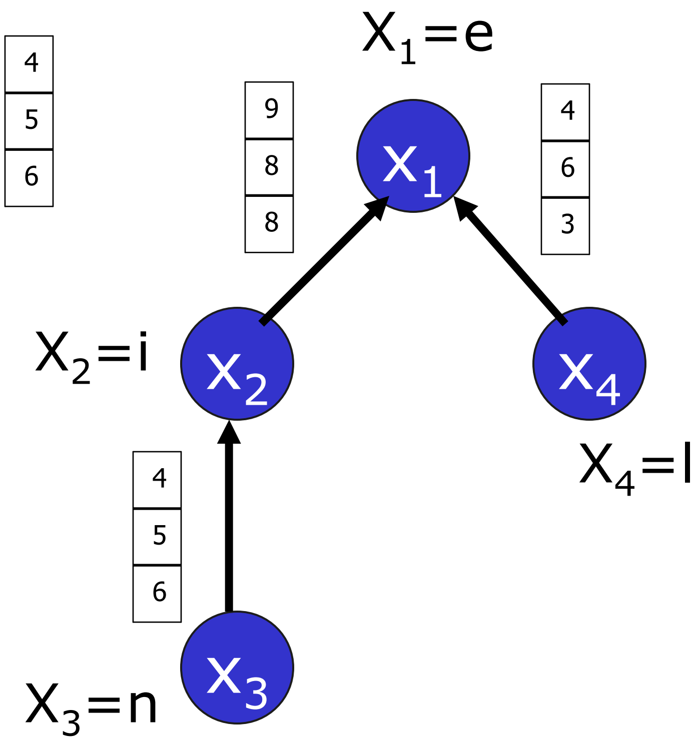
X_2 +

4
5
6

X_2

	g	h	i
m	4	2	2
n	3	5	6
o	1	3	2

X_3



X_1

	d	e	f
j	3	4	2
k	4	2	3
l	2	6	2

X_4

4	+	9	=	13	d
6		8		14	e
3		8		11	f

Outline

- Introduction
  - Optimization tasks for graphical models
  - Solving optimization problems by inference and search
- **Inference**
  - Bucket Elimination, Dynamic Programming
  - **Mini-Bucket Elimination**
- Search
  - Branch-and-Bound and Best-First search
  - Lower-bounding heuristics
- AND/OR search spaces
- Hybrids of search and inference
  - Cutset decomposition
  - Super-Bucket scheme
- Software

Mini-Bucket Approximation

Split a bucket into mini-buckets => bound complexity

bucket (X) =

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

$$h^X = \min_X \sum_{i=1}^{n} h_i$$

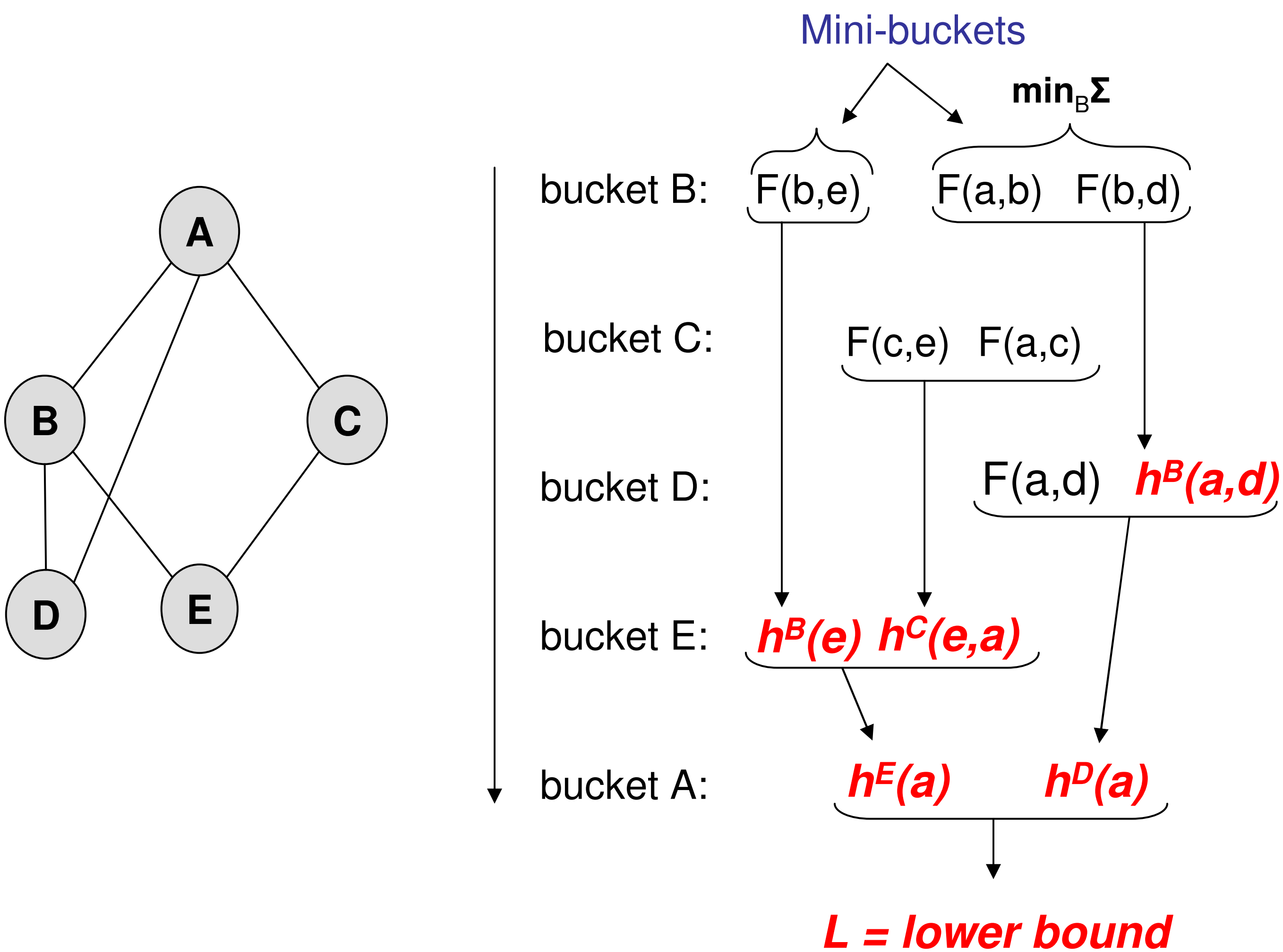
$\{ h_1, \ldots, h_r \}$ **$\{ h_{r+1}, \ldots, h_n \}$**

$$g^X = \left(\min_X \sum_{i=1}^{r} h_i \right) + \left(\min_X \sum_{i=r+1}^{n} h_i \right)$$

$$g^X \leq h^X$$

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

Mini-Bucket Elimination

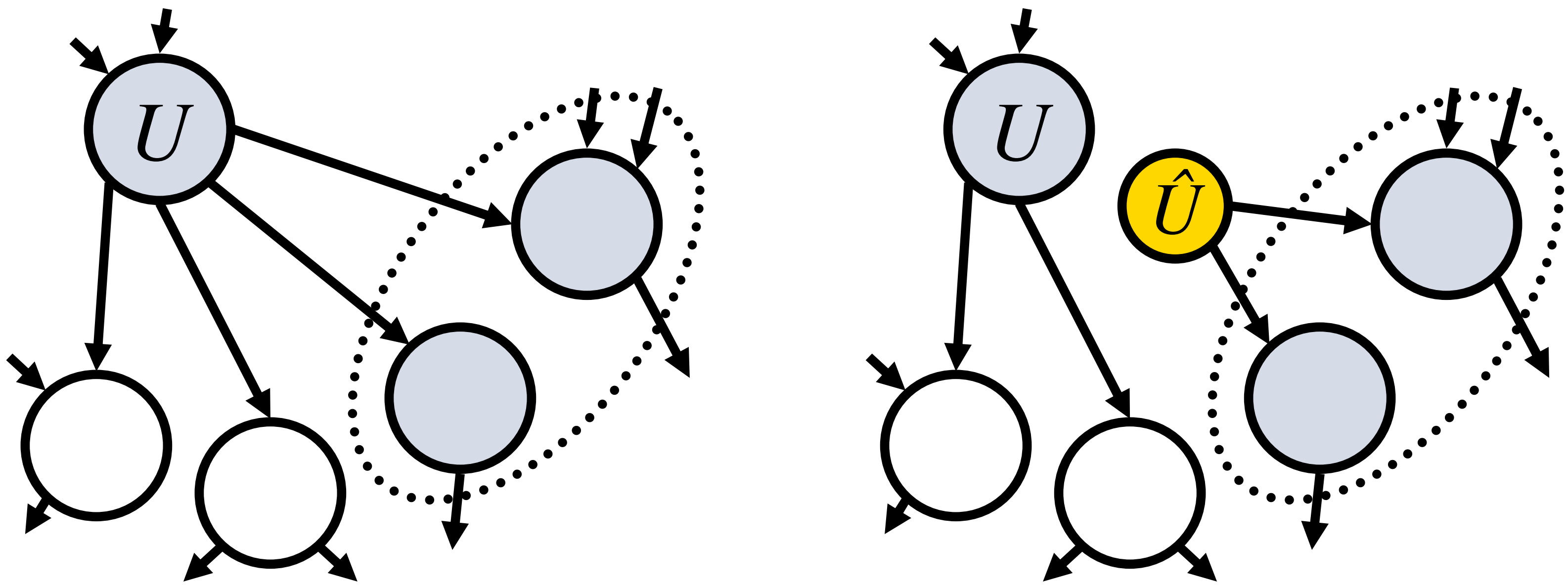


Semantics of Mini-Bucket: Splitting a Node

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

Before Splitting:
Network *N*

After Splitting:
Network *N'*

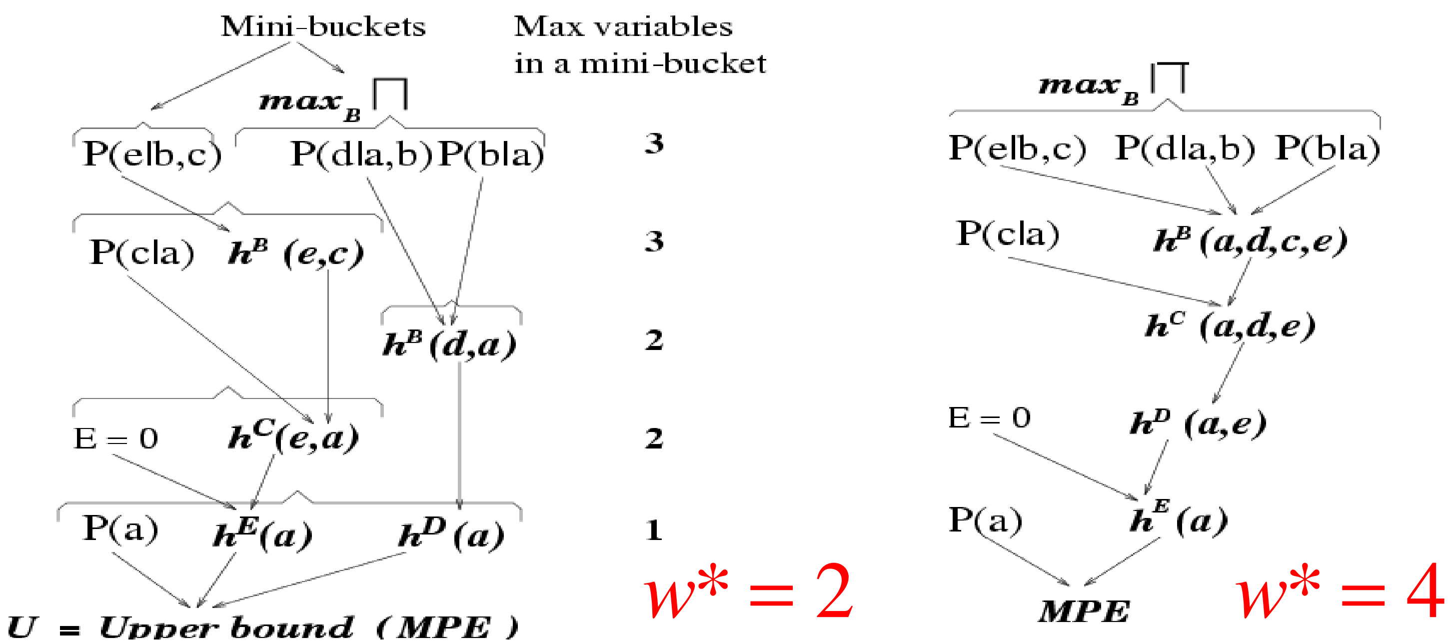


MBE-MPE(i)

Algorithm **Approx-MPE** (Dechter & Rish, 1997)

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

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



Properties of MBE(i)

- **Complexity:** *O(r exp(i))* time and *O(exp(i))* space
- Yields an upper-bound and a lower-bound
- **Accuracy:** determined by upper/lower (U/L) bound
- As ***i*** increases, both accuracy and complexity increase
- Possible use of mini-bucket approximations:
  - As anytime algorithms
  - As heuristics in search
- Other tasks: similar mini-bucket approximations for:
  - Belief updating, MAP and MEU (Dechter & Rish, 1997)

Anytime Approximation

anytime - mpe(ε **)**

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

end

return the largest L and the smallest U

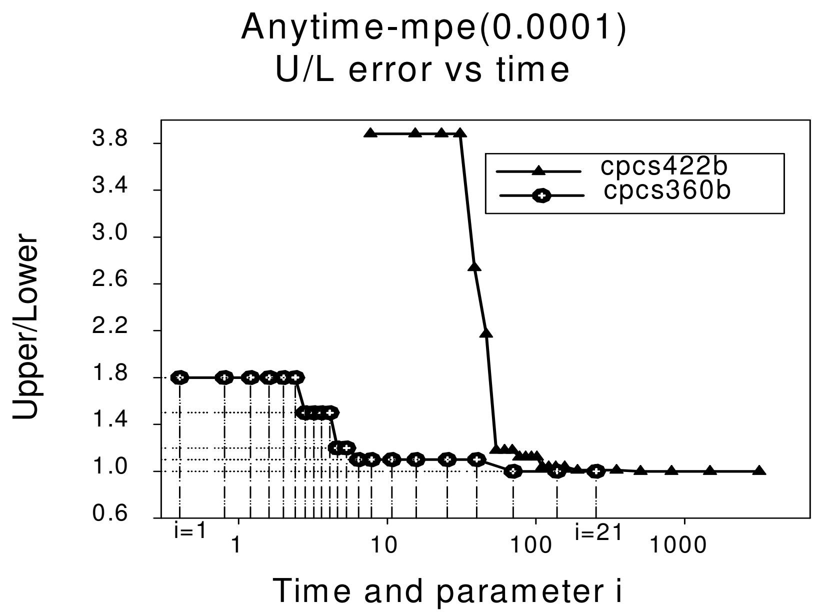
Empirical Evaluation
(Rish & Dechter, 1999)

- **Benchmarks**
  - Randomly generated networks
  - CPCS networks
  - Probabilistic decoding

- **Task**
  - Comparing **approx-mpe** and **anytime-mpe** versus bucket-elimination (**elim-mpe**)

CPCS networks – medical diagnosis (noisy-OR model)

Test case: no evidence



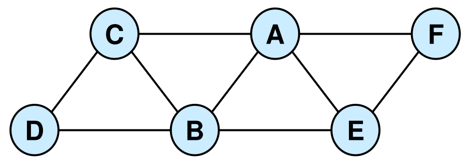
Time (sec)

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

Outline

- Introduction
  - Optimization tasks for graphical models
  - Solving optimization problems by inference and search
- Inference
  - Bucket Elimination, Dynamic Programming
  - Mini-Bucket Elimination
- **Search (OR)**
  - **Branch-and-Bound and Best-First search**
  - **Lower-bounding heuristics**
- AND/OR search spaces
- Hybrids of search and inference
  - Cutset decomposition
  - Super-bucket scheme
- Software

The Search Space



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**

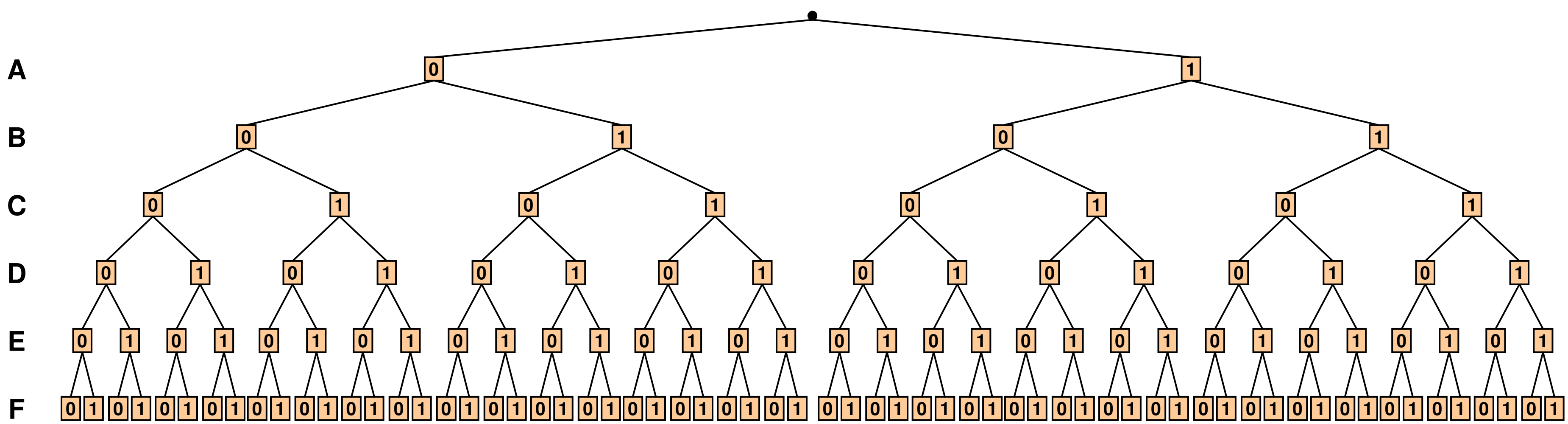
B	D	f_6
0	0	**4**
0	1	**2**
1	0	**1**
1	1	**0**

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

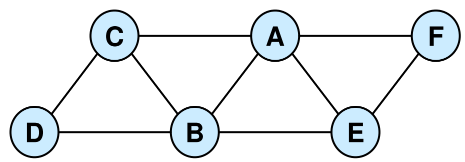
C	D	f_8
0	0	**1**
0	1	**4**
1	0	**0**
1	1	**0**

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

Objective function: $$f(X) = \min_X \sum_{i=1}^{9} f_i(X)$$



The Search Space



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

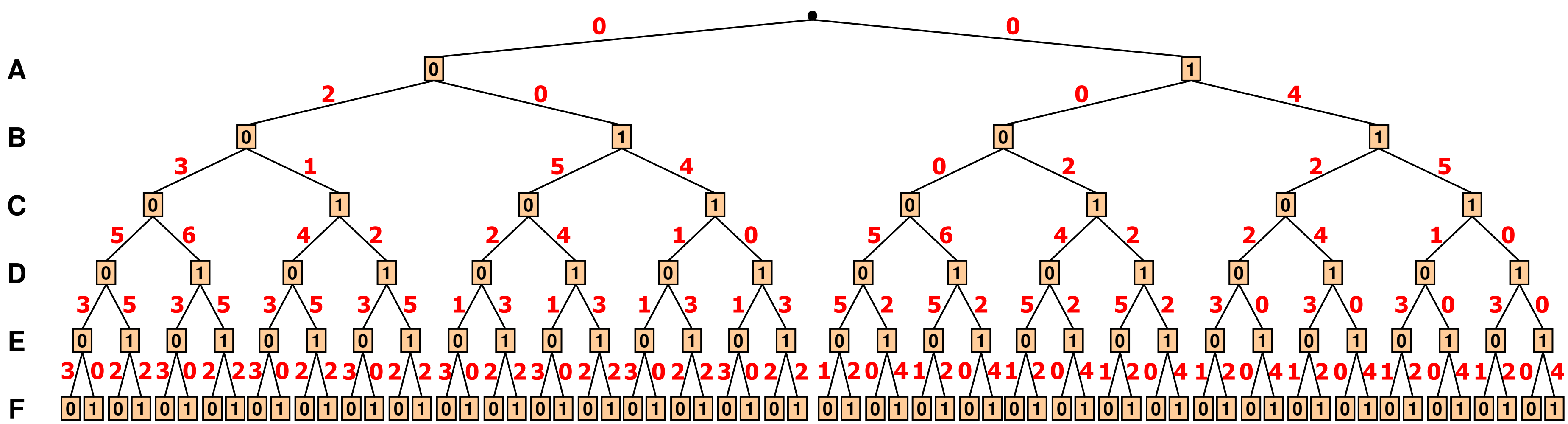
B	D	f_6
0	0	4
0	1	2
1	0	1
1	1	0

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

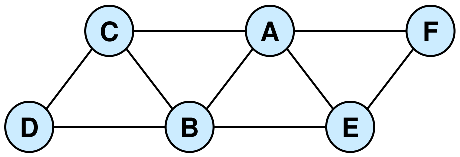
E	F	f_9
0	0	1
0	1	0
1	0	0
1	1	2

$$f(X) = \min_X \sum_{i=1}^{9} f_i(X)$$



Arc-cost is calculated based on cost components.

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**

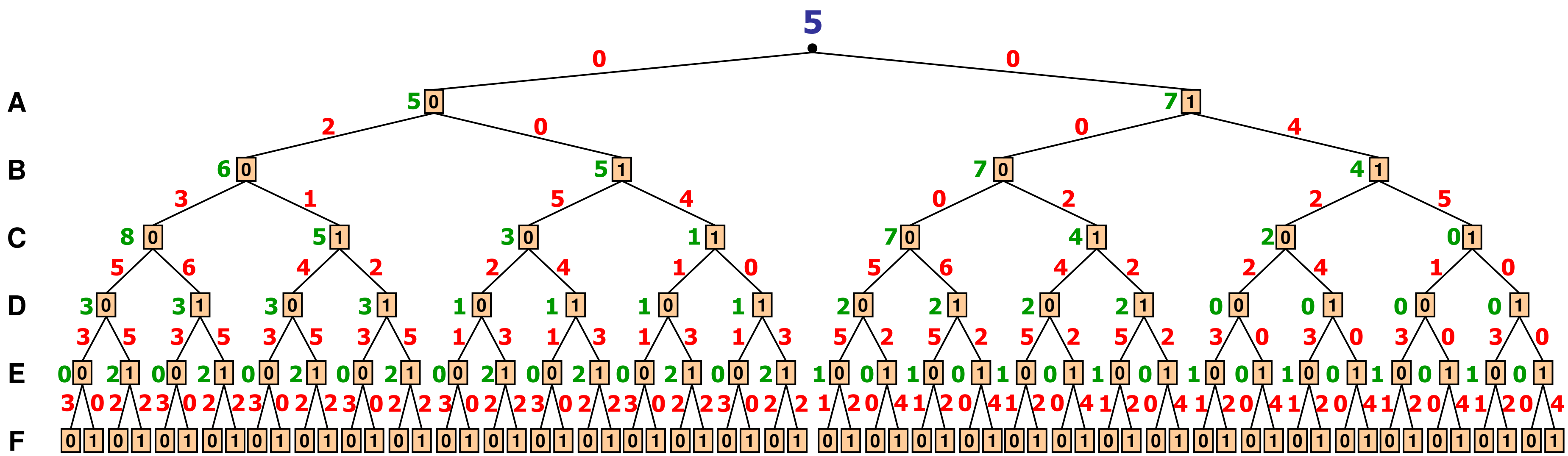
B	D	f_6
0	0	**4**
0	1	**2**
1	0	**1**
1	1	**0**

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**

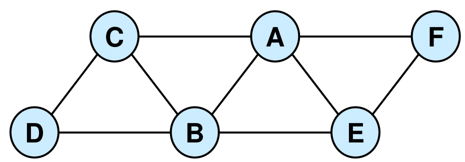
E	F	f_9
0	0	**1**
0	1	**0**
1	0	**0**
1	1	**2**

$$f(X) = \min_X \sum_{i=1}^{9} f_i(X)$$



Value of node = minimal cost solution below it

An 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**

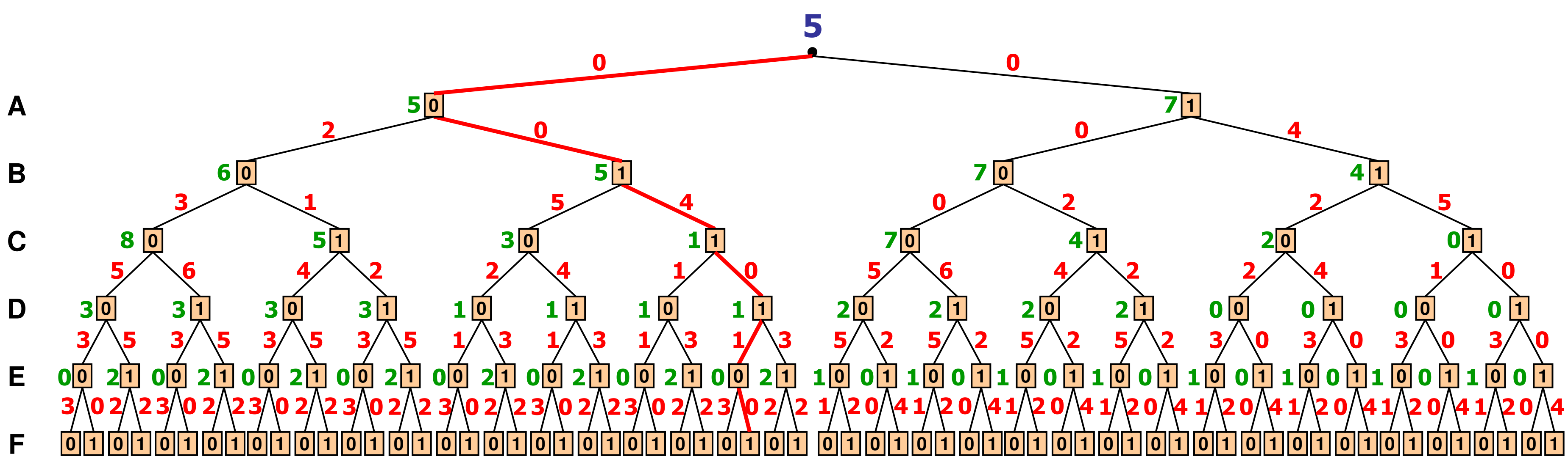
B	D	f_6
0	0	**4**
0	1	**2**
1	0	**1**
1	1	**0**

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**

$$f(\mathrm{X}) = \min_X \sum_{i=1}^{9} f_i(\mathrm{X})$$

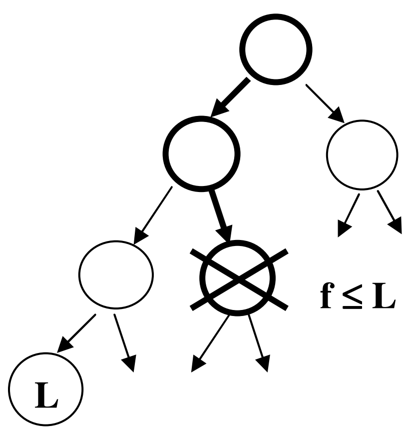


Value of node = minimal cost solution below it

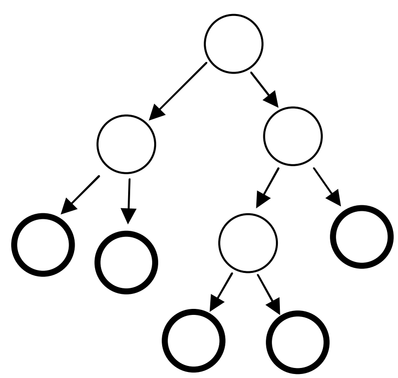
Basic Heuristic Search Schemes

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

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



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

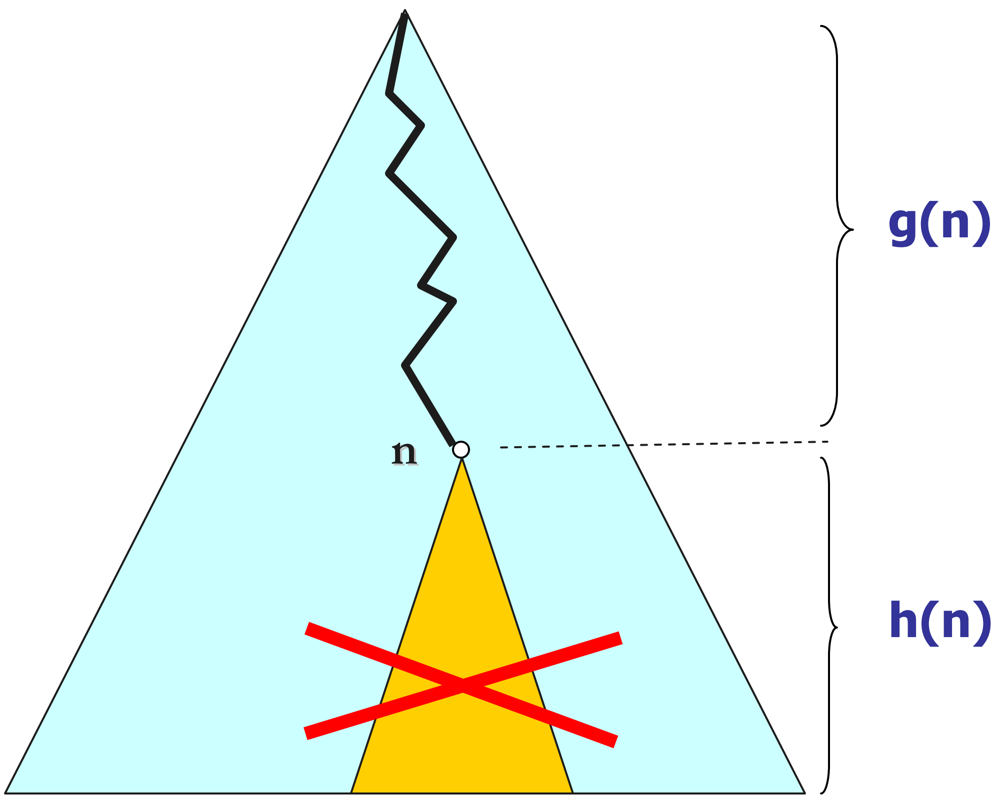


Best-First vs. Depth-first Branch-and-Bound

- **Best-First (A*): (optimal)**
  - Expand least number of nodes given h
  - Requires to store all search tree

- **Depth-first Branch-and-Bound:**
  - Can use only linear space
  - If find an optimal solution early will expand the same space as Best-First (if search space is a tree)
  - B&B can improve heuristic function dynamically

Classic Branch-and-Bound



OR Search Tree

Upper Bound **UB**

Lower Bound **LB**

$LB(n) = g(n) + h(n)$

Prune if $LB(n) \geq UB$

How to Generate Heuristics

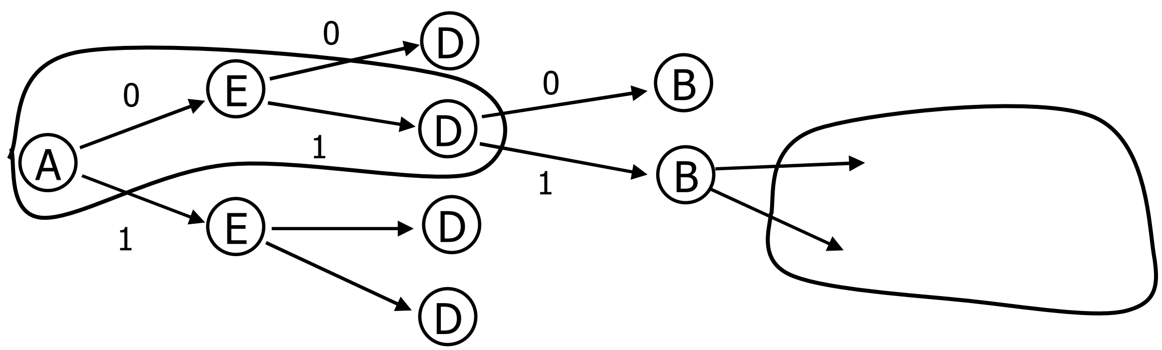
- The principle of relaxed models
  - Linear relaxation for integer programs
  - Mini-Bucket Elimination
  - Bounded directional consistency ideas

Generating Heuristic for Graphical Models
(Kask & Dechter, AIJ'01)

Given a cost function

$$C(a,b,c,d,e) = F(a) + F(b,a) + F(c,a) + F(e,b,c) + F(d,b,a)$$

Define an evaluation function over a partial assignment as the probability of it's best extension



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

$$= \underbrace{F(a)} + \underbrace{\min_{b,c} F(b,a) + F(c,a) + F(e,b,c) + F(d,a,b)}$$

$$= g(a,e,d) \bullet H^*(a,e,d)$$

Generating Heuristics (cont.)

$H^*(a,e,d) = \min_{b,c} F(b,a) + F(c,a) + F(e,b,c) + F(d,a,b)$

$= \min_c [F(c,a) + \min_b [F(e,b,c) + F(b,a) + F(d,a,b)]]$

$>= \min_c [F(c,a) + \min_b F(e,b,c) + \min_b [F(b,a) + F(d,a,b)]]$

$= \min_b [F(b,a) + F(d,a,b)] + \min_c [F(c,a) + \min_b F(e,b,c)]$

$= h^B(d,a) + h^C(e,a)$

$= H(a,e,d)$

$f(a,e,d) = g(a,e,d) + H(a,e,d) <= f^*(a,e,d)$

The heuristic function H is what is compiled during the preprocessing stage of the Mini-Bucket algorithm.

Generating Heuristics (cont.)

$$H^*(a,e,d) = \min_{b,c} F(b,a) + F(c,a) + F(e,b,c) + F(d,a,b)$$

$$= \min_c [F(c,a) + \min_b [F(e,b,c) + F(b,a) + F(d,a,b)]]$$

$$>= \min_c [F(c,a) + \min_b F(e,b,c) + \min_b [F(b,a) + F(d,a,b)]]$$

$$= \min_b [F(b,a) + F(d,a,b)] + \min_c [F(c,a) + \min_b F(e,b,c)]$$

$$= h^B(d,a) + h^C(e,a)$$

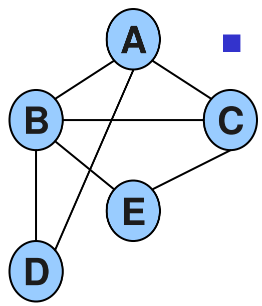
$$= H(a,e,d)$$

$$f(a,e,d) = g(a,e,d) + H(a,e,d) <= f^*(a,e,d)$$

The heuristic function H is what is compiled during the preprocessing stage of the Mini-Bucket algorithm.

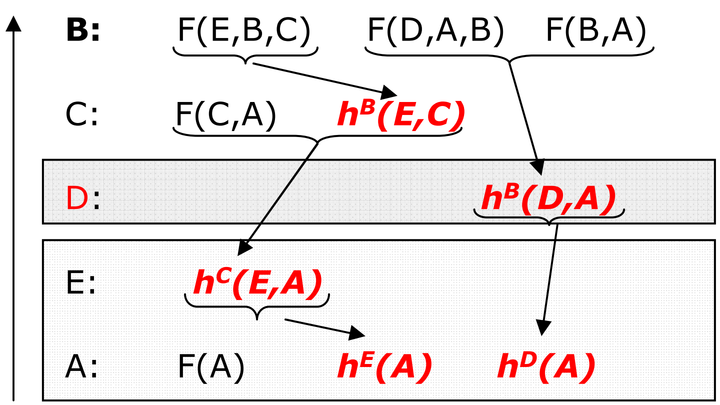
Static MBE Heuristics

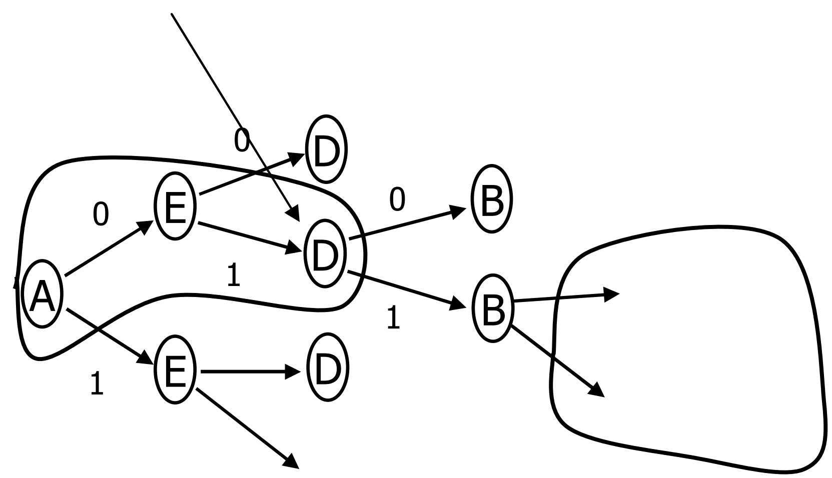
- Given a partial assignment x^p, estimate the cost of the best extension to a full solution
- The evaluation function $f(x^p)$ can be computed using function recorded by the Mini-Bucket scheme



Cost Network

$$f(a,e,D))=g(a,e) + H(a,e,D)$$





$$f(a,e,D) = \underbrace{F(a)}_{g} + \underbrace{h^B(D,a) + h^C(e,a)}_{h \text{ – is admissible}}$$

Heuristics Properties

- MB Heuristic is monotone, admissible
- Computed in linear time

- **IMPORTANT**:
  - Heuristic strength can vary by MB(i)
  - Higher i-bound $\Rightarrow$ more pre-processing $\Rightarrow$ stronger heuristic $\Rightarrow$ less search

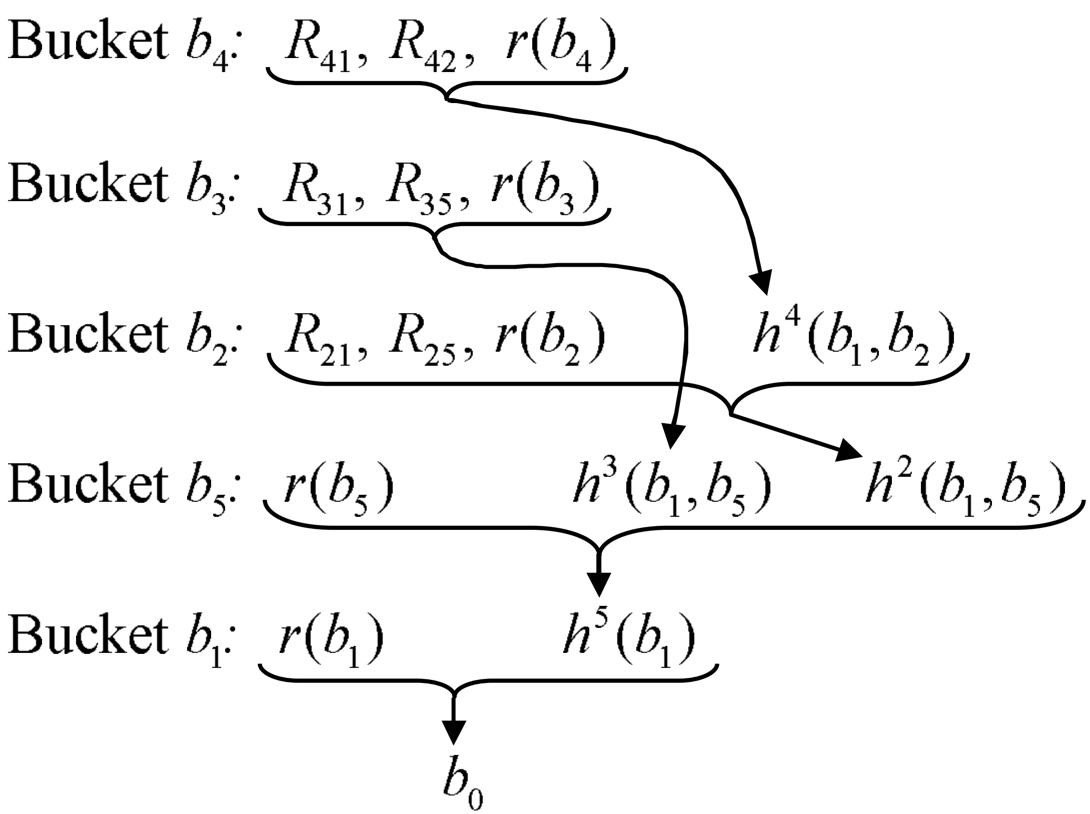
- Allows controlled trade-off between preprocessing and search

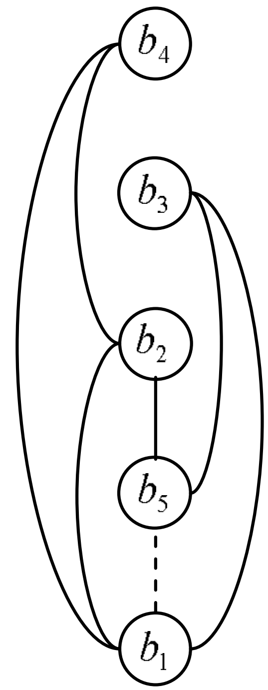
Combinatorial Auctions Example

- BIDS
  - B1 = {1, 2, 3, 4}
  - B2 = {2, 3, 6}
  - B3 = {1, 4, 5}
  - B4 = {2, 8}
  - B5 = {5, 6}
- PRICES
  - P1 = 8
  - P2 = 6
  - P3 = 5
  - P4 = 2
  - P5 = 2

- **Constraint Optimization Problem**
  - **Variables**: b_1, b_2, b_3, b_4, b_5 (i.e., bids)
  - **Domains**: {0, 1}
  - **Constraints**: R_{12}, R_{13}, R_{14}, R_{24}, R_{25}, R_{35}
  - **Cost functions**: $r(b_1)$, $r(b_2)$, $r(b_3)$, $r(b_4)$, $r(b_5)$

Combinatorial Auctions





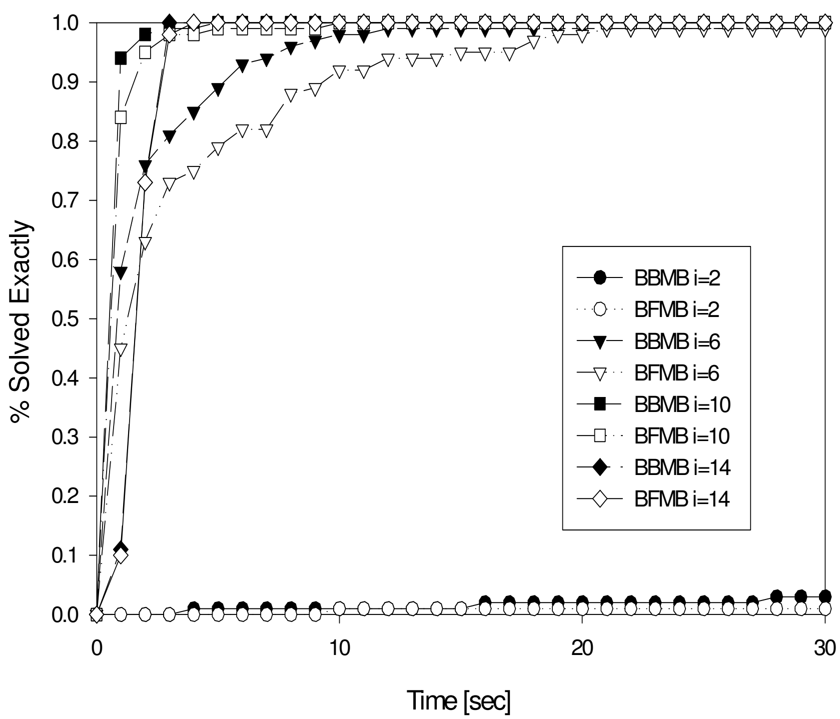
OPT

Experimental Methodology

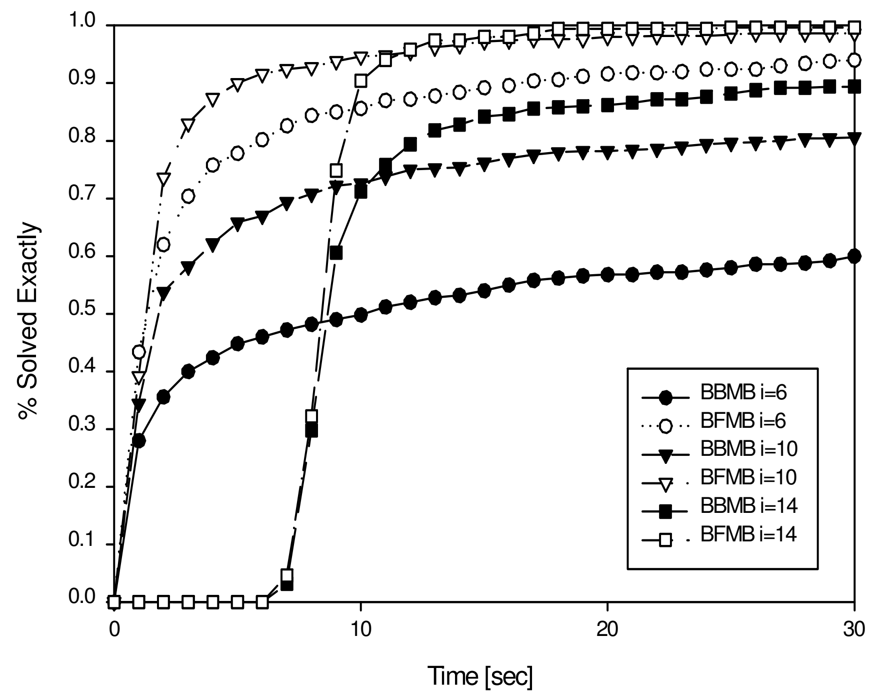
- **Algorithms**
  - BBMB(i) - Branch-and-Bound with MB(i)
  - BBFB(i) - Best-First with MB(i)
  - MBE(i) – Mini-Bucket Elimination
- **Benchmarks**
  - Random Coding (Bayesian)
  - CPCS (Bayesian)
  - Random (CSP)
- **Measures of performance**
  - Compare accuracy given a fixed amount of time
    - i.e., how close is the cost found to the optimal solution
  - Compare trade-off performance as a function of time

Empirical Evaluation of Mini-Bucket heuristics:
Random coding networks (Kask & Dechter, UAI'99)

Random Coding, K=100, noise=0.28



Random Coding, K=100, noise=0.32



Each data point represents an average over 100 random instances

Max-CSP Experiments

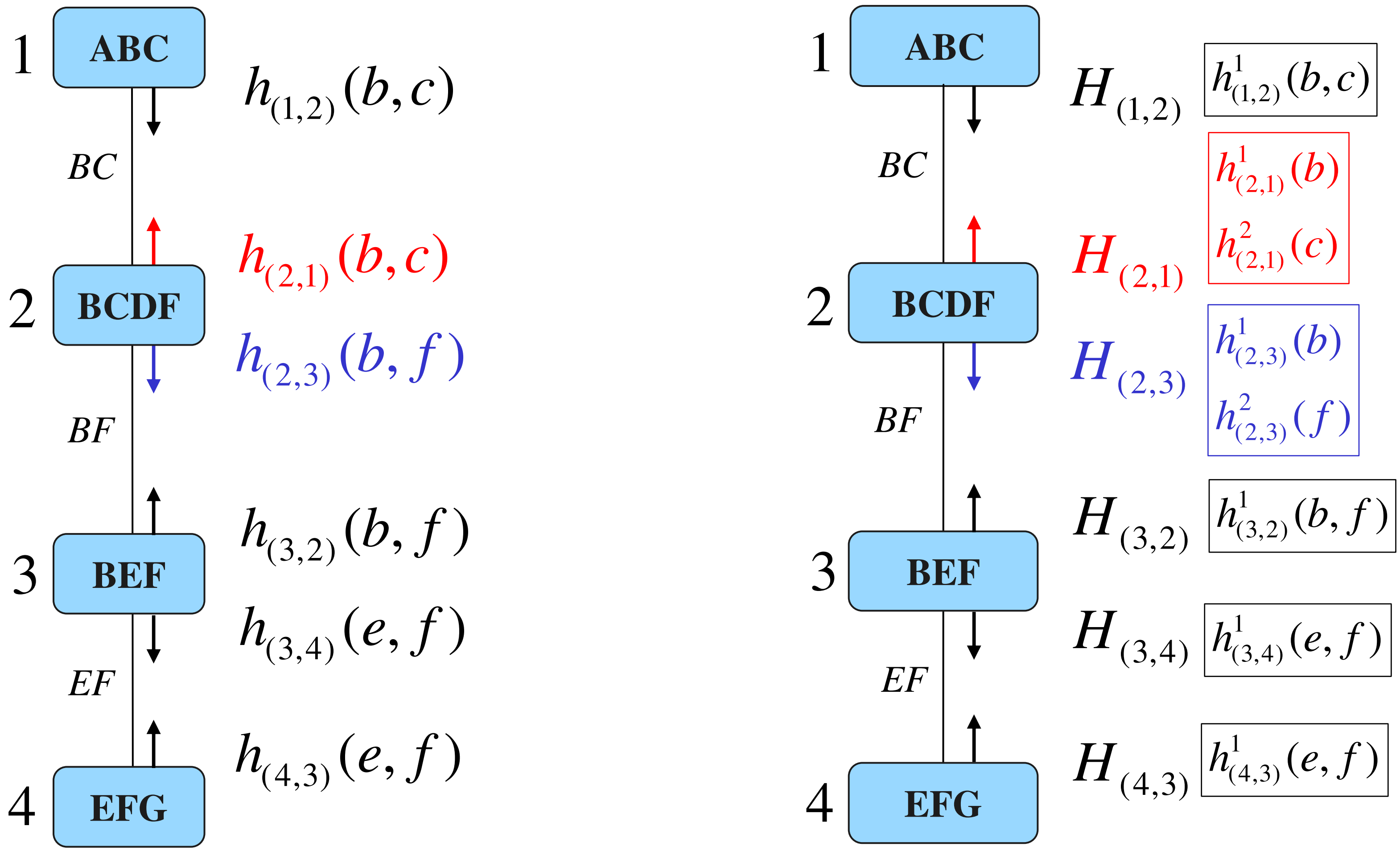
(Kask & Dechter, CP'00)

T	MBE BBMB BFMB i=2 #/time	MBE BBMB BFMB i=4 #/time	MBE BBMB BFMB i=6 #/time	MBE BBMB BFMB i=8 #/time	MBE BBMB BFMB i=10 #/time	MBE BBMB BFMB i=12 #/time	PFC-MRDAC #/time
N=100, K=3, C=200. Time bound 1 hr. Avg w^*=21. Sparse network.							
1	70/0.03 90/12.5 80/0.03	90/0.06 **100/0.07** **100/0.07**	100/0.32 100/0.33 100/0.33	100/2.15 100/2.16 100/2.15	100/15.1 100/15.1 100/15.1	100/116 100/116 100/116	100/0.08
2	0/- 0/- 0/-	0/- 0/- 0/-	4/0.35 96/644 56/131	20/2.28 **92/41** 88/170	20/15.6 96/69 92/135	24/123 100/125 100/130	100/757
3	0/- 0/- 0/-	0/- 0/- 0/-	0/- 100/996 16/597	0/- 100/326 60/462	4/14.4 **100/94.6** 88/344	4/114 100/190 84/216	100/2879
4	0/- 0/- 0/-	0/- 0/- 0/-	0/- 52/2228 4/2934	0/- 88/1042 8/540	4/14.9 92/396 28/365	8/120 **100/283** 60/866	100/7320

Dynamic MB and MBTE Heuristics
(Kask, Marinescu and Dechter, UAI'03)

- Rather than pre-compile compute the heuristics during search
- **Dynamic MB**: use the Mini-Bucket algorithm to produce a bound for any node during search
- **Dynamic MBTE**: We can compute heuristics simultaneously for all un-instantiated variables using mini-bucket-tree elimination
- **MBTE** is an approximation scheme defined over cluster-trees. It outputs multiple bounds for each variable and value extension at once

Mini Bucket Tree Elimination



Branch-and-Bound w/ Mini-Buckets

- BB with static Mini-Bucket Heuristics (s-BBMB)
  - Heuristic information is pre-compiled before search
  - Static variable ordering, prunes current variable

- BB with dynamic Mini-Bucket Heuristics (d-BBMB)
  - Heuristic information is assembled during search
  - Static variable ordering, prunes current variable

- BB with dynamic Mini-Bucket-Tree Heuristics (BBBT)
  - Heuristic information is assembled during search.
  - Dynamic variable ordering, prunes all future variables

Empirical Evaluation

- Algorithms:
  - Complete
    - BBBT
    - BBMB
  - Incomplete
    - DLM
    - GLS
    - SLS
    - IJGP
    - IBP (coding)

- Measures:
  - Time
  - Accuracy (% exact)
  - #Backtracks
  - Bit Error Rate (coding)

- Benchmarks:
  - Coding networks
  - Bayesian Network Repository
  - Grid networks (N-by-N)
  - Random noisy-OR networks
  - Random networks

Real World Benchmarks

(Kask, Marinescu & Dechter, UAI'03)

Network	# vars	avg. dom.	max dom.	BBBT/ BBMB/ IJGP i=2 %[time]	BBBT/ BBMB/ IJGP i=4 %[time]	BBBT/ BBMB/ IJGP i=6 %[time]	BBBT/ BBMB/ IJGP i=8 %[time]	GLS % [time]	DLM % [time]	SLS % [time]
Mildew	35	17	100	**100[0.28]** 30[10.5] 90[3.59]	**100[0.56]** 95[0.18] 97[33.3]	- - -	- - -	15 [30.02]	0 [30.02]	90 [30.02]
Munin2	1003	5	21	95[1.65] 95[30.3] 95[2.44]	95[1.65] 95[30.5] 95[5.17]	95[2.32] 95[31.3] 95[64.9]	**100[1.97]** **100[1.84]** -	0 [30.01]	0 [30.01]	0 [30.01]
Pigs	441	3	3	**90[15.2]** 0[30.01] 80[0.31]	**100[3.73]** 60[4.85] 77[0.53]	**100[2.36]** 80[0.02] 80[1.43]	**100[0.58]** 95[0.04] 83[6.27]	10 [30.02]	0 [30.02]	0 [30.02]
CPCS360b	360	2	2	100[0.17] **100[0.04]** 100[10.6]	100[0.27] **100[0.03]** 100[10.5]	100[0.21] **100[0.03]** 100[9.82]	100[0.19] **100[0.03]** 100[8.59]	100 [30.02]	100 [30.02]	100 [30.02]

Average Accuracy and Time. 30 samples, 10 observations, 30 seconds

Empirical Results: Max-CSP

- **Random Binary Problems**: <N, K, C, T>
  - N: number of variables
  - K: domain size
  - C: number of constraints
  - T: Tightness

- **Task**: Max-CSP

BBBT(i) vs. BBMB(i), N=100

(Kask & Dechter, CP'00)

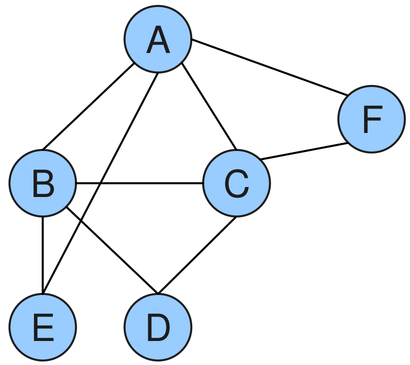
$N = 100$, $K = 5$, $C = 300$. $w^* = 33.9$. 10 instances. time = 600sec.								
	BBMB						BBBT	PFC-MPRDAC
T	i=2	i=3	i=4	i=5	i=6	i=7	i=2	
	# solved time backtracks	# solved time backtracks	# solved time backtracks	# solved time backtracks	# solved time backtracks	# solved time backtracks	# solved time backtracks	# solved time backtracks
3	6 6 150K	6 6 150K	6 6 150K	6 5 115K	8 6.8 115K	8 15 8	10 7.73 60	10 0.03 750
5	2 36 980K	2 32 880K	2 24 650K	2 5.3 130K	3 38 870K	3 33 434K	10 14.3 114	10 0.06 1.5K
7	0	0	0	0	0	0	10 29 331	6 267 1.6M

BBBT(*i*) vs. BBMB(*i*).

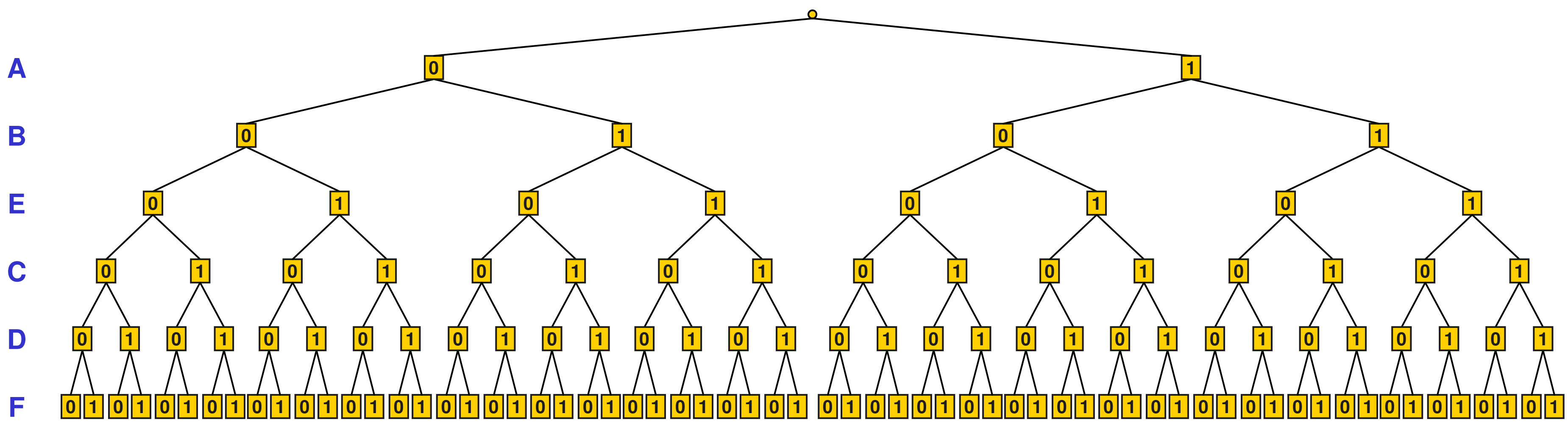
Outline

- Introduction
  - Optimization tasks for graphical models
  - Solving optimization problems by inference and search
- Inference
  - Bucket Elimination, Dynamic Programming
  - Mini-Bucket Elimination
- Search (OR)
  - Branch-and-Bound and Best-First Search
  - Lower-bounding heuristics
- **AND/OR search spaces**
  - **AND/OR Tree search (linear space)**
  - **AND/OR Graph search (caching)**
  - Searching the AND/OR spaces
- Hybrids of search and inference
  - Cutset decomposition
  - Super-bucket scheme
- Software

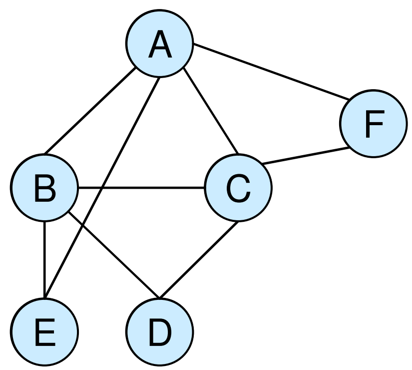
Classic OR Search Space



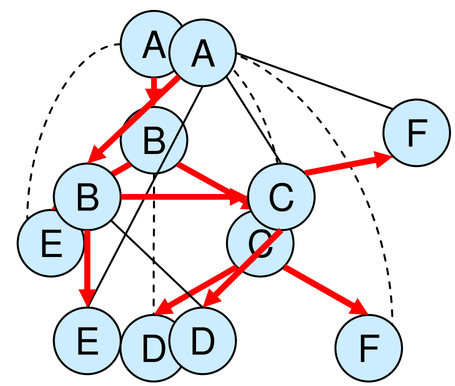
Ordering: A B E C D F



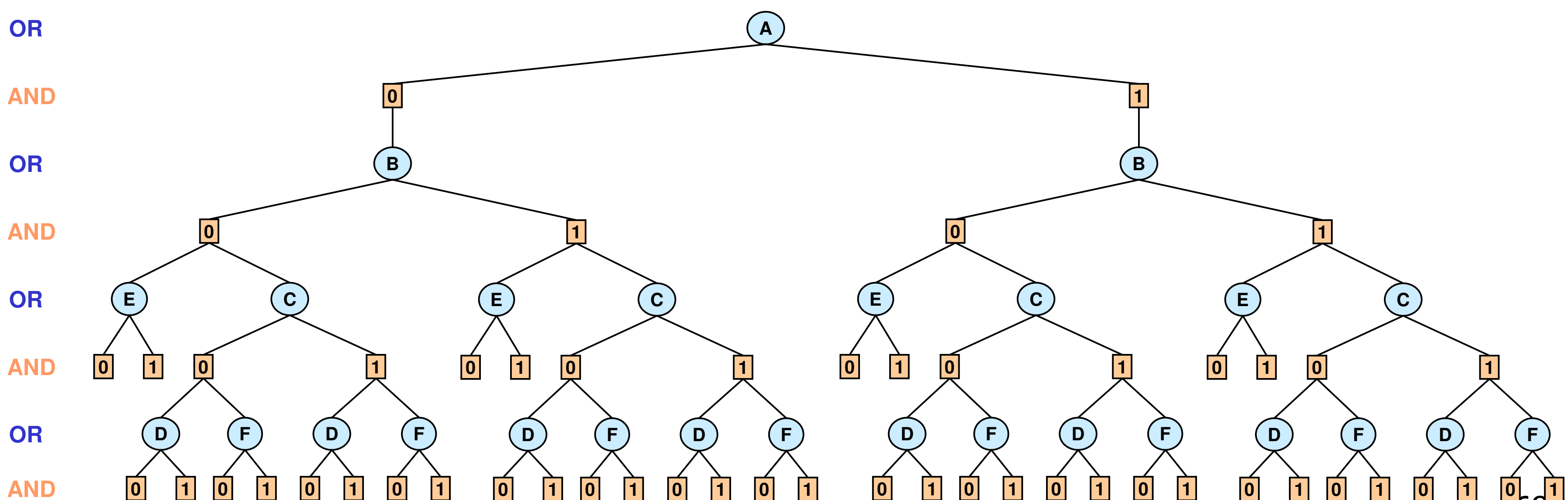
AND/OR Search Space



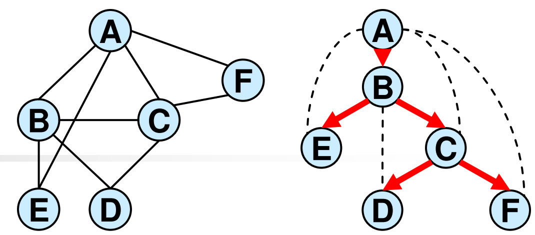
Primal graph

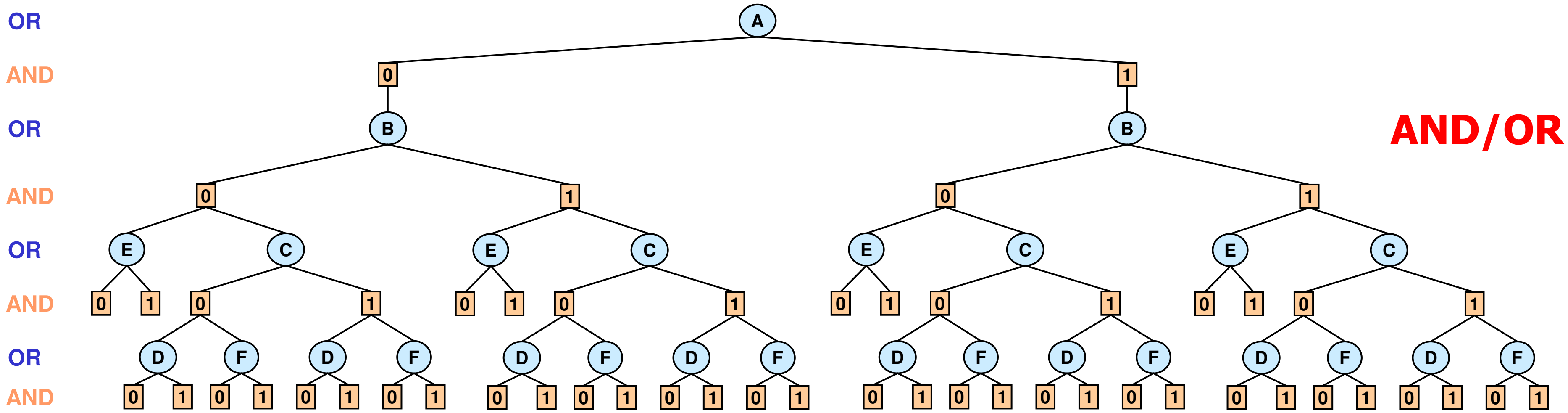


DFS tree

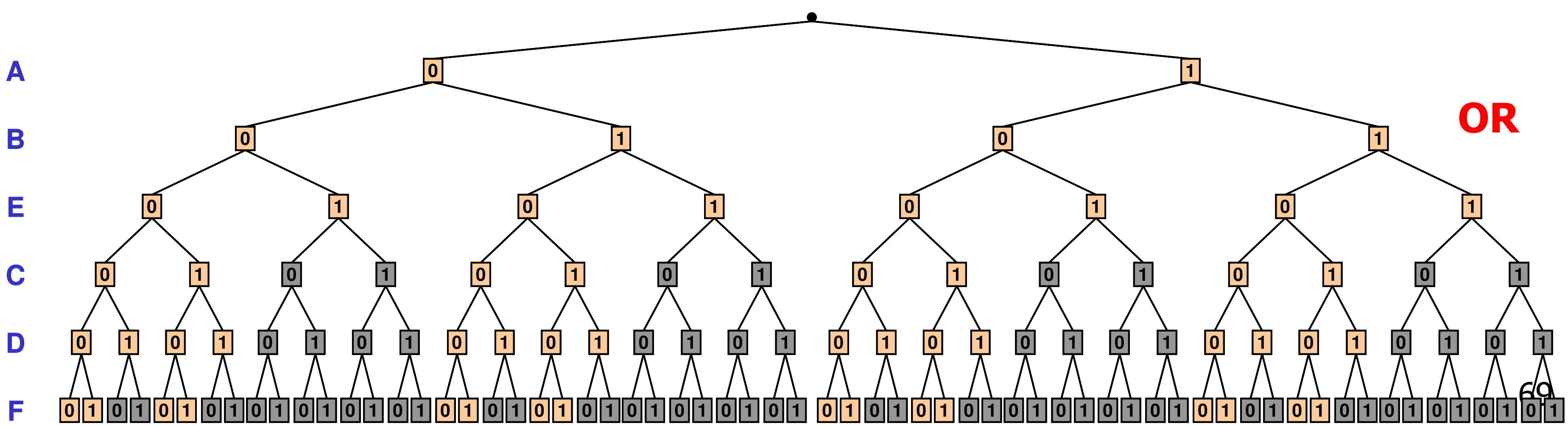


AND/OR vs. OR





AND/OR size: exp(4), OR size exp(6)

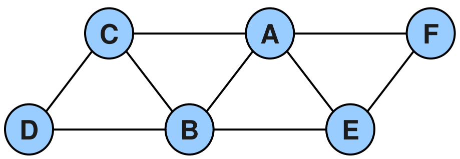


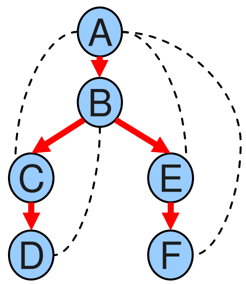
OR space vs. AND/OR space

width	height	OR space			AND/OR space		
		Time (sec.)	Nodes	Backtracks	Time (sec.)	AND nodes	OR nodes
5	10	3.154	2,097,150	1,048,575	0.03	10,494	5,247
4	9	3.135	2,097,150	1,048,575	0.01	5,102	2,551
5	10	3.124	2,097,150	1,048,575	0.03	8,926	4,463
4	10	3.125	2,097,150	1,048,575	0.02	7,806	3,903
5	13	3.104	2,097,150	1,048,575	0.1	36,510	18,255

Random graphs with 20 nodes, 20 edges and 2 values per node.

AND/OR Tree Search for COP





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

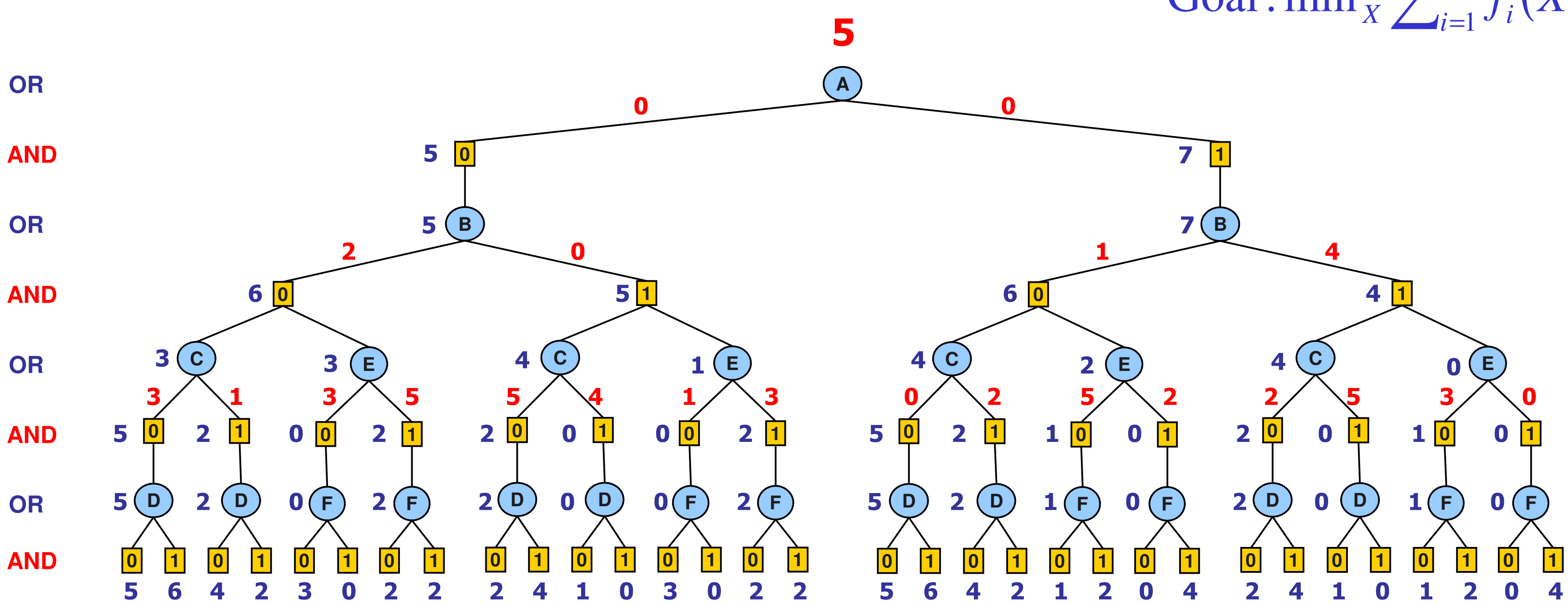
B	D	f_6
0	0	4
0	1	2
1	0	1
1	1	0

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

$$\text{Goal} : \min_X \sum_{i=1}^{9} f_i(X)$$



AND node = Combination operator (summation)

OR node = Marginalization operator (minimization)

Summary of AND/OR Search Trees

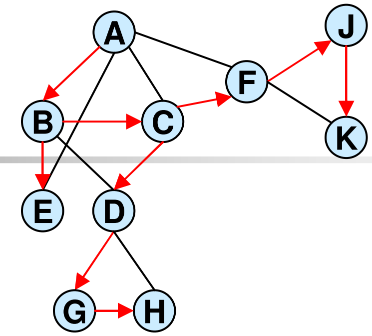
- Based on a backbone pseudo-tree
- A solution is a subtree
- Each node has a **value** – cost of the optimal solution to the subproblem (computed recursively based on the values of the descendants)
- **Solving a task = finding the value of the root node**
- AND/OR search tree and algorithms are

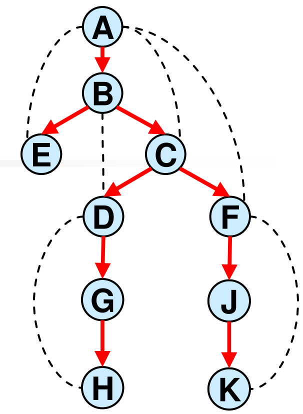
  (Freuder & Quinn, 1985), (Collin, Dechter & Katz, 1991), (Bayardo & Miranker, 1995)
  - Space: **O(n)**
  - Time: **O(exp(m))**, where m is the depth of the pseudo-tree
  - Time: **O(exp(w* log n))**
  - BFS is time and space: **O(exp(w* log n)**

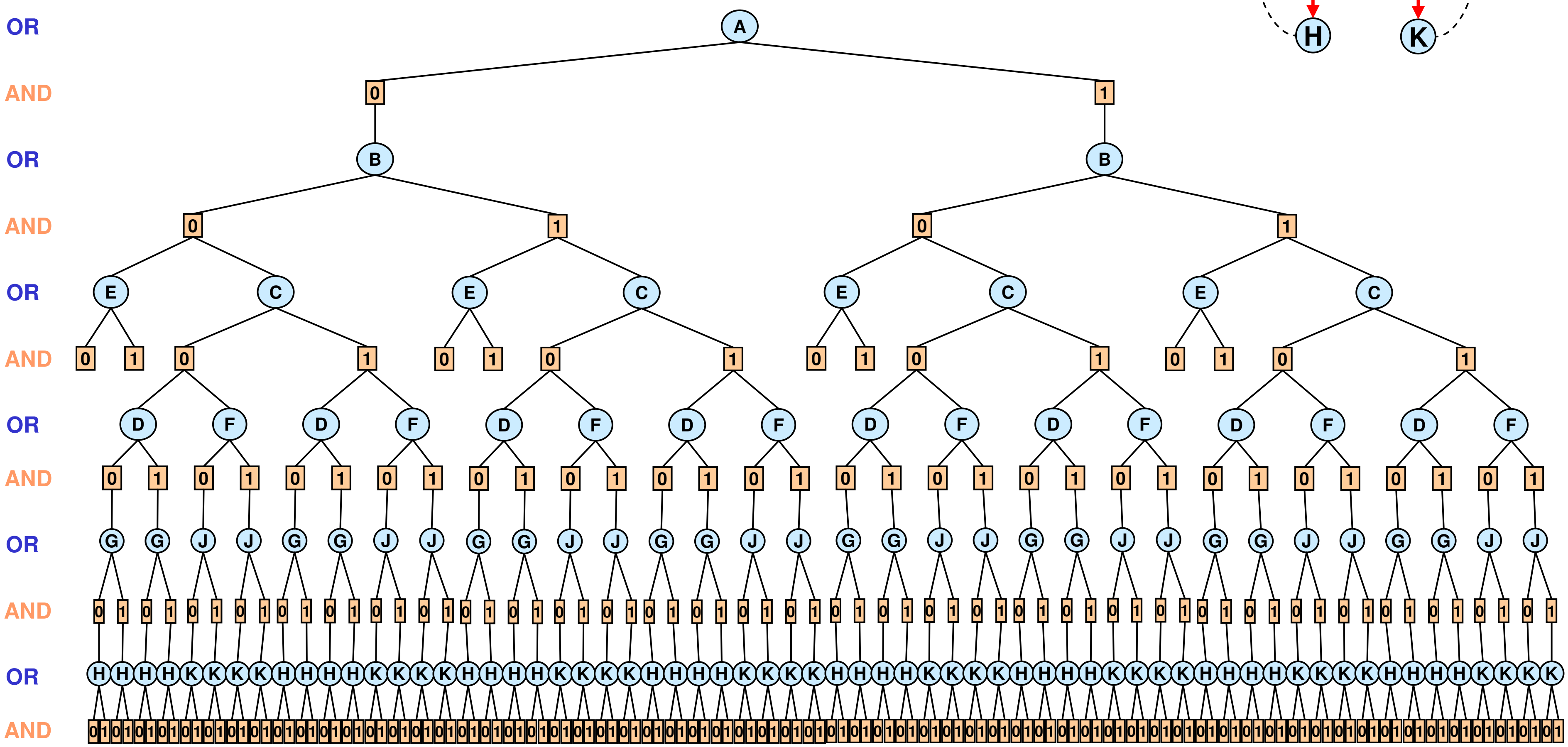
From AND/OR Trees to AND/OR Graphs

- Any two nodes that root identical subtrees or subgraphs can be **merged**
- **Minimal AND/OR search graph**:
  - closure under merge of the AND/OR search tree
- Inconsistent sub-trees can be pruned too
- Some portions can be collapsed or reduced

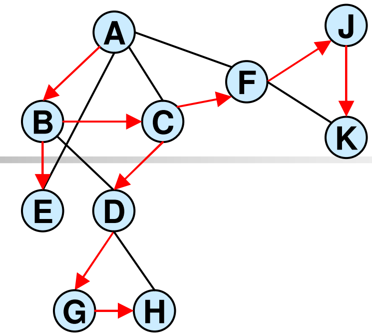
AND/OR Tree

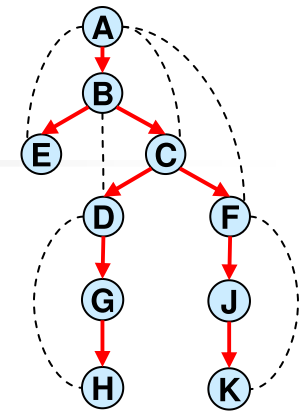

A B C D E F G H J K

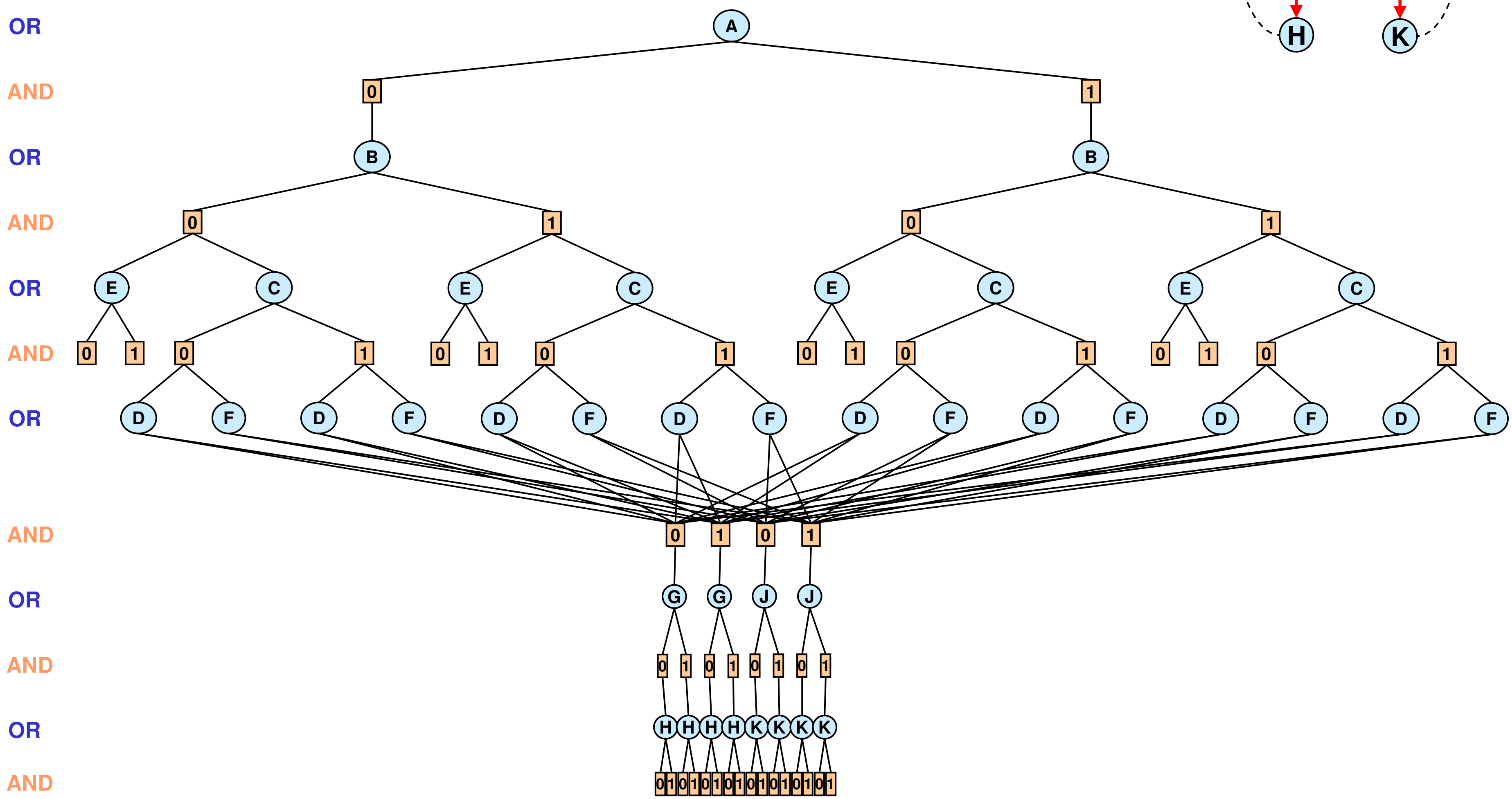

A B E C D F G J H K


OR
AND
OR
AND
OR
AND
OR
AND
OR
AND
OR
AND

An AND/OR Graph:
Caching Goods

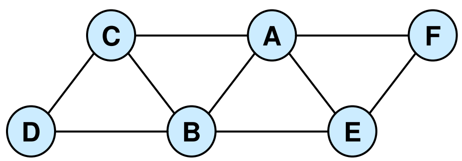




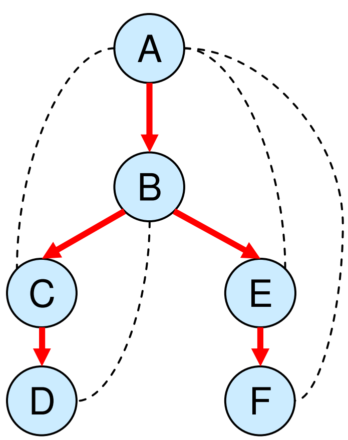


AND/OR Search Graph
(Context-based Caching)

- Identify **unifiable** subtrees
- Context = parent separator set in induced pseudo-tree
  = current variable + ancestors connected to subtree below

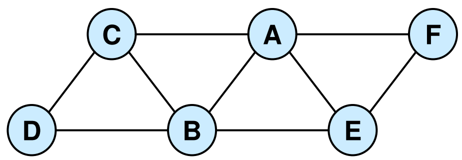


(Dechter & Mateescu, UAI'04), (Mateescu & Dechter, IJCAI'05)

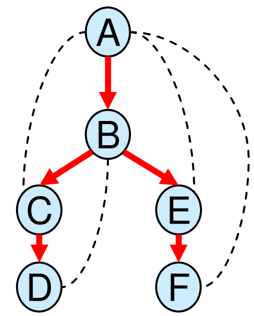


context(A) = {**A**}
context(B) = {**B,A**}
context(C) = {**C,B**}
context(D) = {**D**}
context(E) = {**E,A**}
context(F) = {**F**}

AND/OR Search Graph

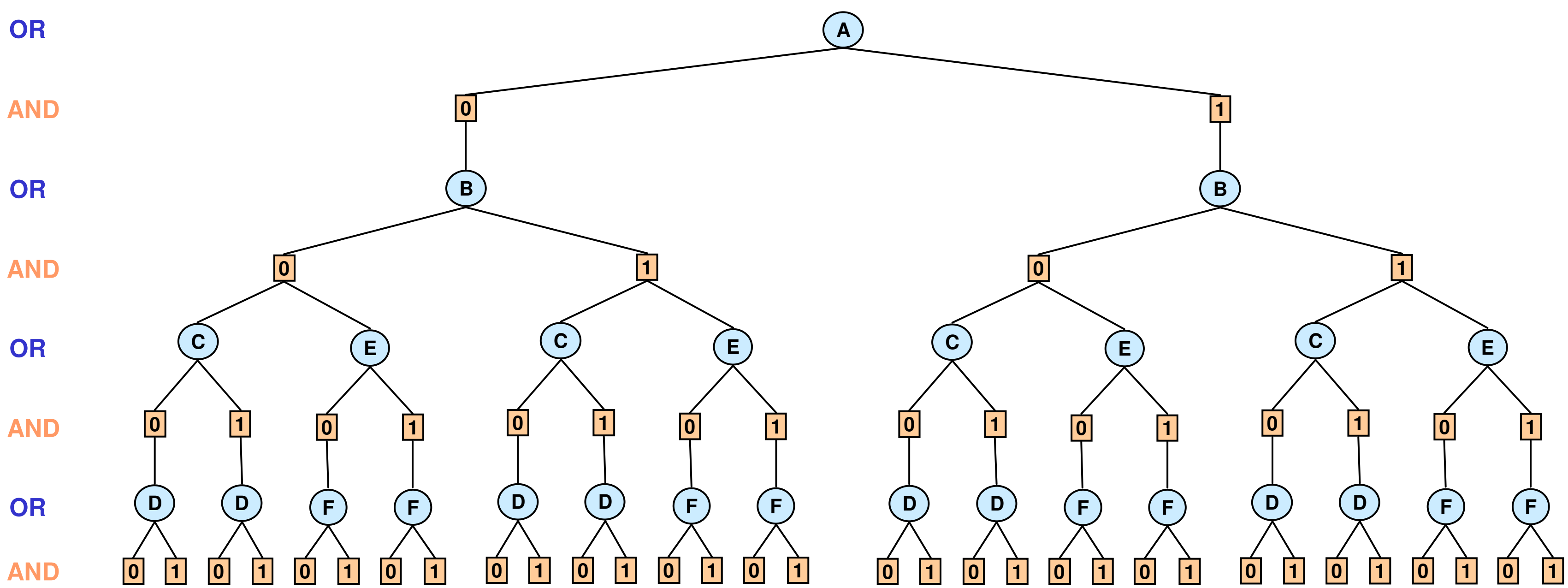


Primal graph

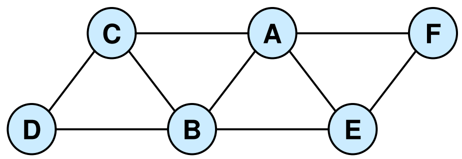


context(A) = {**A**}
context(B) = {**B,A**}
context(C) = {**C,B**}
context(D) = {**D**}
context(E) = {**E,A**}
context(F) = {**F**}

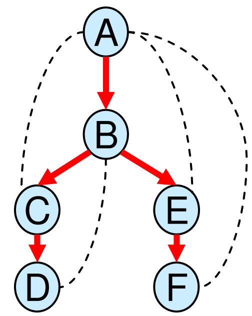
Pseudo-tree



AND/OR Search Graph

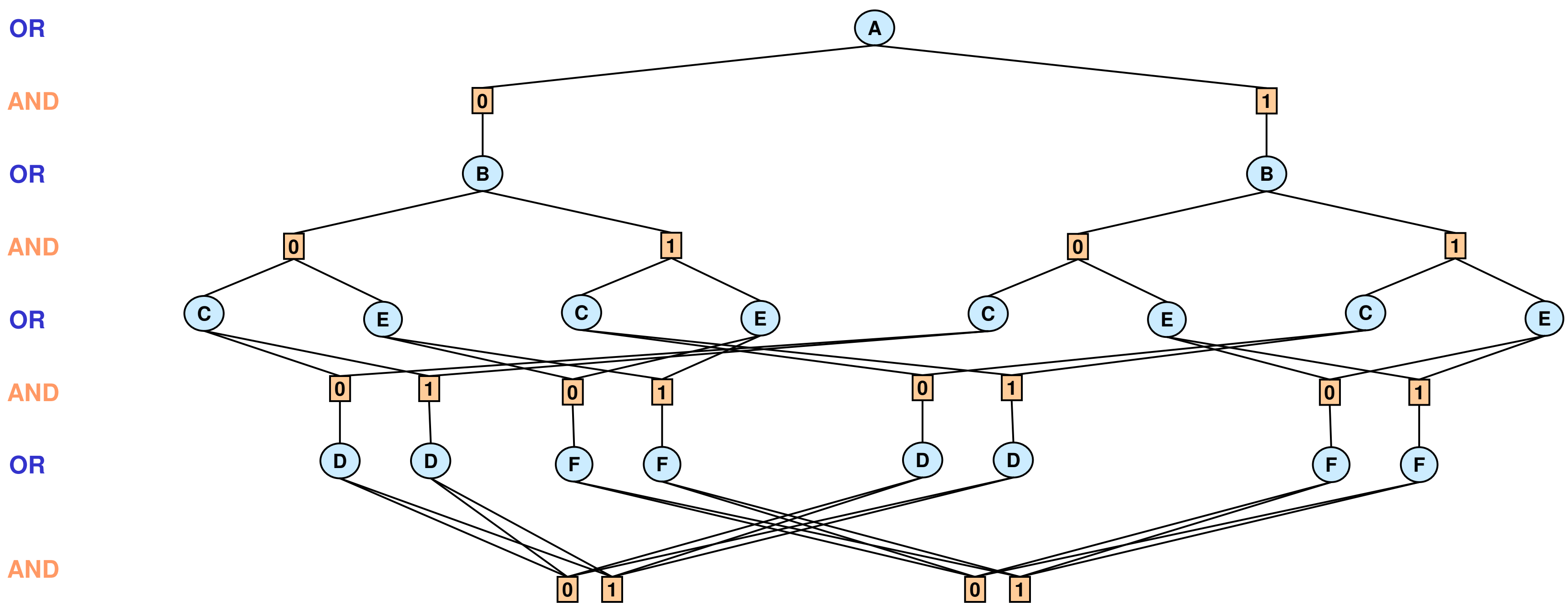


Primal graph

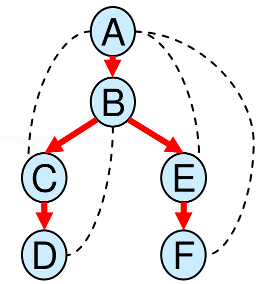


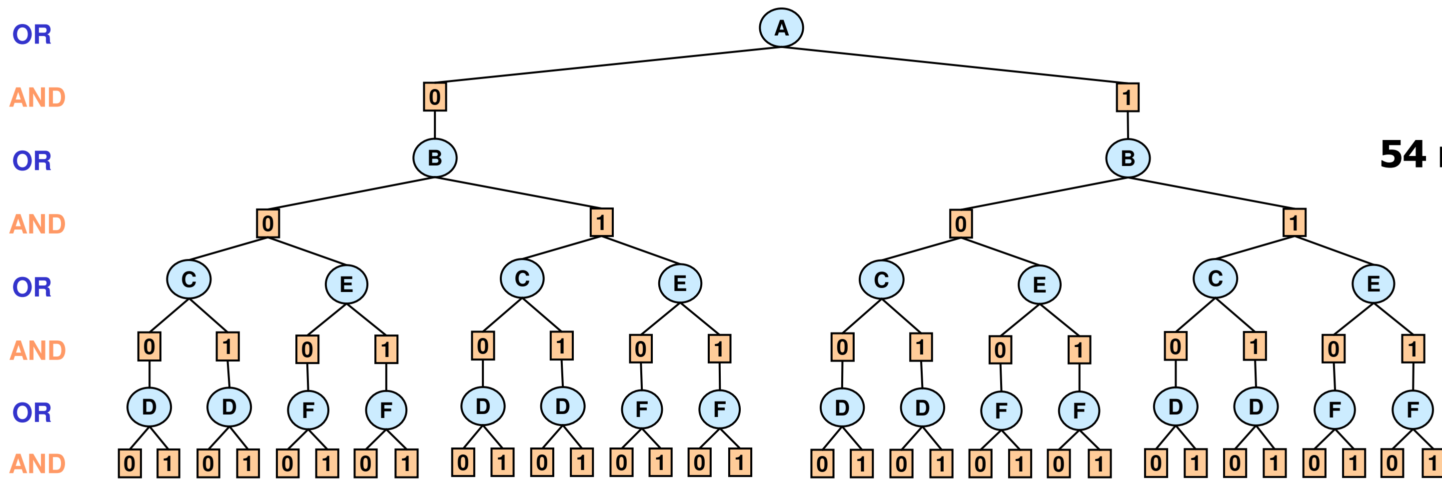
context(A) = {**A**}
context(B) = {**B,A**}
context(C) = {**C,B**}
context(D) = {**D**}
context(E) = {**E,A**}
context(F) = {**F**}

Pseudo-tree



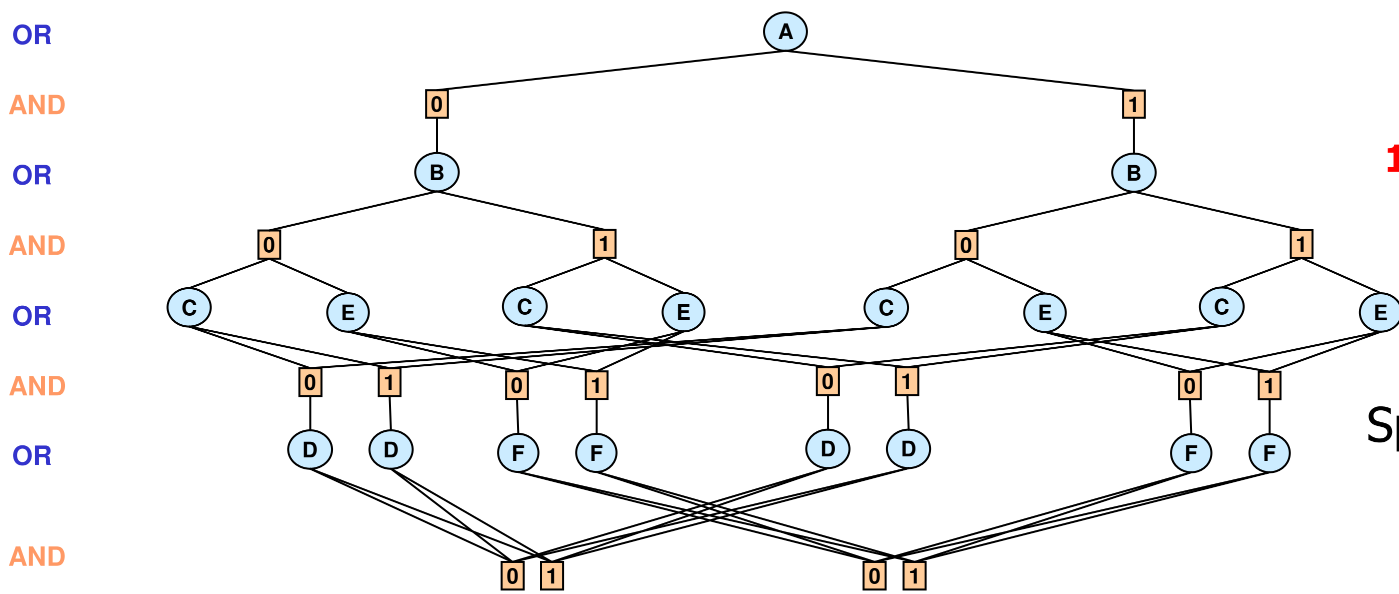
AND/OR Tree vs. Graph





54 nodes

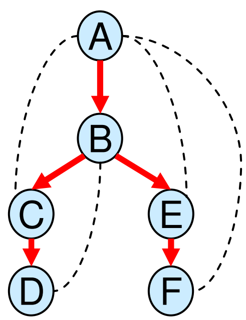
Space: **O(n)**



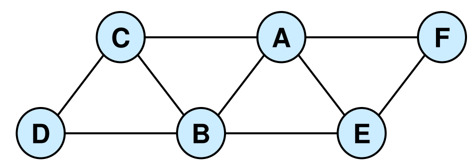
18 nodes

Space: **O(exp(w*))**

Context-based Caching



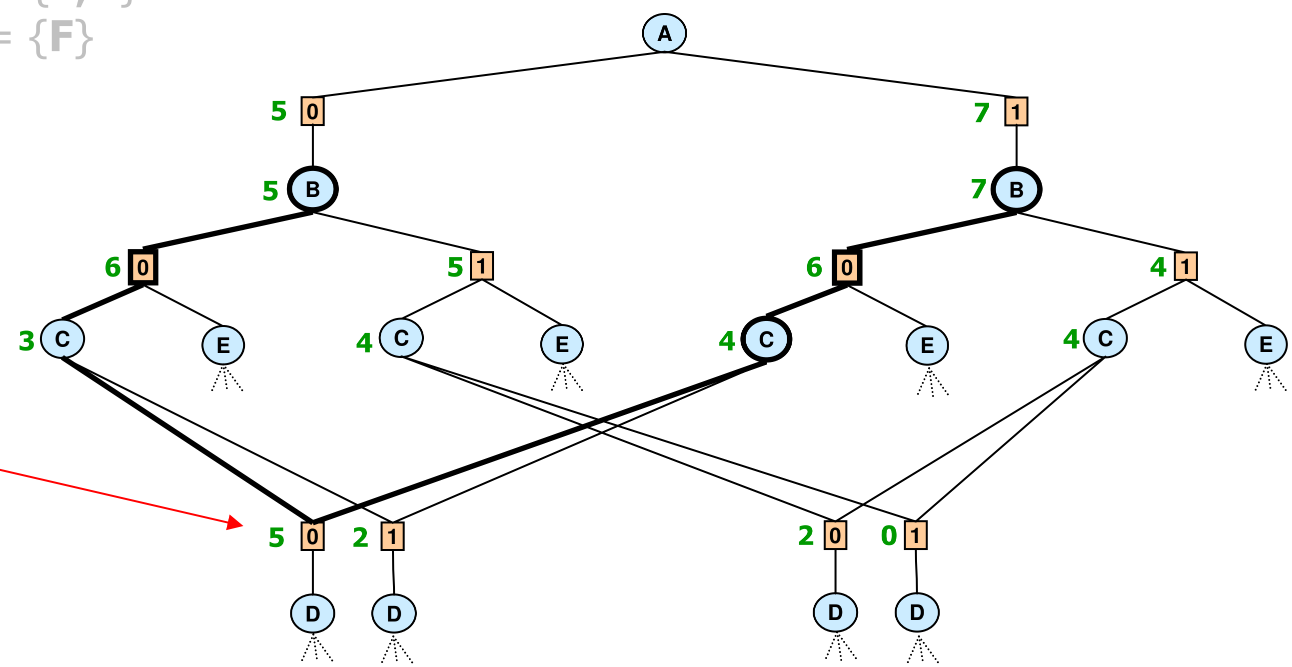
context(A) = {**A**}
context(B) = {**B,A**}
context(C) = {C,B}
context(D) = {**D**}
context(E) = {**E,A**}
context(F) = {**F**}



Primal graph

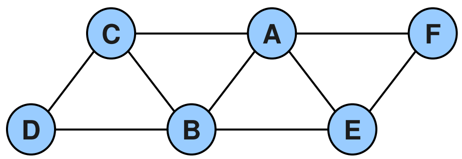
Cache Table (C)

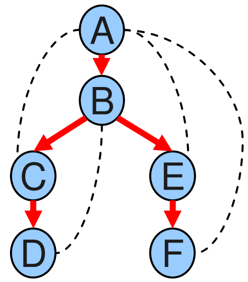
B	C	Value
0	0	5
0	1	2
1	0	2
1	1	0



Space: **O(exp(2))**

Example (graph search)





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

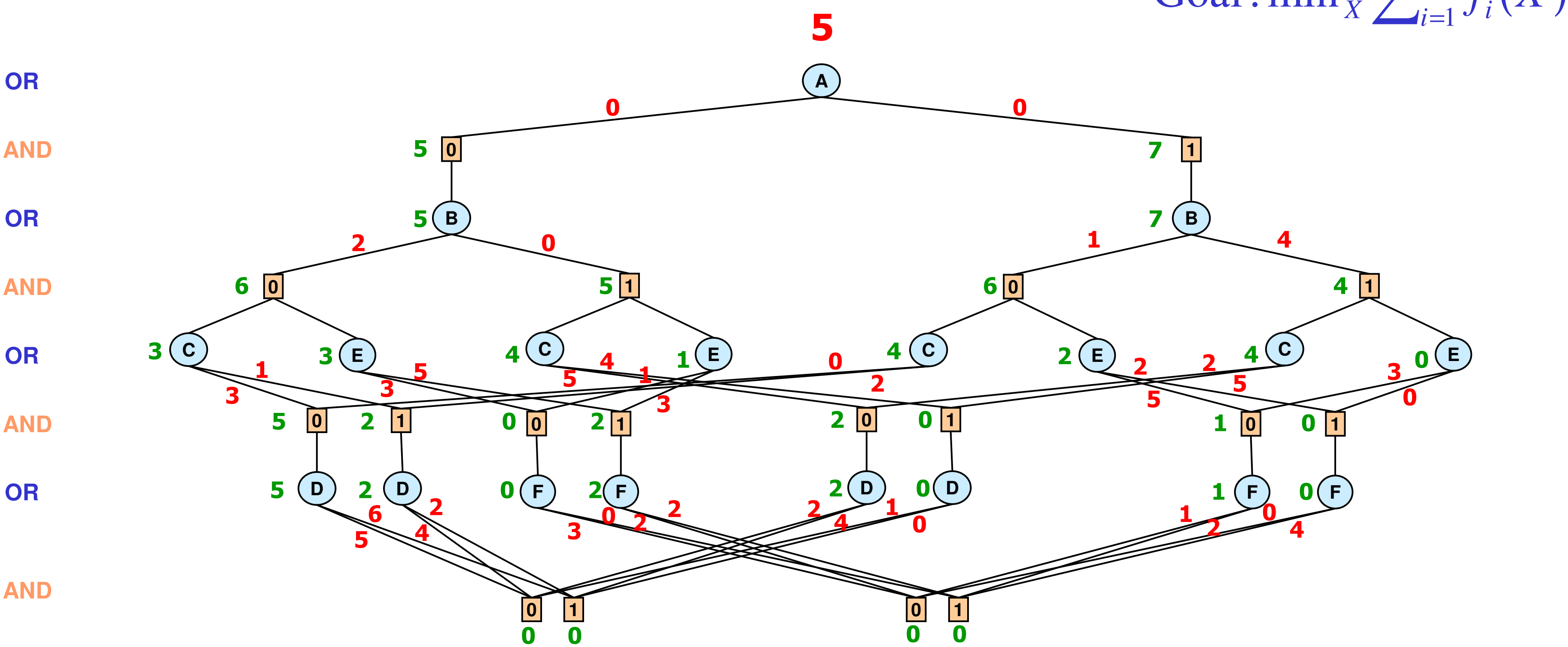
B	D	f_6
0	0	4
0	1	2
1	0	1
1	1	0

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

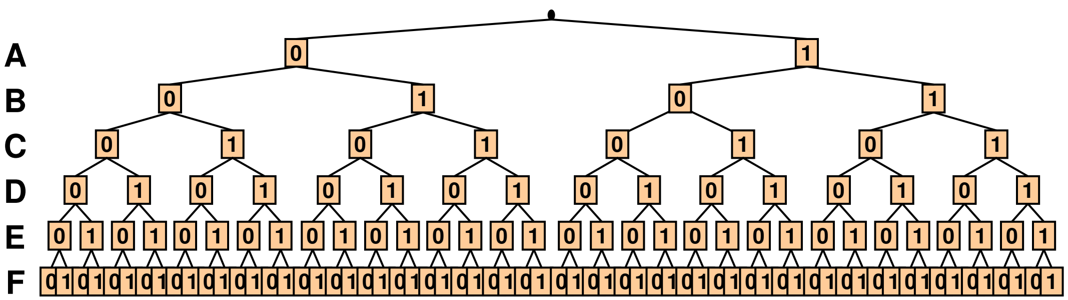
C	D	f_8
0	0	1
0	1	4
1	0	0
1	1	0

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

$$\text{Goal}: \min_X \sum_{i=1}^{9} f_i(X)$$

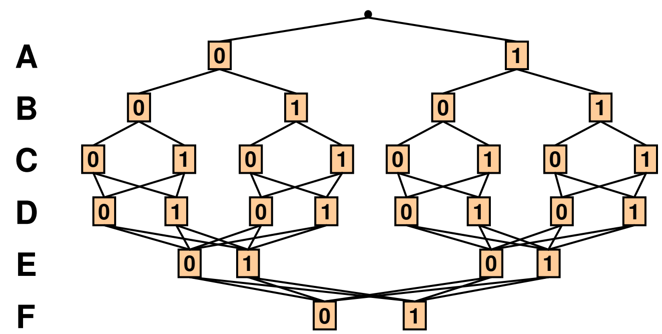


All Four Search Spaces



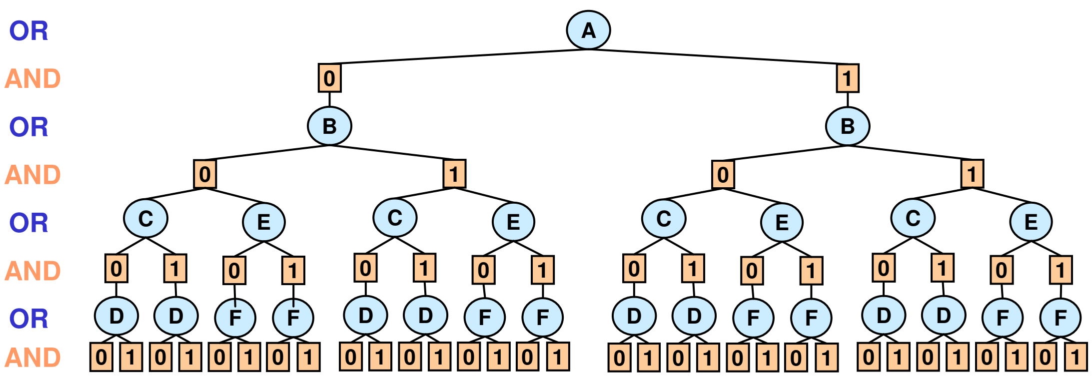
Full OR search tree

126 nodes



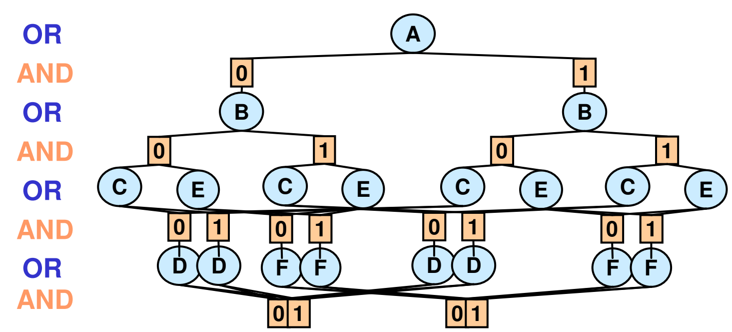
Context minimal OR search graph

28 nodes



Full AND/OR search tree

54 AND nodes



Context minimal AND/OR search graph

18 AND nodes

AND/OR vs. OR DFS Algorithms

k = domain size
m = pseudo-tree depth
n = number of variables
w^* = induced width
pw^* = path width

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

  (Freuder, 1985; Bayardo, 1995; Darwiche, 2001)

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

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

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

Searching AND/OR Graphs

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

j=0	j=w*
Space: O(n)	**Space: O(exp w*)**
Time: O(exp(w* log n))	**Time: O(exp w*)**

AO(j) time complexity?

Pseudo-trees (I)

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

- Finding the minimal context/depth pseudo-tree is a hard problem

- Heuristics
  - Min-fill (min context)
  - Hypergraph separation (min depth)

Pseudo-trees (II)

- **MIN-FILL** (Kjæaerulff, 1990), (Bayardo & Miranker, 1995)
  - Depth-first traversal of the induced graph constructed along some elimination order
  - Elimination order prefers variables with smallest "fill set"

- **HYPERGRAPH** (Darwiche, 2001)
  - Constraints are vertices in the hypergraph and variables are hyperedges
  - Recursive decomposition of the hypergraph while minimizing the separator size (i.e. number of variables) at each step
  - Using state-of-the-art software package **`hMeTiS`**

Quality of Pseudo-trees

Network	hypergraph		min-fill	
	width	depth	width	depth
barley	7	13	7	23
diabetes	7	16	4	77
link	21	40	15	53
mildew	5	9	4	13
munin1	12	17	12	29
munin2	9	16	9	32
munin3	9	15	9	30
munin4	9	18	9	30
water	11	16	10	15
pigs	11	20	11	26

Bayesian Networks Repository

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

SPOT5 Benchmarks

Outline

- Introduction
  - Optimization tasks for graphical models
  - Solving optimization problems by inference and search
- Inference
  - Bucket Elimination, Dynamic Programming
  - Mini-Bucket elimination
- Search (OR)
  - Branch-and-Bound and Best-First search
  - Lower-bounding heuristics
- **AND/OR search spaces**
  - AND/OR Tree search (linear space)
  - AND/OR Graph search (caching)
  - **Searching AND/OR spaces**
  - **Searching the AND/OR tree (linear space)**
    - **with mini-bucket heuristics**
    - **0/1 integer programming with linear programming relaxations**
    - **dynamic variable ordering**
  - **Searching the AND/OR graph (caching)**
- Hybrids of search and inference
  - Cutset decomposition
  - Super-bucket scheme
- Software

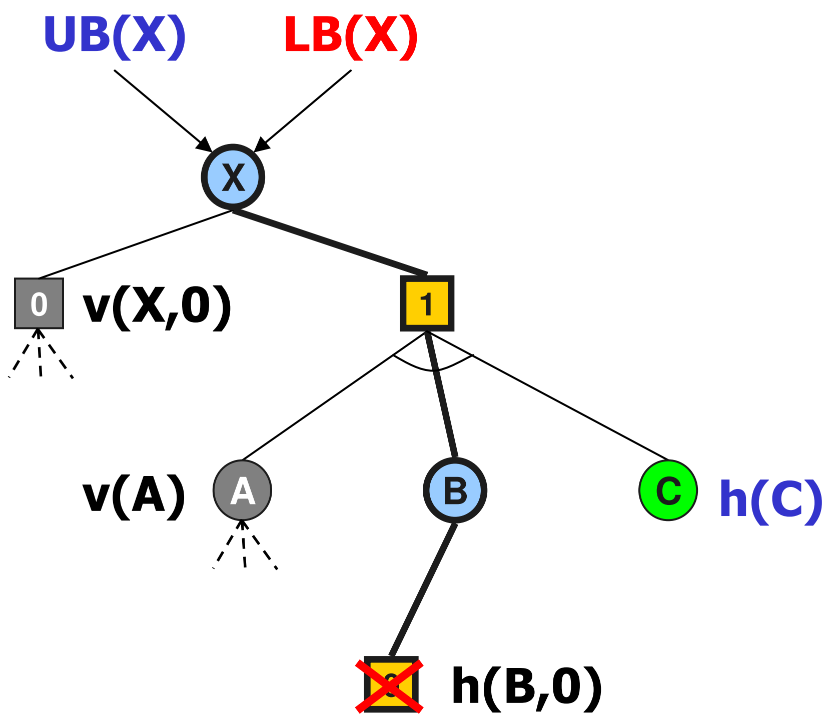
AND/OR Branch-and-Bound (AOBB)

(Marinescu & Dechter, IJCAI'05)

- Associate each node **n** with a static heuristic estimate h(n) of v(n)
  - h(n) is a lower bound on the value v(n)

- For every node **n** in the search tree:
  - **ub(n)** – current best solution cost rooted at n
  - **lb(n)** – lower bound on the minimal cost at n

Lower/Upper Bounds



UB(X) = best cost below X (i.e. v(X,0))

LB(X) = LB(X,1)

LB(X,1) = l(X,1) + v(A) + h(C) + LB(B)

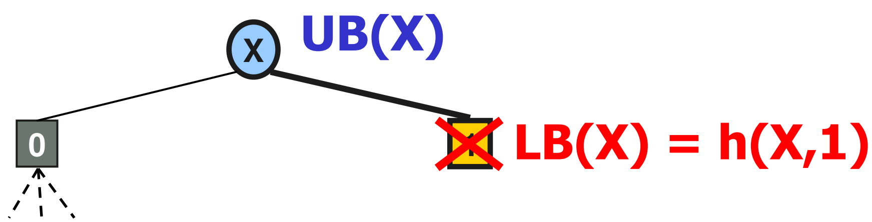
LB(B) = LB(B,0)

LB(B,0) = h(B,0)

Prune below AND node (B,0) if LB(X) ≥ UB(X)

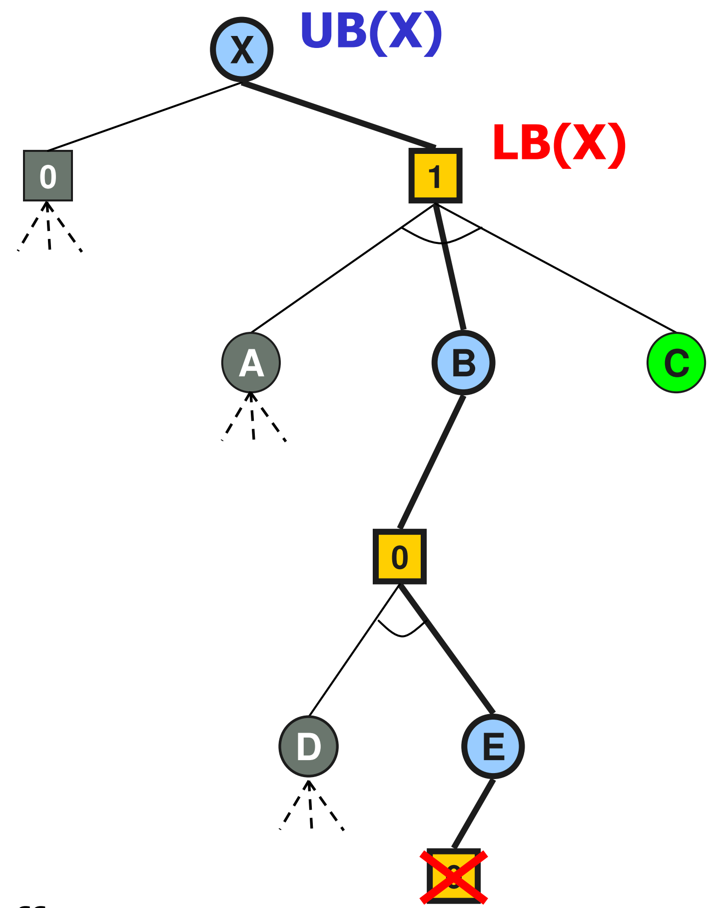
Shallow/Deep Cutoffs

Prune if LB(X) ≥ UB(X)



Shallow cutoff

Reminiscent of **MiniMax** shallow/deep cutoffs



Deep cutoff

Summary of AOBB

- Traverses the AND/OR search tree in a depth-first manner
- Lower bounds computed based on heuristic estimates of nodes at the frontier of search, as well as the values of nodes already explored
- Prunes the search space as soon as an upper-lower bound violation occurs

Heuristics for AND/OR Branch-and-Bound

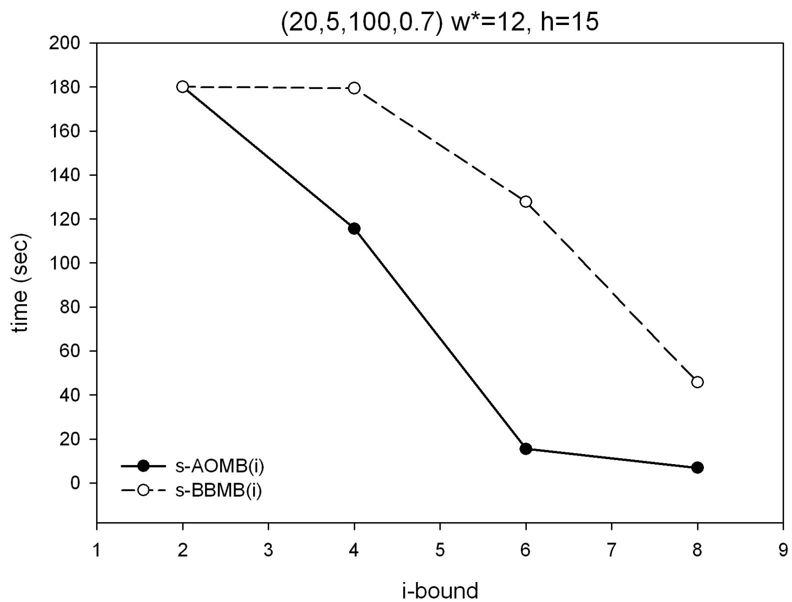
- In the AND/OR search space **h(n)** can be computed using any heuristic. We used:
  - Static Mini-Bucket heuristics
    (Kask & Dechter, AIJ'01), (Marinescu & Dechter, IJCAI'05)
  - Dynamic Mini-Bucket heuristics
    (Marinescu & Dechter, IJCAI'05)
  - Maintaining local consistency
    (Larrosa & Schiex, AAAI'03), (de Givry et al., IJCAI'05)

Empirical Evaluation

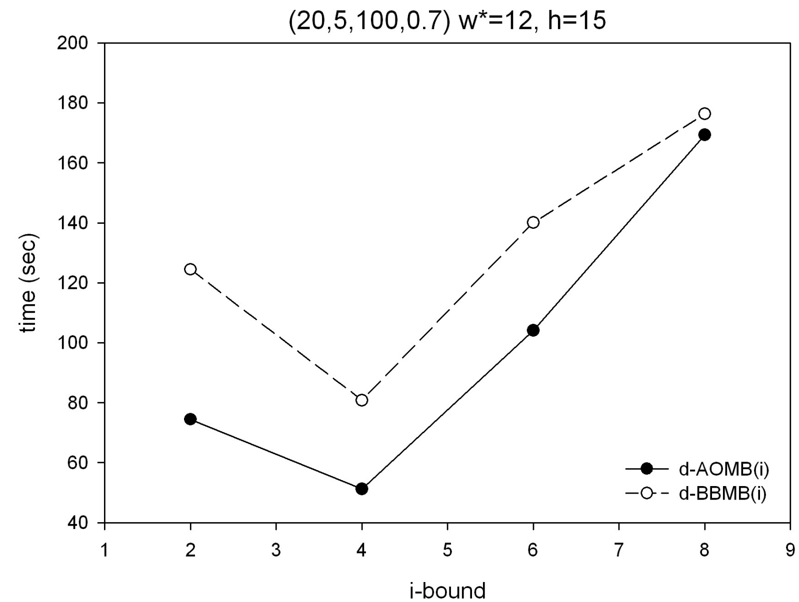
- Tasks
  - Solving Weighted CSPs
  - Finding the MPE in belief networks
- Benchmarks
  - Random binary WCSPs
  - RLFAP networks (CELAR6)
  - Bayesian Networks Repository
- Algorithms
  - AOBB+SMB(i), AOBB+DMB(i), AOBB+FDAC
  - BB+SMB(i), BB+DMB(i), BB+FDAC
  - Static variable ordering (DFS traversal of the pseudo-tree)

Random Binary WCSPs
(Marinescu & Dechter, IJCAI'05)

AOBB+SMB vs. BB+SMB



AOBB+DMB vs. BB+DMB



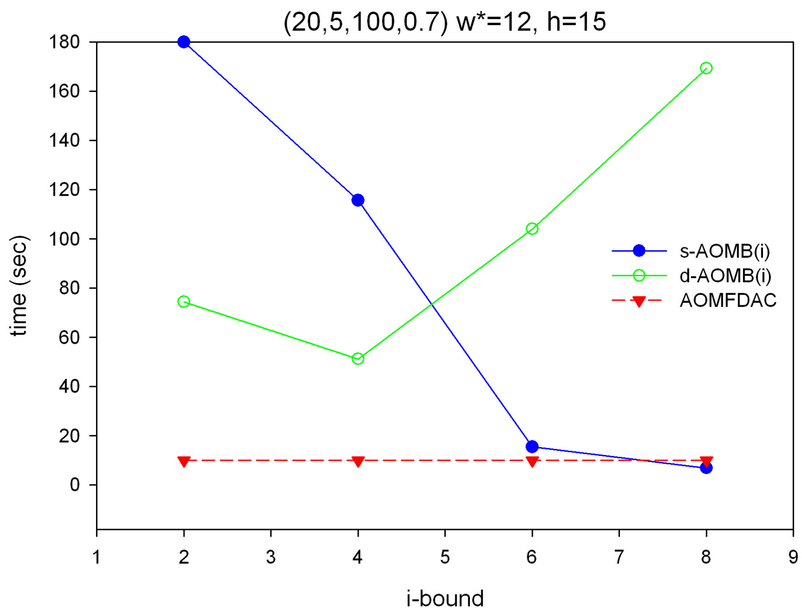
Random networks with n=20 (number of variables), d=5 (domain size), c=100 (number of constraints), t=70% (tightness). Time limit 180 seconds, 20 random instances.

AO search is superior to OR search

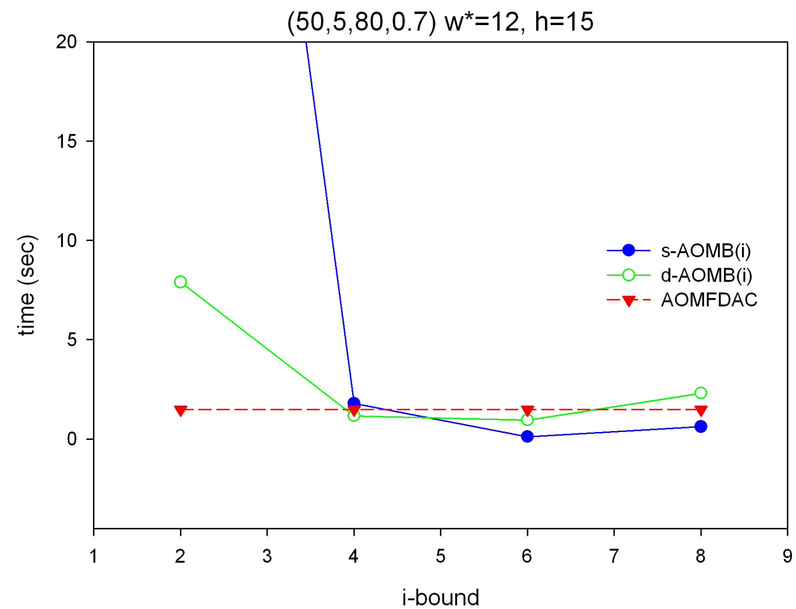
Random Binary WCSPs (contd.)

(Marinescu & Dechter, IJCAI'05)

dense



sparse



n=20 (variables), d=5 (domain size), c=100 (constraints), t=70% (tightness) Time limit 180 seconds, 20 instances.

n=50 (variables), d=5 (domain size), c=80 (constraints), t=70% (tightness) Time limit 180 seconds, 20 instances.

AOBB+SMB for large i is competitive with AOBB+FDAC

Resource Allocation
(Marinescu & Dechter, IJCAI'05)

Radio Link Frequency Assignment Problem (RLFAP)

Instance	BB+FDAC		AOBB+FDAC	
	time (sec)	nodes	time (sec)	nodes
CELAR6-SUB0	2.78	1,871	**1.98**	**435**
CELAR6-SUB1	2,420.93	364,986	**981.98**	**180,784**
CELAR6-SUB2	8,801.12	19,544,182	**1,138.87**	**175,377**
CELAR6-SUB3	38,889.20	91,168,896	**4,028.59**	**846,986**
CELAR6-SUB4	84,478.40	6,955,039	**47,115.40**	**4,643,229**

CELAR6 sub-instances

AOBB+FDAC is superior to BB+FDAC

Bayesian Networks Repository

(Marinescu & Dechter, IJCAI'05)

Network (n,d,w*,h)	Algorithm	i=2 time	i=2 nodes	i=3 time	i=3 nodes	i=4 time	i=4 nodes	i=5 time	i=5 nodes
Barley (48,67,7,17)	**AOBB+SMB(i)**	-	8.5M	-	7.6M	46.22	807K	**0.563**	9.6K
	BB+SMB(i)	-	16M	-	18M	-	17M	-	14M
	AOBB+DMB(i)	-	79K	136.0	23K	12.55	667	45.95	567
	BB+DMB(i)	-	2.2M	-	1M	346.1	76K	-	86K
Munin1 (189,21,11,24)	**AOBB+SMB(i)**	57.36	1.2M	12.08	260K	7.203	172K	**1.657**	43K
	BB+SMB(i)	-	8.5M	-	9M	-	10M	-	8M
	AOBB+DMB(i)	66.56	185K	12.47	8.1K	10.30	1.6K	11.99	523
	BB+DMB(i)	-	405K	-	430K	-	235K	14.63	917
Munin3 (1044,21,7,25)	**AOBB+SMB(i)**	-	5.9M	-	4.9M	1.313	17K	**0.453**	6K
	BB+SMB(i)	-	1.4M	-	1.2M	-	316K	-	1.5M
	AOBB+DMB(i)	-	2.3M	68.64	58K	3.594	5.9K	2.844	3.8K
	BB+DMB(i)	-	33K	-	125K	-	52K	-	31K

Time limit 600 seconds

available at http://www.cs.huji.ac.il/labs/compbio/Repository

AOBB+SMB(i) is better with accurate heuristic (large i)

Outline

- Introduction
  - Optimization tasks for graphical models
  - Solving by inference and search
- Inference
  - Bucket Elimination, Dynamic Programming
  - Mini-Bucket elimination
- Search (OR)
  - Branch-and-Bound and Best-First
  - Lower-bounding heuristics
- **AND/OR search spaces**
  - **Searching the AND/OR tree (linear space)**
    - with mini-bucket heuristics
    - 0/1 integer programming with linear programming relaxations
    - dynamic variable ordering
  - Searching the AND/OR graph (caching)
- Hybrids of search and inference
  - Cutset decomposition
  - Super-bucket scheme
- Software

0/1 Integer Linear Programs
(Marinescu & Dechter, CPAIOR'06)

minimize : $z = c_1 x_1 + c_2 x_2 + \ldots + c_n x_n$

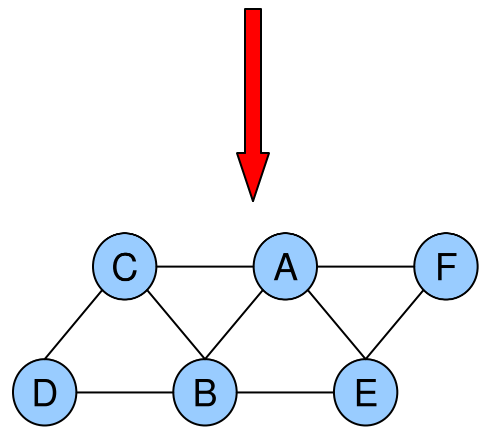
subject to :

$$a_1^1 x_1 + a_2^1 x_2 + \ldots + a_n^1 x_n \le b^1$$
$$a_1^2 x_1 + a_2^2 x_2 + \ldots + a_n^2 x_n \le b^2$$
$$\ldots$$
$$a_1^m x_1 + a_2^m x_2 + \ldots + a_n^m x_n \le b^m$$
$$x_1, x_2, \ldots, x_n \in \{0,1\}$$

minimize: $z = 7A + 3B - 2C + 5D - 6E + 8F$

subject to:

$$3A - 12B + C \le 3$$
$$-2B + 5C - 3D \le -2$$
$$2A + B - 4E \le 2$$
$$A - 3E + F \le 1$$
$$A, B, C, D, E, F \in \{0,1\}$$



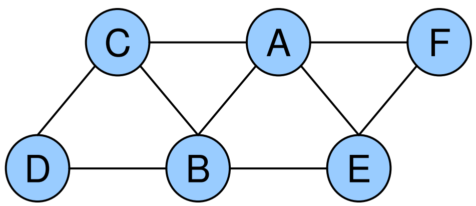
Primal graph

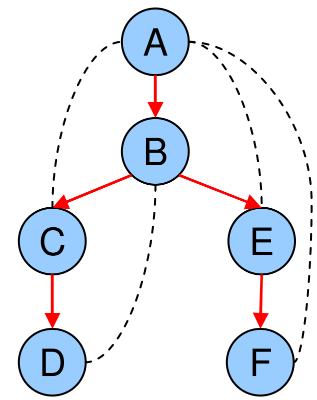
AND/OR Tree Search for 0/1 ILP

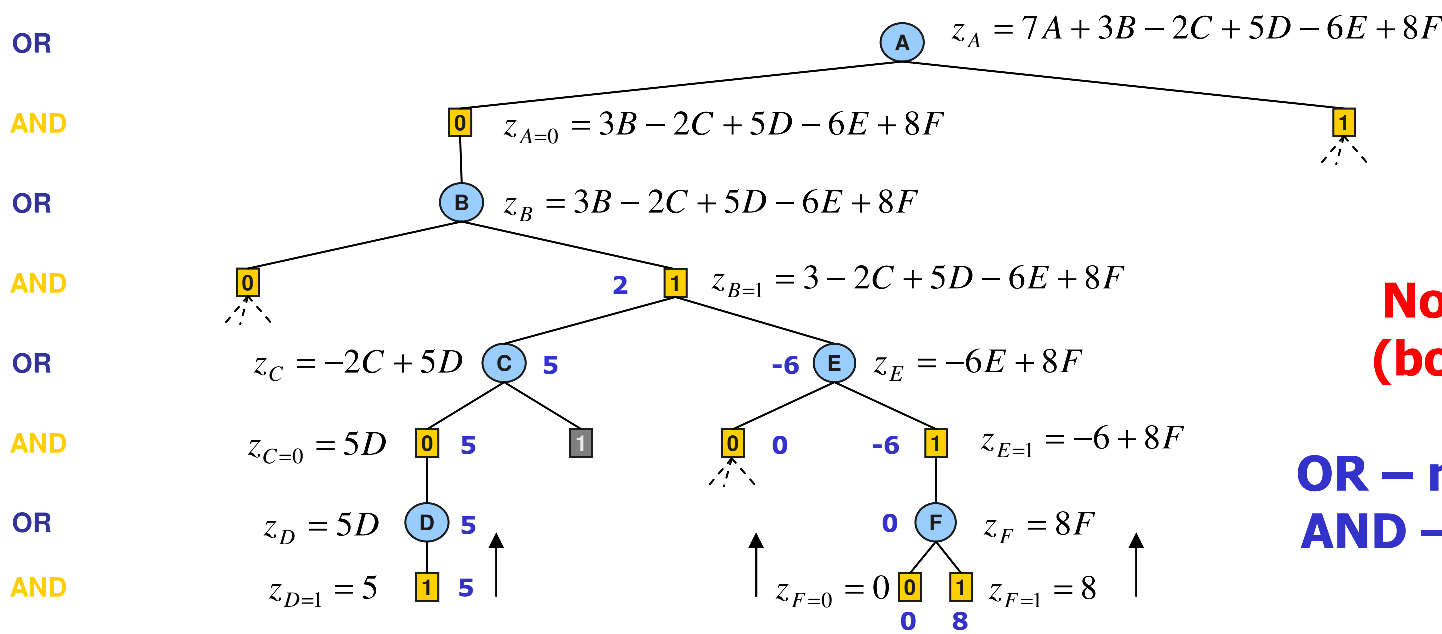
$$\text{minimize: } z = 7A + 3B - 2C + 5D - 6E + 8F$$

subject to:

$$3A - 12B + C \le 3$$
$$-2B + 5C - 3D \le -2$$
$$2A + B - 4E \le 2$$
$$A - 3E + F \le 1$$
$$A, B, C, D, E, F \in \{0,1\}$$







Node Value (bottom up)

OR – minimization
AND – summation

Outline

- **Introduction**
  - Optimization tasks for graphical models
  - Solving by inference and search
- **Inference**
  - Bucket Elimination, Dynamic Programming
  - Mini-Bucket elimination
- **Search (OR)**
  - Branch-and-Bound and Best-First
  - Lower-bounding heuristics
- **AND/OR search spaces**
  - **Searching the AND/OR tree (linear space)**
    - with mini-bucket heuristics
    - 0/1 integer programming with linear programming relaxations
    - dynamic variable ordering
  - Searching the AND/OR graph (caching)
- **Hybrids of search and inference**
  - Cutset decomposition
  - Super-bucket scheme
- **Software**

Dynamic Variable Orderings
(Marinescu & Dechter, ECAI'06)

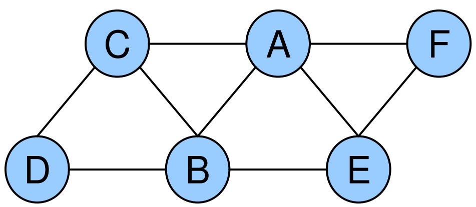
- Variable ordering heuristics:
  - **Semantic-based**
    - Aim at shrinking the size of the search space
      - e.g. min-domain, min-dom/deg, min reduced cost
  - **Graph-based**
    - Aim at maximizing the problem decomposition
      - e.g. pseudo-tree

Orthogonal forces, use one as primary and break ties based on the other

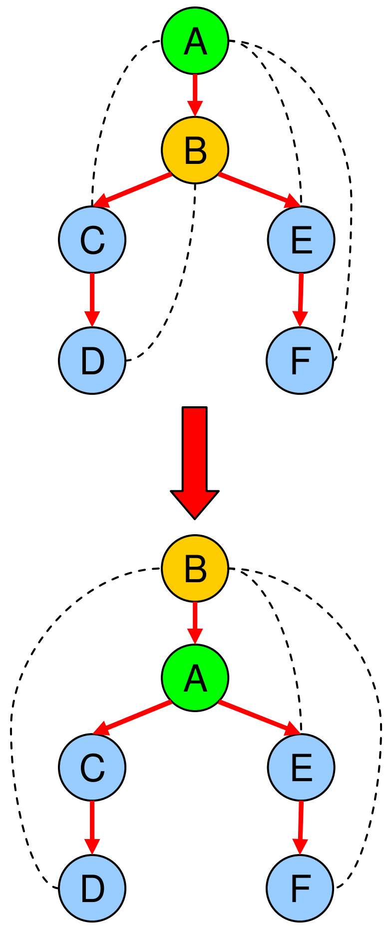
Dynamic AOBB Search

- **Partial Variable Ordering** (AOBB+PVO)
  - Combines the static graph-based decomposition given by a pseudo-tree with a dynamic semantic-based heuristic (giving priority to the first)
- **Dynamic Variable Ordering** (DVO+AOBB)
  - Gives priority to the dynamic semantic-based heuristic and applies problem decomposition as a secondary principle
- **Dynamic Separator Ordering** (AOBB+DSO)
  - Combines a dynamic graph-based decomposition heuristic (separator) with a dynamic semantic variable ordering heuristic (giving priority to the first)

AOBB+PVO Search



Constraint graph



Variable Groups:

- {A,B}
- {C,D}
- {E,F}

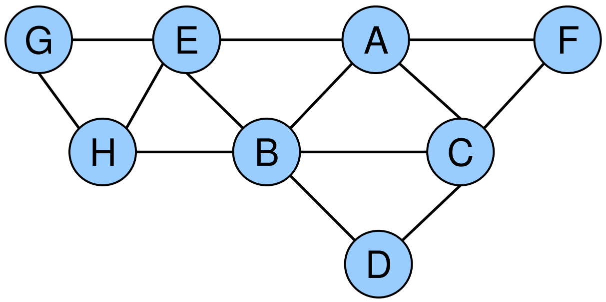
Instantiate {A,B} before {C,D} and {E,F}

*{A,B} is a separator/chain

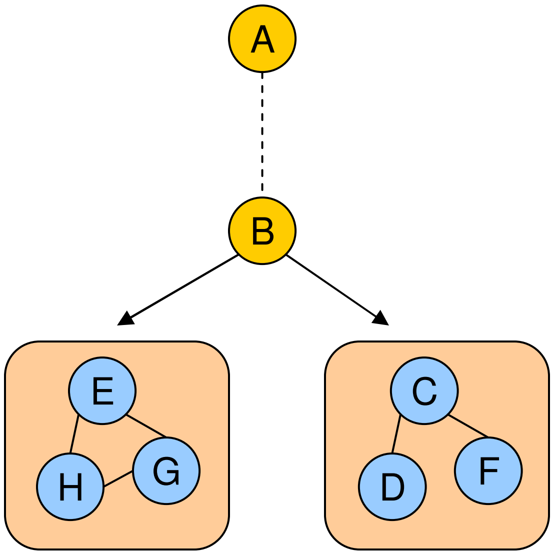
Variables on **chains** in the pseudo-tree can be instantiated dynamically, based on some semantic ordering heuristic

similar idea is exploited by **BTD** (Jegou & Terrioux, 2004)

DVO+AOBB Search



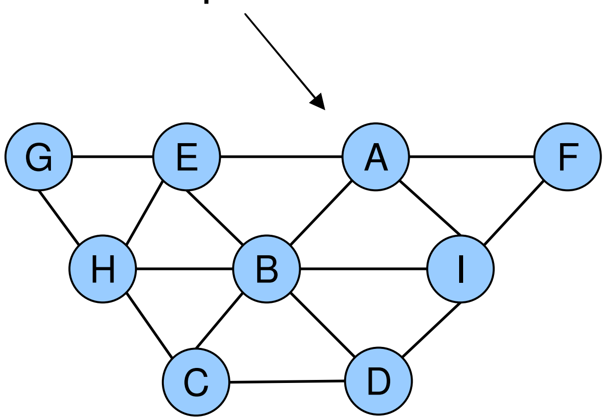
Constraint graph



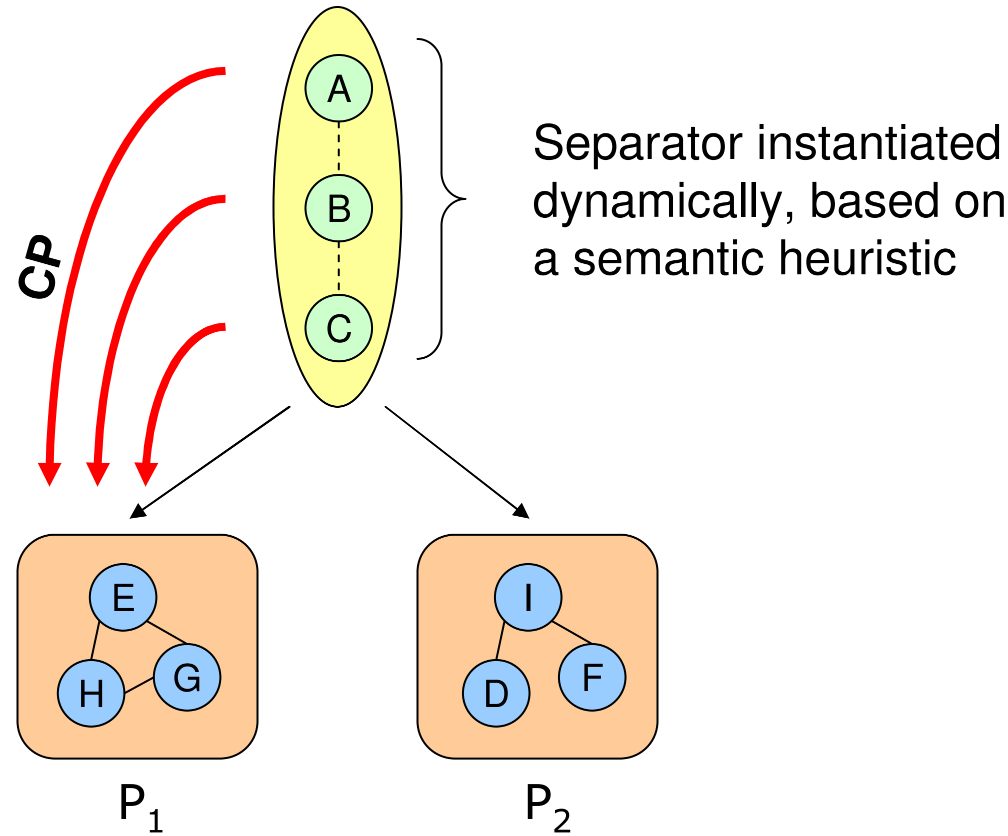
Independent components that are discovered dynamically during search are solved separately and their results combined in an AND/OR manner

[similar idea exploited in #SAT (Bayardo & Pehoushek, 2000)]

AOBB+DSO Search



Constraint graph



Constraint Propagation may create **singleton** variables in **P1** and **P2** (changing the problem's structure), which in turn may yield smaller separators

[also in SAT (Li & van Beek, 2004)]

Summary of Dynamic AOBB Search

- Traverses the AND/OR search tree in a depth-first manner, using dynamic variable ordering heuristics
- Lower bounds computed based on heuristic estimates of nodes at the frontier of search, as well as the values of nodes already explored
- Prunes the search space as soon as an upper-lower bound violation occurs

Experiments

- **Algorithms**
  - AOBB+SVO (static variable ordering)
  - AOBB+PVO (partial variable ordering)
  - AOBB+DVO (dynamic variable ordering)

- **Benchmarks**
  - MIPLIB library
  - Combinatorial Auctions from CATS 2.0 suite
  - Uncapacitated Warehouse-Location Problems

- **Implementation issues**
  - SIMPLEX solver from `lp_solve 5.5` public library
  - Semantic variable selection heuristic based on reduced-costs

0/1 ILP Benchmarks

- MIPLIB
  - Public library of MILP instances commonly used for benchmarking IP algorithms (only pure 0/1 instances)
- Combinatorial Auctions
  - Random combinatorial auctions with **b** bids on **g** goods drawn from the CATS 2.0 and simulating radio spectrum allocation
- Uncapacitated Warehouse Location Problems
  - Random resource allocation problems which consider opening **m** warehouses to supply **n** stores such that the maintenance/supply costs are minimized

MIPLIB Benchmarks
(Marinescu & Dechter, CPAIOR'06)

miplib	n	h	BB		AOBB					
					SVO		PVO		DVO	
			time	nodes	time	nodes	time	nodes	time	nodes
p0033	33	20	6.53	18.1K	0.59	1.9K	**0.39**	1.1K	3.39	9.3K
p0201	201	142	37.4	15.6K	57.9	25.3K	**22.9**	8.9K	42.5	14.5K
lseu	89	69	154	369K	39.7	87.5K	**38.9**	86.1K	153	337K

Results for MIPLIB problem instances

Combinatorial Auctions
(Marinescu & Dechter, CPAIOR'06)

auction	h	BB		AOBB					
				SVO		PVO		DVO	
		time	nodes	time	nodes	time	nodes	time	nodes
u-b200-g50	162	1.71	602	3.28	938	**2.98**	888	1.97	602
u-b250-g75	190	16.3	3.5K	7.32	1.2K	**6.30**	1.1K	18.4	3.5K
u-b300-g100	204	63.3	7.9K	52.8	4.9K	**45.6**	4.8K	63.7	7.9K

Results for CATS 2.0 combinatorial auctions. Time limit 1 hour.

Uncapacitated Warehouse Location Problems

(Marinescu & Dechter, CPAIOR'06)

uwlp	h	BB		AOBB					
				SVO		PVO		DVO	
		time	nodes	time	nodes	time	nodes	time	nodes
50-200-b	123	11.3	53	17.2	60	**5.78**	12	11.7	53
50-200-c	123	73.4	469	15.8	58	**5.83**	10	77.9	469
50-200-d	123	837	4.3K	27.9	116	**11.9**	26	904	4.3K
50-200-e	123	2,502	11.9K	32.7	80	**16.9**	28	2,990	12.7K

Results for UWLP instances. Time limit 1 hour.

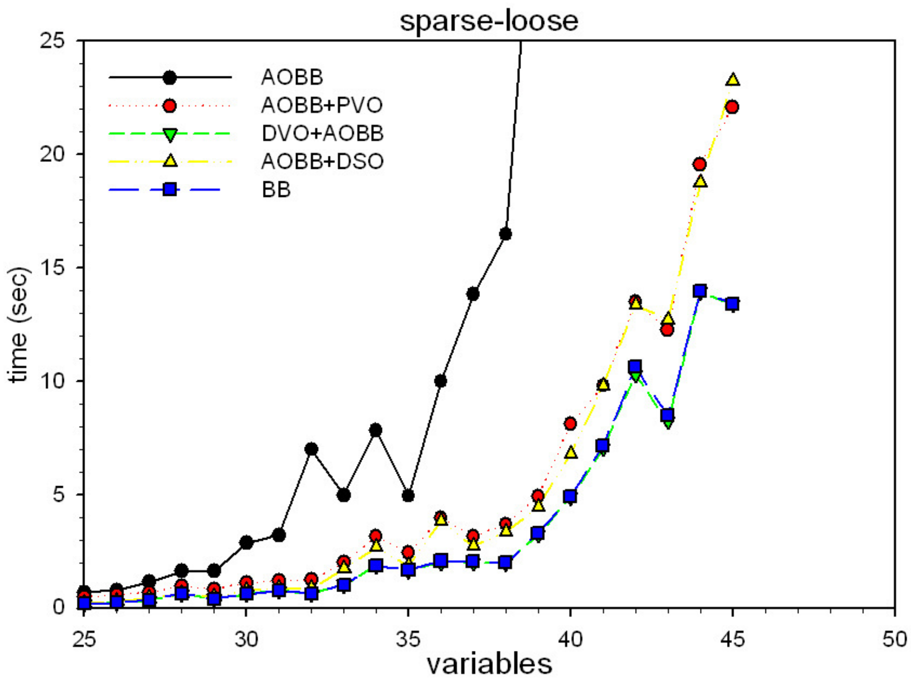
Experiments

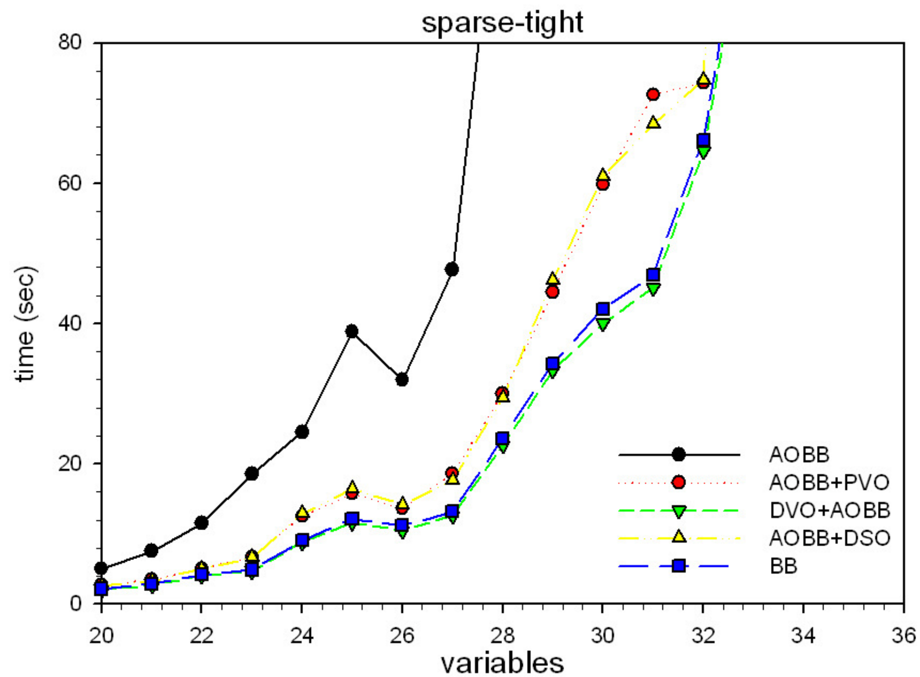
- **Algorithms**
  - AOBB (static variable ordering)
  - AOBB+PVO (partial variable ordering)
  - DVO+AOBB (dynamic variable ordering)
  - AOBB+DSO (dynamic separator ordering)
  - BB (classic OR Branch-and-Bound)

- **Lower Bound Heuristics**
  - EDAC (for WCSP) (de Givry et al., 2005)
  - SIMPLEX (for 0/1 ILP)

- **Benchmarks**
  - Random Binary WCSP
  - SPOT5 (Earth Observing Satellites)
  - RLFAP (Radio Link Frequency Allocation)
  - Uncapacitated Warehouse-Location Problems (0/1 ILP)

Sparse Binary Random WCSPs
(Marinescu & Dechter, ECAI'06)

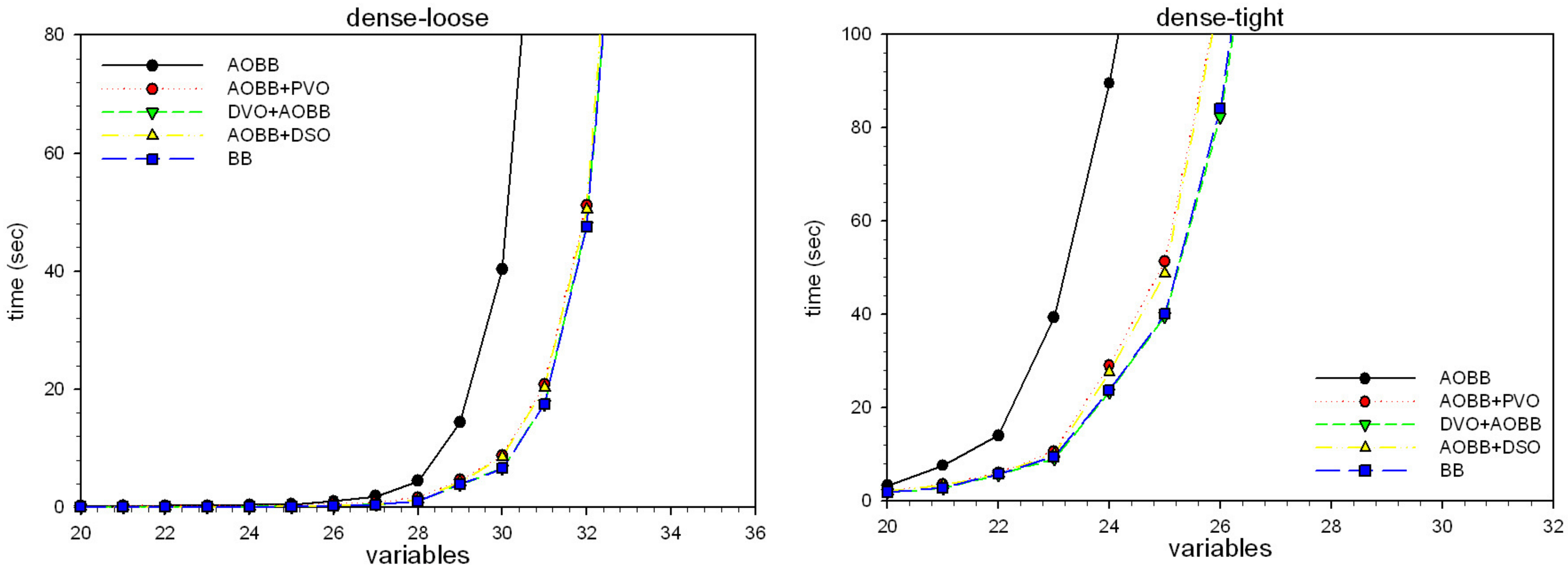




Results for sparse random WCSP. Time limit 3 minutes, 20 instances.

Dense Binary Random WCSPs

(Marinescu & Dechter, ECAI'06)



Results for dense random WCSP. Time limit 3 minutes, 20 instances.

Earth Observing Satellites (SPOT5)

(Marinescu & Dechter, ECAI'06)

spot	w	h	BB		AOBB		AOBB+ PVO		DVO+ AOBB		AOBB+ DSO	
			time	nodes	time	nodes	time	nodes	time	nodes	time	nodes
29	15	22	0.41	2.5K	1.91	3.7K	1.93	3.8K	**0.33**	2.7K	0.68	2.7K
42	38	46	-	-	-	-	-	-	**4,709**	14M	-	-
54	12	16	0.06	543	1.13	2.7K	1.07	1.1K	**0.03**	312	0.08	423
503	11	21	11.7	18K	2.78	1.2K	2.78	1.3K	**0.03**	217	0.13	419
505	27	42	4,010	1.9M	75.4	0.4M	75.4	0.4M	**17.3**	34K	82.7	87K

Results for SPOT5 networks. Time limit 10 hours.

* we solved a simplified Max-CSP (i.e. minimize the number of hard constraint violations)

Radio Link Frequency Assignment
(Marinescu & Dechter, ECAI'06)

celar6	w	h	BB		AOBB		AOBB+ PVO		DVO+ AOBB		AOBB+ DSO	
			time	nodes	time	Nodes	time	nodes	time	nodes	time	nodes
sub0	7	8	0.88	2.7K	0.73	2.6K	0.81	2.3K	0.92	2.8K	**0.71**	3K
sub1	9	9	2,260	4.5M	3,101	4.3M	2,200	3.4M	**2,182**	3.4M	2,246	3.4M
sub1-24	9	9	136	0.7M	171	0.5M	128	0.4M	**127**	0.4M	132	0.4M
sub2	10	12	4,696	19M	10,963	18M	7,047	10M	**4,024**	10M	6,696	9.2M
sub3	10	13	14,687	63M	32,439	39M	28,252	32M	**11,131**	28M	28,407	32M
sub4-20	11	15	681	7.9M	137	0.4M	157	0.7M	**70**	0.2M	179	1M

Results for CELAR6 networks. Time limit 24 hours.

Uncapacitated Warehouse Location Problems
(Marinescu & Dechter, ECAI'06)

uwlp	w	h	BB		AOBB		AOBB+ PVO		DVO+ AOBB	
			time	nodes	time	nodes	time	nodes	time	nodes
50-200-a	50	123	**6.27**	27	15.7	70	6.28	12	7.23	27
50-200-b	50	123	11.3	53	17.2	60	**5.78**	12	11.7	53
50-200-c	50	123	73.4	469	15.8	58	**5.83**	10	77.9	469
50-200-d	50	123	837	4.3K	27.9	116	**11.9**	26	904	4.3K
50-200-e	50	123	2,502	11.9K	32.7	80	**16.9**	28	2,990	12.7K

Results for UWLP networks (10,500 variables). Time limit 1 hour.

Outline

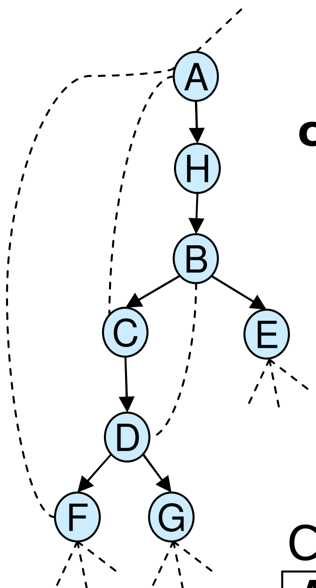
- Introduction
  - Optimization tasks for graphical models
  - Solving by inference and search
- Inference
  - Bucket Elimination, Dynamic Programming
  - Mini-Bucket elimination
- Search (OR)
  - Branch-and-Bound and Best-First
  - Lower-bounding heuristics
- **AND/OR search spaces**
  - Searching the AND/OR tree (linear space)
  - **Searching the AND/OR graph (caching)**
    - **Depth-First AND/OR Branch-and-Bound Search**
    - Best-First AND/OR Search
- Hybrids of search and inference
  - Cutset decomposition
  - Super-bucket scheme
- Software

Graph AND/OR Branch-and-Bound - AOBB(j)

(Marinescu & Dechter, AAAI'06)

- Associate each node **n** with a static heuristic estimate h(n) of v(n)
  - h(n) is a lower bound on the value v(n)
- For every node **n** in the search tree:
  - **ub(n)** – current best solution cost rooted at n
  - **lb(n)** – lower bound on the minimal cost at n
- During search, cache nodes based on context
  - maintain cache tables of size **O(exp(j))**, where **j** is a bound on the size of the context.

Full Context-based Caching

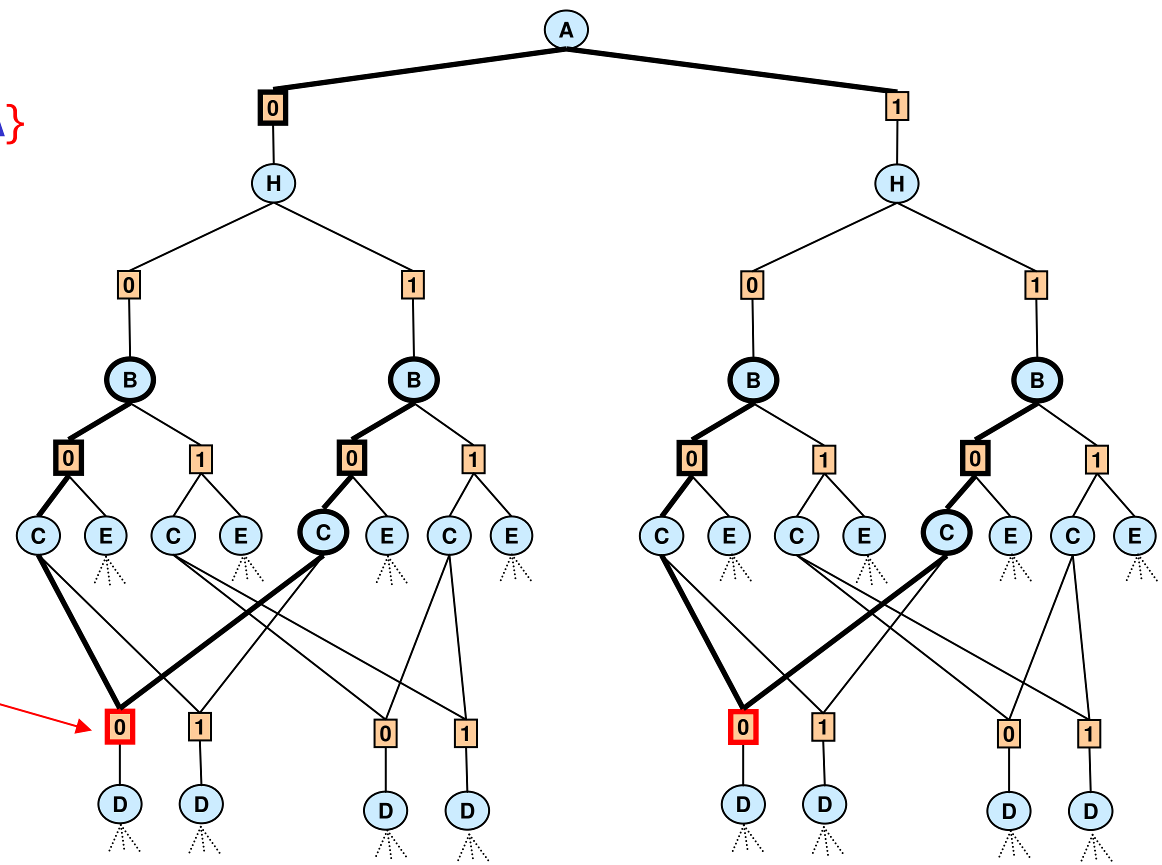


context(C) = {**C,B,A**}

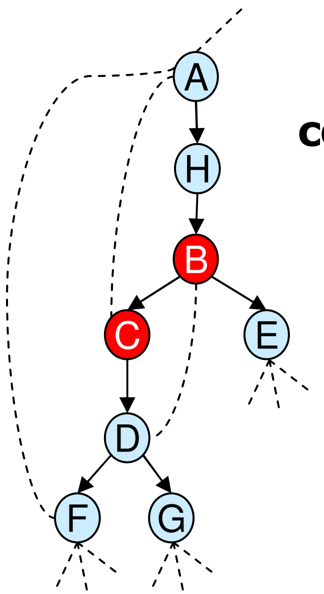
Cache Table (C)

A	B	C	Value
0	0	0	**5**
0	0	1	**2**
0	1	0	**2**
0	1	1	**0**
…			

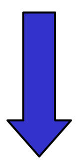
Space: **O(exp(3))**



Adaptive Caching



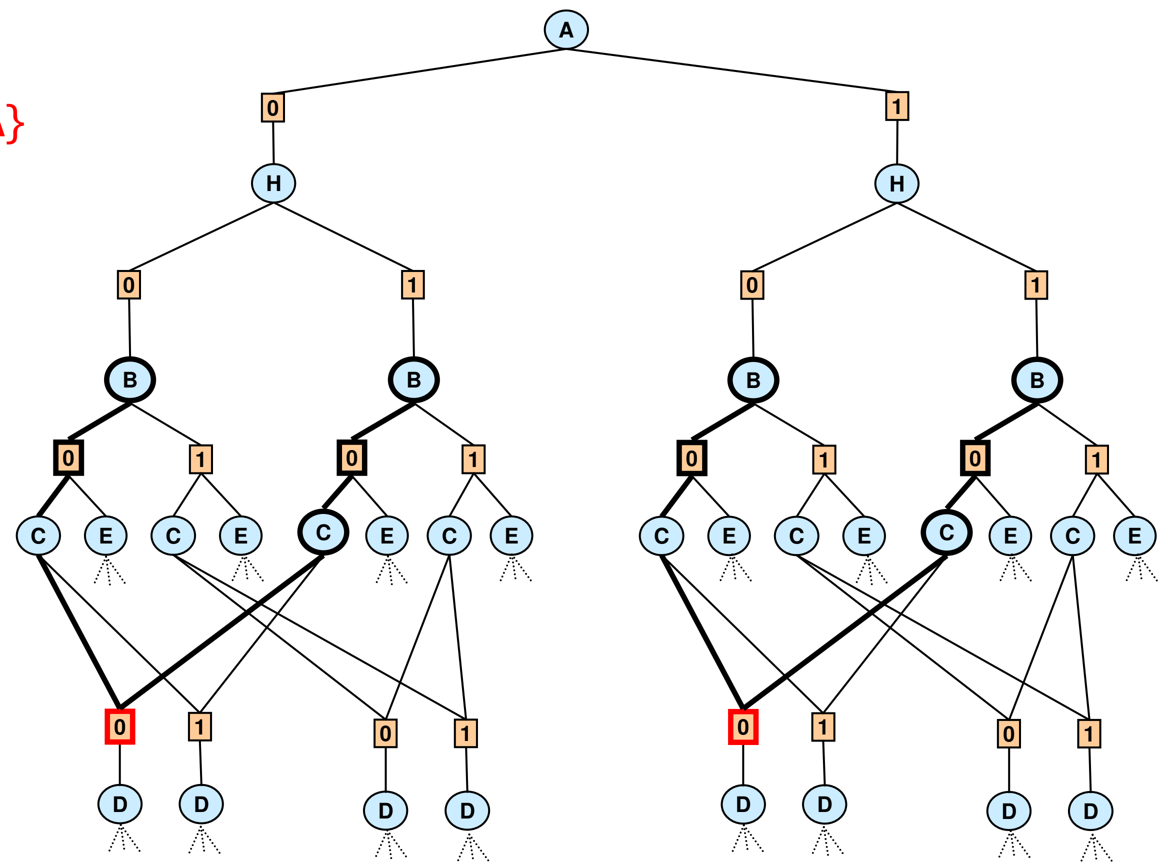
context(C) = {C,B,A}



Cache table on **[C,B]** only!

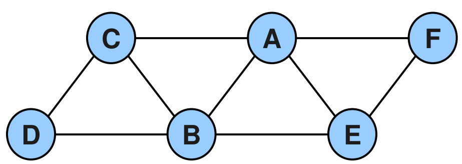
B	C	Value
0	0	5
0	1	2
1	0	2
1	1	0

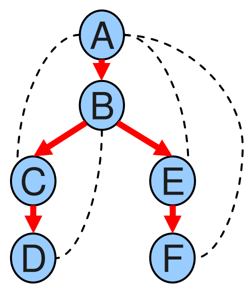
Space: **O(exp(2))**



Reset cache table when A changes its value!

Example (graph search)





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

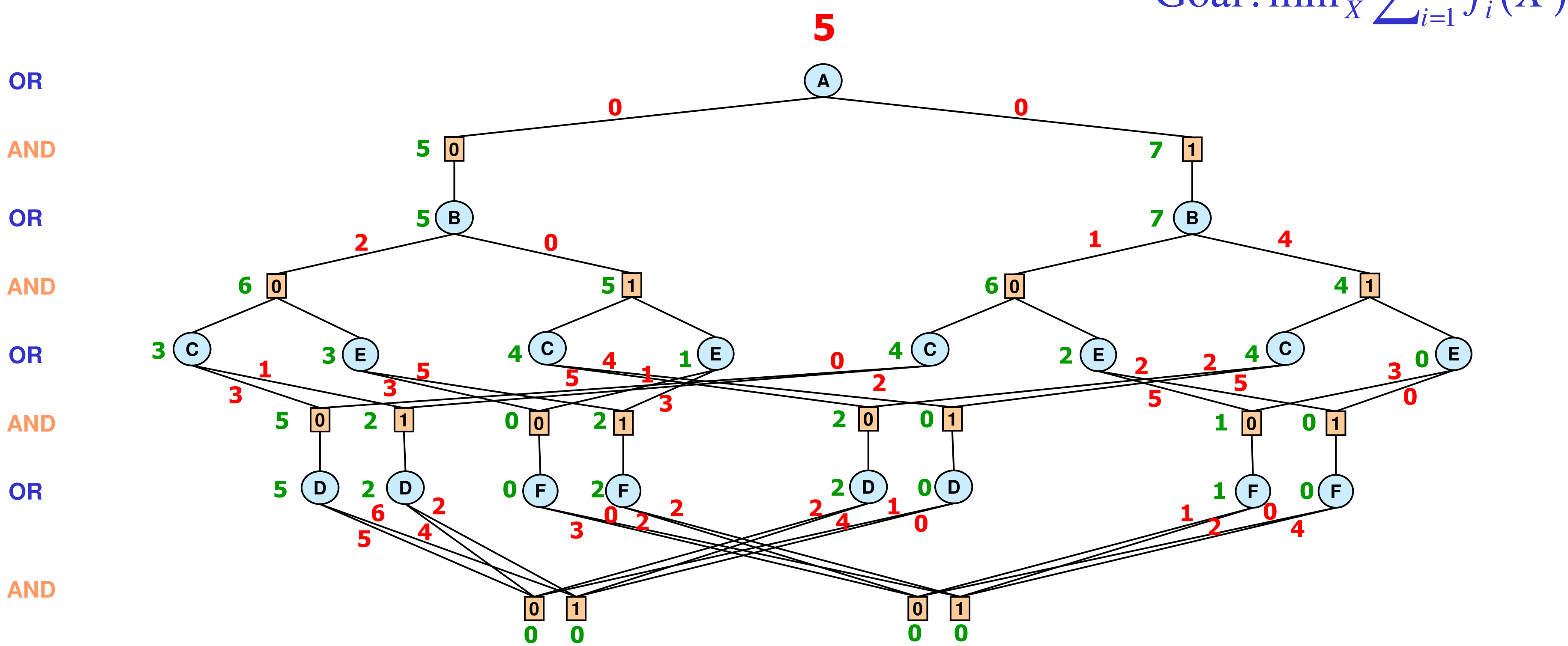
B	D	f_6
0	0	4
0	1	2
1	0	1
1	1	0

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

$$\text{Goal}: \min_X \sum_{i=1}^{9} f_i(X)$$



Heuristics

- In the AND/OR search space **h(n)** can be computed using:
  - Static Mini-Bucket heuristics (SMB)
    (Kask & Dechter, AIJ'01), (Marinescu & Dechter, IJCAI'05)
  - Dynamic Mini-Bucket heuristics (DMB)
    (Marinescu & Dechter, IJCAI'05)

Empirical Evaluation

- **Tasks**
  - Solving WCSPs
  - Finding the MPE in belief networks

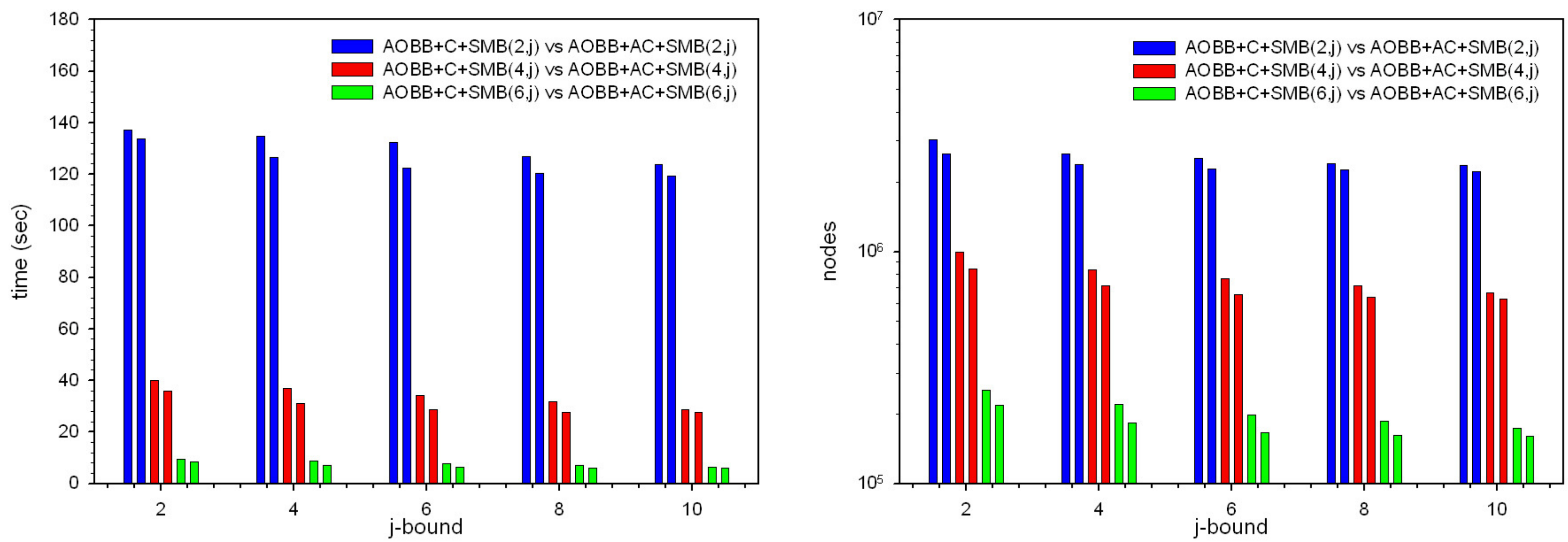
- **Benchmarks**
  - Random networks
  - Resource allocation (SPOT5)
  - Genetic linkage analysis
  - ISCAS'89 Benchmark circuits

- **Algorithms**
  - AOBB+C+SMB(i,j), AOBB+C+DMB(i,j)
    - i.e., context-based caching
  - AOBB+AC+SMB(i,j), AOBB+AC+DMB(i,j)
    - i.e., adaptive caching
  - j is the cache bound
  - Static variable ordering

Random Belief Networks (MPE)

(Marinescu & Dechter, AAAI'06)

Naïve versus Adaptive Caching

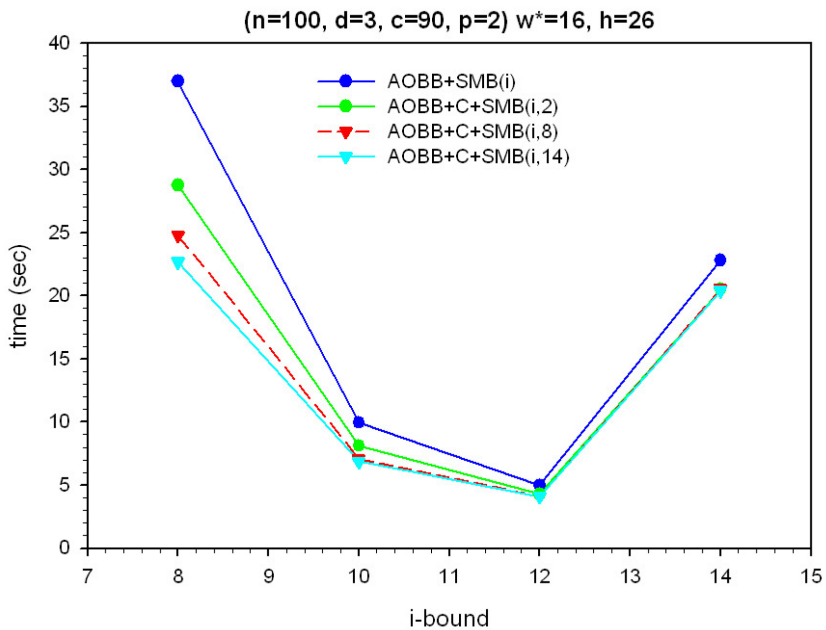


Random Belief Networks with 120 nodes, domain size 2, 2 parents per CPT and 10 root nodes. Time limit 180 seconds. Average induced width w*=22, average pseudo-tree depth h= 32. Each data point is an average over 20 random samples.

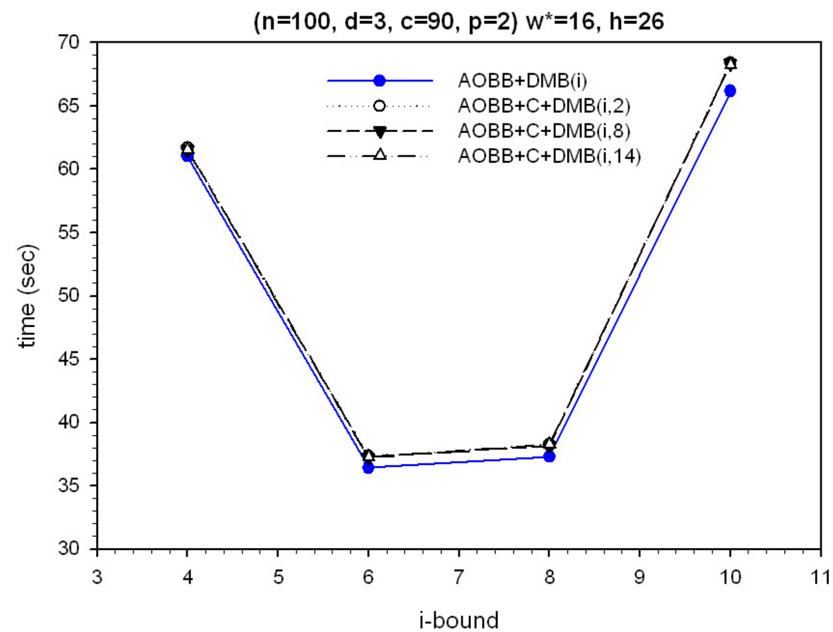
Random Belief Networks (MPE)

(Marinescu & Dechter, AAAI'06)

Static Mini-Bucket Heuristics



Dynamic Mini-Bucket Heuristics



Caching helps for static heuristics with small i-bounds

Random belief networks with n=100 (number of variables), d=3 (domain size), c=90 (number of CPTs), p=2 (number of parents per CPT). Time limit 180 seconds (each data point is an average over 20 random samples).

SPOT5 Benchmarks (WCSP)

(Marinescu & Dechter, AAAI'06)

spot5 (n, c, w*, h)	AOBB+SMB(i) AOBB+C+SMB(i) i=4		AOBB+SMB(i) AOBB+C+SMB(i) i=6		AOBB+SMB(i) AOBB+C+SMB(i) i=8		AOBB+SMB(i) AOBB+C+SMB(i) i=10		AOBB+SMB(i) AOBB+C+SMB(i) i=12		toolbar AOEDAC+DVO	
	time	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time	nodes
29	8.77	86,058	5.05	45,509	0.66	2,738	3.70	1,405	22.02	246	4.56	218,846
(83, 476, 14, 42)	5.53	48,995	3.66	29,702	**0.56**	2,267	3.64	1,165	21.67	110	0.81	8,698
42b	-	-	-	-	1842.32	9,606,846	587.83	3,371,806	134.39	689,402	-	-
(191, 1341, 18, 62)	-	-	-	-	1804.76	9,410,729	**553.47**	3,191,205	116.98	584,838	6825.40	27,698,614
54	113.19	1,106,598	1.59	17,757	0.39	3,616	0.69	329	1.27	329	0.31	21,939
(68, 283, 11, 33)	18.42	198,712	0.23	2,477	0.16	591	0.69	120	1.25	120	**0.06**	688
404	430.99	3,969,398	151.99	1,373,846	14.83	144,535	2.78	23,557	1.44	3,273	151.11	6,215,135
(100, 710, 19, 42)	174.09	1,396,321	51.88	529,002	2.55	23,565	**0.55**	1,704	1.16	598	12.09	88,079
408b	-	-	-	-	-	-	-	-	715.35	4,784,407	-	-
(201, 1847, 24, 59)	-	-	-	-	7507.10	54,826,929	515.94	3,114,294	**75.08**	408,619	-	-
503	-	-	435.26	5,102,299	421.10	4,990,898	0.44	641	0.44	641	-	-
(144, 639, 9, 39)	-	-	189.39	2,442,998	291.72	4,050,474	**0.42**	256	**0.42**	256	10005.00	44,495,545
505b	-	-	-	-	-	-	-	-	-	-	-	-
(240, 1721, 16, 98)	-	-	-	-	-	-	-	-	**1180.48**	8,905,473	-	-

Results for SPOT5 benchmarks. Time limit 3 hours.

ISCAS'89 Circuits (WCSP)

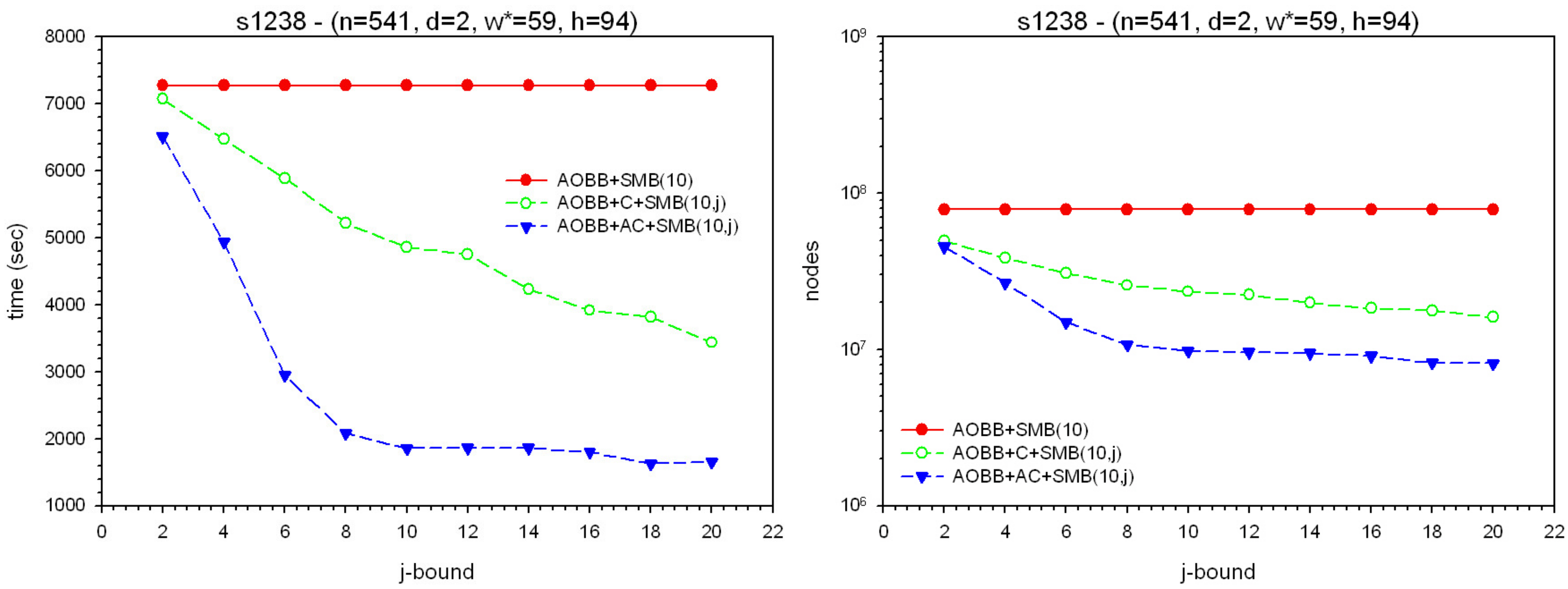
(Marinescu & Dechter, AAAI'06)

iscas (n, d, w*, h)	AOBB+SMB(i) AOBB+C+SMB(i) i=6		AOBB+SMB(i) AOBB+C+SMB(i) i=8		AOBB+SMB(i) AOBB+C+SMB(i) i=10		AOBB+SMB(i) AOBB+C+SMB(i) i=12		AOBB+SMB(i) AOBB+C+SMB(i) i=14		AOBB+SMB(i) AOBB+C+SMB(i) i=16	
	time	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time	nodes
c432	-	-	2010.53	23,355,897	148.39	1,713,265	5.94	76,346	5.84	75,420	0.70	1,958
(432, 2, 27, 45)	-	-	422.08	2,945,230	40.91	337,574	0.89	6,254	0.89	6,010	**0.64**	914
c880	3533.28	39,448,762	1698.08	19,992,512	1316.73	15,247,946	505.75	5,835,825	1134.61	13,568,696	245.06	2,837,010
(881, 2, 27, 67)	488.05	1,936,422	100.66	516,056	91.66	446,893	31.06	169,138	59.35	316,124	**14.78**	78,268
s935	-	-	2559.30	21,438,706	342.80	3,074,516	-	-	41.34	348,699	7.86	51,441
(441, 2, 66, 101)	-	-	1285.07	6,623,608	143.53	763,933	-	-	22.28	128,372	**4.80**	15,010
s1196	-	-	-	-	1347.95	12,392,442	-	-	1949.37	15,775,180	384.20	3,318,953
(562, 2, 54, 101)	-	-	3347.38	13,554,137	503.30	2,425,152	2299.72	11,488,366	734.66	3,524,780	**149.81**	793,417
s1238	3335.01	32,501,292	-	-	-	-	1722.53	18,302,873	1394.86	14,213,319	38.08	360,788
(541, 2, 59, 94)	1219.65	5,336,572	1897.37	8,386,634	1682.99	7,431,223	281.05	1,350,933	248.27	1,220,658	**12.64**	59,635
s1494	261.82	1,758,093	954.37	6,267,644	17.70	155,334	70.30	484,010	15.49	109,016	18.47	114,355
(661, 2, 48, 69)	120.94	334,047	364.80	953,945	**5.64**	17,279	27.64	80,895	6.92	23,131	9.02	20,004

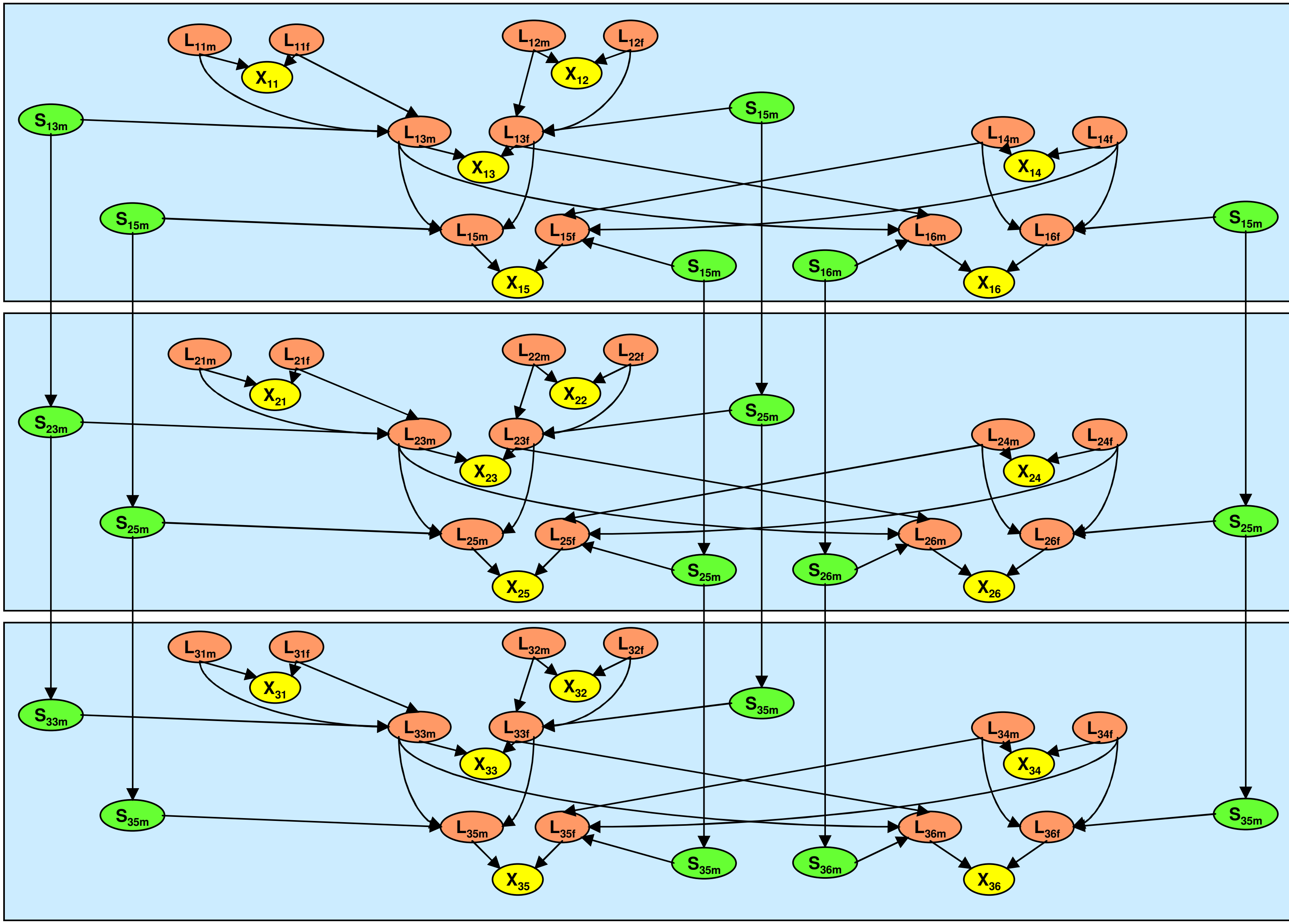
Results for ISCAS'89 benchmarks. Time limit 1 hours.

s1238 Circuit (WCSP)

Naïve versus Adaptive Caching



Pedigree: 6 people, 3 markers



Genetic Linkage Analysis (BN)

(Marinescu & Dechter, AAAI'06)

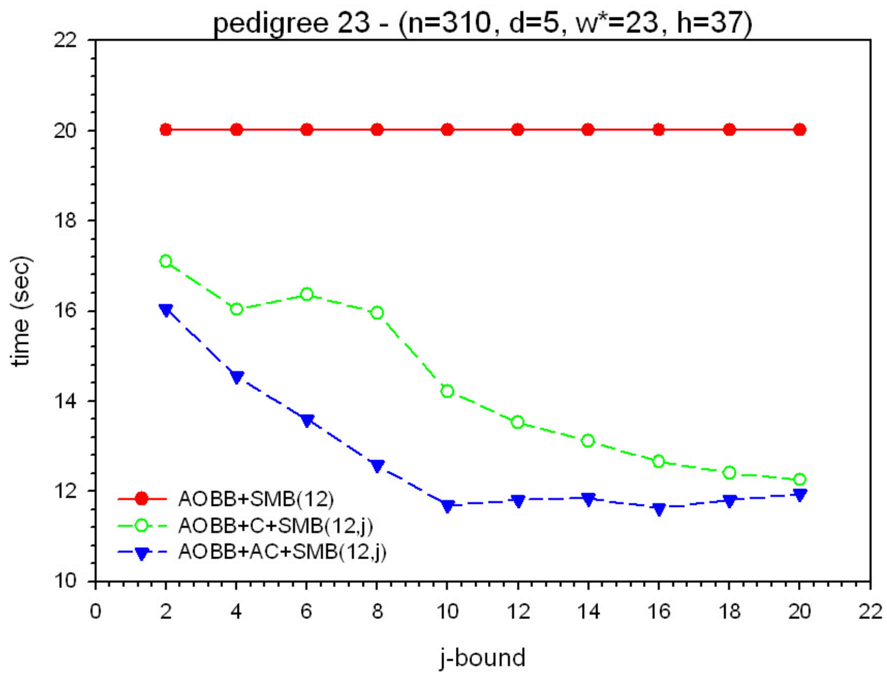
ped (n,d,w*,h)	SamIam v. 2.3.2	Superlink v. 1.6	AOBB+SMB(i) AOBB+C+SMB(i) i=12 time	nodes	AOBB+SMB(i) AOBB+C+SMB(i) i=14 time	nodes	AOBB+SMB(i) AOBB+C+SMB(i) i=16 time	nodes	AOBB+SMB(i) AOBB+C+SMB(i) i=18 time	nodes	AOBB+SMB(i) AOBB+C+SMB(i) i=20 time	nodes
ped18	157.05	139.06	-	-	2177.81	28,651,103	270.96	2,555,078	100.61	682,175	**20.27**	7,689
(1184,5,21,119)			-	-	406.88	3,567,729	52.91	397,934	23.83	118,869	20.60	2,972
ped20	out	**14.72**	-	-	-	-	38.75	311,385	96.02	555,872		
(388,5,23,42)			7243.43	63,530,037	5560.63	46,858,127	**37.28**	279,804	95.13	554,623		
ped30	out	13095.83	5563.22	63,068,960	1397.14	15,336,772	1811.34	20,275,620	550.57	5,535,261	82.25	588,558
(1016,5,25,51)			1440.26	11,694,534	597.88	5,580,555	1023.90	10,458,174	151.96	1,179,236	**43.83**	146,896
ped39	out	322.14	-	-	-	-	4041.56	52,804,044	386.13	2,171,470	141.23	407,280
(1272,5,23,94)			-		-		968.03	7,880,928	**61.20**	313,496	93.19	83,714
ped42	out	**561.31**	-	-	-	-	-	-				
(448,5,25,76)			-		-		2364.67	22,595,247				
ped25	out	-	-	-	-	-	-	-	8415.18	45,825,494	1894.17	11,709,153
(994,5,29,53)			-		-		-		2041.64	6,117,320	**693.74**	1,925,152
ped33	out	-	2335.28	32,444,818	806.12	11,403,812	62.91	807,071	67.92	701,030	76.47	320,279
(581,5,26,48)			886.05	8,426,659	370.41	4,032,864	**26.31**	229,856	33.11	219,047	54.89	83,360

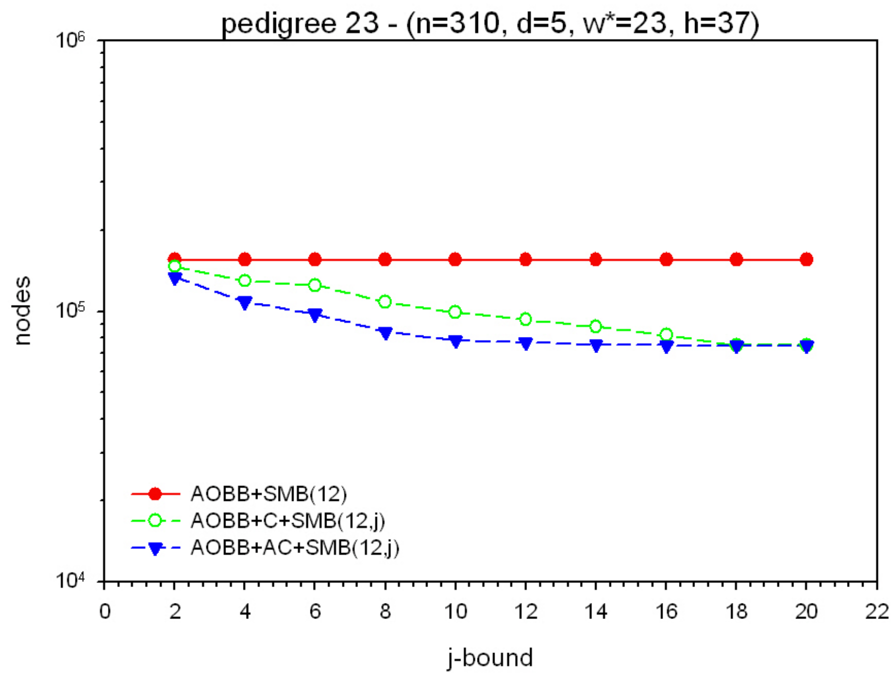
Results for pedigree benchmarks. Time limit 3 hours.

Superlink: Variable Elimination + Conditioning hybrid
Exploits determinism in the network (Fishelson & Geiger, 2005)
SamIam: Recursive Conditioning (Darwiche, 2001)

Pedigree 23

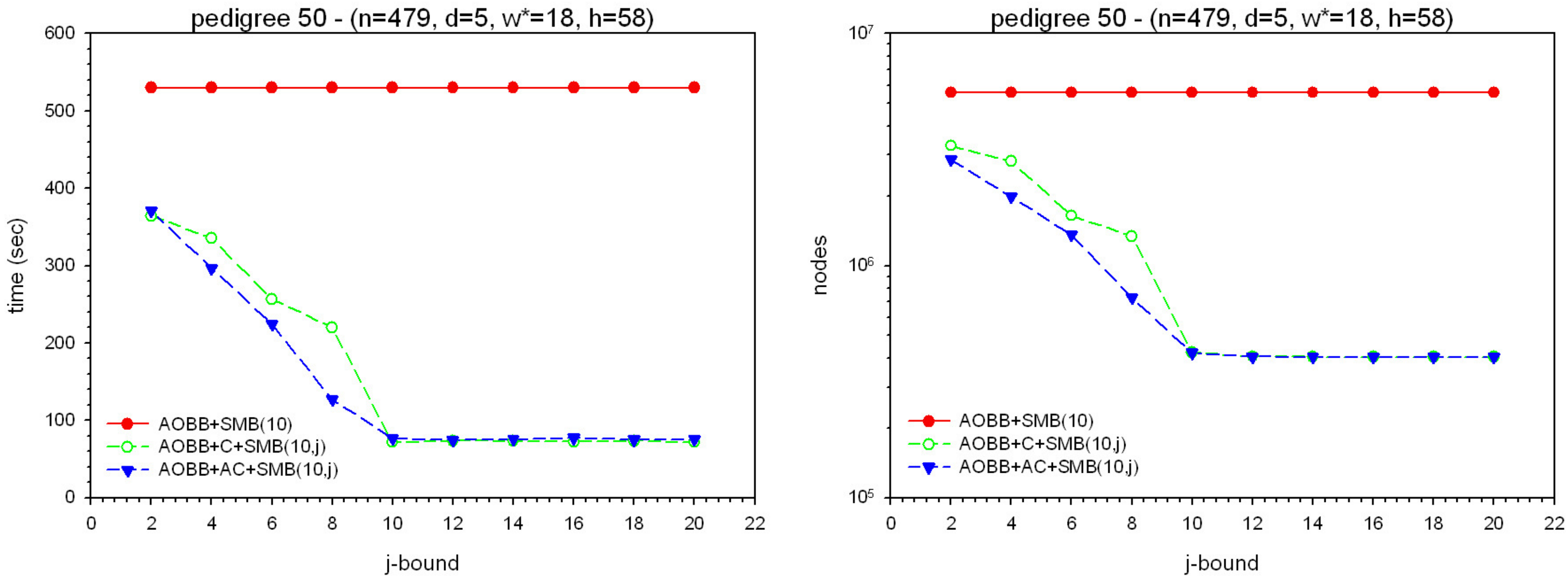
Naïve versus Adaptive Caching





Pedigree 50

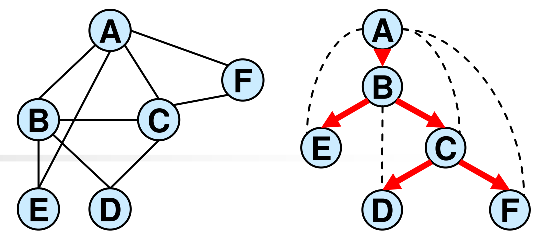
Naïve versus Adaptive Caching

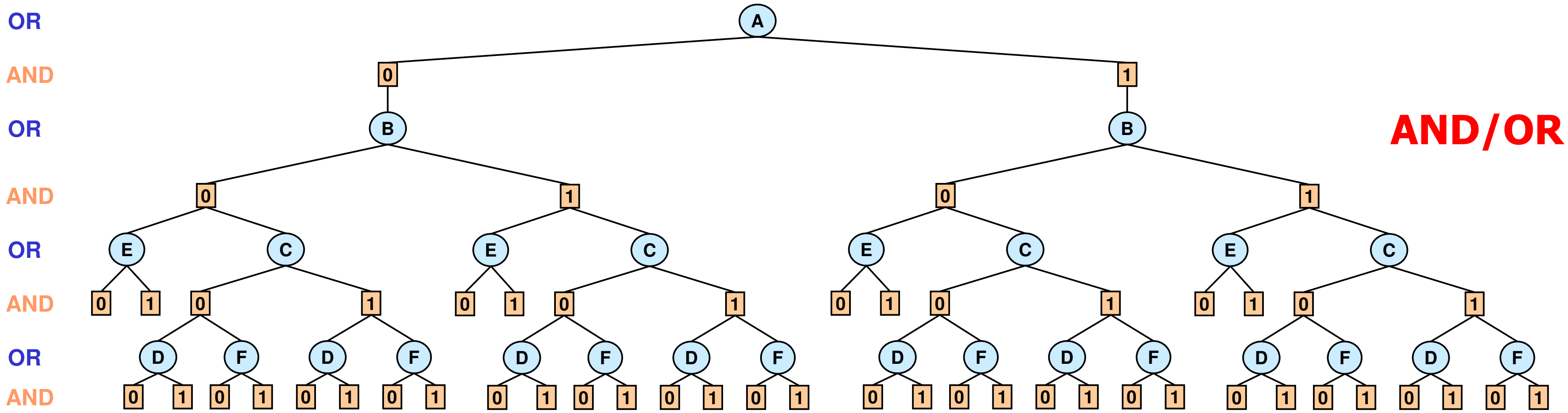


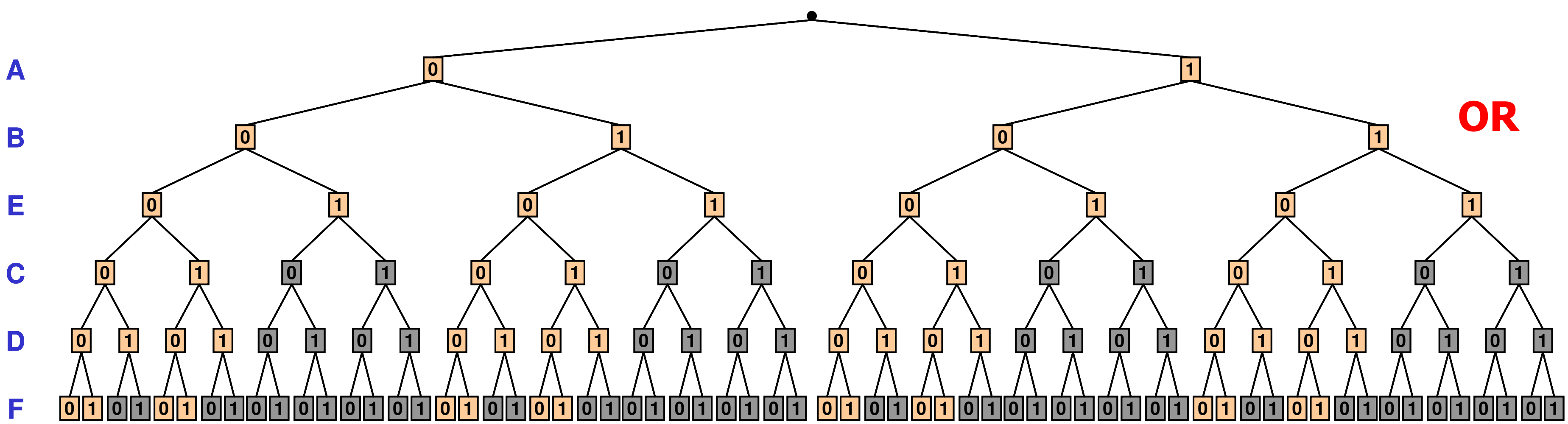
Exploiting Constraint Processing in Optimization

- Constraints should be recognized
- Constraint processing should be incorporated into optimization search:
  - Constraint propagation in look-ahead
  - Back-jumping
  - No-good recording

AND/OR vs. OR

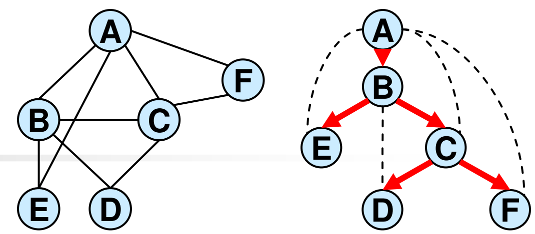


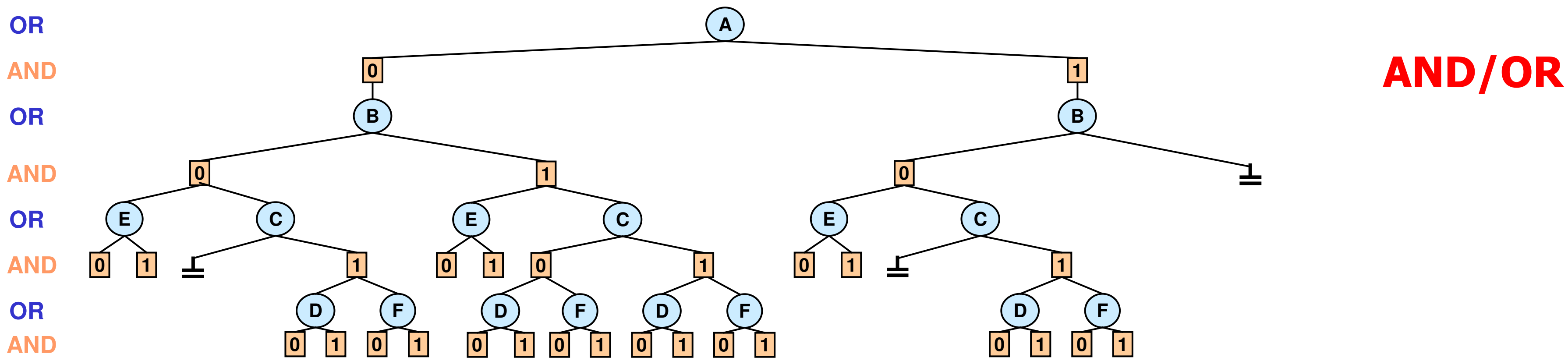


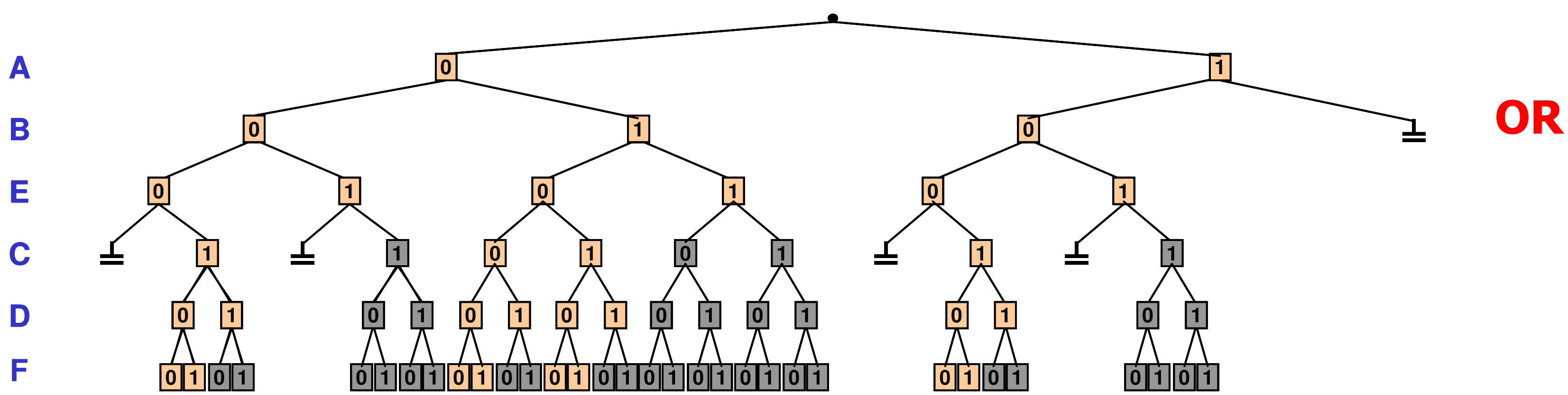


AND/OR vs. OR

(A=1,B=1)
(B=0,C=0)

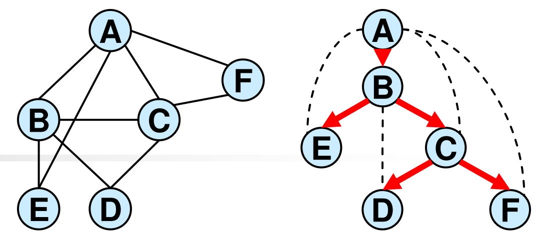


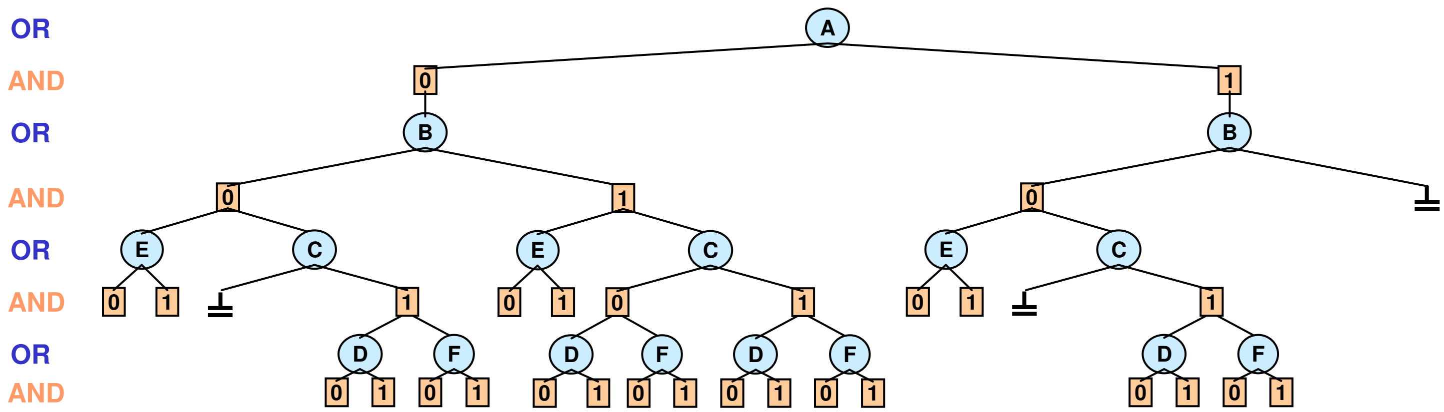




AND/OR vs. OR

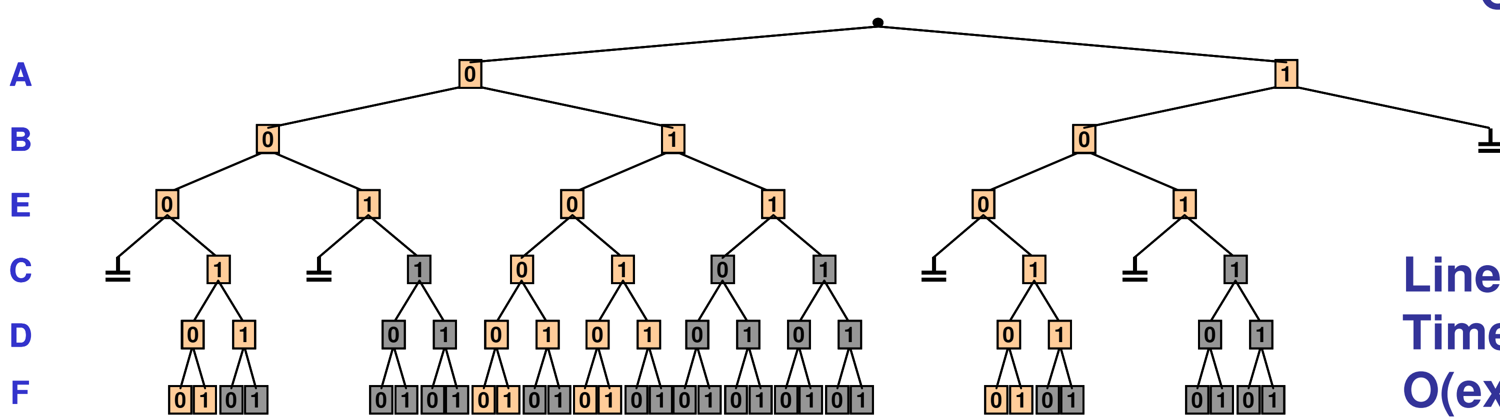
(A=1,B=1)
(B=0,C=0)





AND/OR

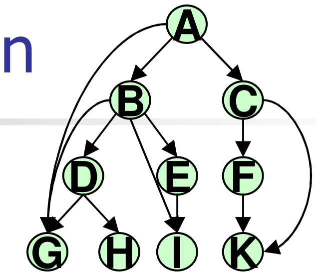
Space: linear
Time:
O(exp(m))
O(w* log n)

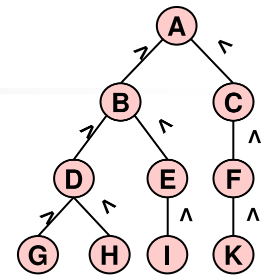


OR

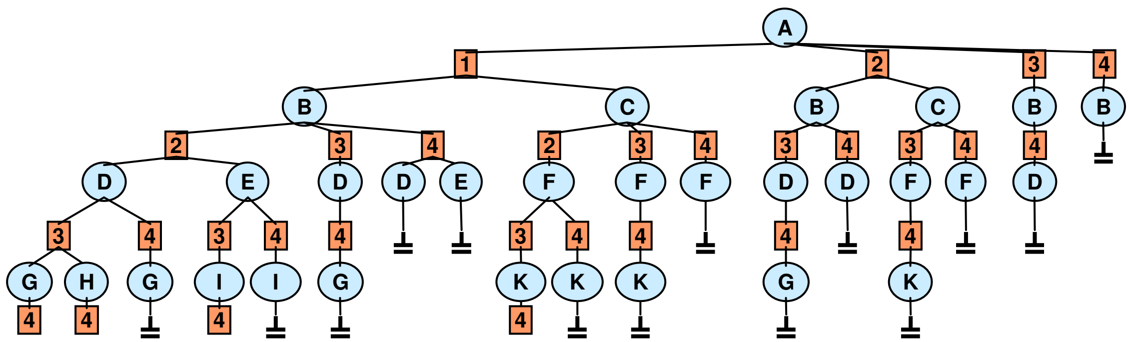
Linear space,
Time:
O(exp(n))

The Effect of Constraint Propagation

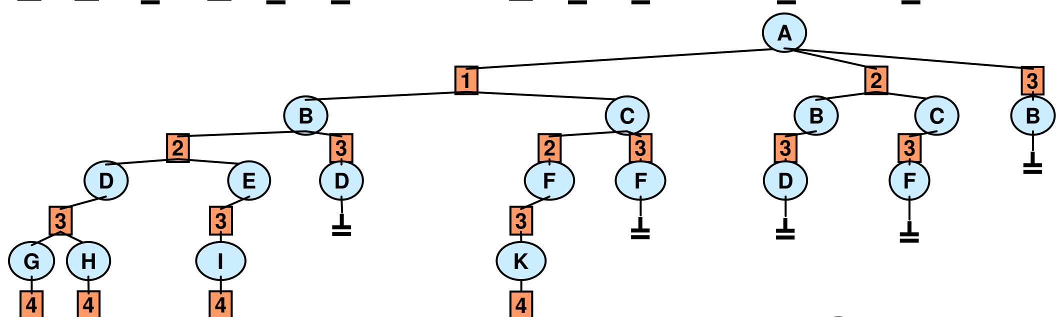




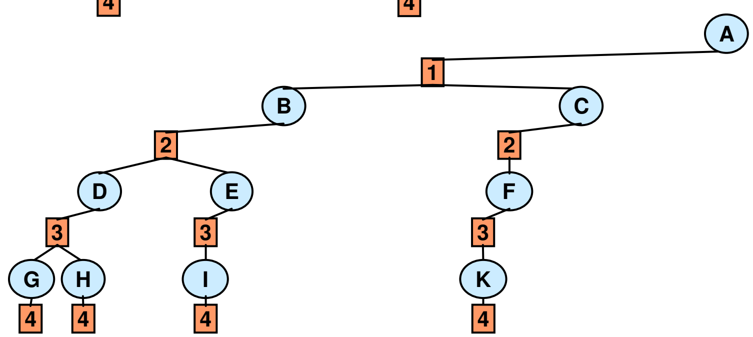
Domains are {1,2,3,4}



CONSTRAINTS ONLY



FORWARD CHECKING



MAINTAINING ARC CONSISTENCY

Outline

- Introduction
  - Optimization tasks for graphical models
  - Solving by inference and search
- Inference
  - Bucket Elimination, Dynamic Programming
  - Mini-Bucket elimination
- Search (OR)
  - Branch-and-Bound and Best-First
  - Lower-bounding heuristics
- **AND/OR search spaces**
  - Searching the AND/OR tree (linear space)
  - Searching the AND/OR graph (caching)
    - Depth-First AND/OR Branch-and-Bound Search
    - **Best-First AND/OR Search**
- Hybrids of search and inference
  - Cutset decomposition
  - Super-bucket scheme
- Software

Best-First Principle

- Best-first search expands first the node with the best heuristic evaluation function among all node encountered so far

- It **never** expands nodes whose cost is beyond the optimal one, unlike depth-first search algorithms (Dechter & Pearl, 1985)

- Superior among memory intensive algorithms employing the **same heuristic function**

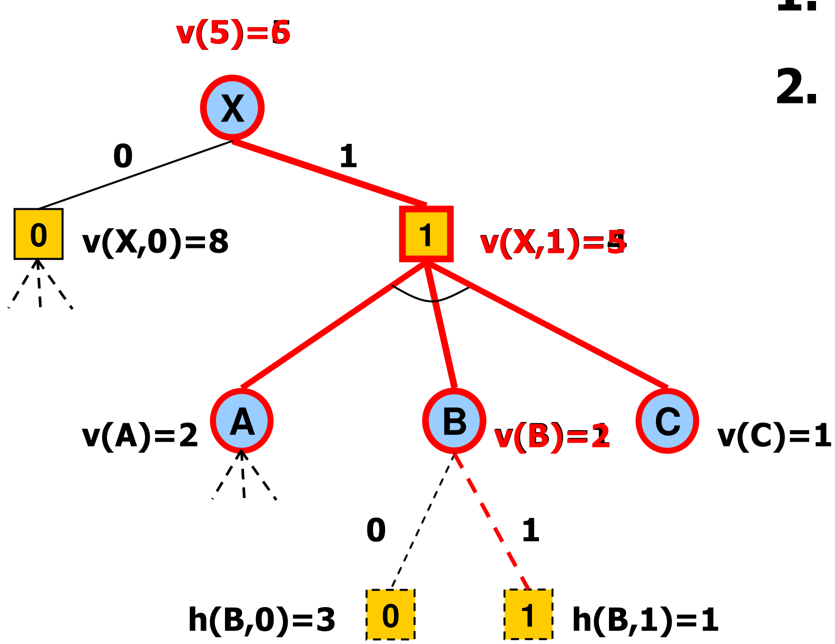
Best-First AND/OR Search (AOBF)

(Marinescu & Dechter, CPAIOR'07, AAAI'07, UAI'07)

- **Maintains the set of best partial solution trees**
- **Top-down Step (EXPAND)**
  - Traces down marked connectors from root
    - i.e., best partial solution tree
  - Expands a tip node **n** by generating its successors **n'**
  - Associate each successor with heuristic estimate **h(n')**
    - Initialize **v(n') = h(n')**
- **Bottom-up Step (REVISE)**
  - Updates node values **v(n)**
    - OR nodes: **minimization**
    - AND nodes: **summation**
  - Marks the most promising solution tree from the root
  - Label the nodes as SOLVED:
    - OR is SOLVED if marked child is SOLVED
    - AND is SOLVED if all children are SOLVED
- **Terminate when root node is SOLVED**

(specializes Nilsson's AO* to solving COP) (Nilsson, 1984)

Cost Revision



1. **Best Partial Solution Tree: X, (X,1), A, B, C**
2. **Expand node B: generate (B,0) and (B,1)**

$$v(B) = \min(w(B,0) + h(B,0),\ w(B,1) + h(B,1)) = \min(3,2) = 2$$

-> mark (B,1) best successor of B

$$v(X,1) = v(A) + v(B) + h(C) = 2+2+1=5$$

$$v(X) = \min(w(X,0) + v(x,0),\ w(X,1) + v(X,1)) = \min(8,6) = 6$$

-> mark (X,1) best successor of X

AOBF versus AOBB

- **AOBF** with the same heuristic as **AOBB** is likely to expand the smallest search space
- **AOBB** improves its heuristic function dynamically, whereas **AOBF** uses only **h(n)**
- **AOBB** can use far less memory by avoiding for example dead-caches, whereas **AOBF** keeps in memory the explicated search graph
- **AOBB** is any-time, whereas **AOBF** is not

Heuristics

- **AOBF can be guided by:**
  - Static Mini-Bucket heuristics
    (Kask & Dechter, AIJ'01), (Marinescu & Dechter, IJCAI'05)
  - Dynamic Mini-Bucket heuristics
    (Marinescu & Dechter, IJCAI'05)
  - LP Relaxations
    (Nemhauser & Wosley, 1988)

Experiments

- **Algorithms**
  - **AOBB –** depth-first AND/OR Branch-and-Bound search
  - **AOBF –** best-first AND/OR search

- **Benchmarks**
  - Weighted CSP
    - SPOT5 benchmarks
    - ISCAS'89 circuits
  - 0/1 ILP
    - Combinatorial auctions
  - Belief Networks
    - Random Networks
    - Coding
    - Grids
    - UAI'06 Evaluation Dataset

Earth Observing Satellites (WCSP)

(Marinescu & Dechter, AAAI'07)

spot5 (n, c, w*, h)	AOBB+SMB(i) AOBF+SMB(i) i=4		AOBB+SMB(i) AOBF+SMB(i) i=6		AOBB+SMB(i) AOBF+SMB(i) i=8		AOBB+SMB(i) AOBF+SMB(i) i=10		AOBB+SMB(i) AOBF+SMB(i) i=12		toolbar AOEDAC+DVO	
	time	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time	nodes
29	5.53	48,995	3.66	29,702	0.56	2,267	3.64	1,165	21.67	110	4.56	218,846
(83, 476, 14, 42)	6.42	36,396	2.23	12,801	**0.47**	757	3.59	323	21.77	96	0.81	8,698
42b	-	-	-	-	1804.76	9,410,729	553.47	3,191,205	116.98	584,838	-	-
(191, 1341, 18, 62)	35.42	118,085	29.11	106,648	20.80	82,611	**19.13**	67,538	38.91	43,127	6825.40	27,698,614
54	18.42	198,712	0.23	2,477	0.16	591	0.69	120	1.25	120	0.31	21,939
(68, 283, 11, 33)	0.41	2,714	0.11	631	0.16	312	0.69	68	0.69	68	**0.06**	688
404	174.09	1,396,321	51.88	529,002	2.55	23,565	**0.55**	1,704	1.16	598	151.11	6,215,135
(100, 710, 19, 42)	1.45	7,251	1.20	6,399	1.02	5,140	0.62	1,303	1.22	576	12.09	88,079
408b	-	-	-	-	7507.10	54,826,929	515.94	3,114,294	75.08	408,619	-	-
(201, 1847, 24, 59)	208.41	185,935	52.53	175,366	44.99	145,901	25.20	98,616	**16.97**	39,238	-	-
503	-	-	189.39	2,442,998	291.72	4,050,474	0.42	256	0.42	256	-	-
(144, 639, 9, 39)	5.28	16,114	1.56	9,929	1.59	9,186	**0.42**	144	**0.42**	144	10005.00	44,495,545
505b	-	-	-	-	-	-	-	-	1180.48	8,905,473	-	-
(240, 1721, 16, 98)	51.86	149,928	42.73	144,723	**29.25**	111,223	31.20	108,256	54.09	31,692	-	-

Results for SPOT5 networks. Time limit 3 hours.

AOBB+SMB(i) – AND/OR Branch&Bound with Static Mini-Bucket heuristics
AOBF+SMB(i) – Best-First AND/OR search with Static Mini-Bucket heuristics

AOEDAC+DVO – AND/OR Branch&Bound with EDAC consistency and DVO (Marinescu & Dechter, 2006b)
toolbar – OR Branch&Bound with EDAC consistency and DVO (de Givry et al., 2005)

ISCAS'89 Circuits (WCSP)

(Marinescu & Dechter, AAAI'07)

iscas (n, d, w*, h)	AOBB+SMB(i) AOBF+SMB(i) i=6		AOBB+SMB(i) AOBF+SMB(i) i=8		AOBB+SMB(i) AOBF+SMB(i) i=10		AOBB+SMB(i) AOBF+SMB(i) i=12		AOBB+SMB(i) AOBF+SMB(i) i=14		AOBB+SMB(i) AOBF+SMB(i) i=16	
	time	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time	nodes
c432	-	-	422.08	2,945,230	40.91	337,574	0.89	6,254	0.89	6,010	0.64	914
(432, 2, 27, 45)	40.89	195,918	39.33	196,892	0.52	2,154	**0.31**	1,007	0.38	847	0.67	445
c880	488.05	1,936,422	100.66	516,056	91.66	446,893	31.06	169,138	59.35	316,124	14.78	78,268
(881, 2, 27, 67)	2.16	6,929	1.36	4,454	0.91	2,792	**0.81**	2,231	1.19	2,862	1.44	1,589
s935	-	-	1285.07	6,623,608	143.53	763,933	-	-	22.28	128,372	4.80	15,010
(441, 2, 66, 101)	63.20	244,719	6.16	25,493	1.22	4,087	**1.19**	3,319	1.22	2,216	2.42	883
s1196	-	-	3347.38	13,554,137	503.30	2,425,152	2299.72	11,488,366	734.66	3,524,780	149.81	793,417
(562, 2, 54, 101)	23.16	75,617	22.67	72,075	**2.89**	9,336	13.02	40,210	7.27	21,989	3.56	2,090
s1238	1219.65	5,336,572	1897.37	8,386,634	1682.99	7,431,223	281.05	1,350,933	248.27	1,220,658	12.64	59,635
(541, 2, 59, 94)	75.88	291,101	34.09	137,960	29.41	111,205	12.31	53,095	6.64	26,101	**4.63**	7,142
s1494	120.94	334,047	364.80	953,945	5.64	17,279	27.64	80,895	6.92	23,131	9.02	20,004
(661, 2, 48, 69)	0.89	2,794	1.44	5,694	**0.59**	1,472	0.95	2,311	1.50	1,476	3.81	985

Results for ISCAS'89 networks. Time limit 1 hour.

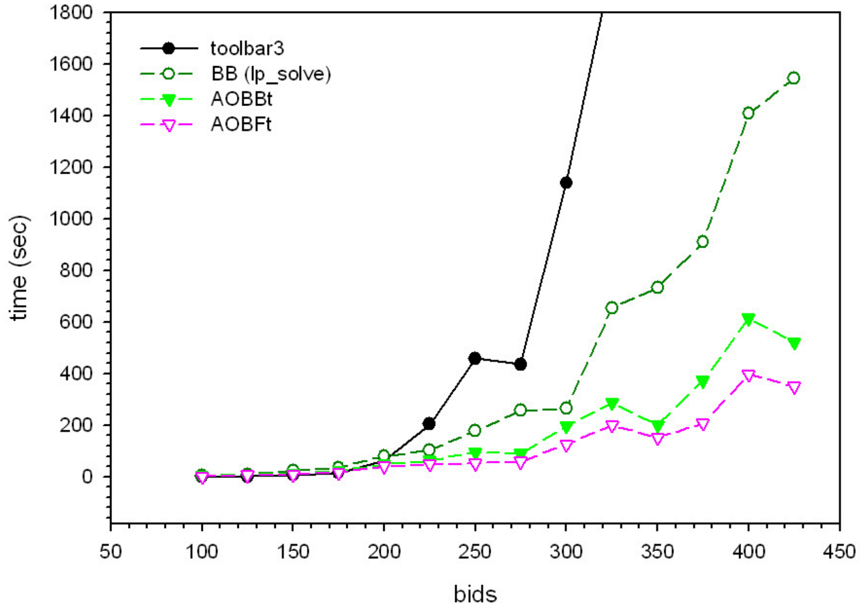
AOBB+SMB(i) – AND/OR Branch&Bound with Static Mini-Bucket heuristics
AOBF+SMB(i) – Best-First AND/OR search with Static Mini-Bucket heuristics

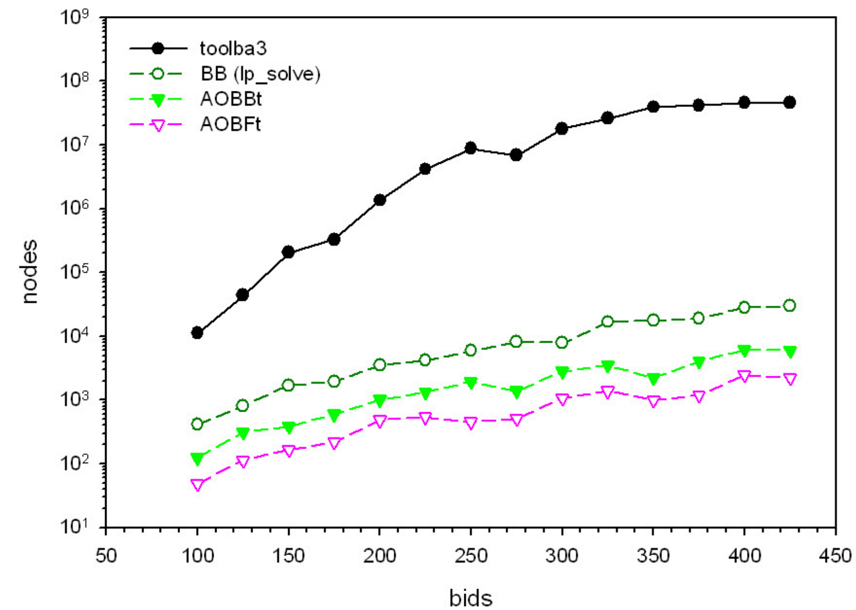
AOEDAC+DVO – AND/OR Branch&Bound with EDAC consistency and DVO (Marinescu & Dechter, 2006b)
toolbar – OR Branch&Bound with EDAC consistency and DVO [de Givry et al., 2005]

AOEDAC+DVO and **toolbar** could not solve any of these instances within the time limit!

Combinatorial Auctions (ILP)

(Marinescu & Dechter, CPAIOR'07)

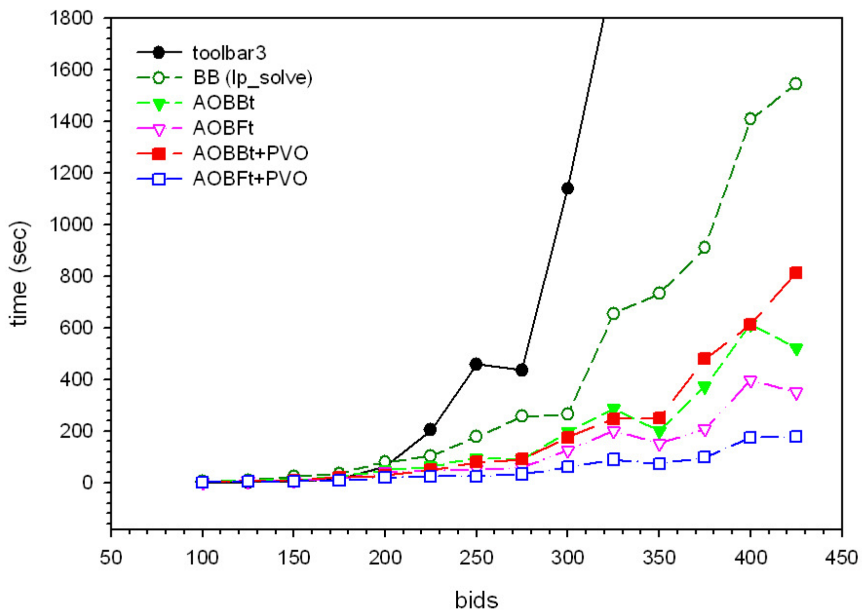


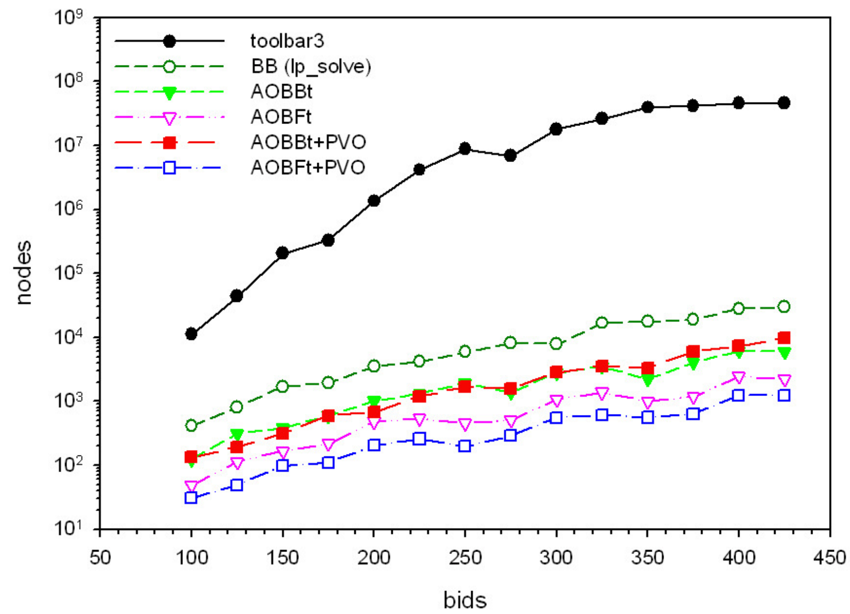


Combinatorial auctions from **regions-upv** distribution with 100 goods and increasing number of bids. Time limit 1 hour (each data point is an average over 10 random samples).

Very large treewidth ∈ [68, 184]

Dynamic Variable Orderings (ILP)
(Marinescu & Dechter, CPAIOR'07)



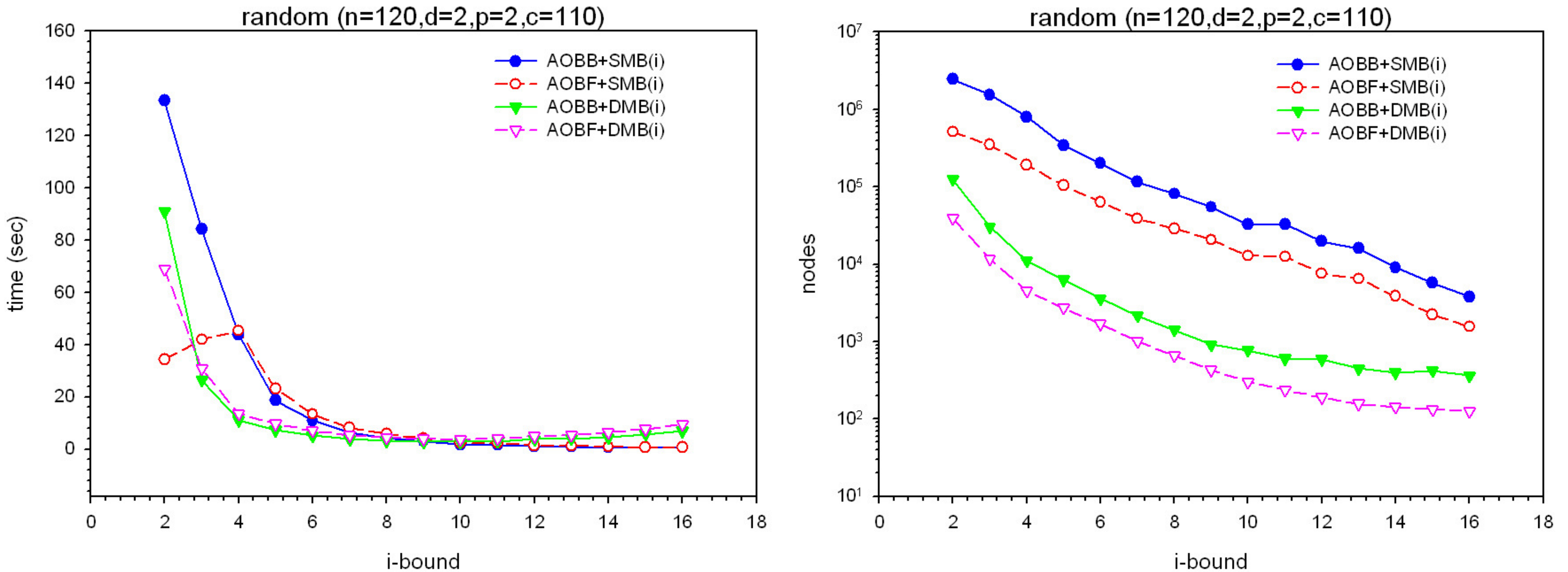


Combinatorial auctions from **regions-upv** distribution with 100 goods and increasing number of bids. Time limit 1 hour (each data point is an average over 10 random samples).

Very large treewidth ∈ [68, 184]

Random Belief Networks (BN)

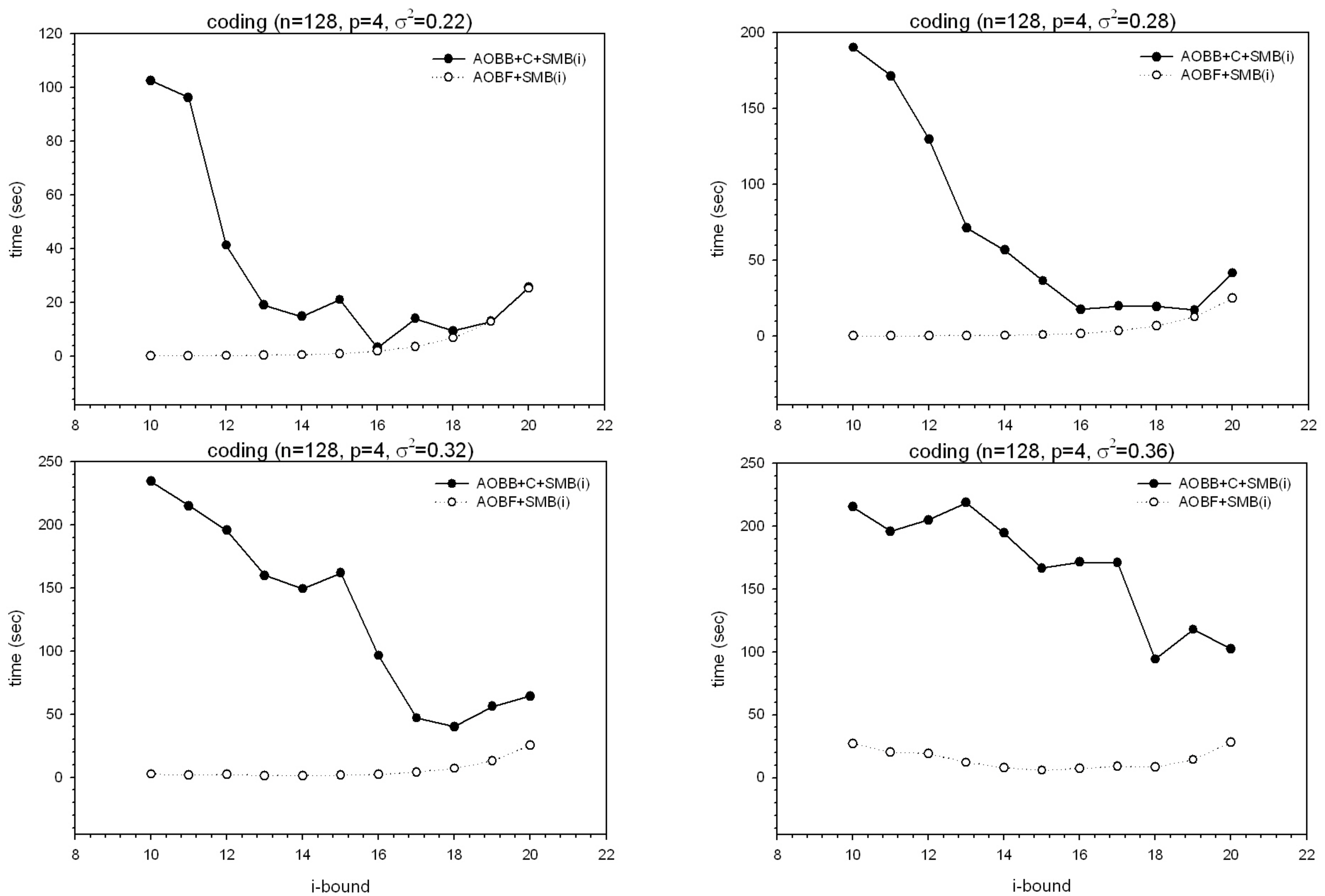
(Marinescu & Dechter, UAI'07)



CPU time in seconds (left) and number of nodes visited (right) for solving random belief networks with 120 nodes. Time limit 180 seconds, average induced width w* = 20 (each data point is an average over 20 random samples).

Coding Networks (BN)

(Marinescu & Dechter, UAI'07)



CPU time in seconds for solving coding networks with channel noise variance $\sigma^2 \in \{0.22, 0.28, 0.32, 0.36\}$. Time limit 300 seconds, average induced width w* = 54 (each data point is an average over 20 random samples).

Grid Networks (BN)

(Marinescu & Dechter, UAI'07)

grid	n e	w* h		SamIam v. 2.3.2	AOBB+SMB(i) i=8	i=10	i=12	i=14	i=16	AOBF+SMB(i) i=8	i=10	i=12	i=14	i=16
90-10-1	100	16	t	0.13	0.23	0.19	**0.08**	0.11	0.19	0.22	0.14	**0.08**	0.09	0.19
	0	26	#		4,396	3,681	1,231	760	101	1,788	1,046	517	312	100
90-14-1	196	23	t	11.97	19.95	12.52	8.83	1.22	0.78	8.24	5.97	2.20	1.02	**0.70**
	0	37	#		215,723	156,387	112,962	14,842	4,209	46,153	35,537	13,990	5,137	1,163
90-16-1	256	26	t	147.19	1223.55	130.47	11.09	11.25	**2.38**	133.19	47.72	9.91	10.53	2.97
	0	42	#		13,511,366	1,469,593	135,746	123,841	18,230	673,238	250,098	55,112	52,644	11,854
					i=12	i=14	i=16	i=18	i=20	i=12	i=14	i=16	i=18	i=20
90-24-1	576	36	t	out	1237.19	285.63	75.02	22.83	20.78	34.21	38.35	13.49	**9.08**	21.00
	20	61	#		6,922,516	2,051,503	547,401	110,144	15,400	125,962	149,445	49,261	14,390	8,155
90-26-1	676	35	t	out	-	-	634.59	85.11	49.97	out	out	57.66	**29.08**	32.95
	40	64	#				4,254,454	455,404	169,942			190,527	66,429	24,487
90-30-1	900	38	t	out	-	-	365.69	145.86	37.39	out	out	40.80	40.67	**36.00**
	60	68	#				2,837,671	936,463	32,637			136,576	121,561	13,217
90-34-1	1154	43	t	out	-	-	974.65	534.10	522.05	494.69	175.85	88.24	**59.39**	90.19
	80	79	#				5,555,182	2,647,012	2,430,599	705,922	303,782	189,340	112,955	115,553
90-38-1	1444	47	t	out	-	81.27	657.91	734.46	133.06	478.02	**22.80**	47.14	43.74	78.05
	120	86	#			259,405	1,505,849	1,478,903	161,156	580,623	38,376	80,177	52,209	35,294

CPU time in seconds and number of nodes visited for solving grid networks. Time limit 1 hour.
(we ran a single MPE query with *e* variables set as evidence uniformly at random)

AOBB+SMB(i) – AND/OR Branch&Bound with Static Mini-Bucket heuristics
AOBF+SMB(i) – Best-First AND/OR search with Static Mini-Bucket heuristics

SamIam – Recursive Conditioning (Darwiche, 2001)

Genetic Linkage Analysis (BN)

(Marinescu & Dechter, UAI'07)

ped (n,d,w*,h)	SamIam v. 2.3.2	Superlink v. 1.6	AOBB+SMB(i) AOBF+SMB(i) i=12 time	nodes	AOBB+SMB(i) AOBF+SMB(i) i=14 time	nodes	AOBB+SMB(i) AOBF+SMB(i) i=16 time	nodes	AOBB+SMB(i) AOBF+SMB(i) i=18 time	nodes	AOBB+SMB(i) AOBF+SMB(i) i=20 time	nodes
ped18 (1184,5,21,119)	157.05	139.06	- out	-	406.88 127.41	3,567,729 542,156	52.91 42.19	397,934 171,039	23.83 **19.85**	118,869 53,961	20.60 19.91	2,972 2,027
ped20 (388,5,23,42)	out	**14.72**	7243.43 out	63,530,037	5560.63 out	46,858,127	37.28 33.33	279,804 144,212	95.13 121.91	554,623 466,817		
ped30 (1016,5,25,51)	out	13095.83	1440.26 186.77	11,694,534 692,870	597.88 58.38	5,580,555 253,465	1023.90 85.53	10,458,174 350,497	151.96 49.38	1,179,236 179,790	43.83 **33.03**	146,896 37,705
ped39 (1272,5,23,94)	out	322.14	- out		- out		968.03 68.52	7,880,928 218,925	61.20 **41.69**	313,496 79,356	93.19 87.63	83,714 14,479
ped42 (448,5,25,76)	out	561.31	- out		- out		2364.67 **133.19**	22,595,247 93,831				
ped25 (994,5,29,53)	out	-	- out		- out		- out		2041.64 out	6,117,320	693.74 **198.49**	1,925,152 468,723
ped33 (581,5,26,48)	out	-	886.05 out	8,426,659	370.41 194.78	4,032,864 975,617	26.31 **24.16**	229,856 102,888	33.11 32.55	219,047 101,862	54.89 58.52	83,360 57,593

Results for genetic linkage analysis networks. Time limit 3 hours.

AOBB+SMB(i) – AND/OR Branch&Bound with Static Mini-Bucket heuristics
AOBF+SMB(i) – Best-First AND/OR search with Static Mini-Bucket heuristics

SamIam – Recursive Conditioning (Darwiche, 2001)
Superlink – Variable Elimination + Conditioning hybrid (Fishelson & Geiger, 2005)

UAI'06 Evaluation Networks (BN)

(Marinescu & Dechter, UAI'07)

bn	n	w* h		SamIam v. 2.3.2	AOBB+SMB(i) i=16	i=18	i=20	i=21	i=22	AOBF+SMB(i) i=16	i=18	i=20	i=21	i=22
BN_031	1153	46	t	out	1183.49	541.82	217.80	83.08	145.55	187.95	125.94	83.89	**71.53**	132.55
		160	#		3,990,212	2,131,977	889,782	94,507	97,721	427,788	292,293	114,046	25,392	30,067
BN_033	1441	43	t	-	1717.53	157.17	190.77	129.74	154.16	80.58	**41.25**	73.70	94.52	143.58
		163	#		2,156,432	210,552	256,191	89,308	46,312	124,453	41,865	49,760	22,256	14,894
BN_035	1441	41	t	-	67.74	133.28	58.81	80.64	157.83	**27.25**	36.75	51.20	75.53	158.17
		168	#		174,370	243,533	65,657	58,973	45,758	31,460	34,987	15,953	18,048	18,461
BN_037	1441	45	t	-	34.77	21.28	45.20	90.35	144.60	**12.80**	19.25	45.88	90.30	146.61
		169	#		69,326	33,475	8,815	16,400	12,507	16,304	11,046	4,315	5,610	4,798
BN_039	1441	48	t	-	-	1727.89	475.26	246.60	653.83	out	254.25	113.97	**112.69**	211.84
		162	#			3,448,072	1,043,378	518,011	3,045,139		725,738	213,676	127,872	239,838
BN_041	1441	49	t	-	257.96	56.66	54.36	78.74	130.94	36.22	**22.20**	43.56	69.91	121.24
		164	#		354,822	77,653	38,467	31,763	38,088	94,220	20,485	16,549	11,648	16,533
BN_127	512	57	t	out	1798.57	-	-	128.55	113.06	**54.03**	58.84	64.53	66.34	121.53
		74	#		17,583,748			860,026	93,543	235,416	251,134	166,741	84,007	70,351
BN_129	512	52	t	out	640.29	-	1439.32	222.17	155.63	out	200.47	**135.60**	out	231.95
		68	#		6,150,175		13,437,762	1,747,613	671,931		922,831	537,371		622,449
BN_131	512	48	t	out	-	43.06	51.16	-	156.11	**19.67**	50.58	36.66	65.75	99.20
		72	#			396,234	303,818		759,649	82,780	209,748	73,163	120,153	46,662
BN_134	512	52	t	out	-	-	-	-	234.38	out	**86.80**	96.21	97.28	112.63
		70	#						1,438,986		373,081	377,064	214,591	102,530

Results for UAI'06 Evaluation Dataset networks. Time limit 30 minutes.

AOBB+SMB(i) – AND/OR Branch&Bound with Static Mini-Bucket heuristics
AOBF+SMB(i) – Best-First AND/OR search with Static Mini-Bucket heuristics

SamIam – Recursive Conditioning (Darwiche, 2001)

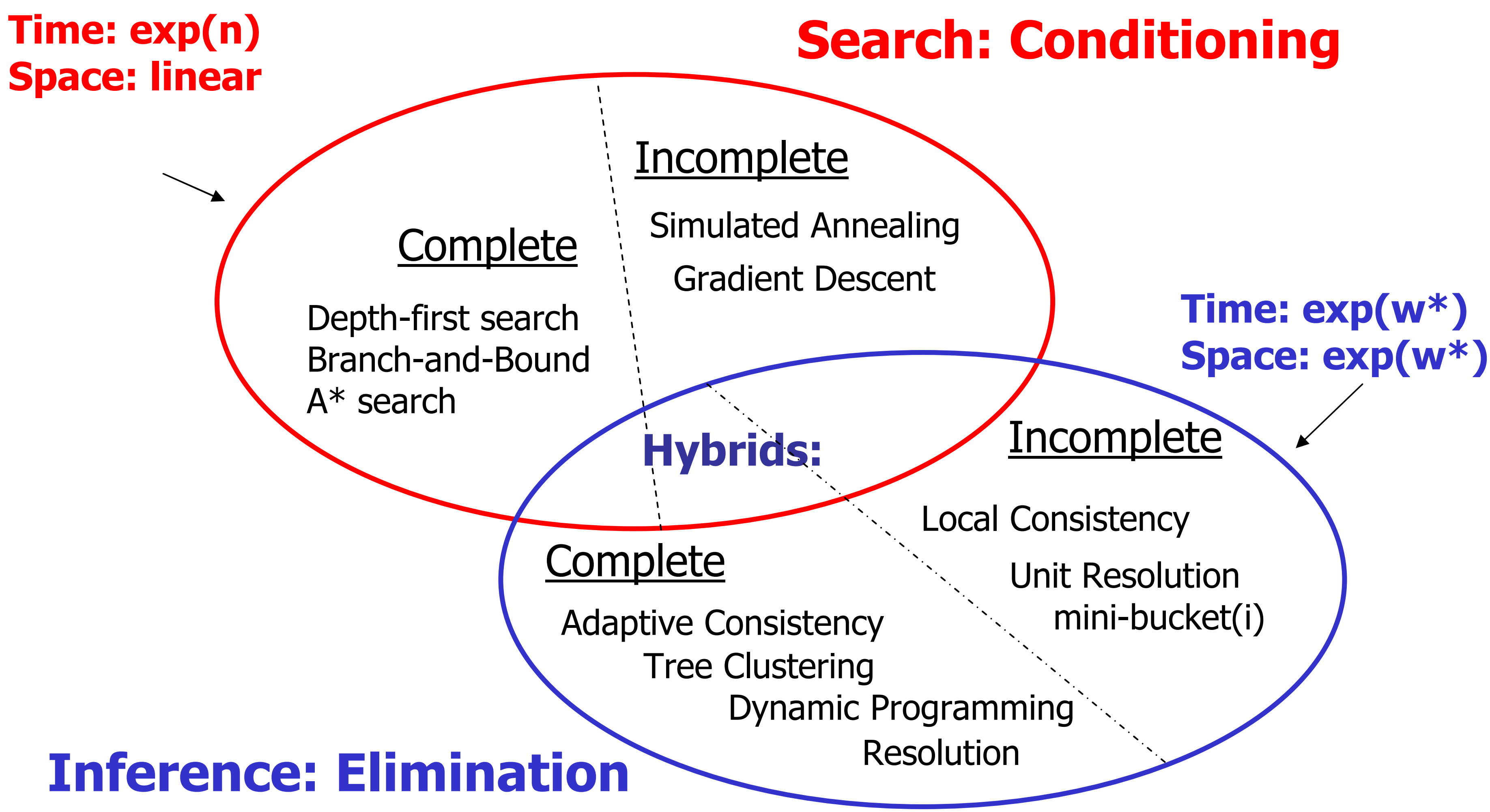
Summary

- New Best-First AND/OR search algorithm for solving optimization tasks in graphical models
- AOBF search incorporates dynamic variable ordering heuristics, thus exploring a dynamic AND/OR search tree
- Superior to classic OR Branch-and-Bound as well as AND/OR Branch-and-Bound on various benchmarks
- **Future Work**
  - Extend both AOBB and AOBF algorithms to incorporate cutting planes / no-good recording during search
  - Address several implementation issues so that the solvers can be competitive with commercial ones (e.g., CPLEX)

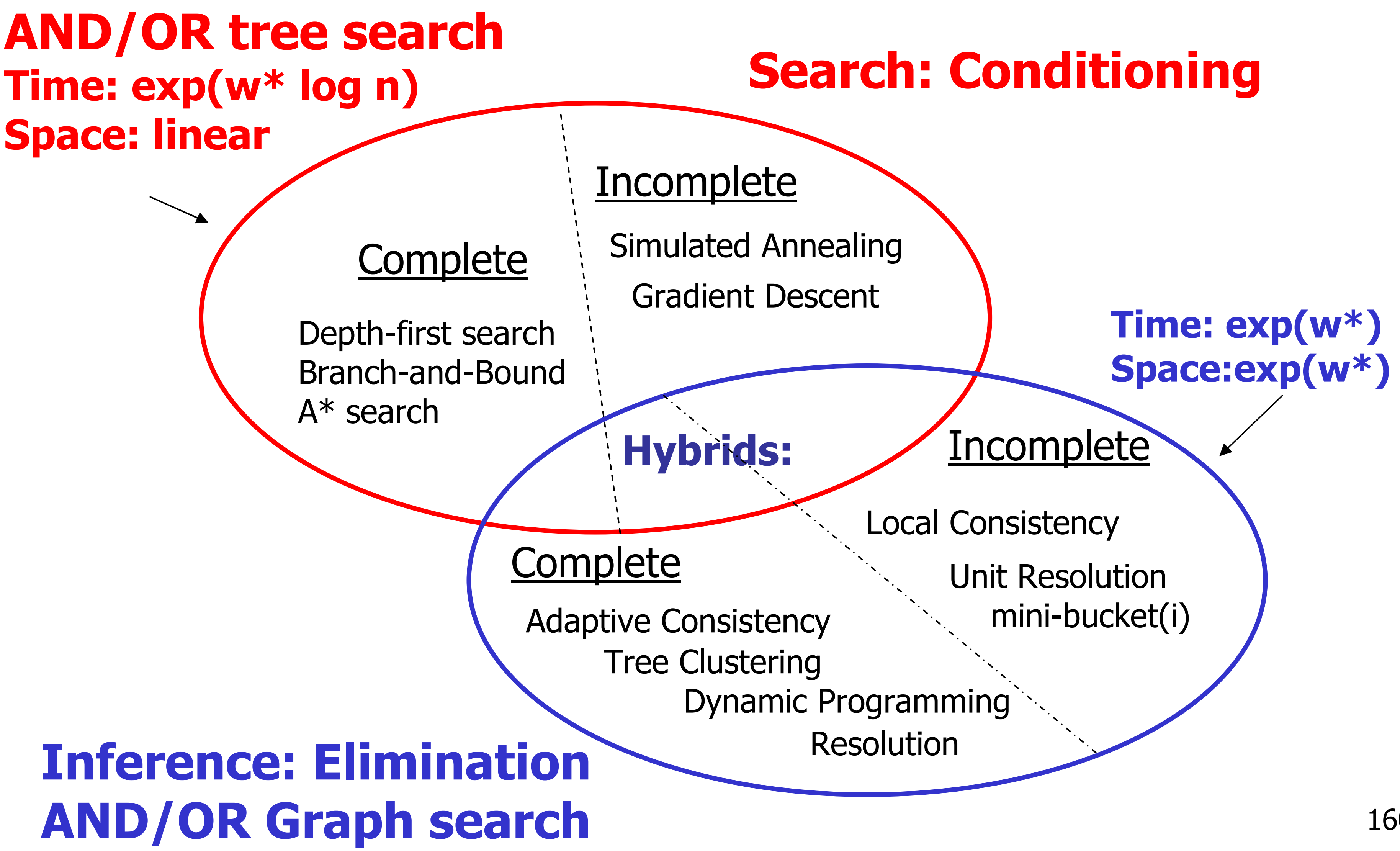
Outline

- **Introduction**
  - Optimization tasks for graphical models
  - Solving optimization problems by inference and search
- **Inference**
  - Bucket elimination, dynamic programming
  - Mini-bucket elimination, belief propagation
- **Search**
  - Branch-and-Bound and Best-First search
  - Lower-bounding heuristics
  - AND/OR search spaces
- **Hybrids of search and inference**
  - Cutset decomposition
  - Super-bucket scheme
- **Software**

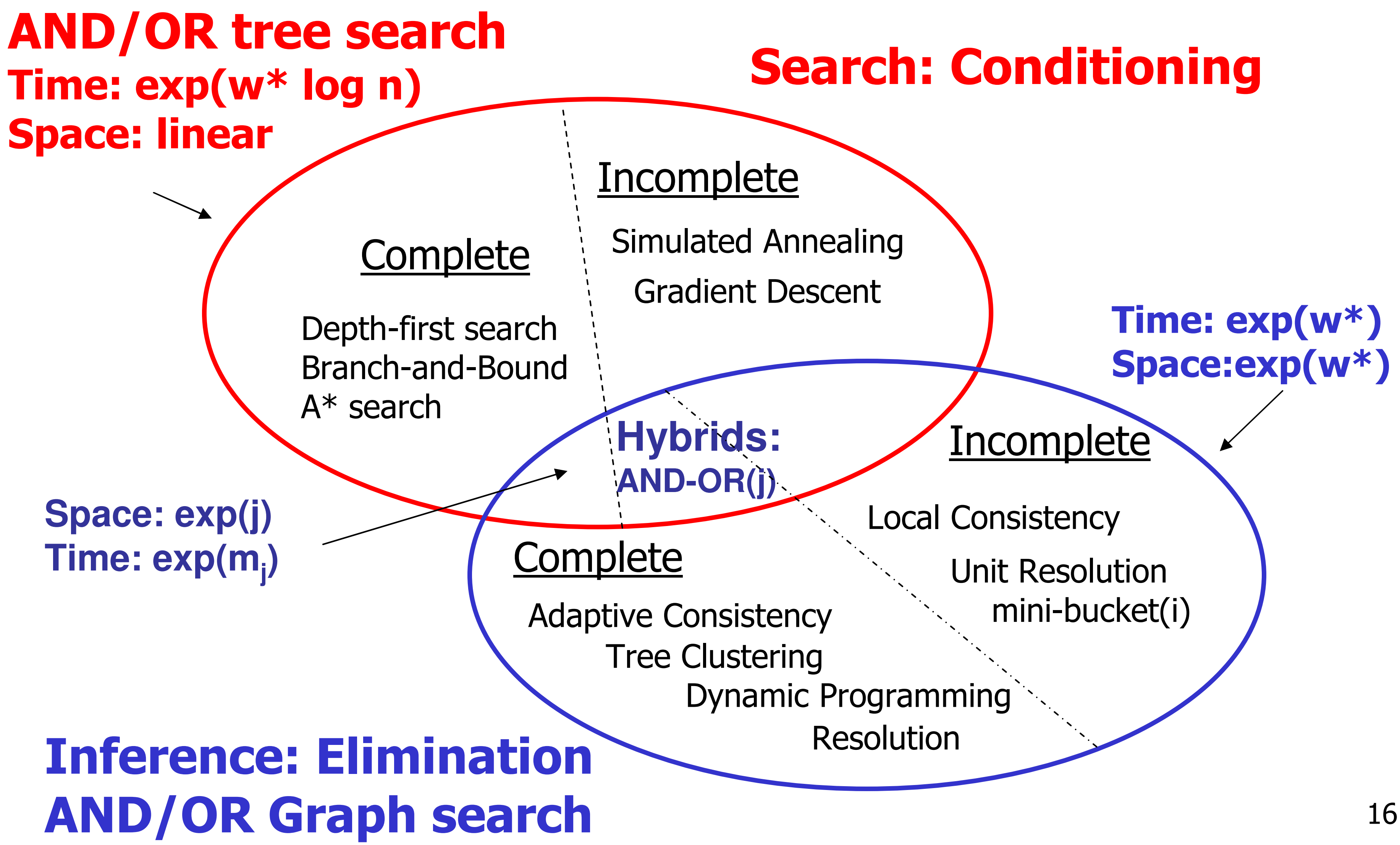
Solution Techniques



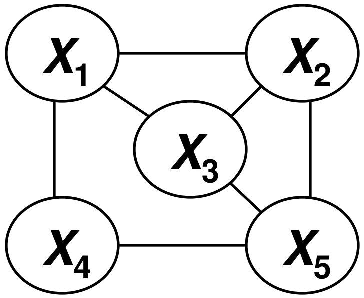
Solution Techniques



Solution Techniques

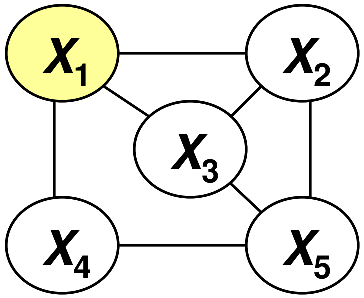


Search Basic Step: Conditioning

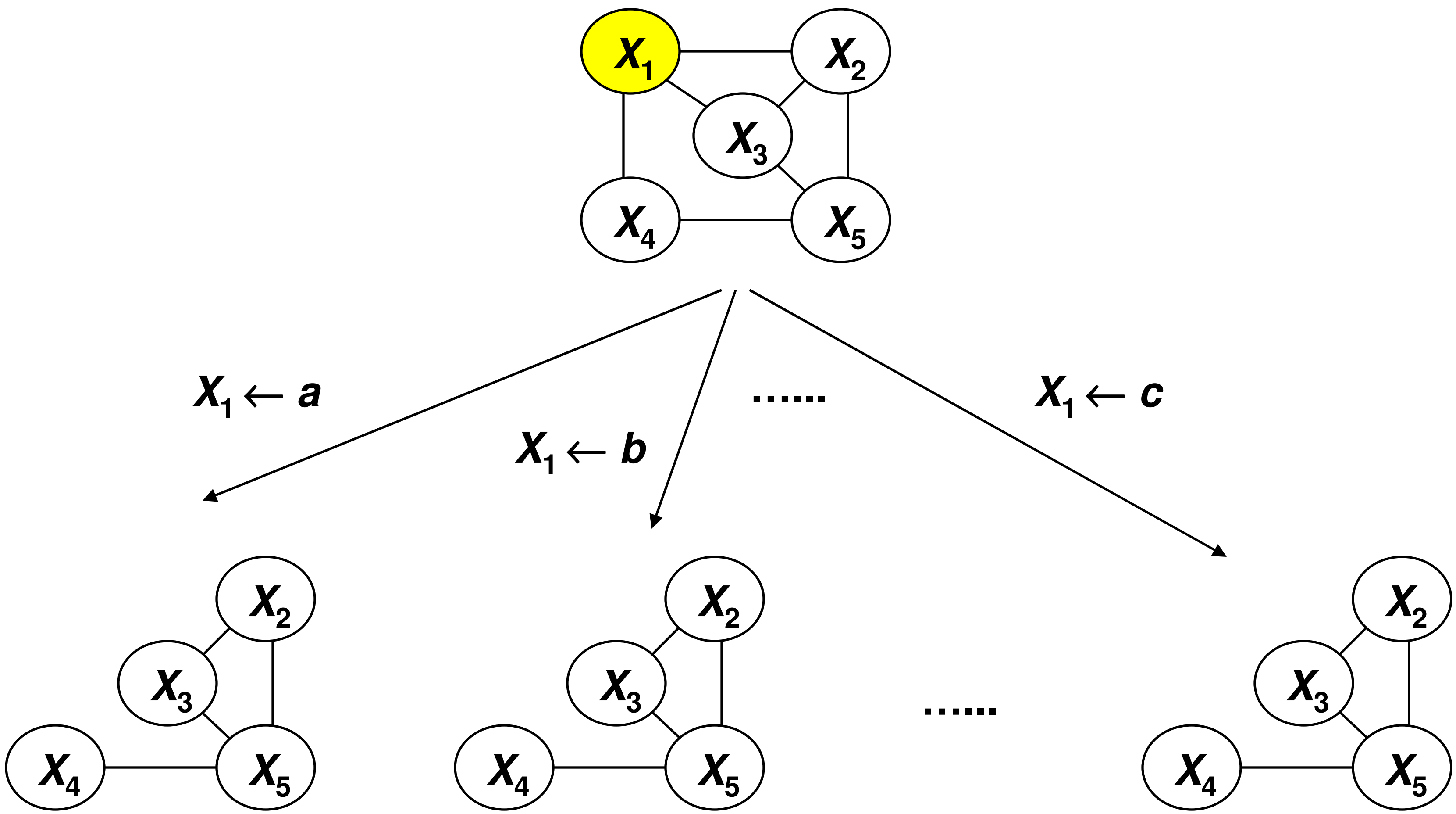


Search Basic Step: Conditioning

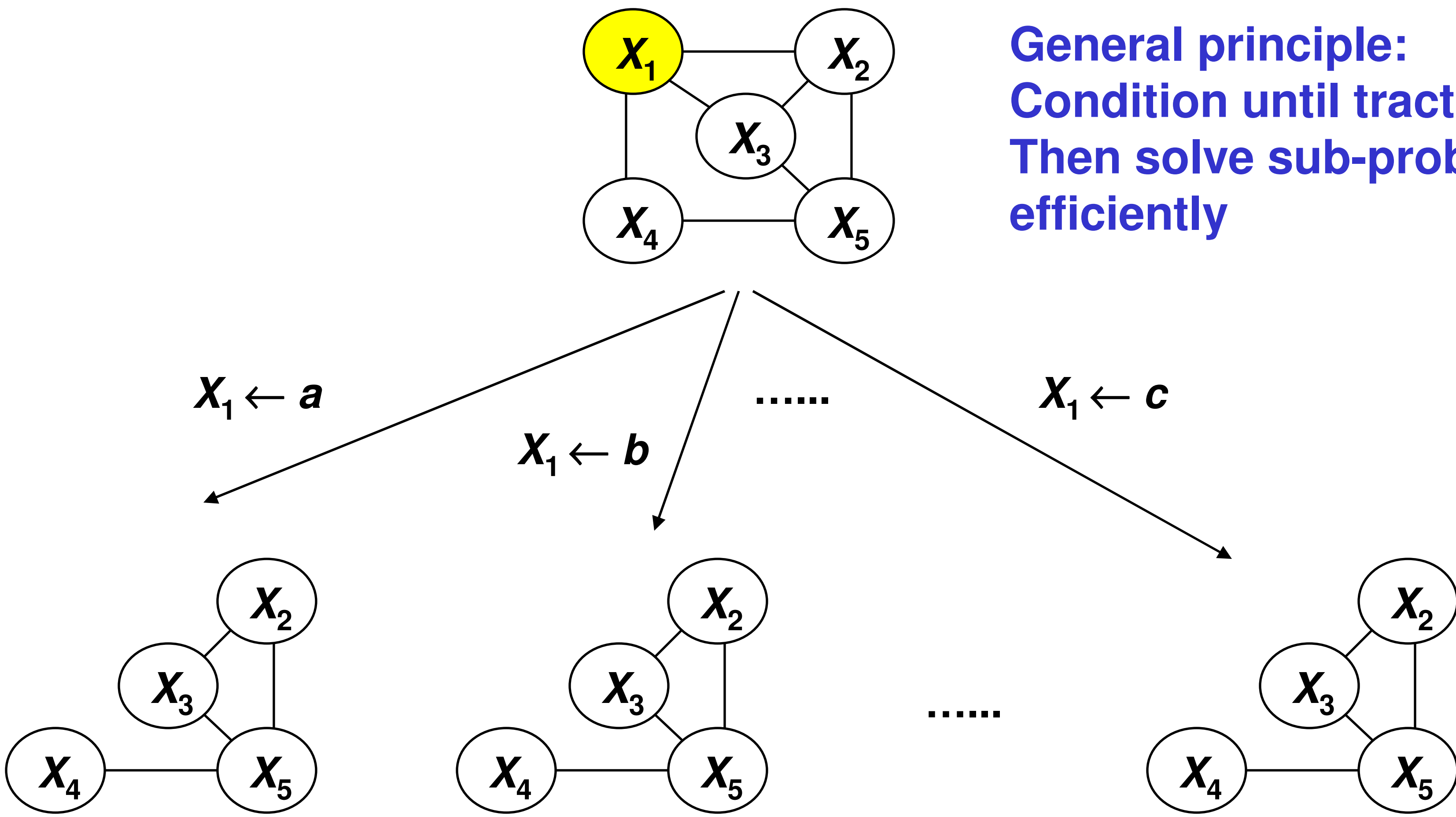
- **Select a variable**



Search Basic Step: Conditioning

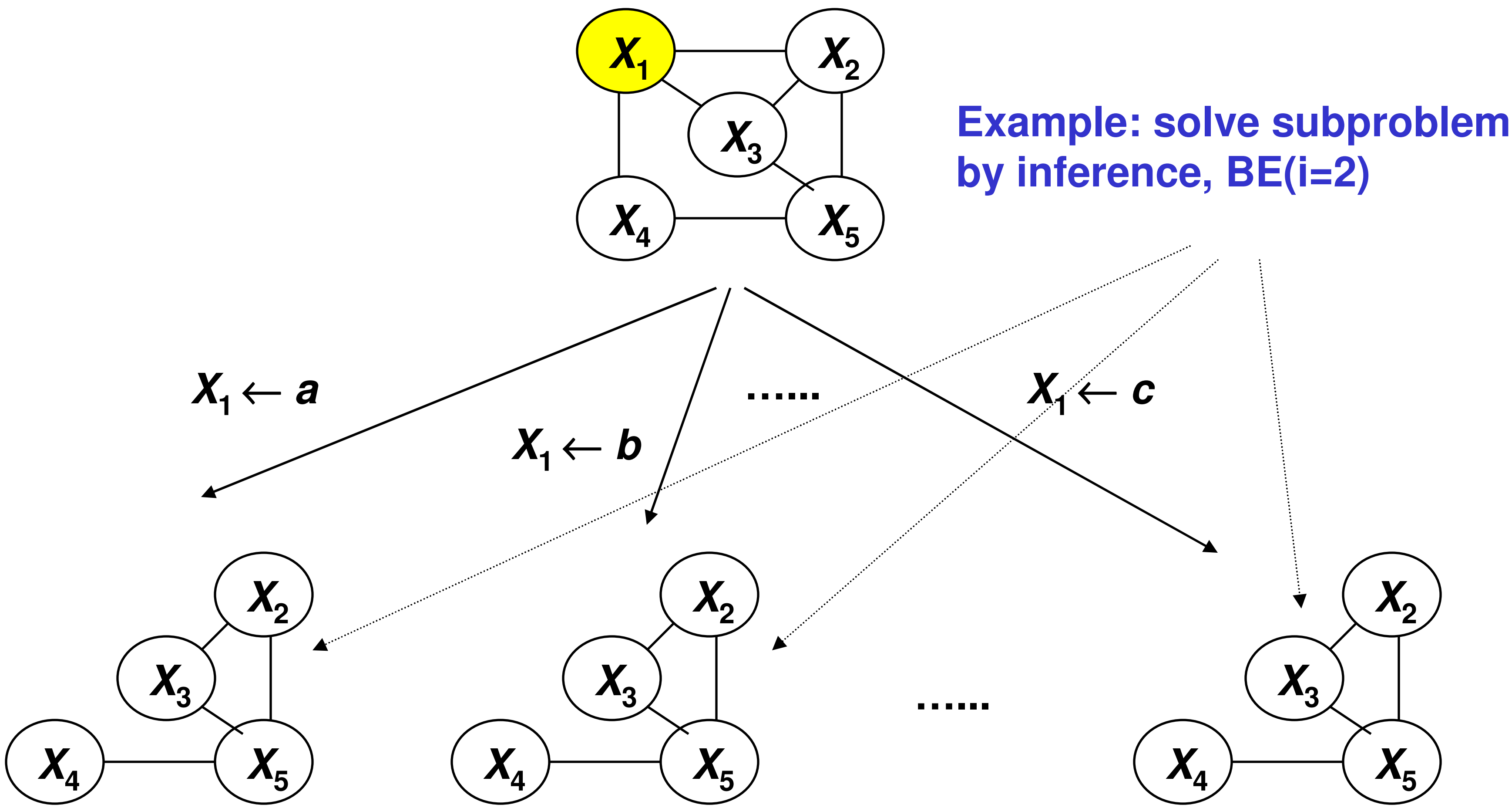


Search Basic Step:
Variable Branching by Conditioning



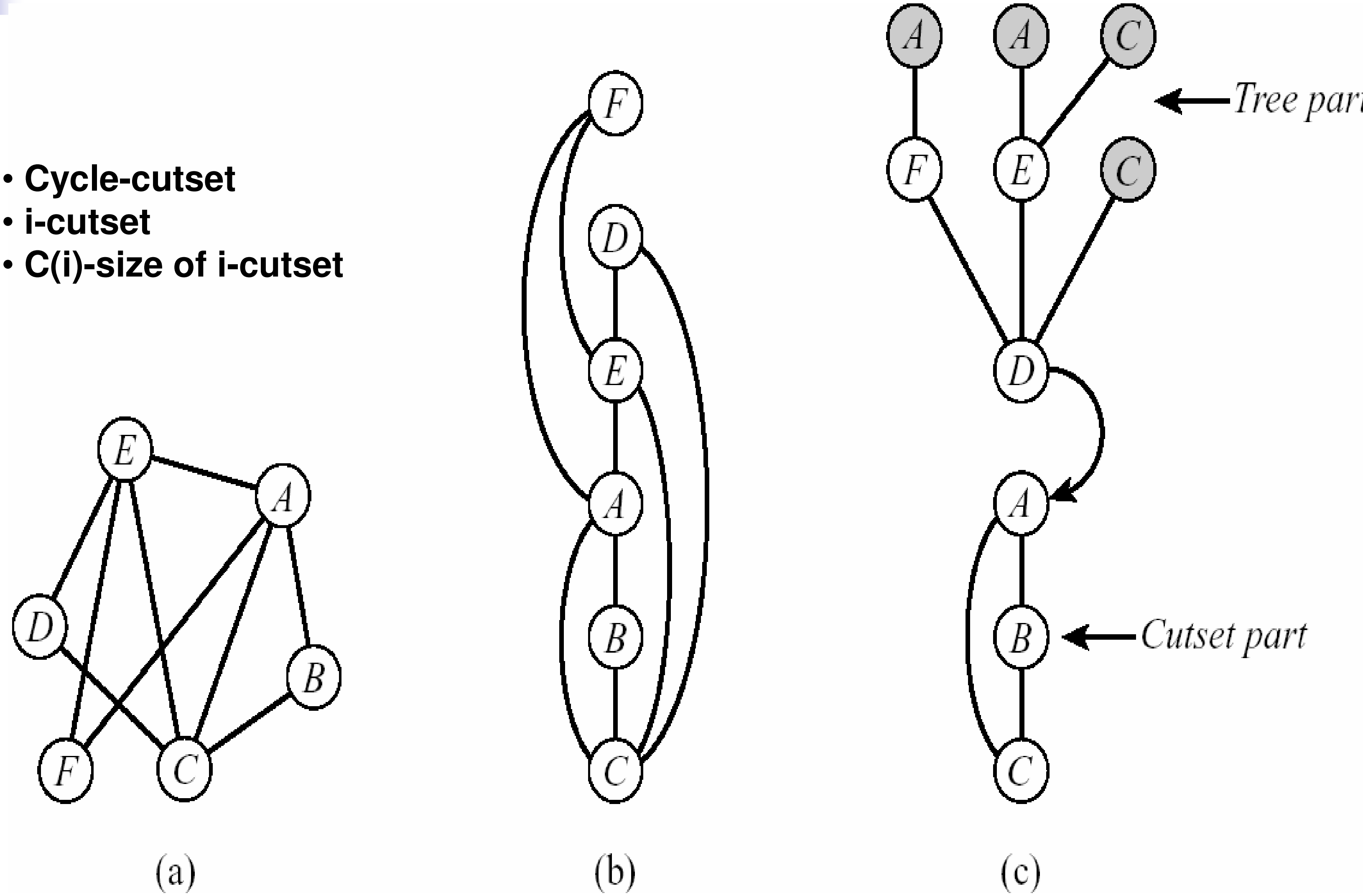
General principle:
Condition until tractable
Then solve sub-problems efficiently

Search Basic Step:
Variable Branching by Conditioning



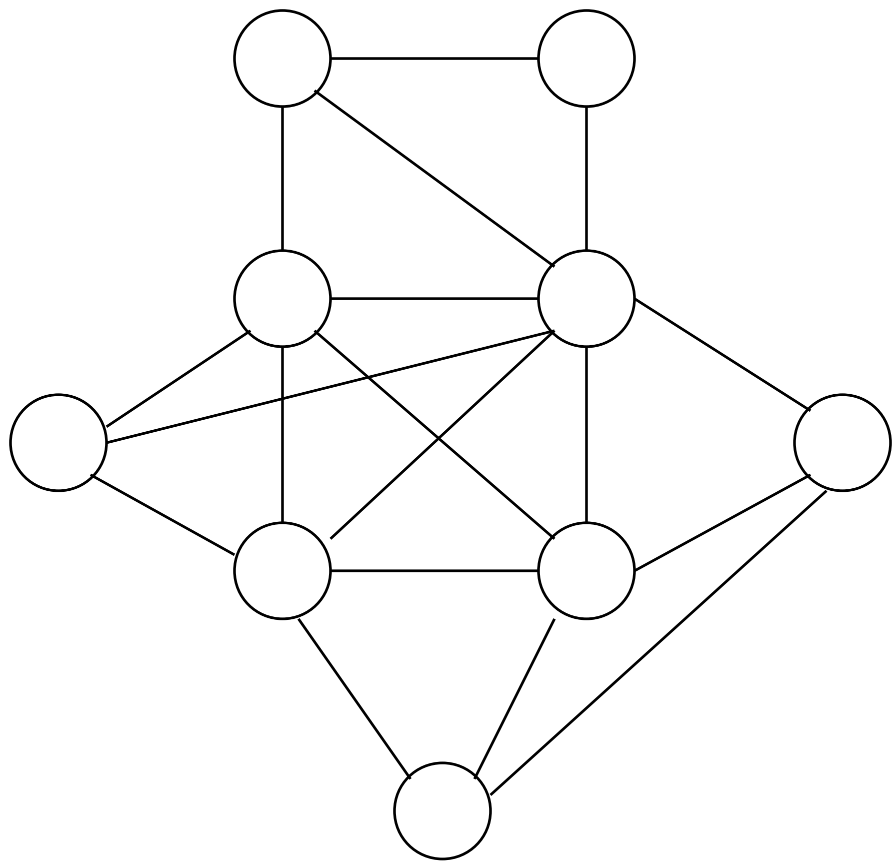
The Cycle-Cutset Scheme:
Condition Until Treeness

- **Cycle-cutset**
- **i-cutset**
- **C(i)-size of i-cutset**

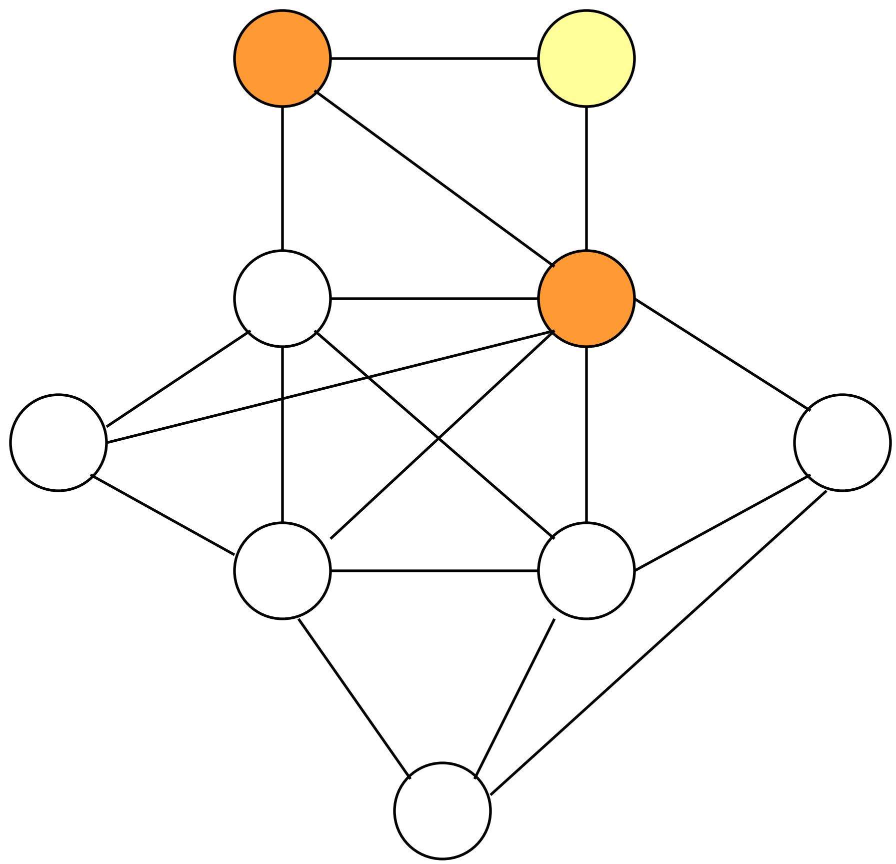


Space: exp(i), Time: O(exp(i+c(i))

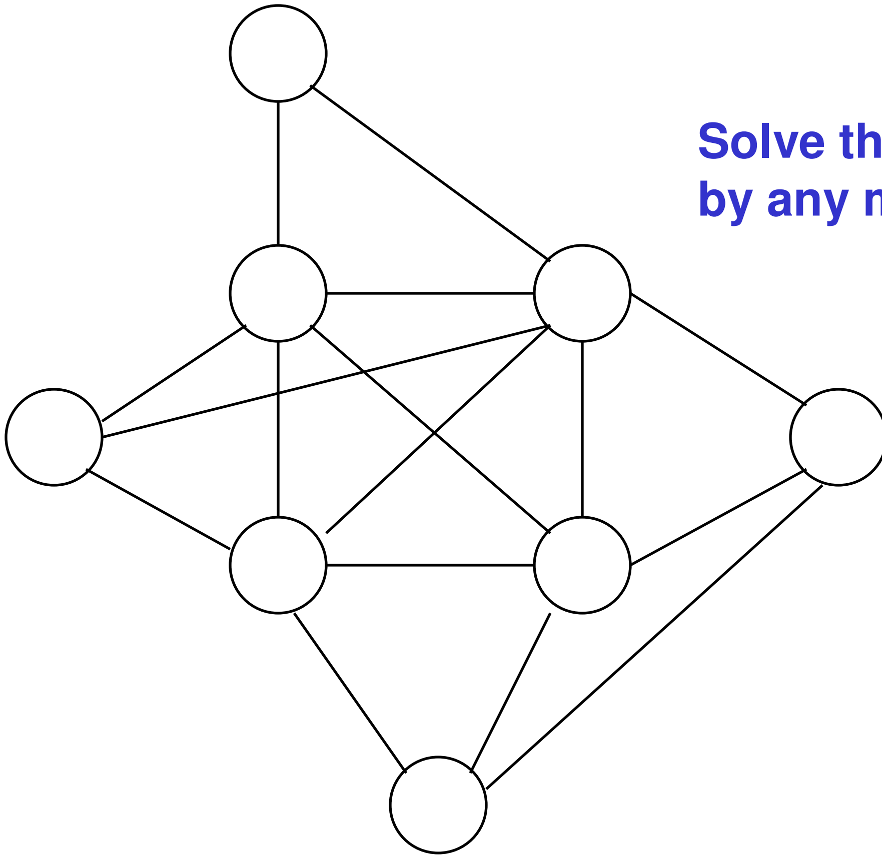
Eliminate First



Eliminate First



Eliminate First



Solve the rest of the problem by any means

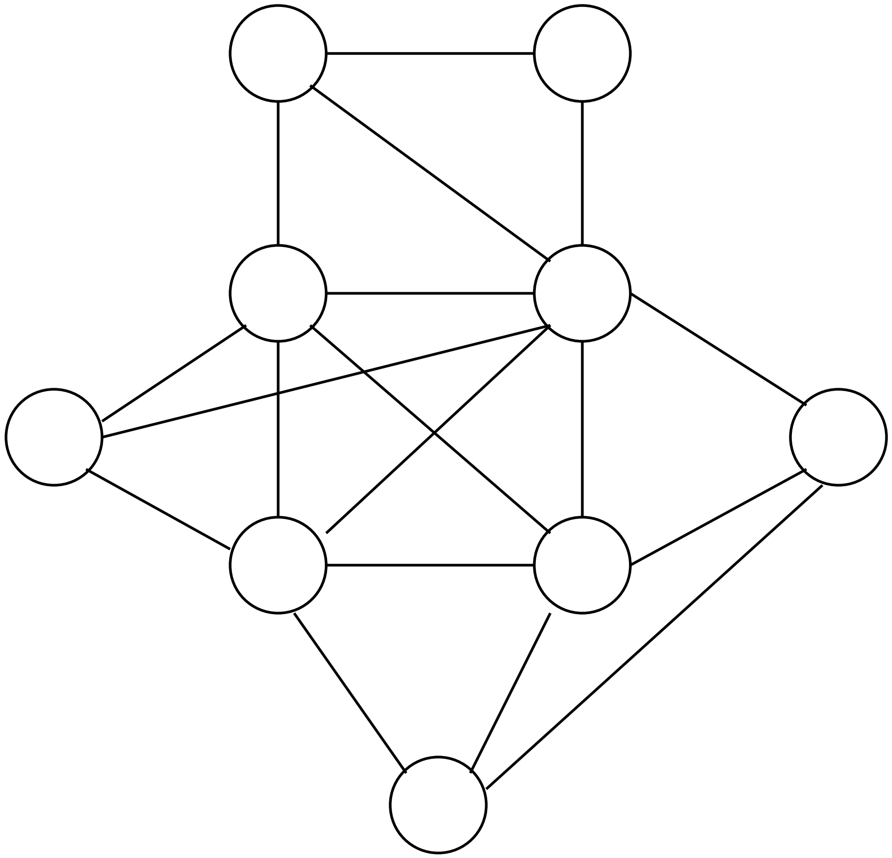
Hybrids Variants

- Condition, condition, condition … and then only eliminate (w-cutset, cycle-cutset)

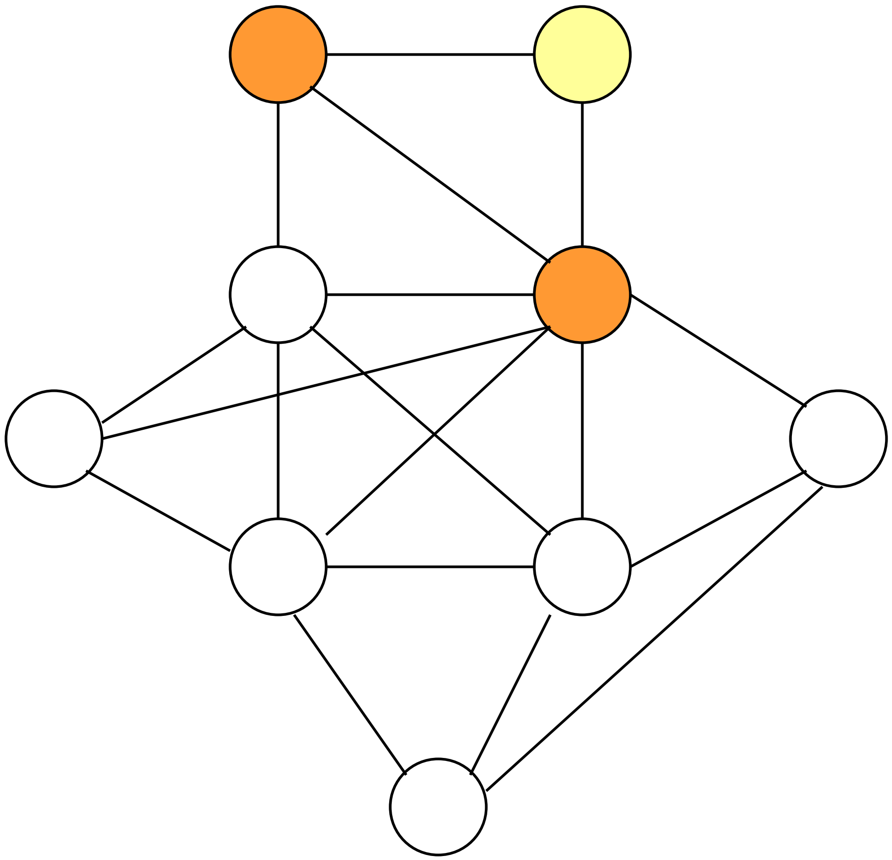
- Eliminate, eliminate, eliminate … and then only search

- Interleave conditioning and elimination (elim-cond(i), VE+C)

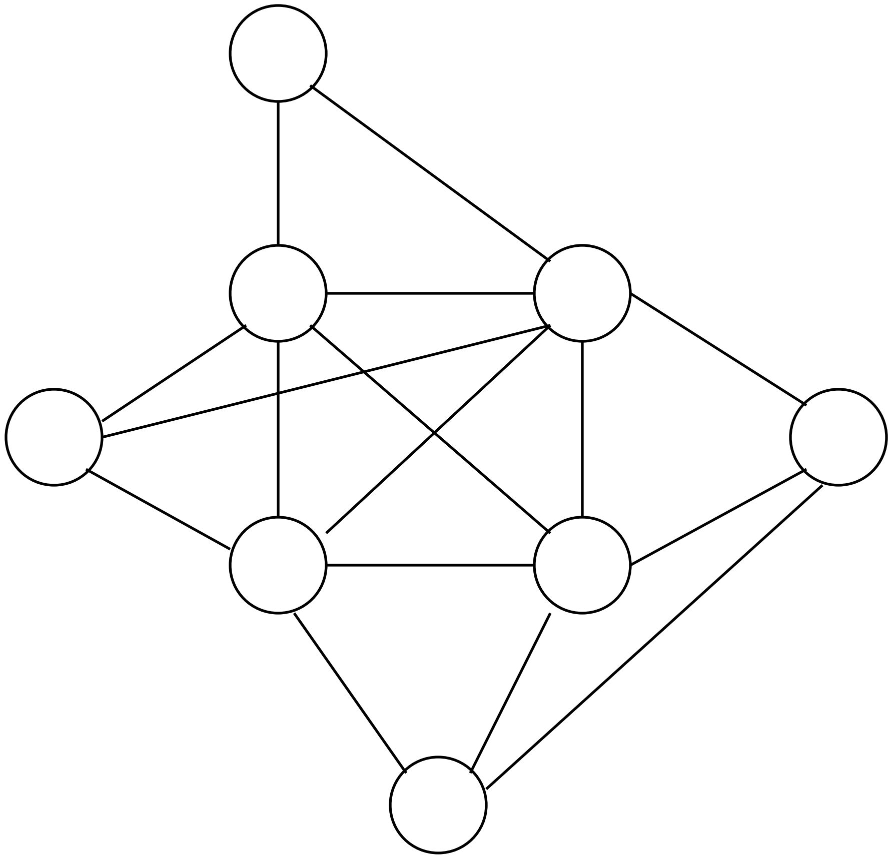
Interleaving Conditioning and Elimination
(Larrosa & Dechter, CP'02)



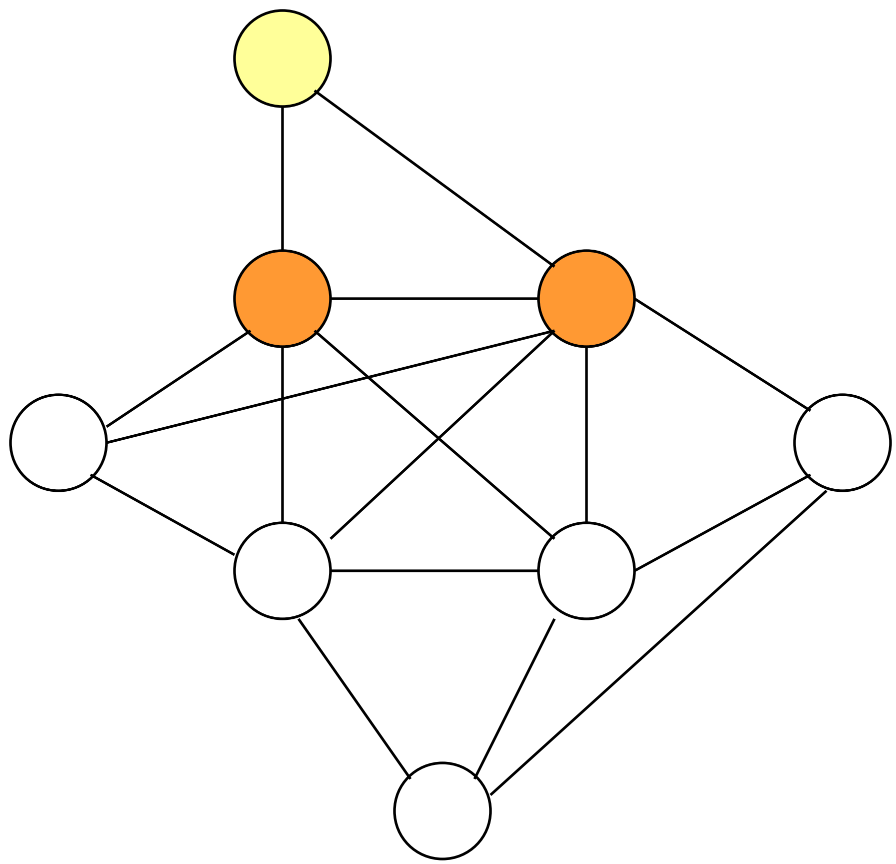
Interleaving Conditioning and Elimination



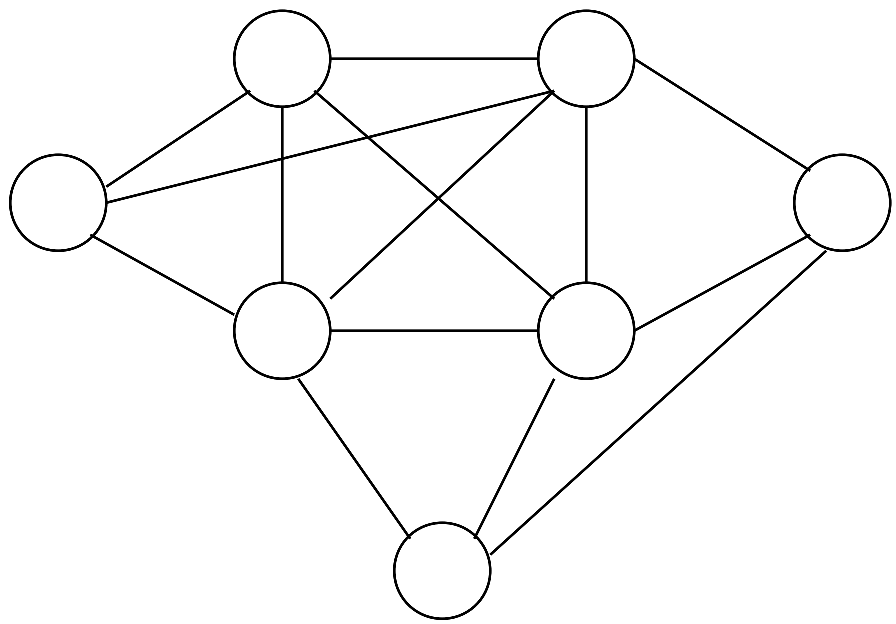
Interleaving Conditioning and Elimination



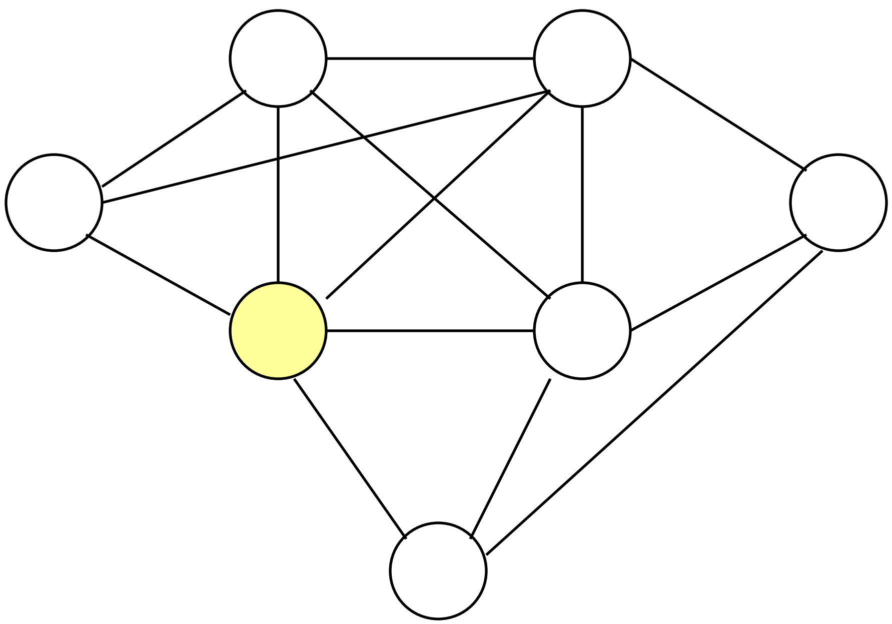
Interleaving Conditioning and Elimination



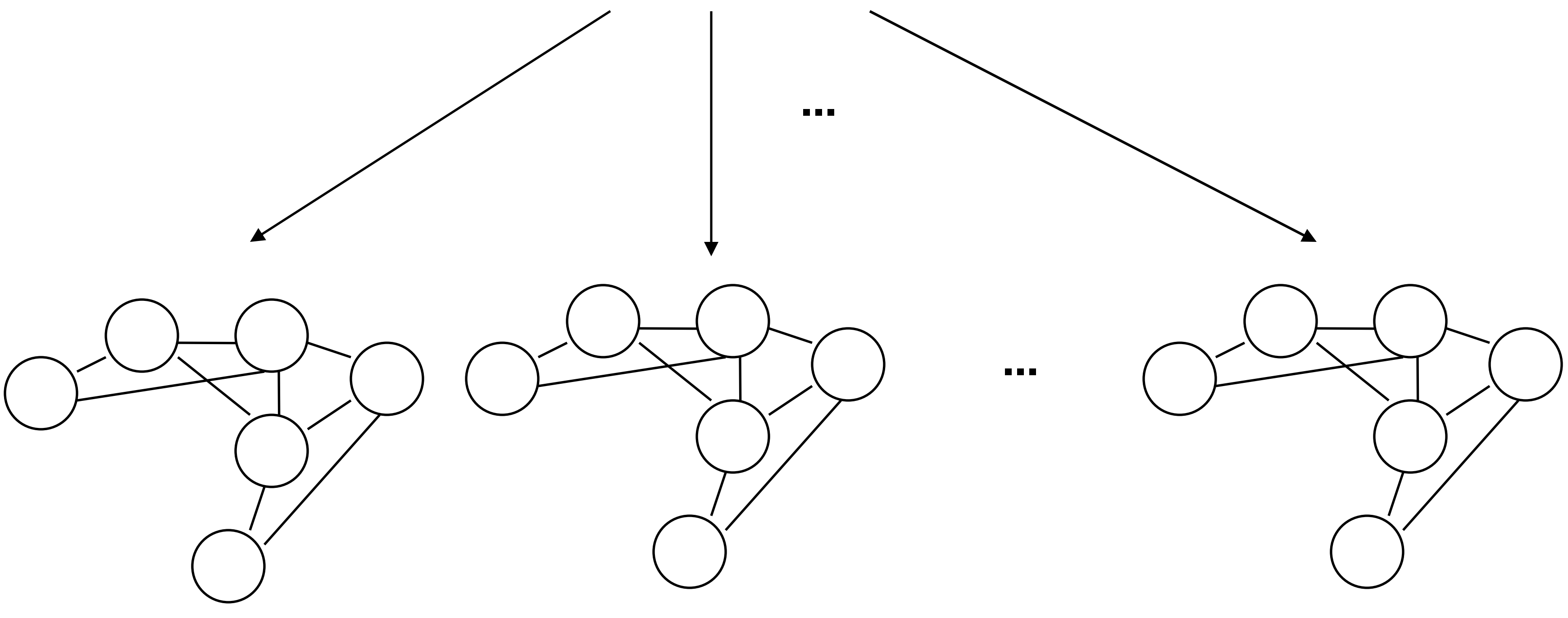
Interleaving Conditioning and Elimination



Interleaving Conditioning and Elimination



Interleaving Conditioning and Elimination



Time-space Tradeoffs

- AO(j): DFS search of AO, caches j-context
  - j = max number of variables in a context
- *AO j-cutset*
- Elimination-conditioning(j)

j=0 **j** j=w*

Space: O(n)

Time: O(exp(w* log n))

Space: O(exp(j))

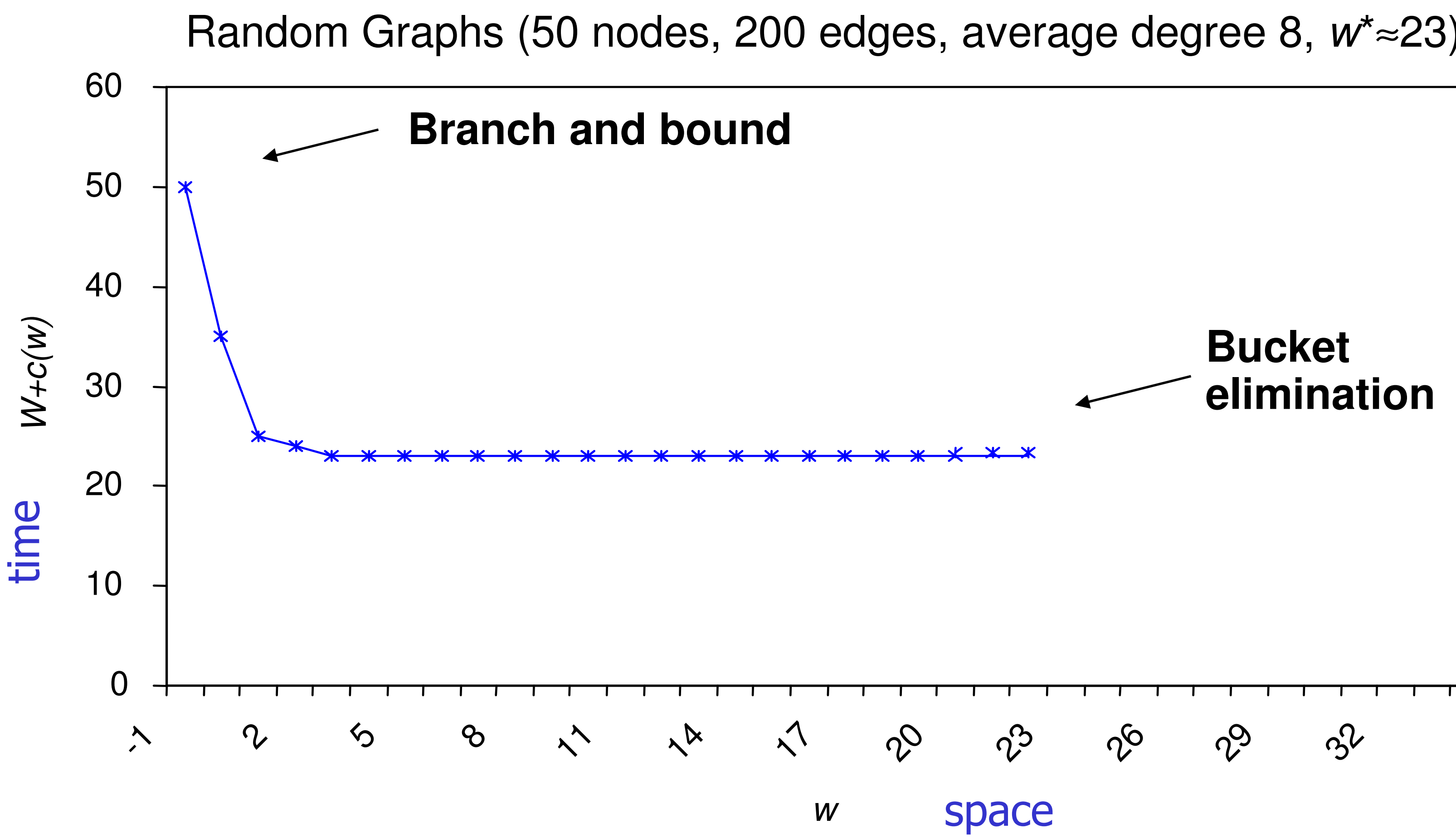
Time: O(exp(m(j)+j)

m(j) - AO depth of w-cutset

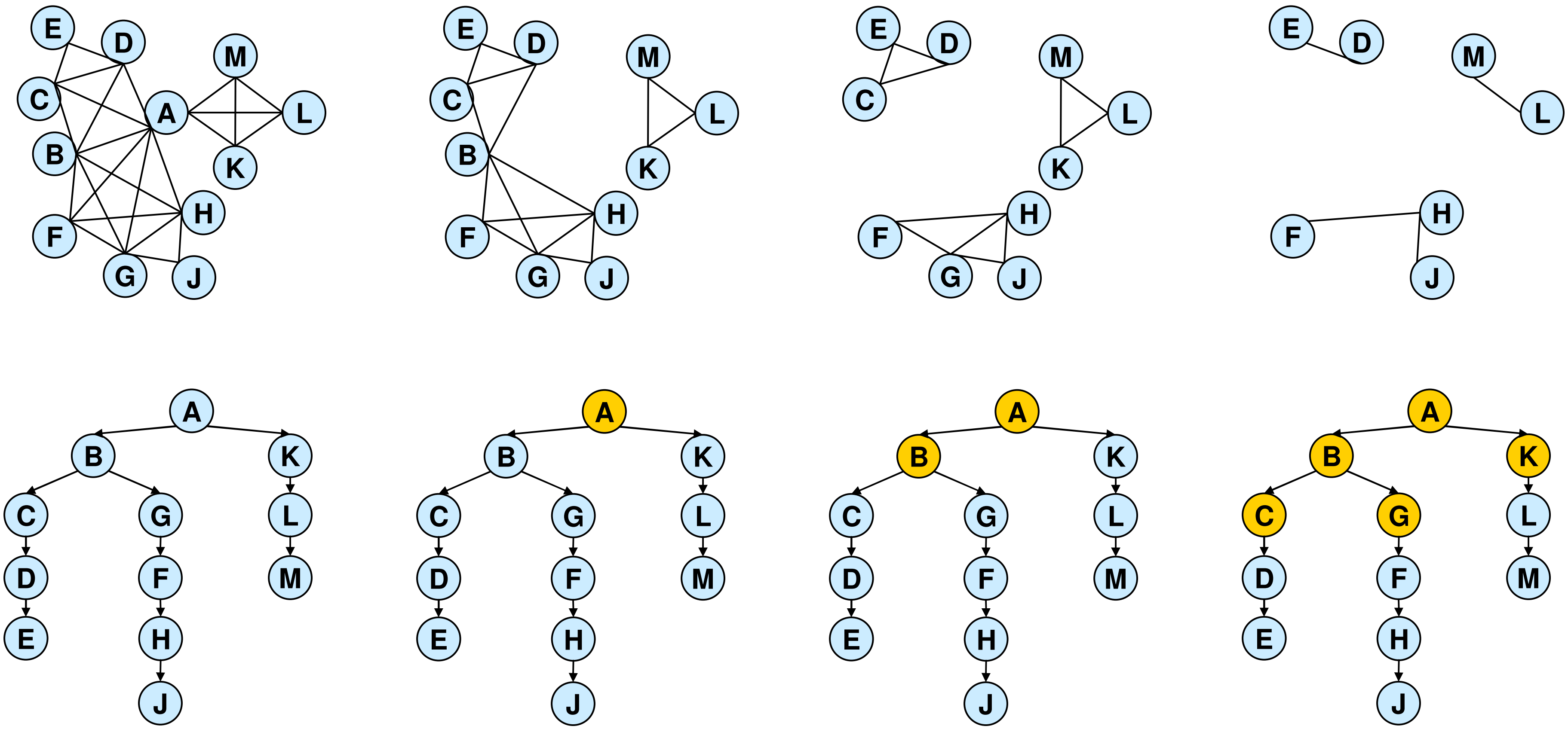
Space: O(exp w*)

Time: O(exp w*)

Time vs. Space for w-cutset



AND/OR i-cutset

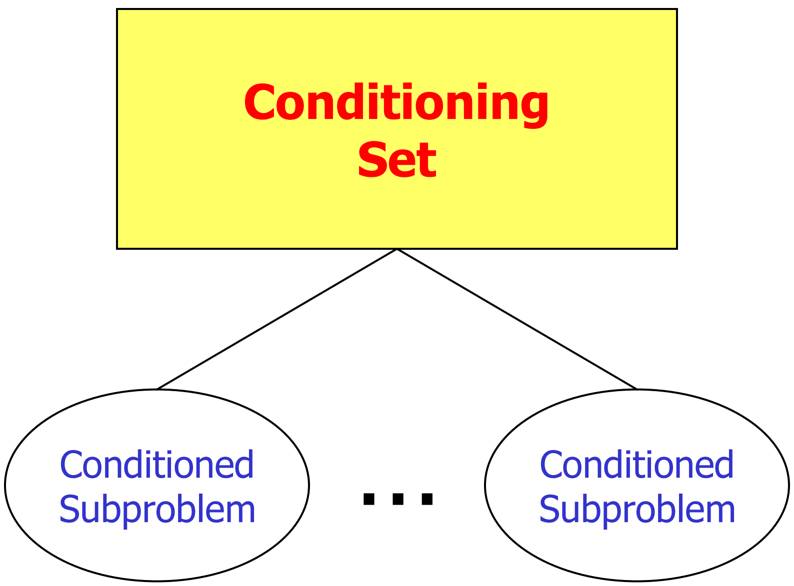


3-cutset

2-cutset

1-cutset

How to find small cutset?



- Complexity of cutset scheme:
  - Complexity of enumerating the conditioning set solutions

    multiplied by
  - Complexity of solving a conditioned subproblem

What is the best conditioning set?

Time-Space Complexity

- **Space**: O(exp(j))
  - j-cutset: a set that when removed the induced-width is j.
  - $c(j)$: size of j-cutset.
  - m(j): depth of AO j-cutset
- **Time**: O(exp($j+c(j)$)) on OR space
- **Time**: O(exp($j+m(j)$)) on AND/OR space and m(j) <= c(j)

Time-Space Complexity

- **Space**: O(exp(i))
  - ***i* - cutset: a set that when removed the induced-width is i.**
  - ***c*(*i*): size of i-cutset.**
  - **m(i): depth of AO i-cutset**
- **Time**: O(exp(i+c(i))) on OR space
- **Time**: O(exp(i+m(i))) on AND/OR space and m(i) <= c(i)

$$c(1)+1 \geq 2+c(2) \geq ... \geq i+c(i) \geq ...w^*+c(w^*) = tw^*$$

$$tw^* \leq m(i)+i \leq c(i)+i,$$

Summary: Hybrid Time/Space in Variable-models

- Cutset schemes
  - i-cutset
  - Condition-elim(i)
  - AO(i)
- Super-bucket(i)
  - AO(i) avoids some dead-caches
- Combine:
  - i-cutset in an i super-bucket…

Algorithms for AND/OR Space

- **Back-jumping** for CSPs
  (Gaschnig 1977), (Dechter 1990), (Prosser, Bayardo & Mirankar, 1995)
- **Pseudo-search re-arrangement**, for any CSP task
  (Freuder & Quinn 1985)
- **Recursive Conditioning**
  (Darwiche, 2001), explores the AND/OR tree or graph for any query
- **BTD: Searching tree-decompositions** for optimization
  (Jeagou & Terrioux, 2000)
- **Pseudo-tree search for soft constraints**
  (Larrosa, Meseguer & Sanchez, 2002)
- **Valued-elimination**
  (Bacchus, Dalmao & Pittasi, 2003)
- **Arc-consistency for soft constraints**
  (Larrosa & Schiex, 2003)

Conclusions

- **Only a few principles:**

1. Inference and search should be combined → time-space
2. AND/OR search should be used
3. Heuristics using approximation of inference (mini-bucket, GBP)
4. Caching in search should be used

Outline

- **Introduction**
  - Optimization tasks for graphical models
  - Solving optimization problems by inference and search
- **Inference**
  - Bucket elimination, dynamic programming
  - Mini-bucket elimination, belief propagation
- **Search**
  - Branch-and-Bound and Best-First search
  - Lower-bounding heuristics
  - AND/OR search spaces
- **Hybrids of search and inference**
  - Cutset decomposition
  - Super-bucket scheme
- **Software**

Software

- **Reports on competitions**
  - UAI'06 Inference Evaluation
    - 57 MPE instances
  - CP'06 Competition
    - 686 2-ary MAX-CSP instances
    - 135 n-ary MAX-CSP instances
- **How to use the software**
  - http://csp.ics.uci.edu/group/Software

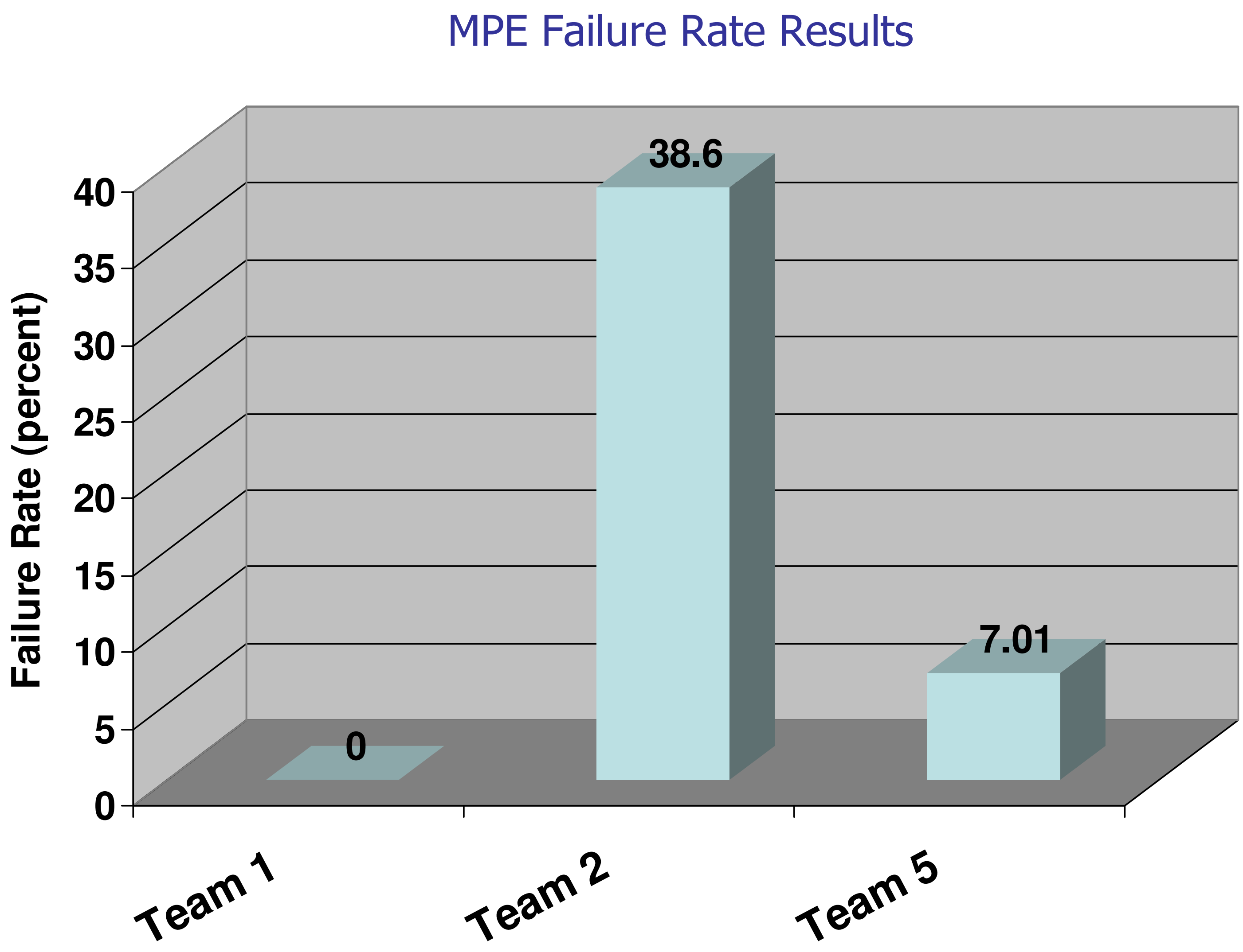
UAI'06 Inference Evaluation

- MPE solver – **AOMB(i, j)**
  - **Node value v(n)**: most probable explanation of the sub-problem rooted by n
  - **Caching**: identical sub-problems rooted at AND nodes (identified by their **contexts**) are solved once and the results cached
    - **j-bound** (context size) controls the memory used for caching
  - **Heuristics**: pruning is based on heuristics estimates which are pre-computed by bounded inference (i.e. mini-bucket approximation)
    - **i-bound** (mini-bucket size) controls the accuracy of the heuristic
  - **No constraint propagation**

UAI'06 Competitors

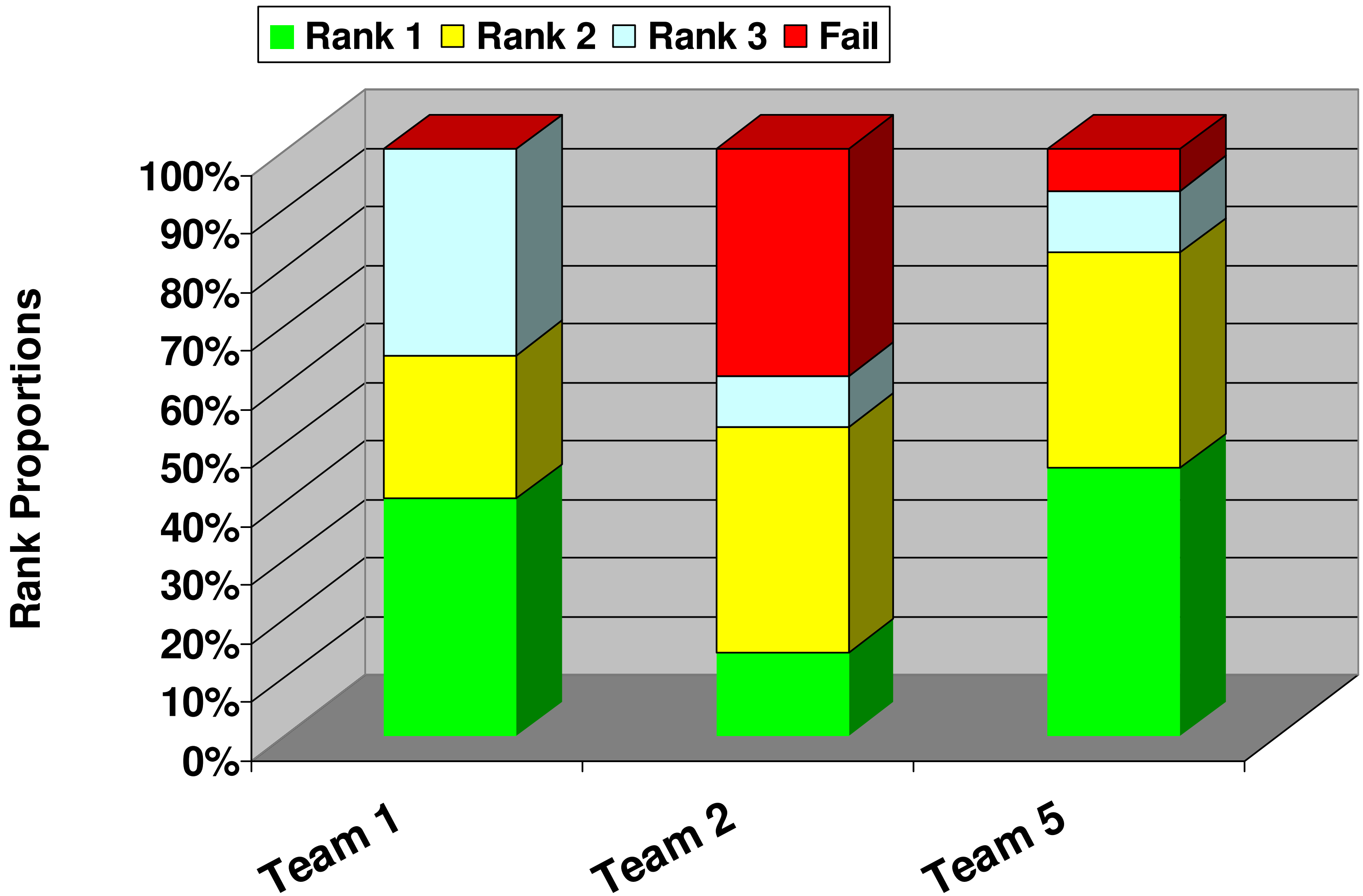
- **Team 1**
  - UCLA
    - David Allen, Mark Chavira, Arthur Choi, Adnan Darwiche
- **Team 2**
  - IET
    - Masami Takikawa, Hans Dettmar, Francis Fung, Rick Kissh
- **Team 5 (ours)**
  - UCI
    - Radu Marinescu, Robert Mateescu, Rina Dechter

UAI'06 Results



UAI'06 Results

Rank Proportions (how often was each team a particular rank, rank 1 is best)



CP'06 Competition

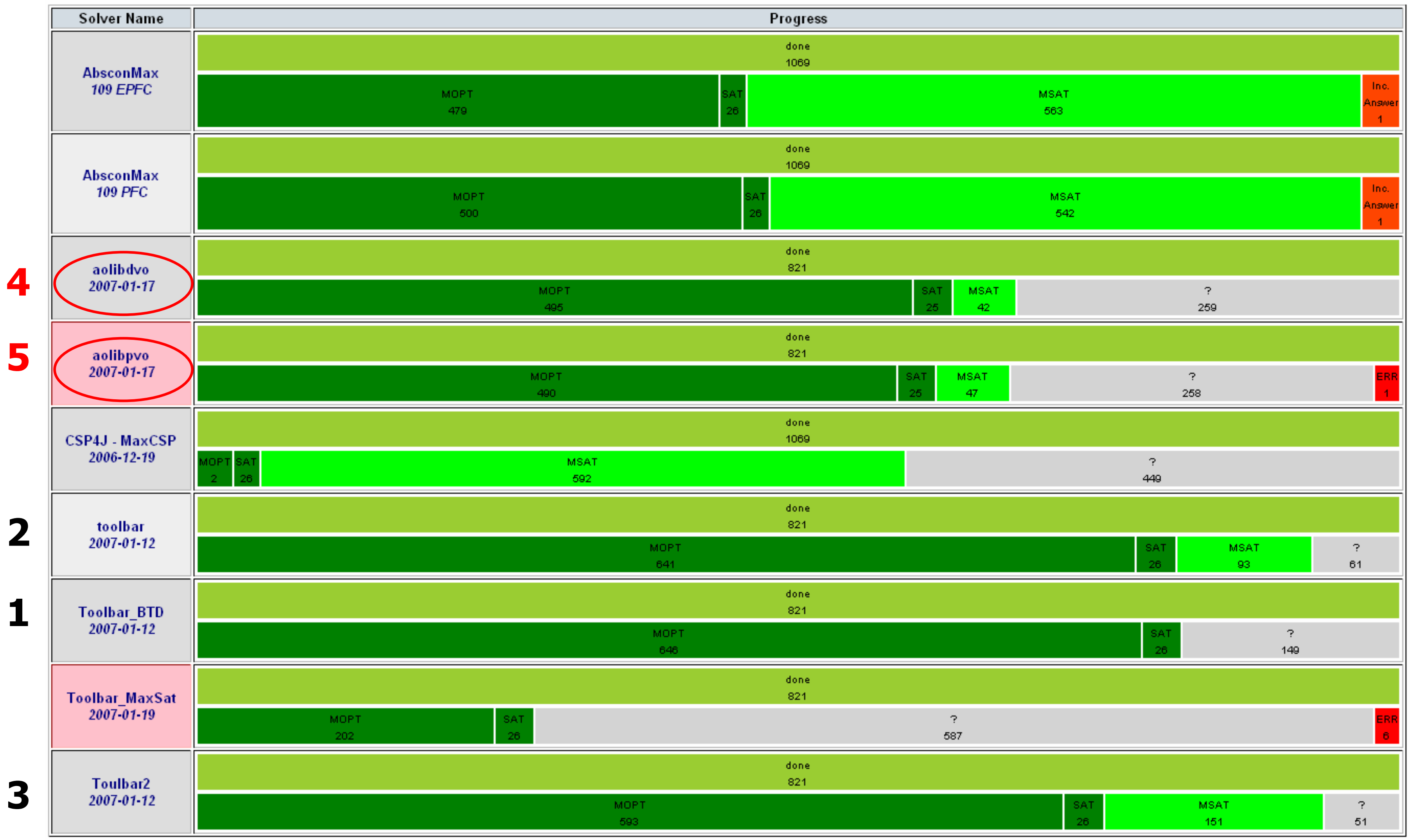
- **MAX-CSP solver**
  - AND/OR Branch-and-Bound **tree** search with EDAC heuristics and dynamic variable orderings
    - **aolibdvo** v. 0.5
      - **AOBB + EDAC + DVO** (dynamic variable ordering)
    - **aolibpvo** v. 0.5
      - **AOBB + EDAC + PVO** (partial variable ordering)
  - No constraint propagation

CP'06 Competitors

- Solvers
  - AbsconMax
  - aolibdvo (ours)
  - aolibpvo (ours)
  - CSP4J-MaxCSP
  - Toolbar
  - Toolbar_BTD
  - Toolbar_MaxSAT
  - Toulbar2

CP'06 Results

Overall ranking on all selected competition benchmarks



The longest dark green bar wins

Outline/Summary

- **Introduction**
  - Optimization tasks for graphical models
  - Planning as optimization
  - Solving optimization problems with inference and search
- **Inference**
  - Bucket Elimination, Dynamic Programming
  - Mini-Bucket Elimination
- **Search (OR)**
  - Branch-and-Bound and Best-First Search
  - Lower-bounding heuristics
- **AND/OR search spaces**
- **Hybrids of search and inference**
  - Cutset decomposition
  - Super-Bucket scheme
- **Software**