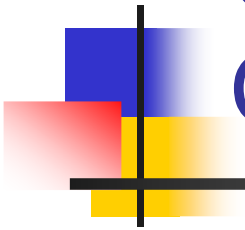
http://4c.ucc.ie/~rmarines/talks/tutorial-IJCAI-09-syllabus.pdf



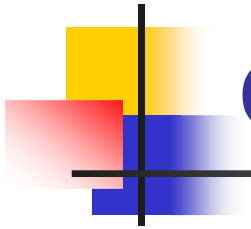
Combinatorial Optimization for Graphical Models

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

Radu Marinescu
Cork Constraint Computation Centre
University College Cork, Ireland

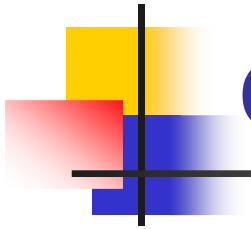
Simon de Givry & **Thomas Schiex**
Dept. de Mathématique et Informatique Appliquées
INRA, Toulouse, France

with contributed slides by Javier Larrosa (UPC, Spain)



Outline

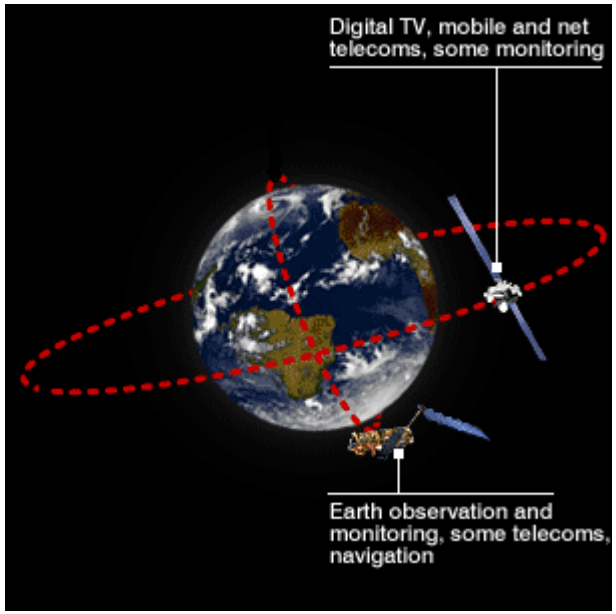
- **Introduction**
  - Graphical models
  - Optimization tasks for graphical models
- **Inference**
  - Variable Elimination, Bucket Elimination
- **Search (OR)**
  - Branch-and-Bound and Best-First Search
- **Lower-bounds and relaxations**
  - Bounded variable elimination and local consistency
- **Exploiting problem structure in search**
  - AND/OR search spaces (trees, graphs)
- **Software**



Outline

- **Introduction**
  - Graphical models
  - Optimization tasks for graphical models
  - Solving optimization problems by inference and search
- **Inference**
- **Search (OR)**
- **Lower-bounds and relaxations**
- **Exploiting problem structure in search**
- **Software**

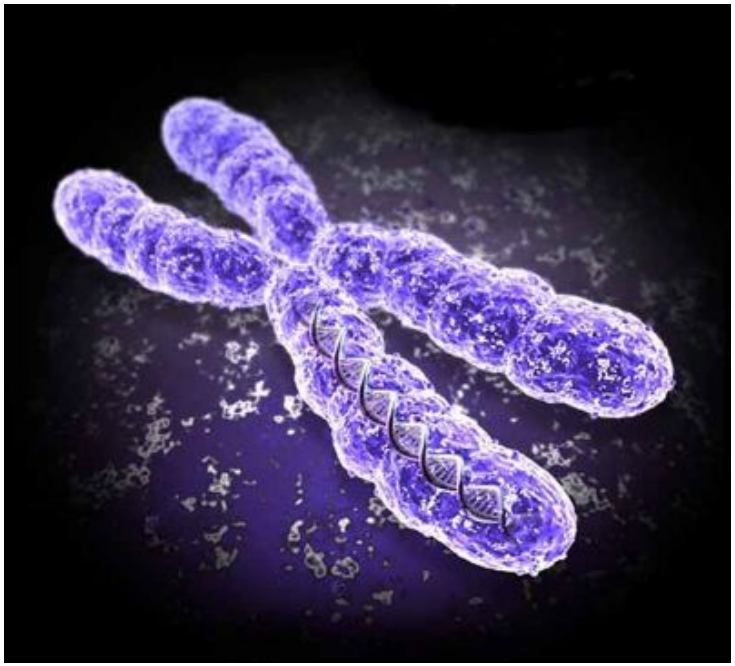
Combinatorial Optimization



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

Earn 8 cents per invested dollar such that the investment risk is minimized

Combinatorial Optimization

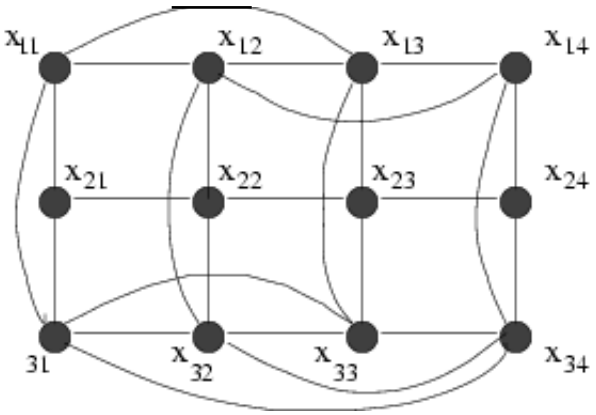


Assign frequencies to a set of radio links such that interferences are minimized

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

Constrained Optimization

Example: power plant scheduling



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

Variables $= \{X_1,...,X_n\}$, 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 $\text{TotalFuelCost}(X_1,..., X_N)$

Constraint Optimization Problems
for Graphical Models

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

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

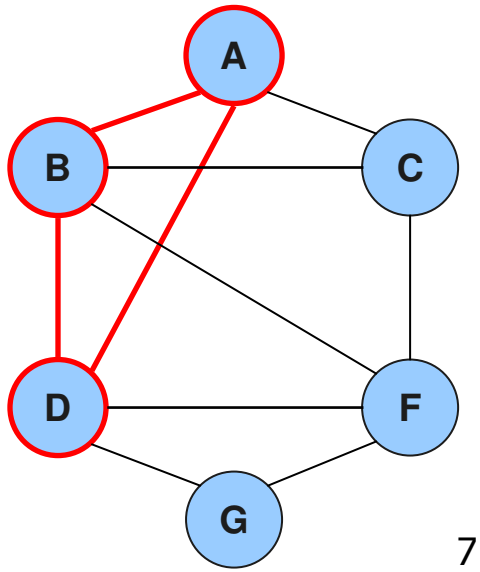
$D = \{D_1, \ldots, D_n\}$ - domains

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

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

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

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



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

Global Cost Function

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

Constraint Networks

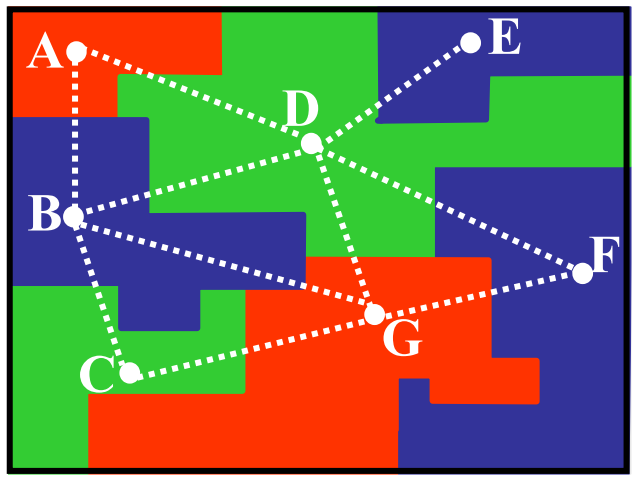
Map coloring

Variables: countries (A B C etc.)

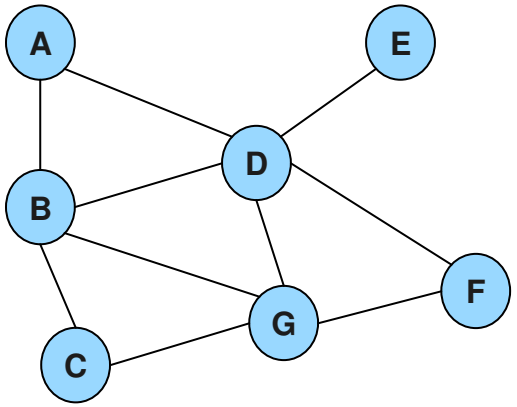
Values: colors (red green blue)

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

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



Constraint graph



Constraint Networks

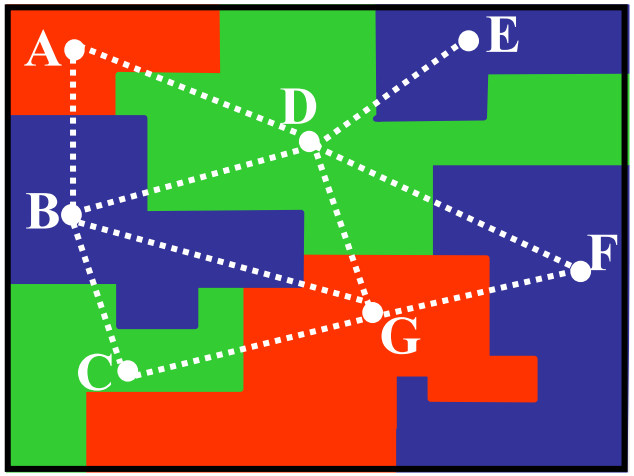
Map coloring

Variables: countries (A B C etc.)

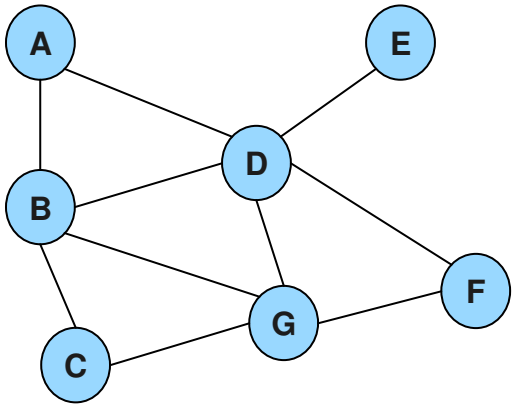
Values: colors (red green blue)

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

A	B	
red	green	0
red	yellow	0
green	red	0
green	yellow	0
yellow	green	0
yellow	red	0
Others		∞

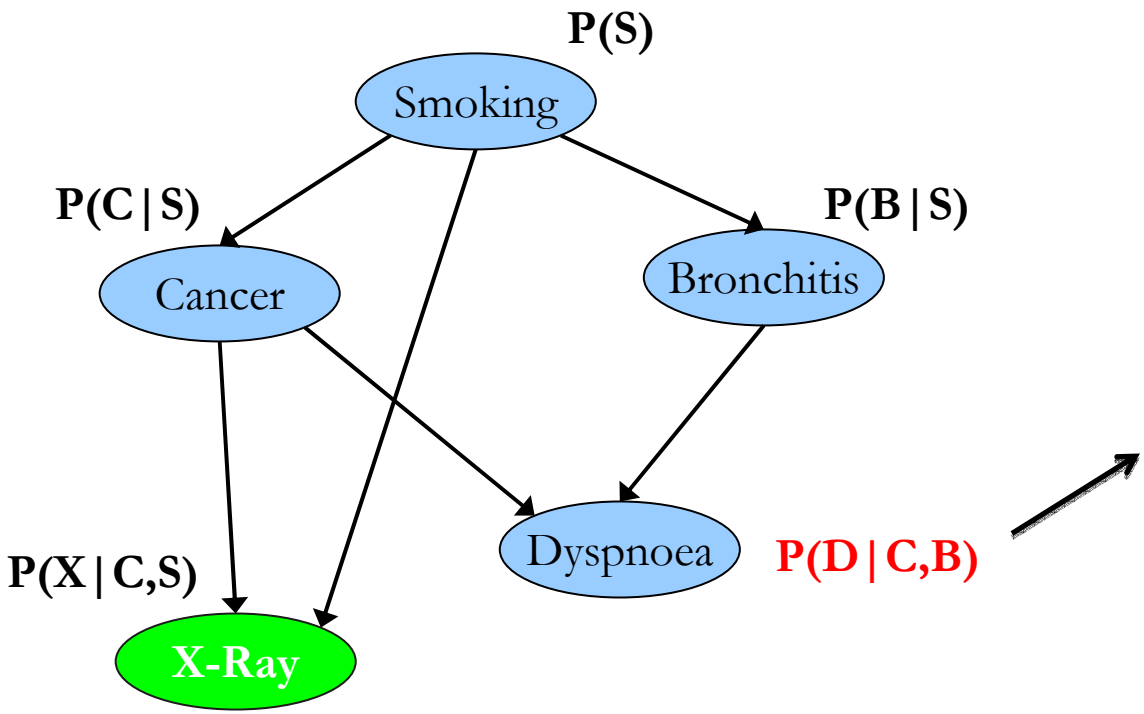


Constraint graph



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

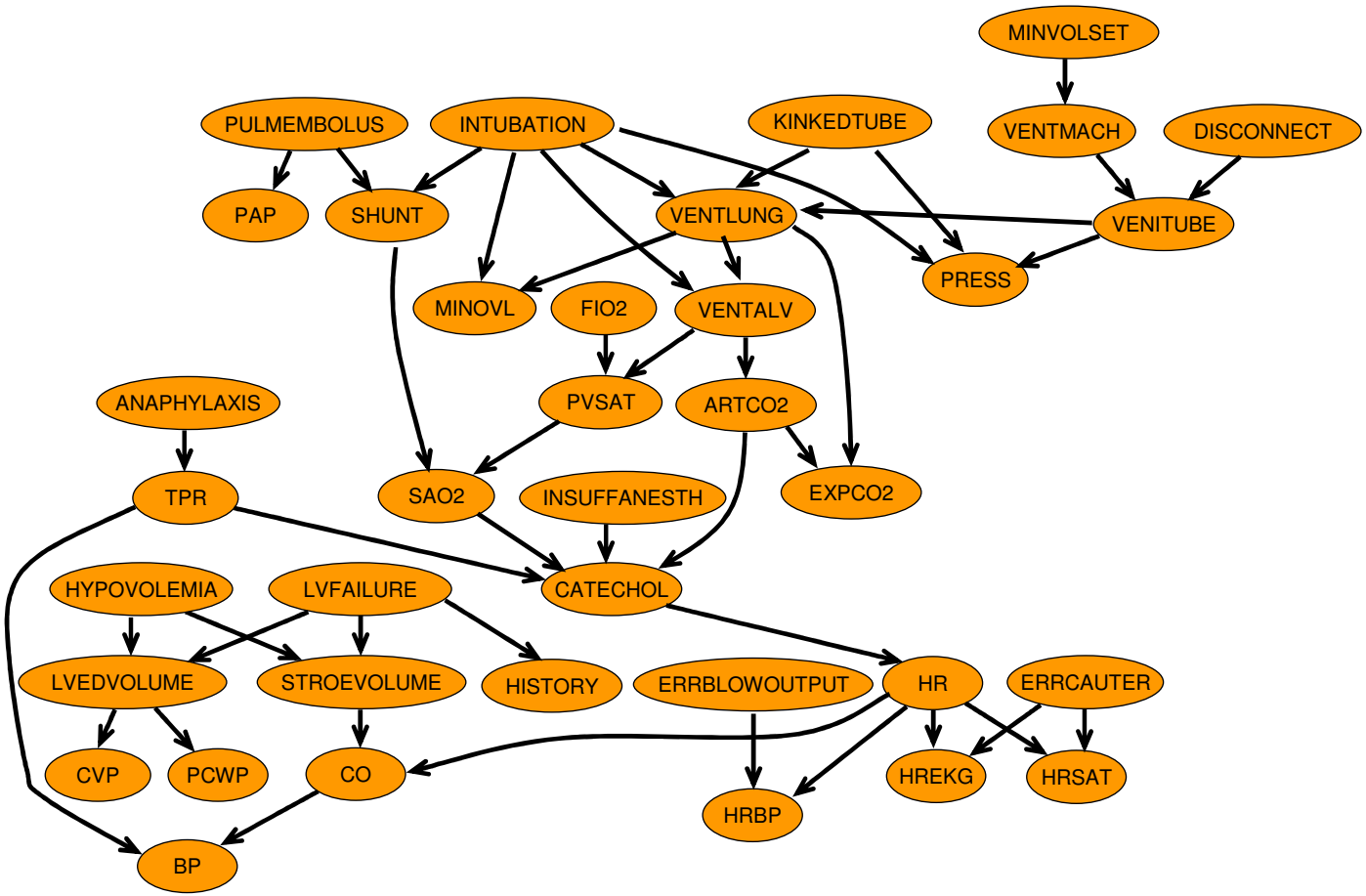
$$P(S,C,B,X,D) = P(S) \cdot P(C|S) \cdot P(B|S) \cdot P(X|C,S) \cdot P(D|C,B)$$

MPE= Find a maximum probability assignment, given evidence

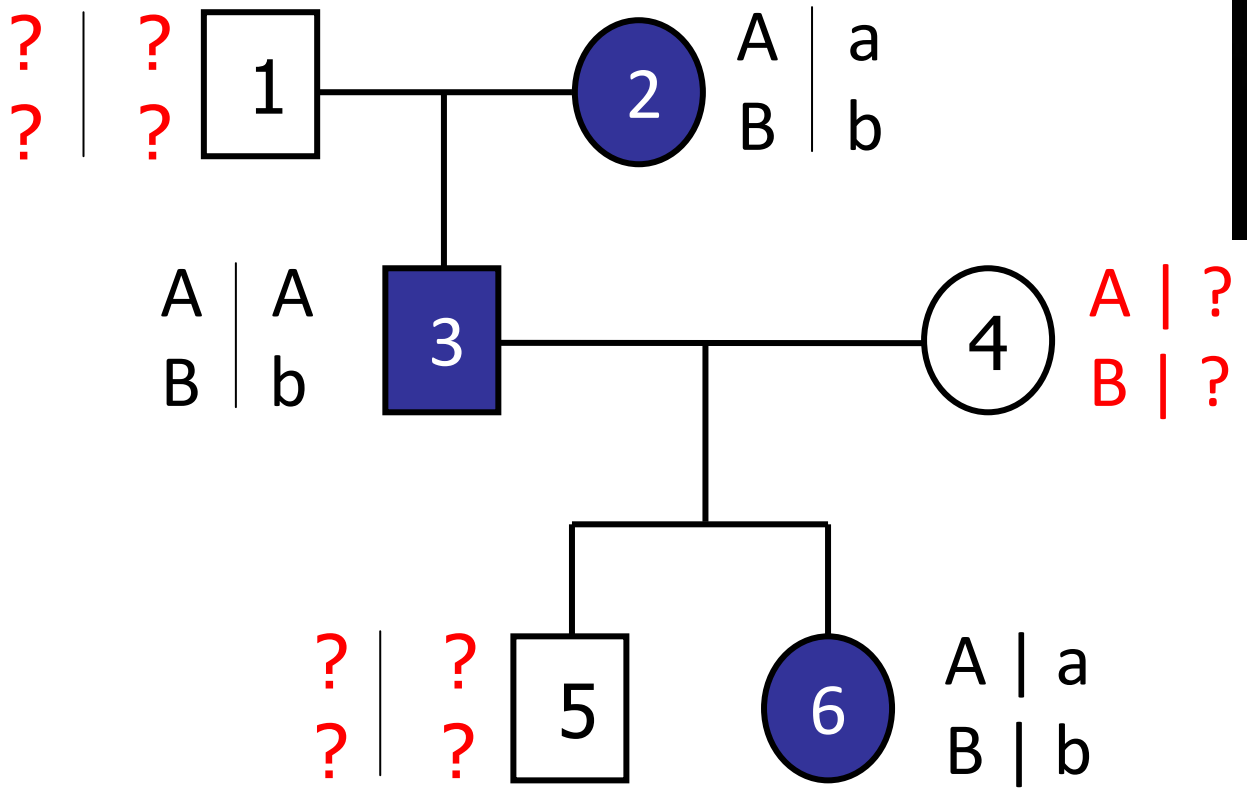
MPE= find argmax P(S)· P(C|S)· P(B|S)· P(X|C,S)· P(D|C,B)

Monitoring Intensive-Care Patients

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

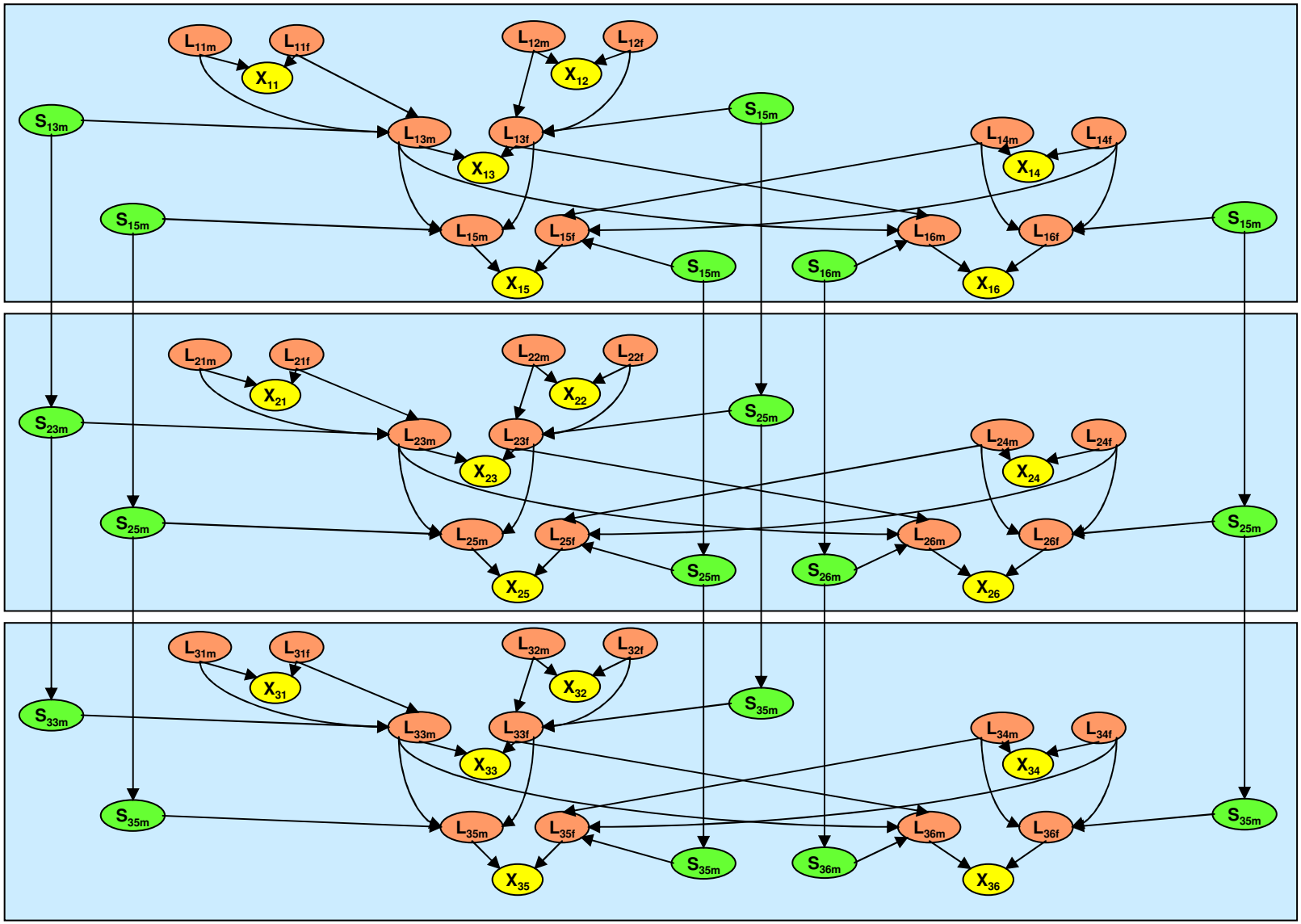


Linkage Analysis



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

Pedigree: 6 people, 3 markers

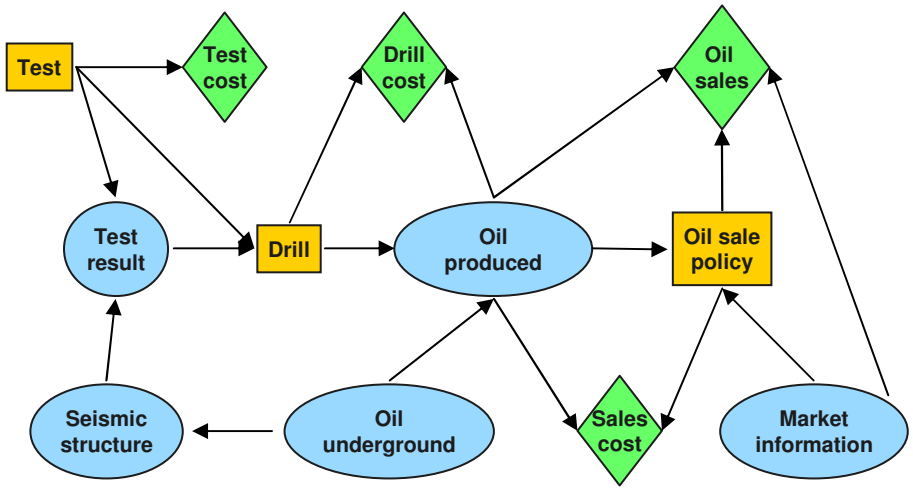


Influence Diagrams

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

Task: find optimal policy:

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



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

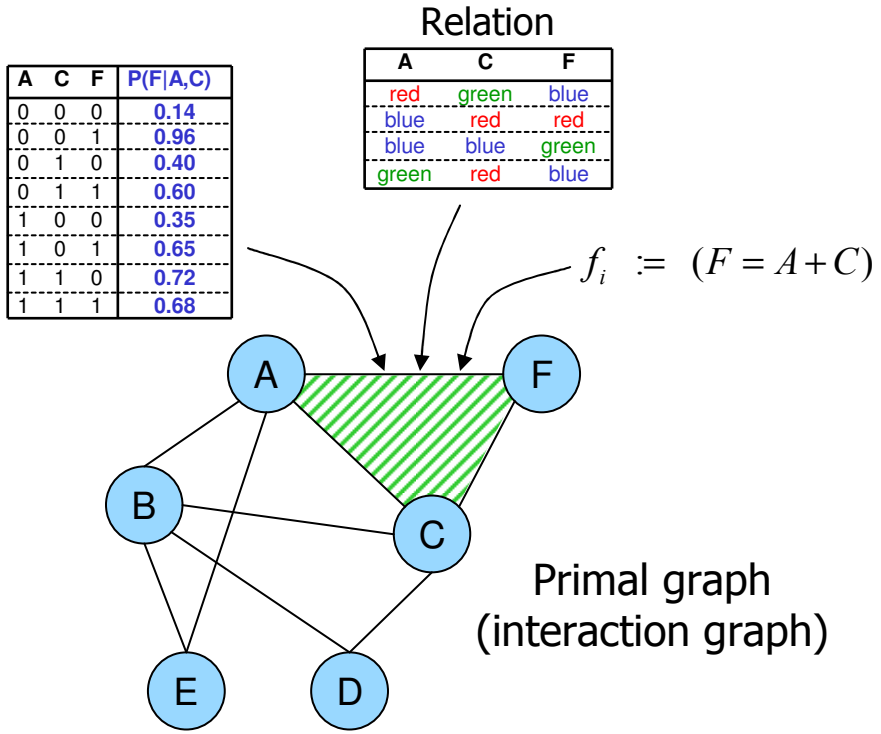
Utility function: $u = \sum_i r_i$

Graphical Models

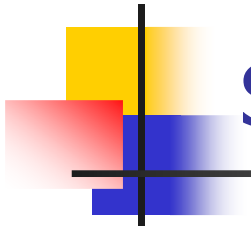
- A graphical model (**X**,**D**,**F**):
  - **X** = $\{X_1,\ldots X_n\}$ variables
  - **D** = $\{D_1, \ldots D_n\}$ domains
  - **F** = $\{f_1,\ldots,f_m\}$ functions
    (constraints, CPTS, CNFs …)

- 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**: $\min_X \Sigma_j f_j$



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



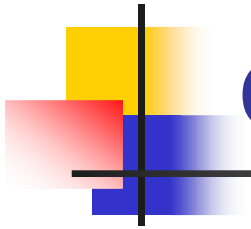
Sample Domains for Graphical M

- 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

...

Type of constrained optimization:

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



Outline

- **Introduction**
  - Graphical models
  - Optimization tasks for graphical models
  - Solving optimization problems by inference and search
- **Inference**
- **Search (OR)**
- **Lower-bounds and relaxations**
- **Exploiting problem structure in search**
- **Software**

Solution Techniques

AND/OR search

Time: exp(treewidth*log n)

Space: linear

Space: exp(treewidth)

Time: exp(treewidth)

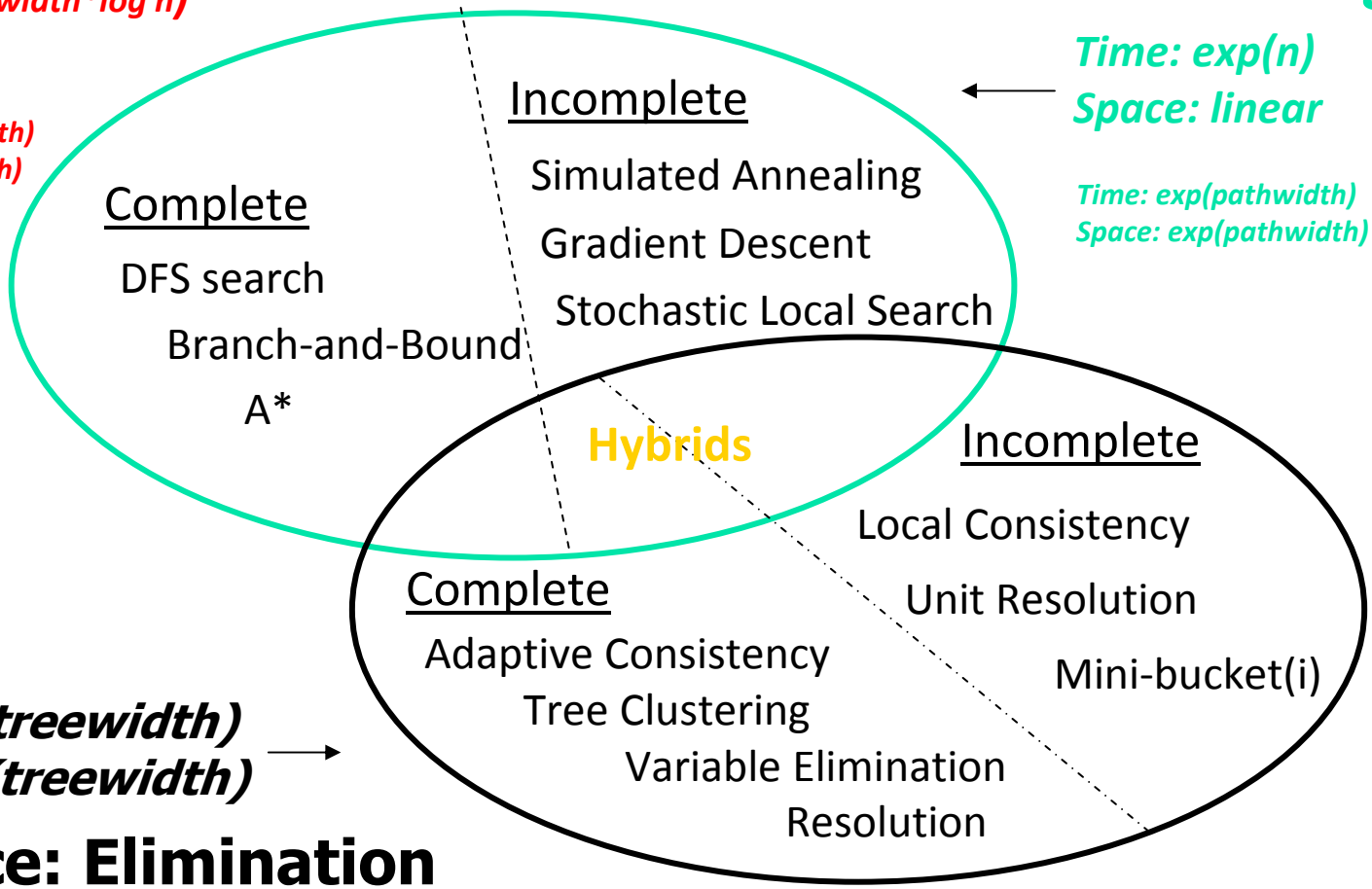
Search: Conditioning

Time: exp(n)

Space: linear

Time: exp(pathwidth)

Space: exp(pathwidth)



Time: exp(treewidth)

Space:exp(treewidth)

Inference: Elimination

Combination of cost functions

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

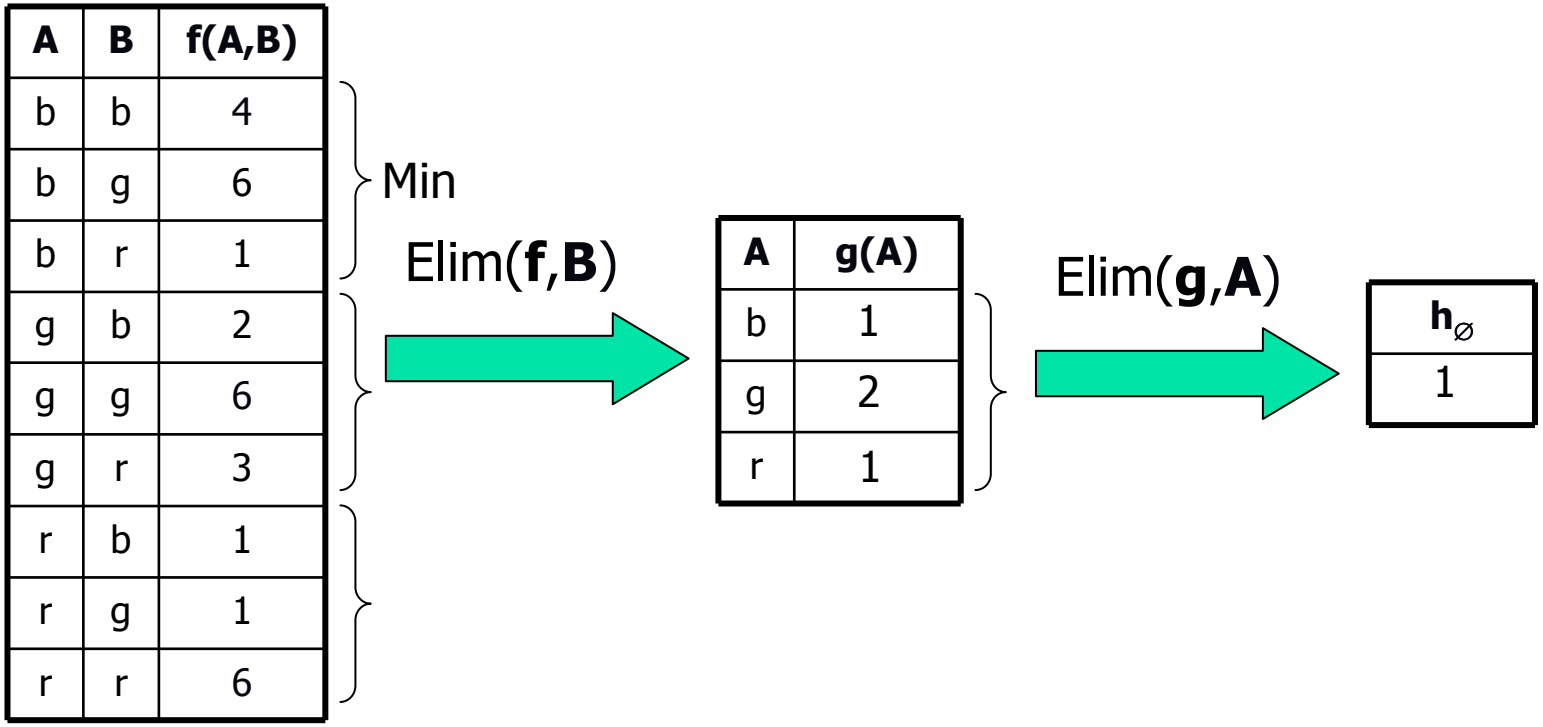
+

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

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

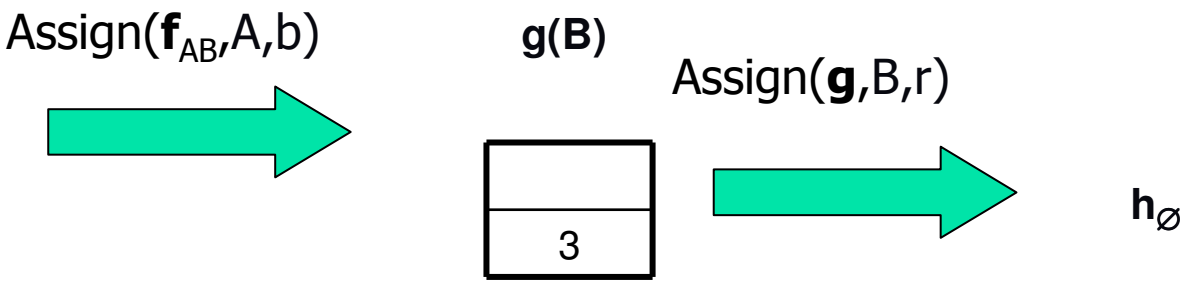
= 0 + 6

Elimination in a cost function



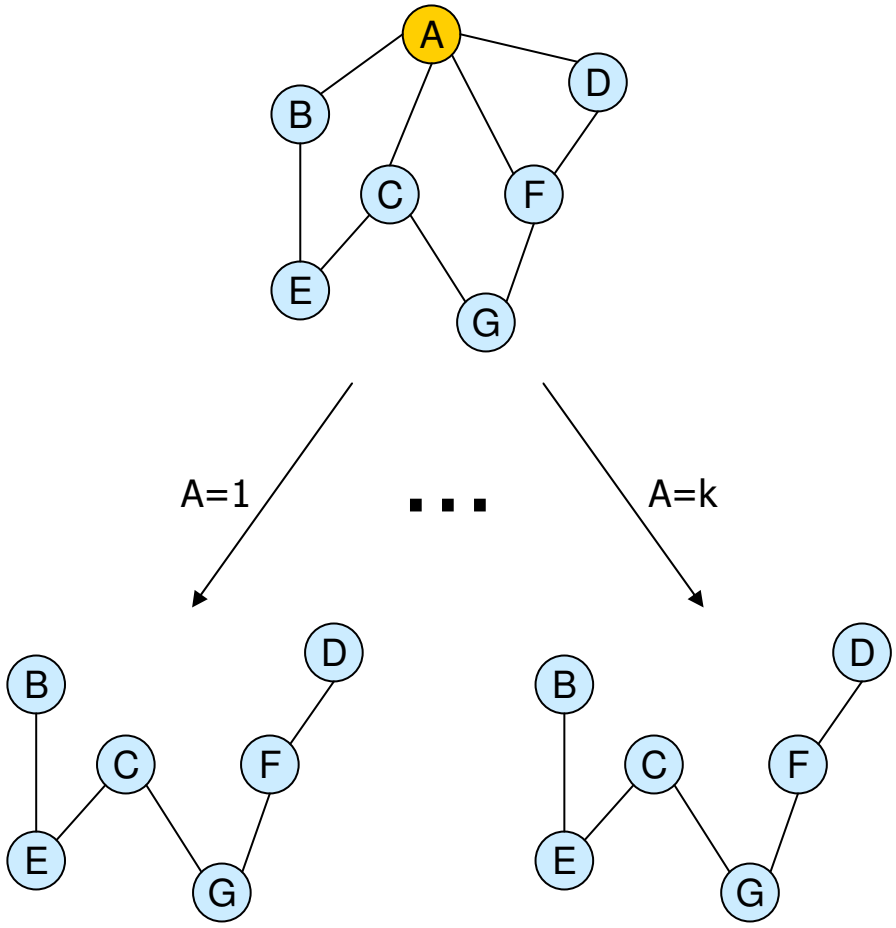
Conditioning a cost function

A	B	f(A,B)
b	b	6
b	g	0
b	r	3
g	b	0
g	g	6
g	r	0
r	b	0
r	g	0
r	r	6



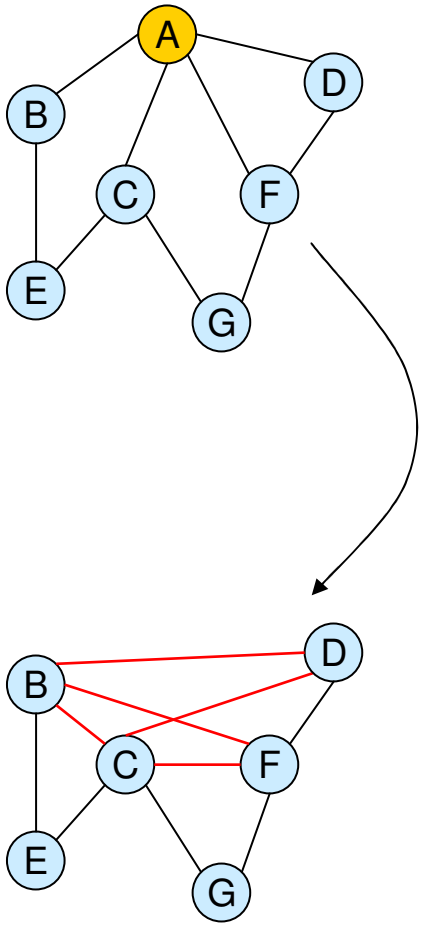
Conditioning vs. Elimination

Conditioning (search)

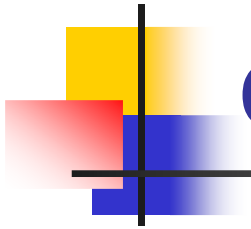


k "sparser" problems

Elimination (inference)



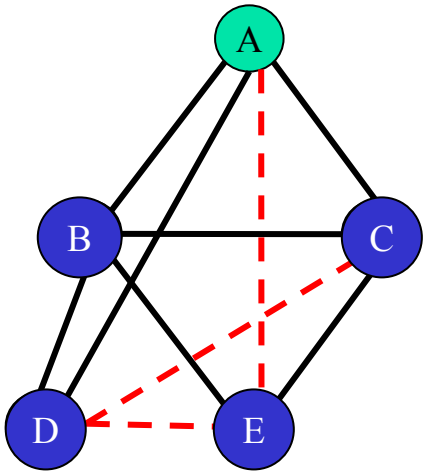
1 "denser" problem



Outline

- Introduction
- **Inference**
  - Variable Elimination, Bucket Elimination
- Search (OR)
- Lower-bounds and relaxations
- Exploiting problem structure in search
- Software

Computing the Optimal Cost Solution



Constraint graph

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

Combination

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

Variable Elimination

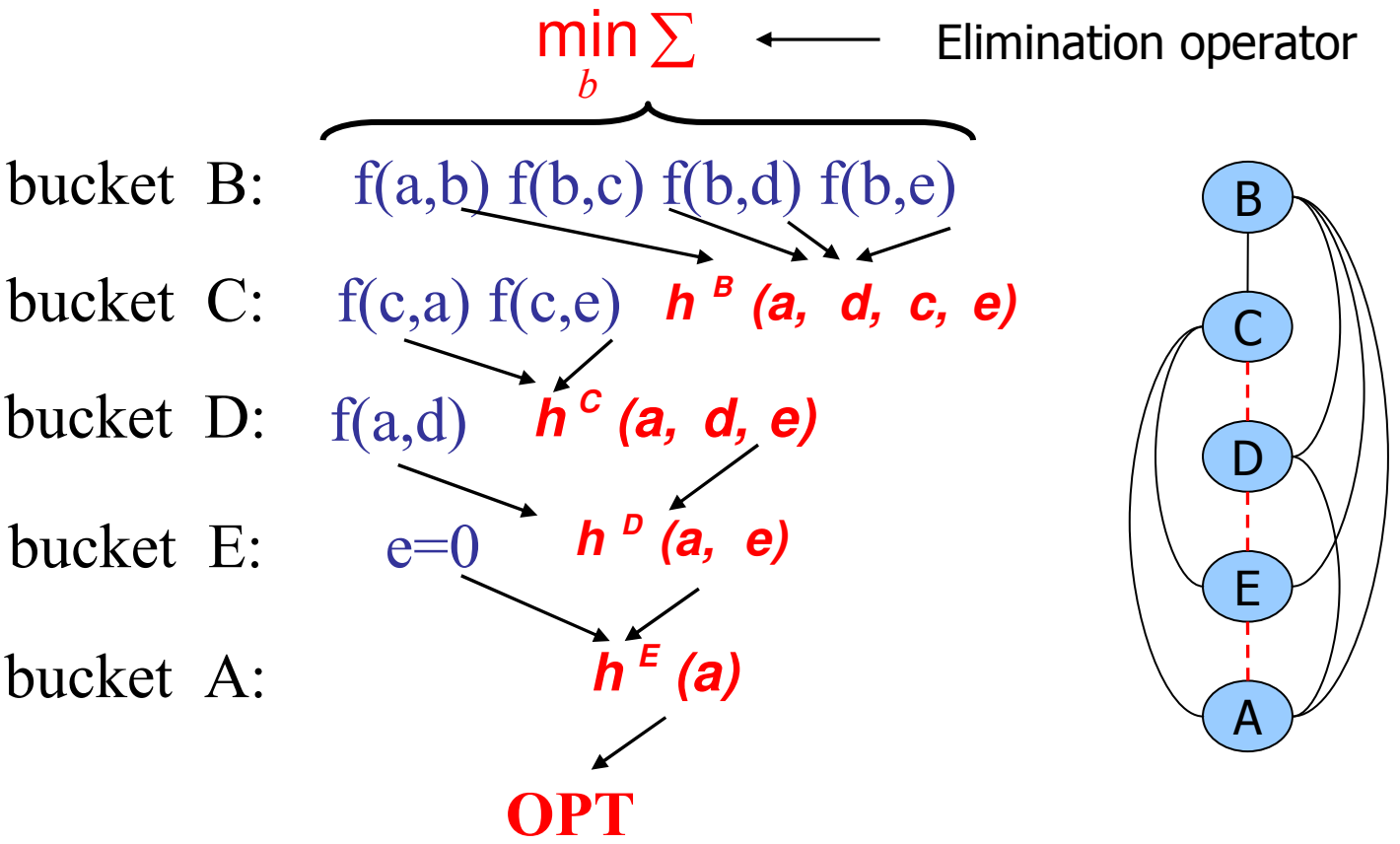
$$h^{B}(a,d,c,e)$$

Finding

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

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

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



Generating the Optimal Assignment

5. $b' = \arg\min_b f(a',b) + f(b,c') + f(b,d') + f(b,e')$

4. $c' = \arg\min_c f(c,a') + f(c,e') + h^B(a',d',c,e')$

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

2. $e' = 0$

1. $a' = \arg\min_a h^E(a)$

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 $h^D(a, e)$

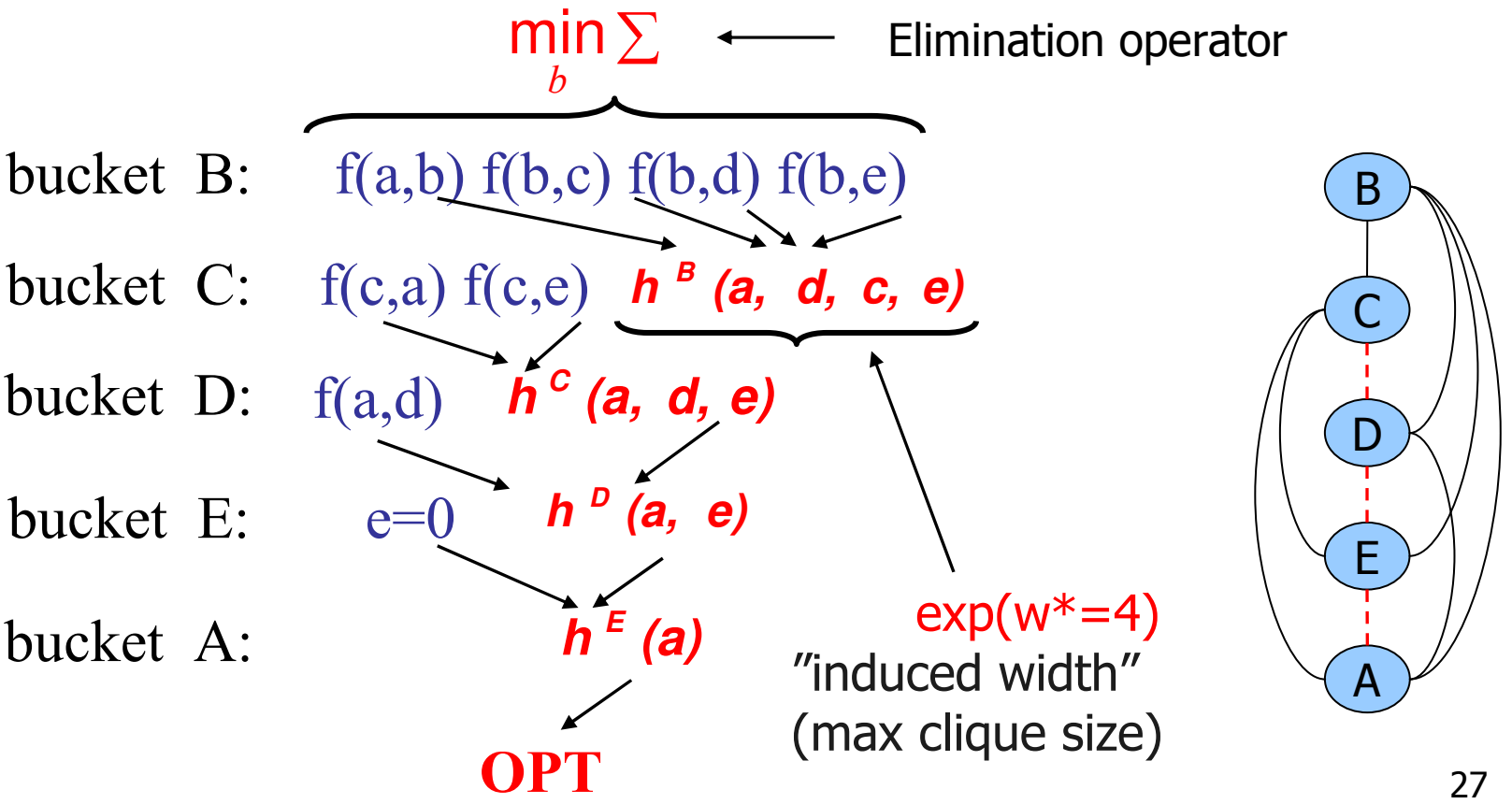
A: $h^E(a)$

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

Complexity

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

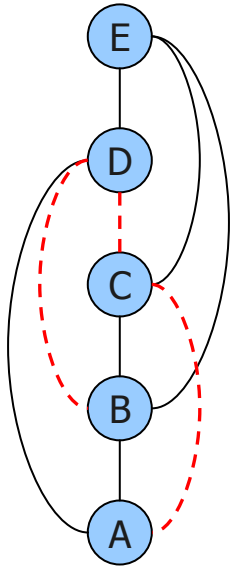
$$OPT = \min_{a,e,d,c,b} F(a,b) + F(a,c) + F(a,d) + F(b,c) + F(b,d) + F(b,e) + F(c,e)$$



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

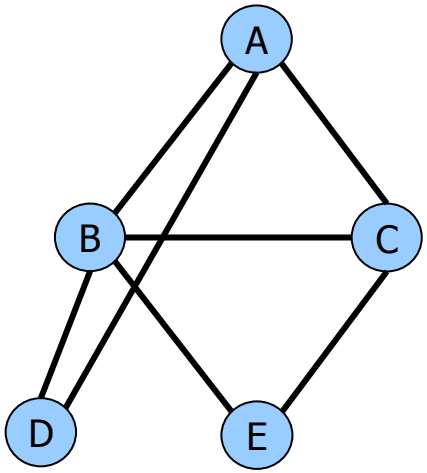
Bucket-Elimination is **time** and **space**

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

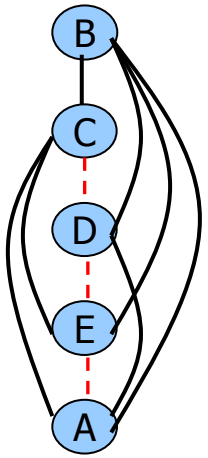
$w^*(d)$ – the induced width of the primal graph along ordering d

r = number of functions

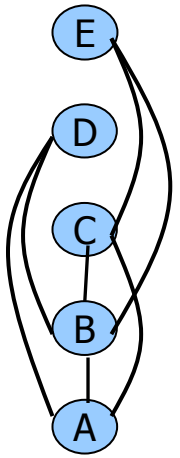
The effect of the ordering:



constraint graph

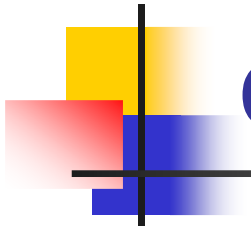


$w^*(d_1) = 4$



$w^*(d_2) = 2$

Finding smallest induced-width is hard!



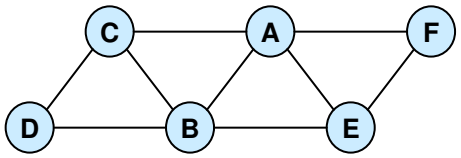
Outline

- Introduction
- Inference

- **Search (OR)**
  - Branch-and-Bound and Best-First search

- Lower-bounds and relaxations
- Exploiting problem structure in search
- 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**

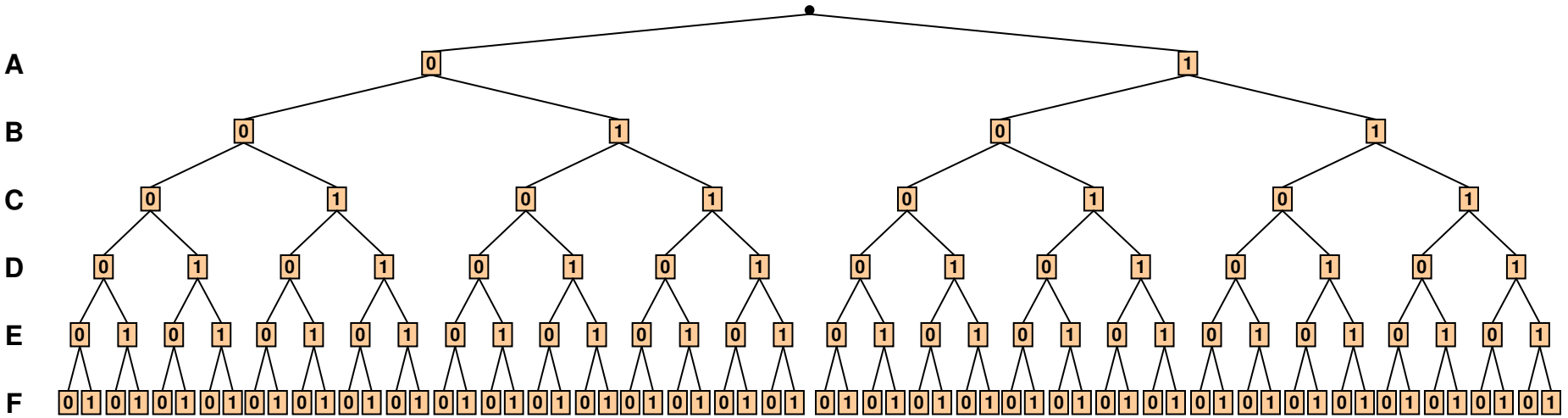
B	D	f_6
0	0	**4**
0	1	**2**
1	0	**1**
1	1	**0**

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

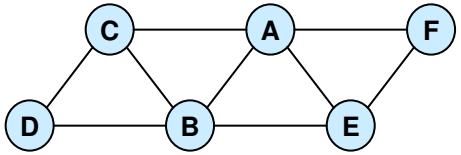
C	D	f_8
0	0	**1**
0	1	**4**
1	0	**0**
1	1	**0**

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

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



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

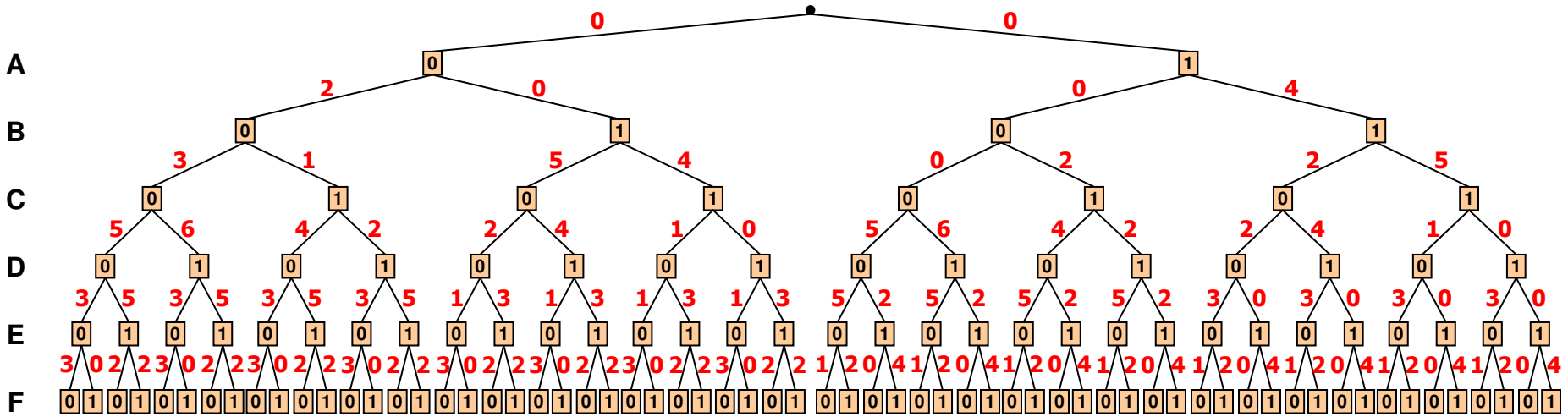
B	D	f_6
0	0	4
0	1	2
1	0	1
1	1	0

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

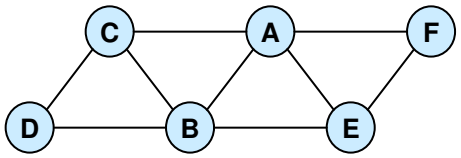
E	F	f_9
0	0	1
0	1	0
1	0	0
1	1	2

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



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

The Value Function



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

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

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

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

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

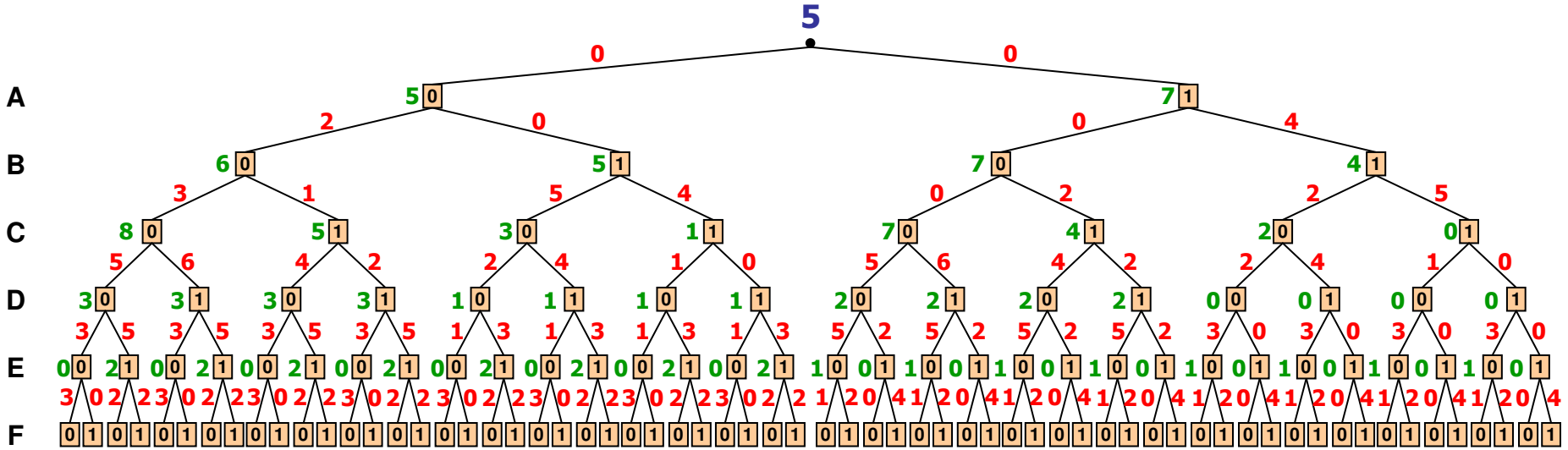
B	D	f_6
0	0	4
0	1	2
1	0	1
1	1	0

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

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

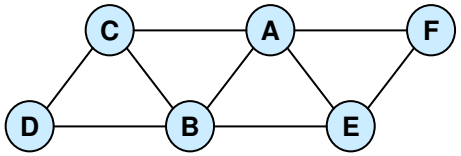
E	F	f_9
0	0	1
0	1	0
1	0	0
1	1	2

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



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**

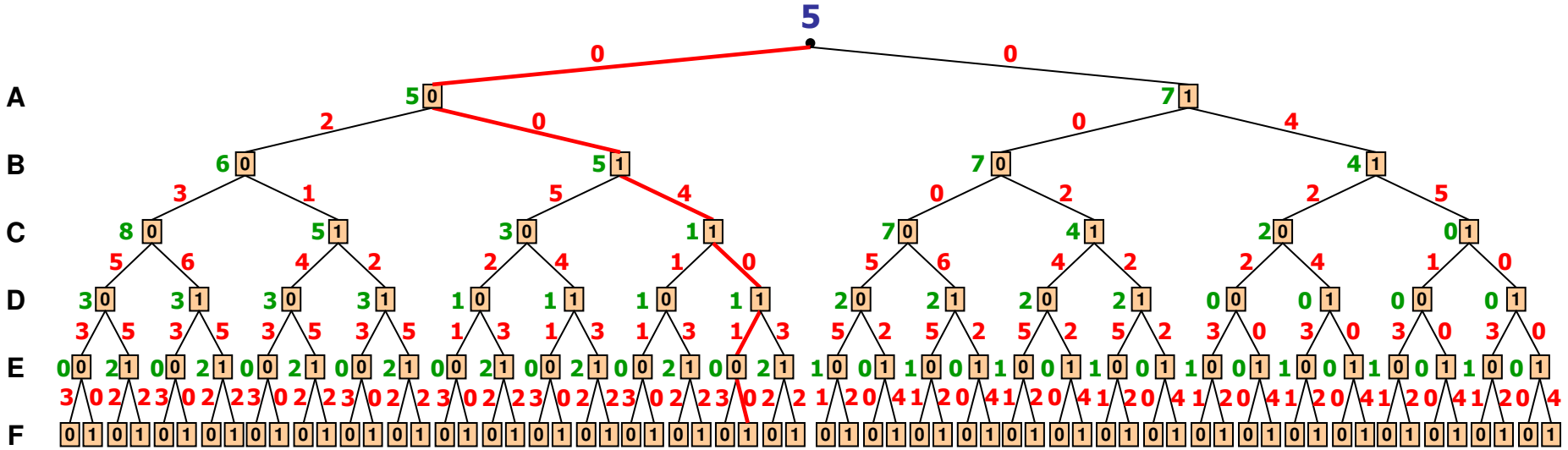
B	D	f_6
0	0	**4**
0	1	**2**
1	0	**1**
1	1	**0**

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

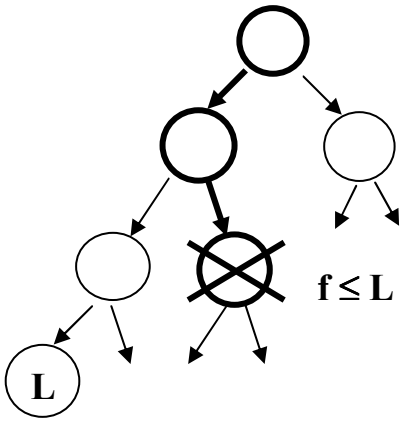
Basic Heuristic Search Schemes

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

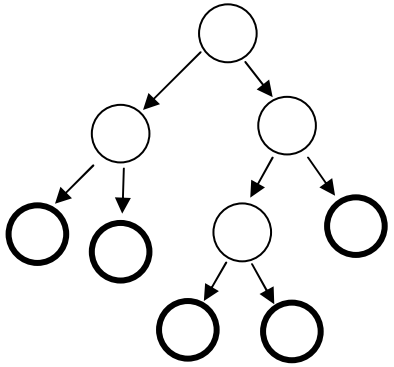
Linear space



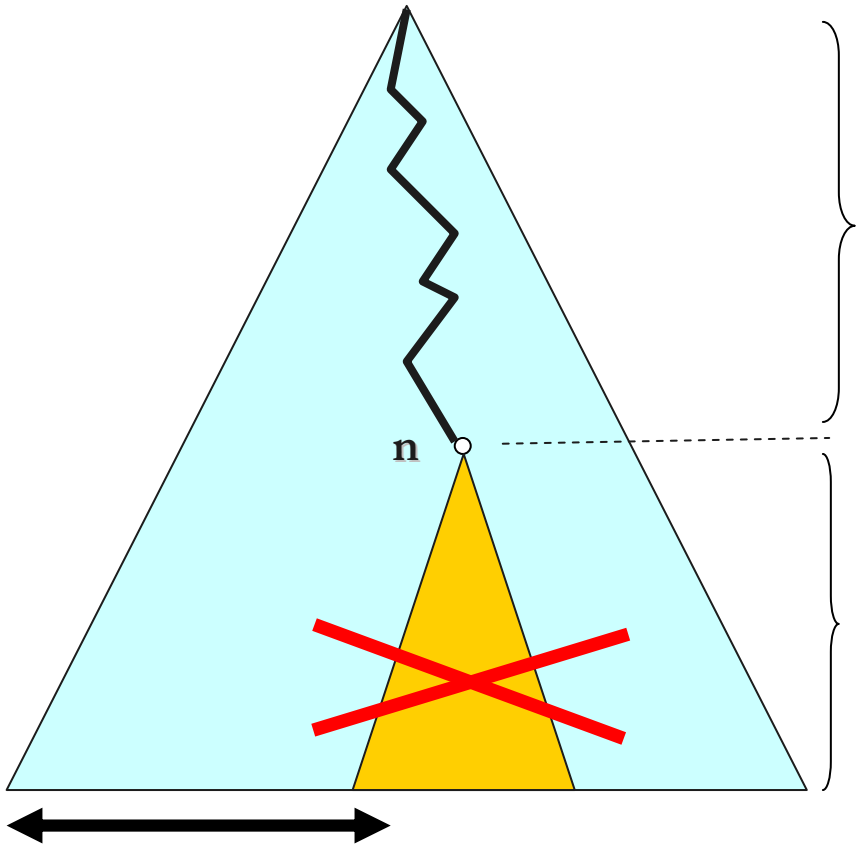
2. Best-First Search

Always expand the node with the highest heuristic value **$f(x^p)$**

Needs lots of memory



Classic Branch-and-Bound



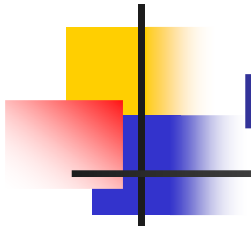
Each node is a COP subproblem
(defined by current conditioning)

f(n) = g(n) + h(n)
f(n) = lower bound

Prune if f(n) ≥ UB

h(n) - under-estimates
Optimal cost below n

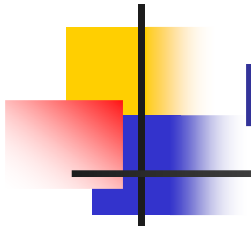
(UB) Upper Bound = best solution so far



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

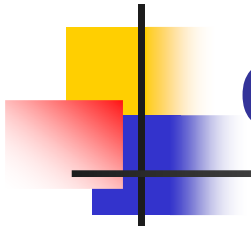
- **Best-First (A*): (optimal)**
  - Expand least number of nodes given h
  - Requires to store all search tree

- **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



How to Generate Heuristics

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



Outline

- Introduction
- Inference
- Search (OR)

- **Lower-bounds and relaxations**
  - **Bounded variable elimination**
    - Mini-Bucket Elimination
    - Generating heuristics using mini-bucket elimination
  - **Local consistency**

- Exploiting problem structure in search
- Software

Mini-Bucket Approximation

Split a bucket into mini-buckets => bound complexity

bucket (X) =
{ h_1, …, h_r, h_{r+1}, …, h_n }

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

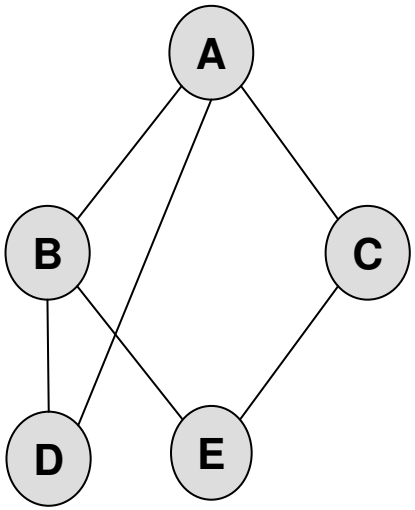
{ h_1, …, h_r } **{ h_{r+1}, …, h_n }**

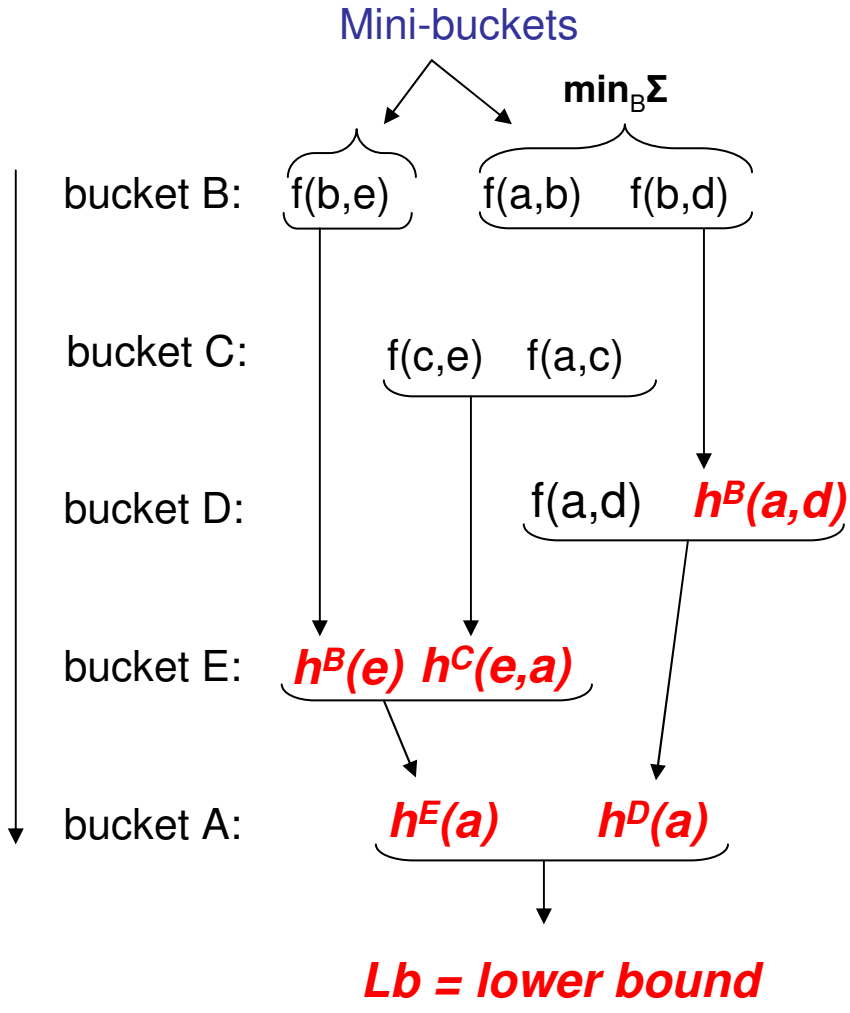
$$g^X = \left(\min_X \sum_{i=1}^{r} h_i \right) + \left(\min_X \sum_{i=r+1}^{n} h_i \right)$$

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

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

Mini-Bucket Elimination

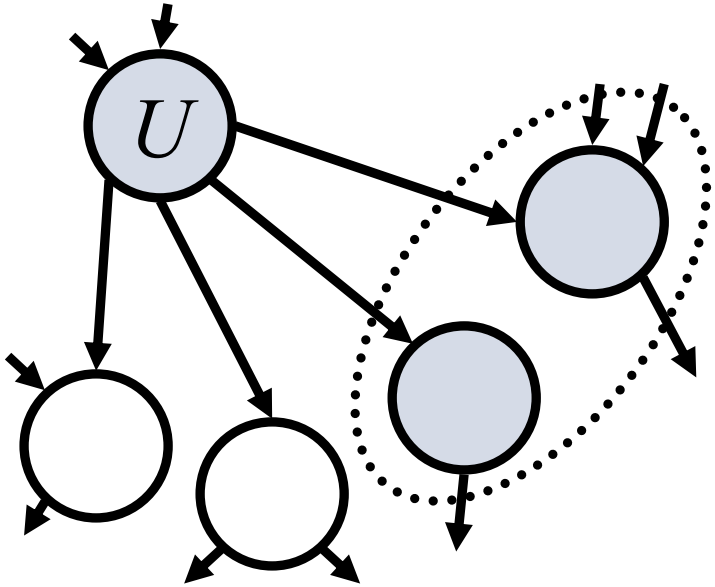




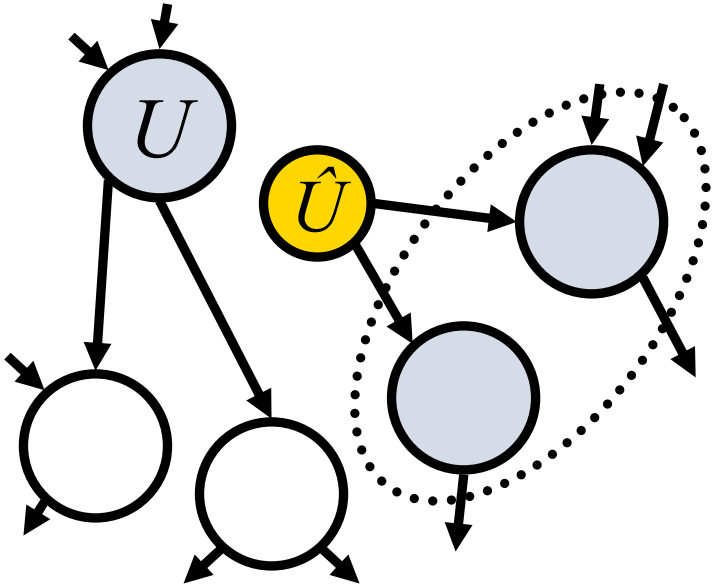
Semantics of Mini-Bucket: Splitting a Node

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

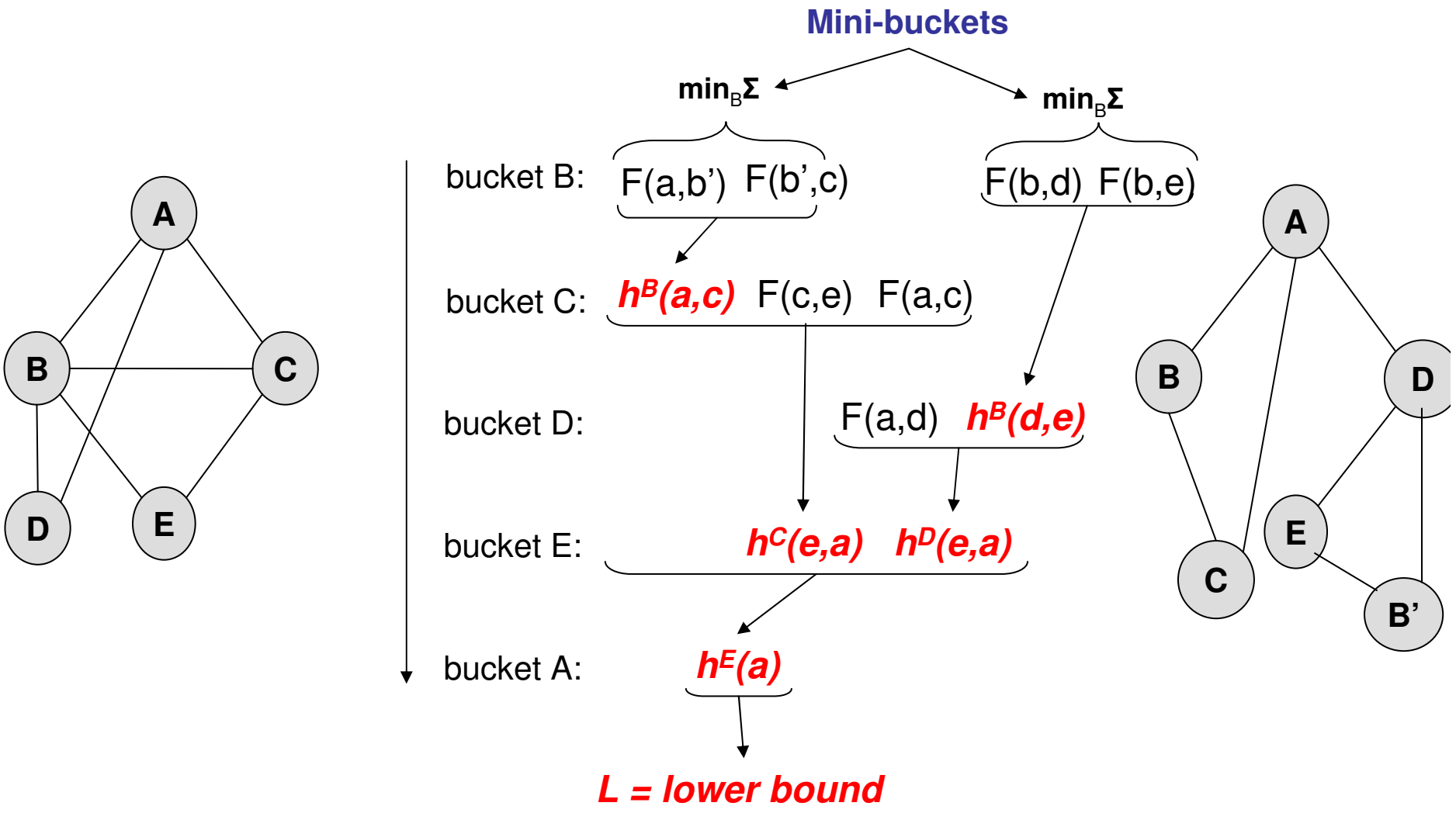
Before Splitting:
Network *N*



After Splitting:
Network *N'*



Mini-Bucket Elimination semantic

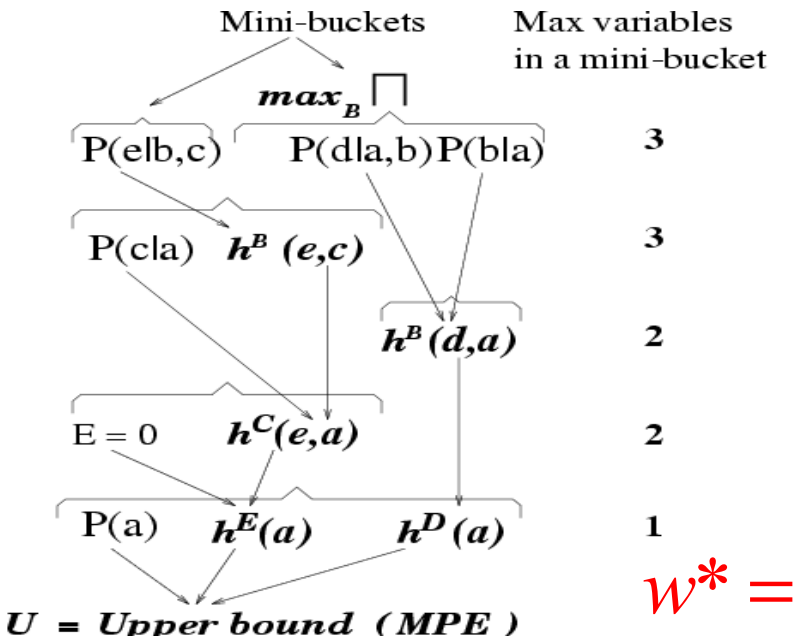


MBE-MPE(i)

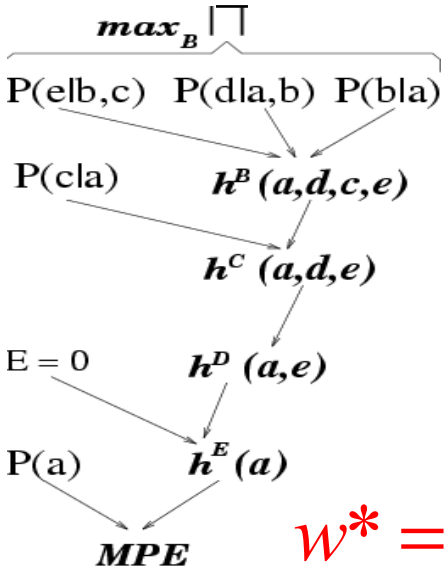
Algorithm **Approx-MPE** (Dechter & Rish, 1997)

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

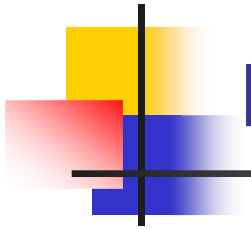
Example: **approx-mpe(3)** versus **elim-mpe**



$w^* = 2$



$w^* = 4$



Properties of MBE(i)

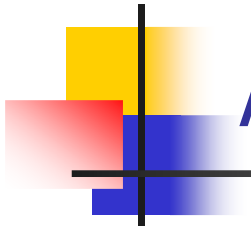
- **Complexity:** *O(r exp(i))* time and *O(exp(i))* space
- Yields an upper-bound and a lower-bound

- **Accuracy:** determined by upper/lower (U/L) bound

- As ***i*** increases, both accuracy and complexity increase

- Possible use of mini-bucket approximations:
  - As anytime algorithms
  - As heuristics in search

- Other tasks: similar mini-bucket approximations for:
  - Belief updating, MAP and MEU (Dechter & 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)*

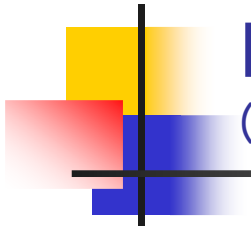
$L \leftarrow$ lower bound computed by *approx - mpe(i)*

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



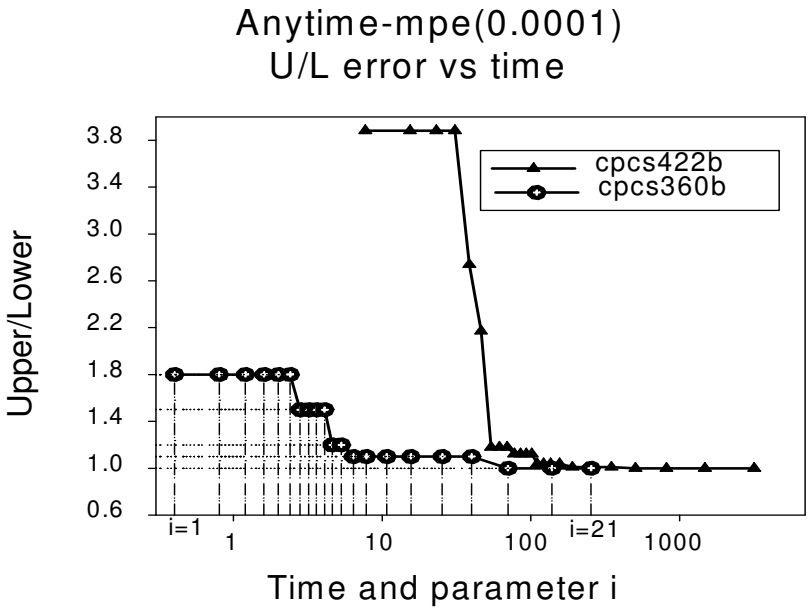
Empirical Evaluation
(Rish & Dechter, 1999)

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

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

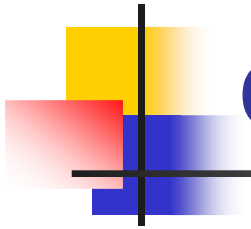
CPCS networks – medical diagnosis (noisy-OR model)

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
- Inference
- Search (OR)

- **Lower-bounds and relaxations**
  - **Bounded variable elimination**
    - Mini-Bucket Elimination
    - Generating heuristics using mini-bucket elimination
  - Local consistency

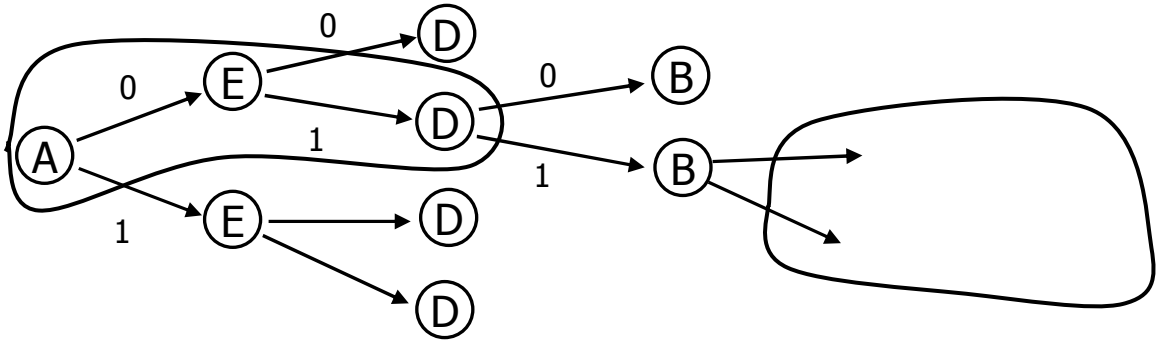
- Exploiting problem structure in search
- Software

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

Given a cost function

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

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



$$f^*(a,e,D) = \min_{b,c} F(a,B,C,D,e) =$$
$$= f(a) + \min_{b,c} f(B,a) + f(C,a) + f(e,B,C) + f(D,a,B)$$
$$= g(a,e,D) \bullet H^*(a,e,D)$$

Generating Heuristics (cont.)

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

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

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

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

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

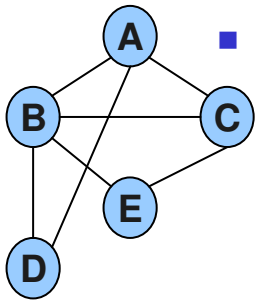
$$= H(a,e,d)$$

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

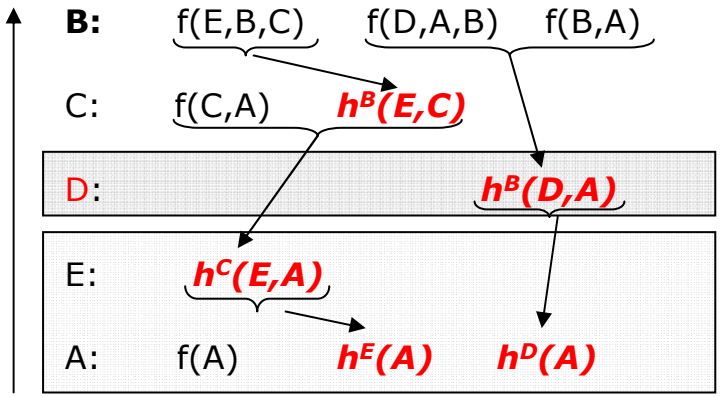
The heuristic function H is what is compiled during the preprocessing stage of the Mini-Bucket algorithm.

Static MBE Heuristics

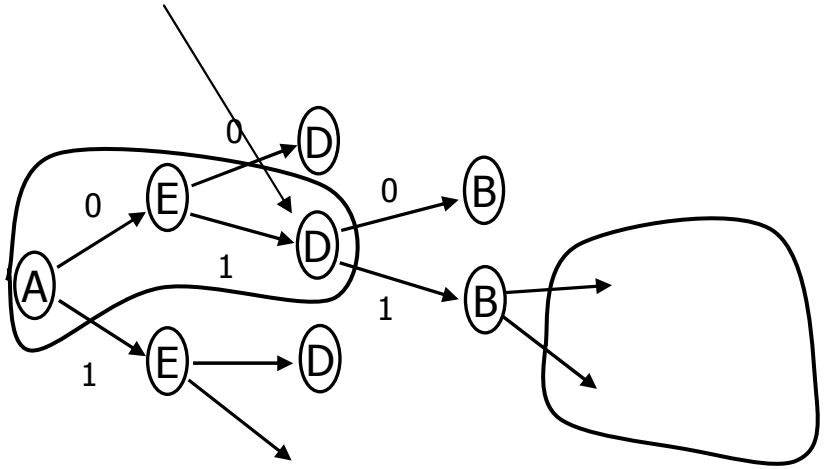
- Given a partial assignment $\mathbf{x^p}$, estimate the cost of the best extension to a full solution
- The evaluation function $\mathbf{f(x^p)}$ can be computed using function recorded by the Mini-Bucket scheme



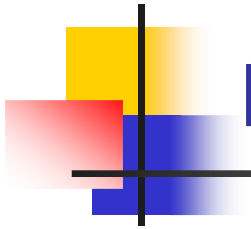
Cost Network



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



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

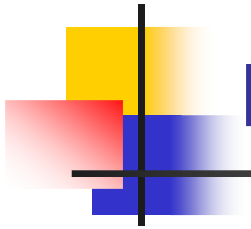


Heuristics Properties

- 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

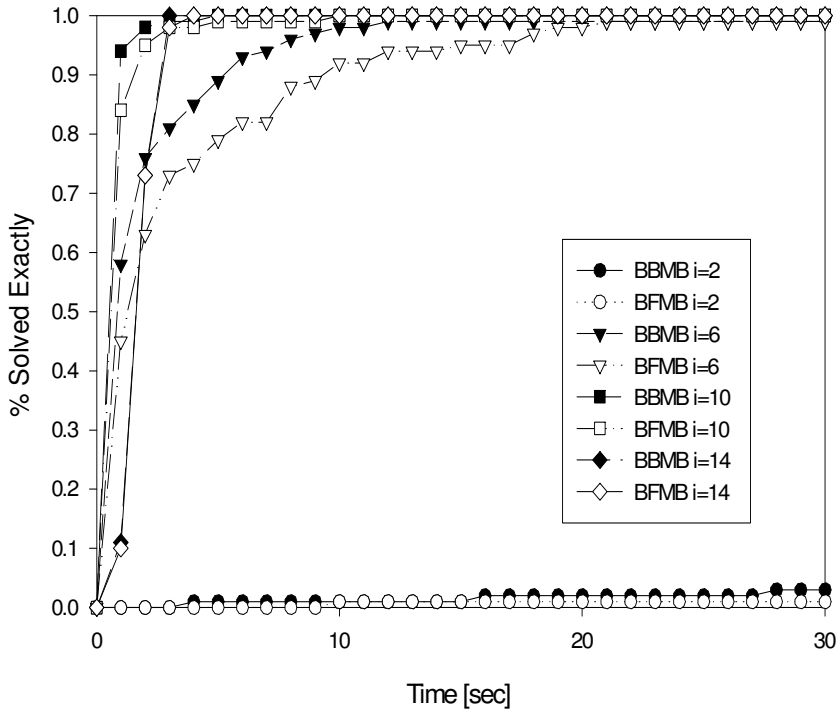


Experimental Methodology

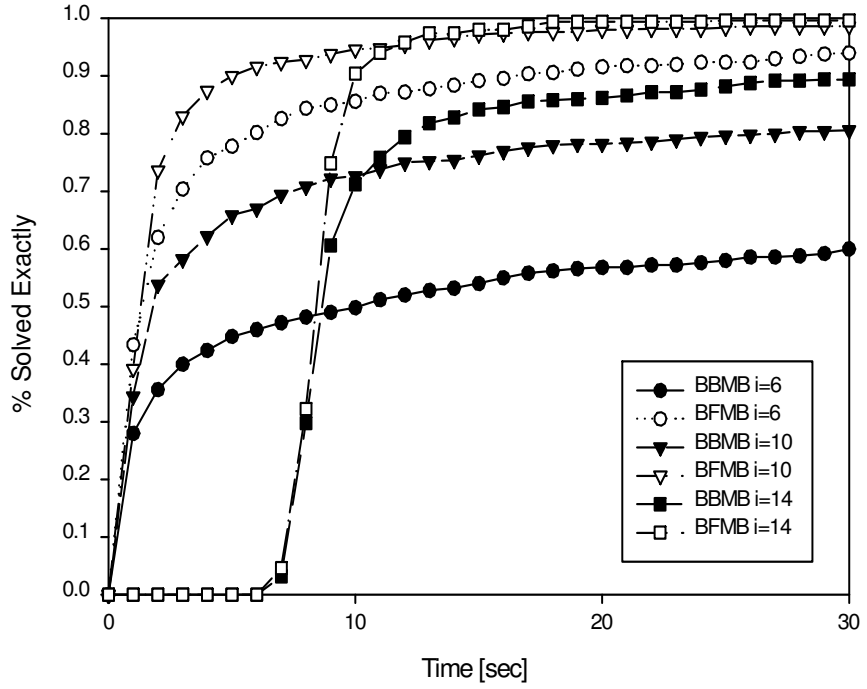
- **Algorithms**
  - BBMB(i) - Branch-and-Bound with MB(i)
  - BBFB(i) - Best-First with MB(i)
  - MBE(i) – Mini-Bucket Elimination
- **Benchmarks**
  - Random Coding (Bayesian)
  - CPCS (Bayesian)
  - Random (CSP)
- **Measures of performance**
  - Compare accuracy given a fixed amount of time
    - 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, Aij 2000)

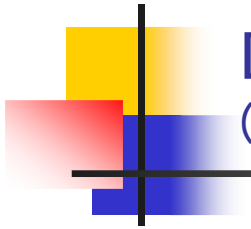
Random Coding, K=100, noise=0.28



Random Coding, K=100, noise=0.32



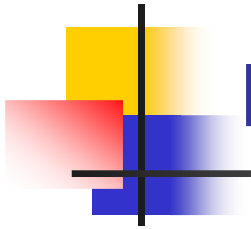
Each data point represents an average over 100 random instances



Dynamic MB and MBTE Heuristics

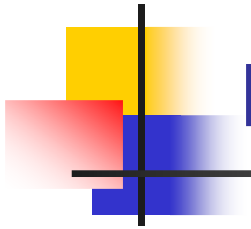
(Kask, Marinescu and Dechter, UAI'03)

- 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



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

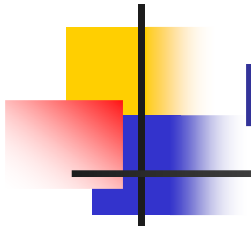
- Algorithms:
  - Complete
    - BBBT
    - BBMB
  - Incomplete
    - DLM
    - GLS
    - SLS
    - IJGP
    - IBP (coding)
- Measures:
  - Time
  - Accuracy (% exact)
  - #Backtracks
  - Bit Error Rate (coding)
- Benchmarks:
  - Coding networks
  - Bayesian Network Repository
  - Grid networks (N-by-N)
  - Random noisy-OR networks
  - Random networks

Real World Benchmarks

(Marinescu, Kask & 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]

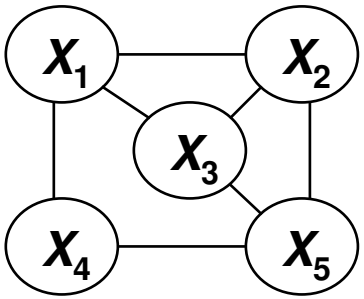
Average Accuracy and Time. 30 samples, 10 observations, 30 seconds



Hybrid of Variable-elimination and Search

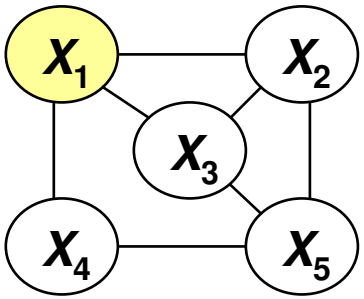
- Tradeoff space and time

Search Basic Step: Conditioning

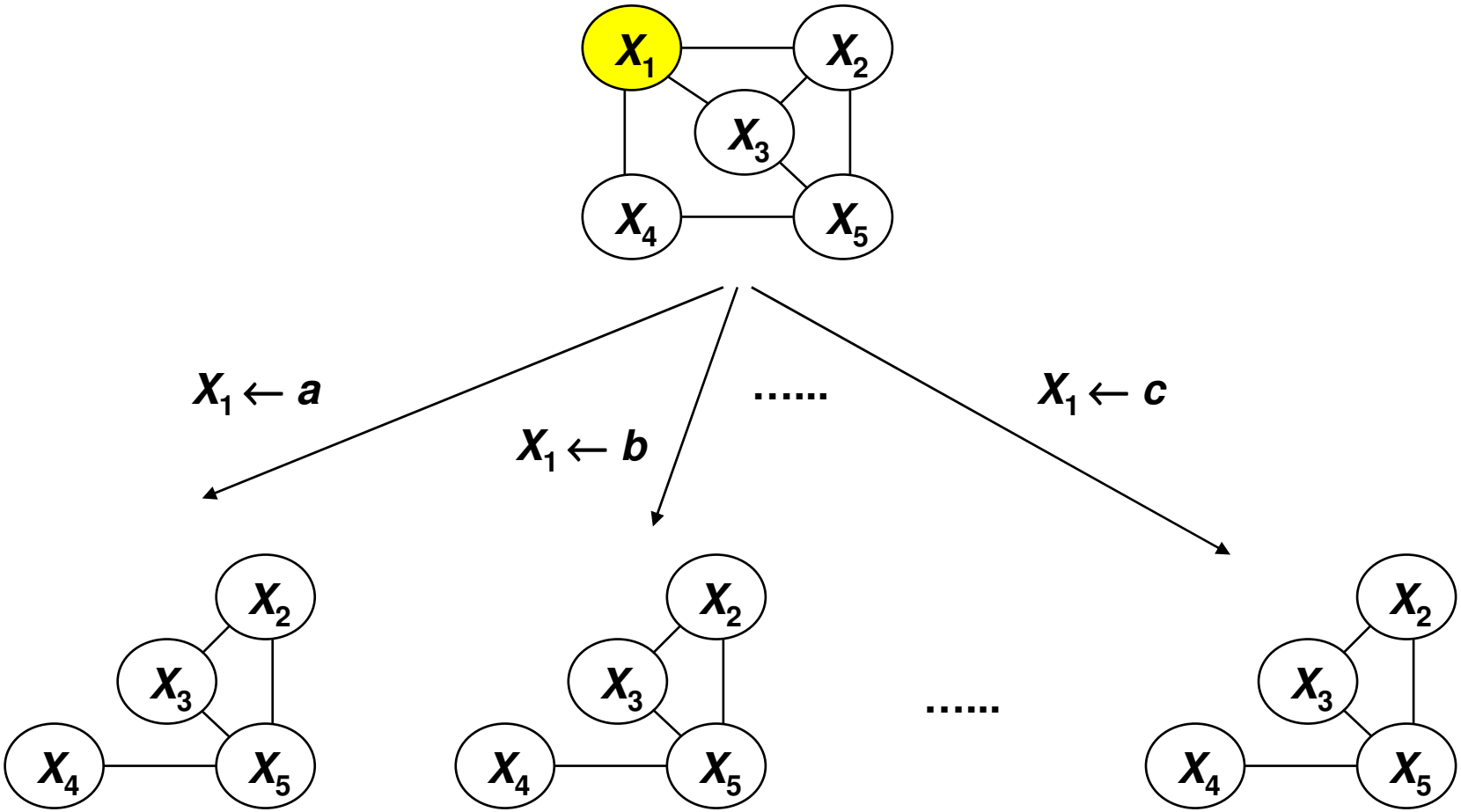


Search Basic Step: Conditioning

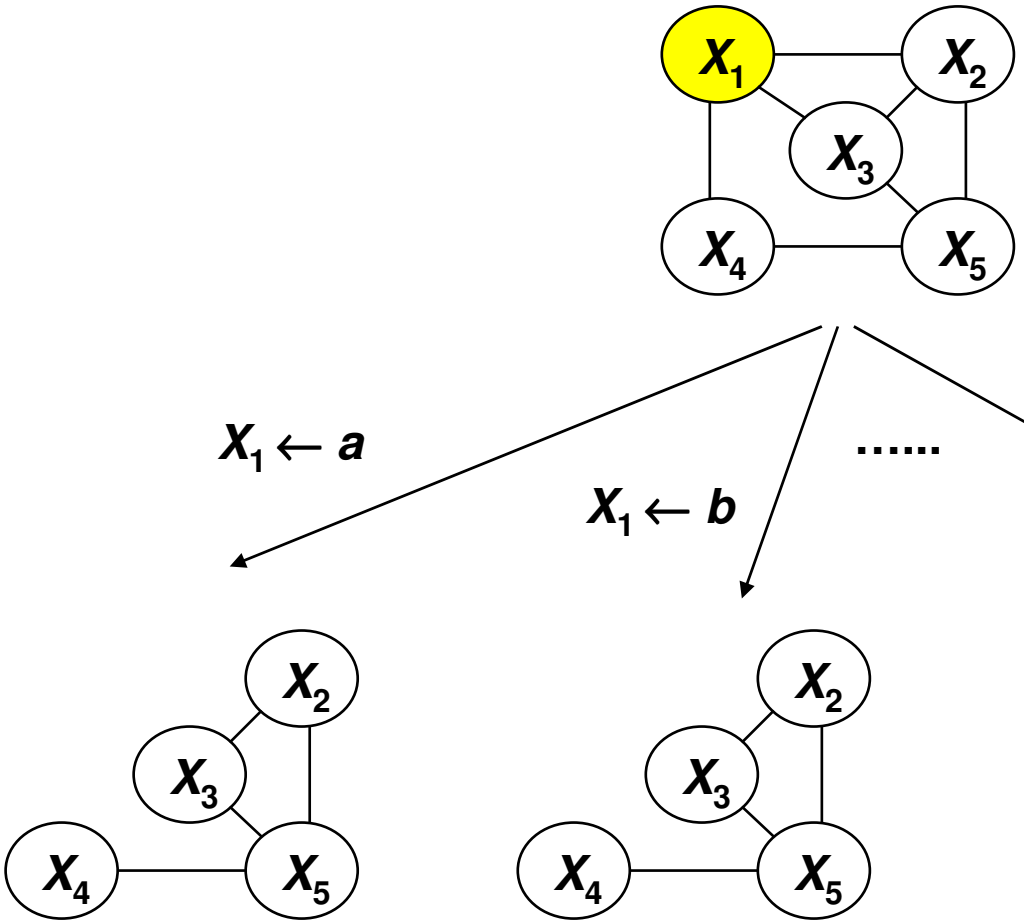
- **Select a variable**



Search Basic Step: Conditioning

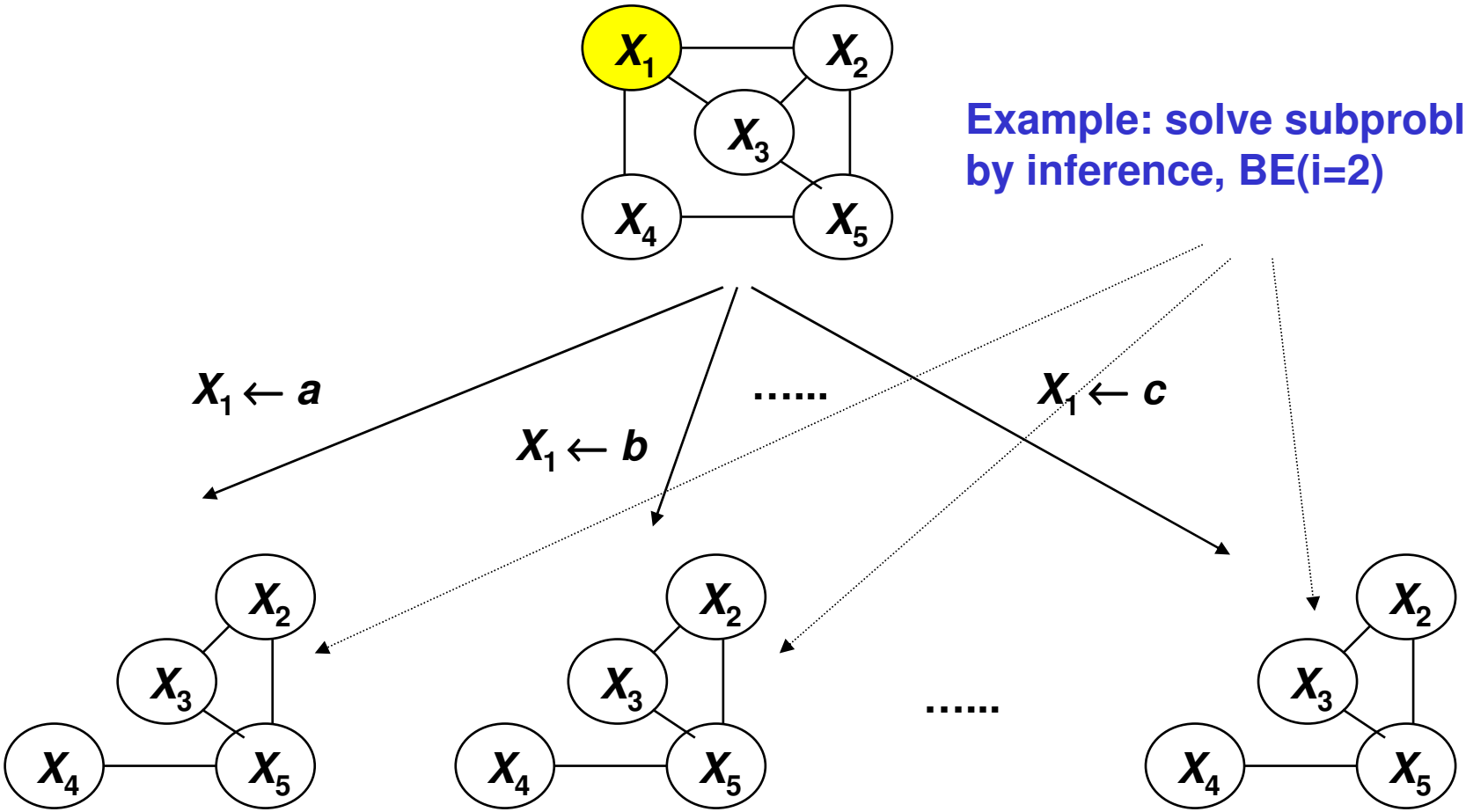


Search Basic Step:
Variable Branching by Conditioning



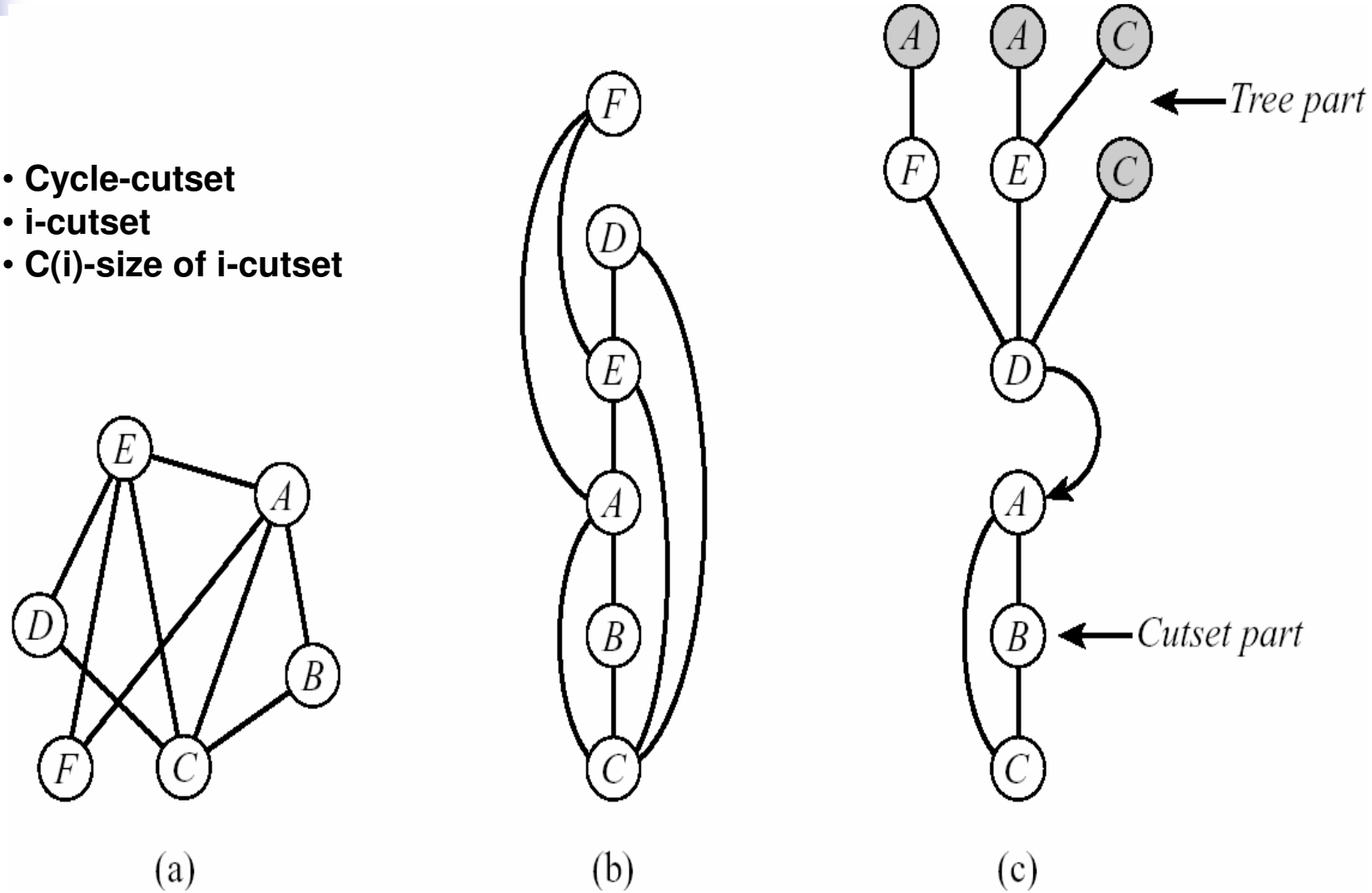
General principle:
Condition until tractable
Then solve sub-problems efficiently

Search Basic Step:
Variable Branching by Conditioning



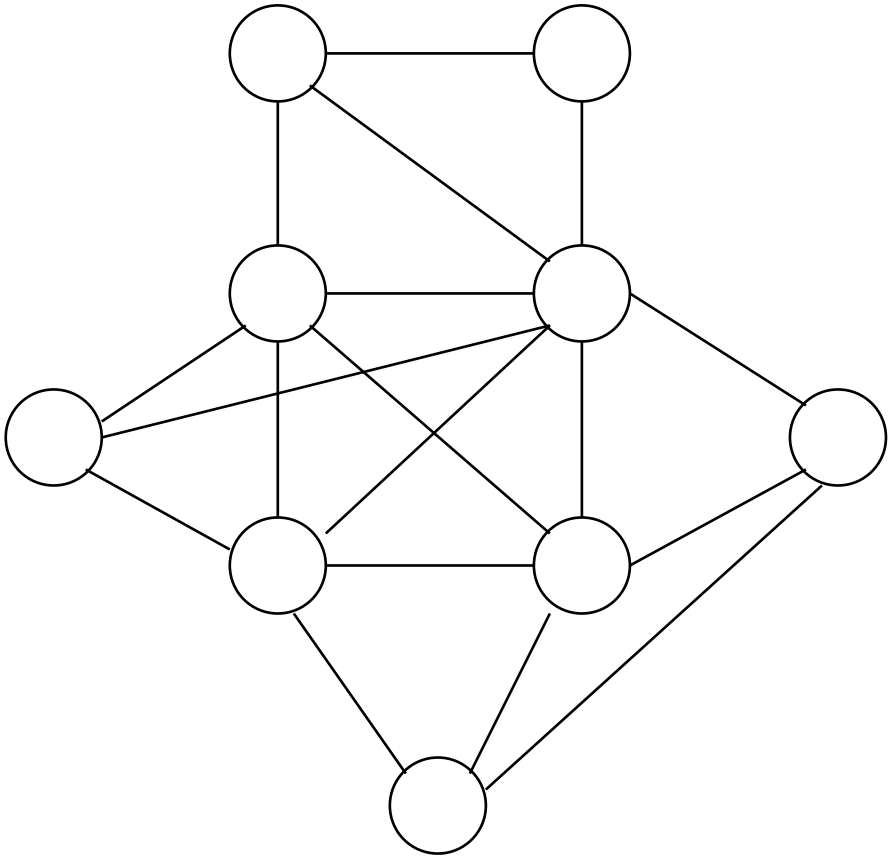
The Cycle-Cutset Scheme:
Condition Until Treeness

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

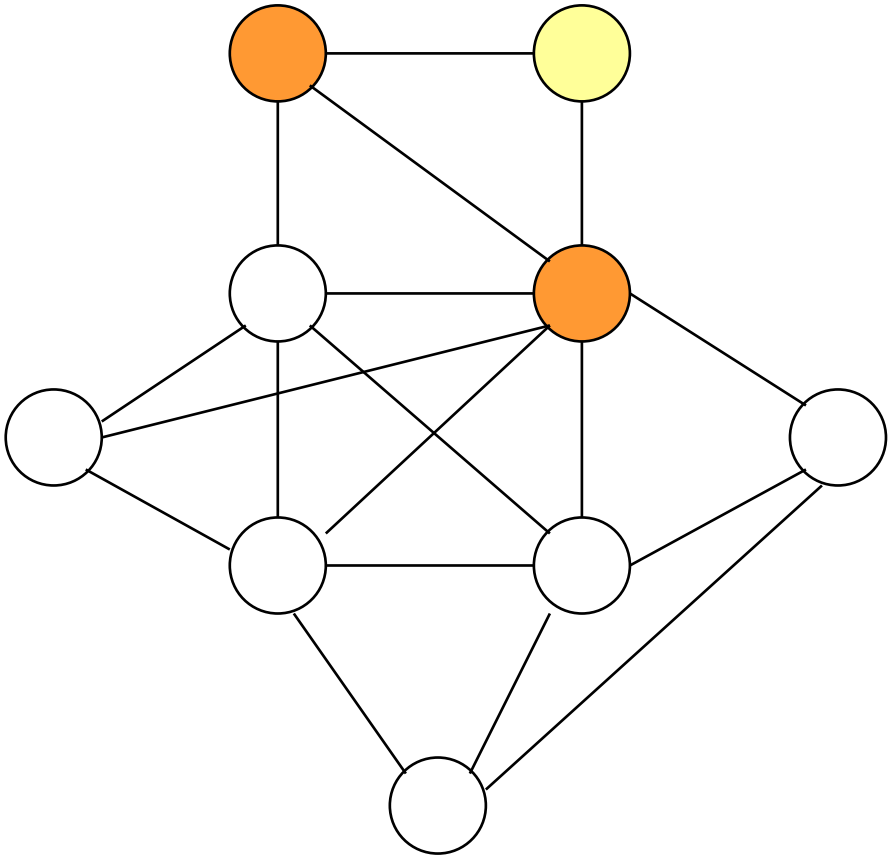


Space: exp(i), Time: O(exp(i+c(i))

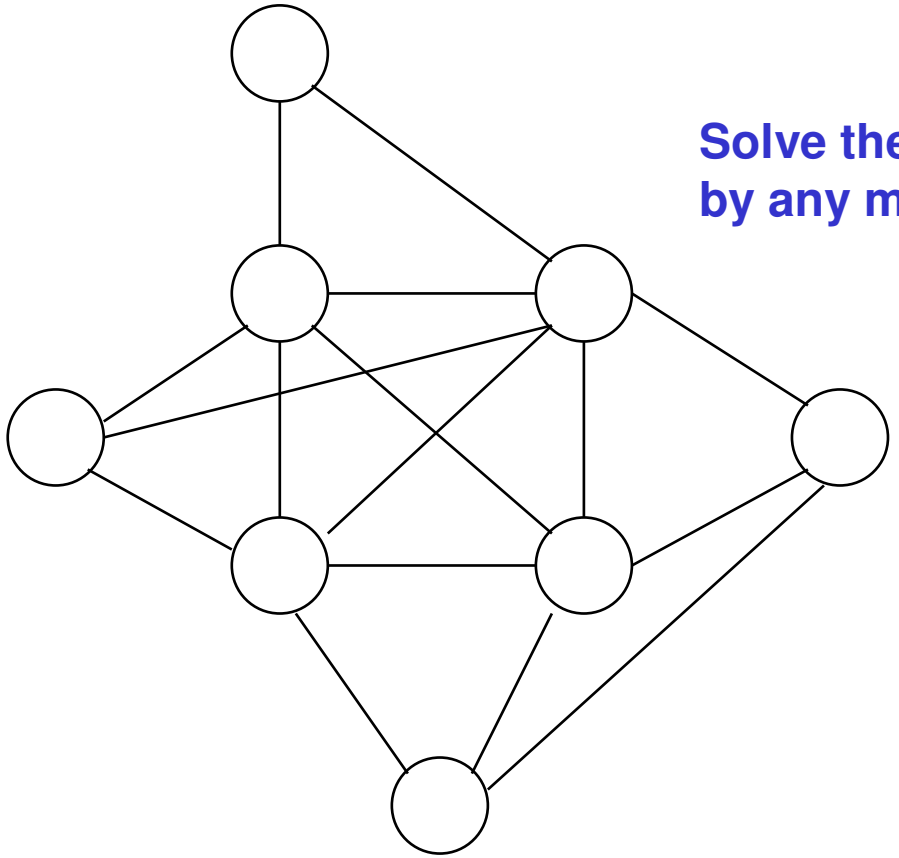
Eliminate First



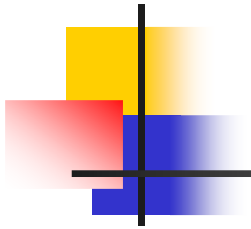
Eliminate First



Eliminate First



Solve the rest of the problem by any means

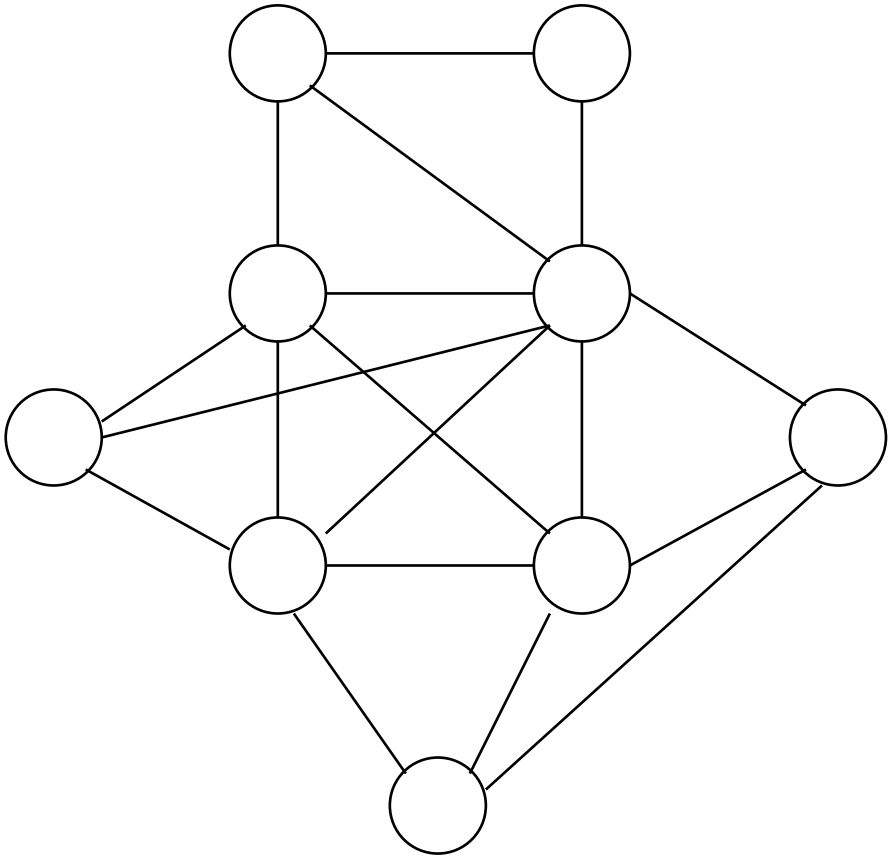


Hybrids Variants

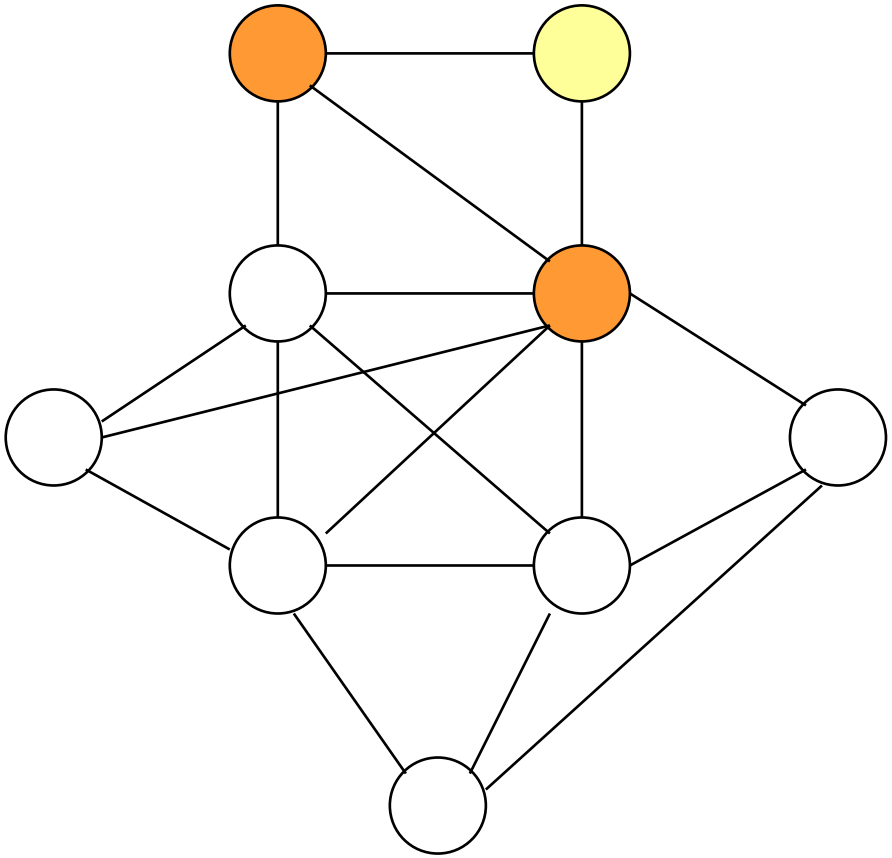
- Condition, condition, condition … and then only eliminate (w-cutset, cycle-cutset)
- Eliminate, eliminate, eliminate … and then only search
- Interleave conditioning and elimination (elim-cond(i), VE+C)

Interleaving Conditioning and Elimination

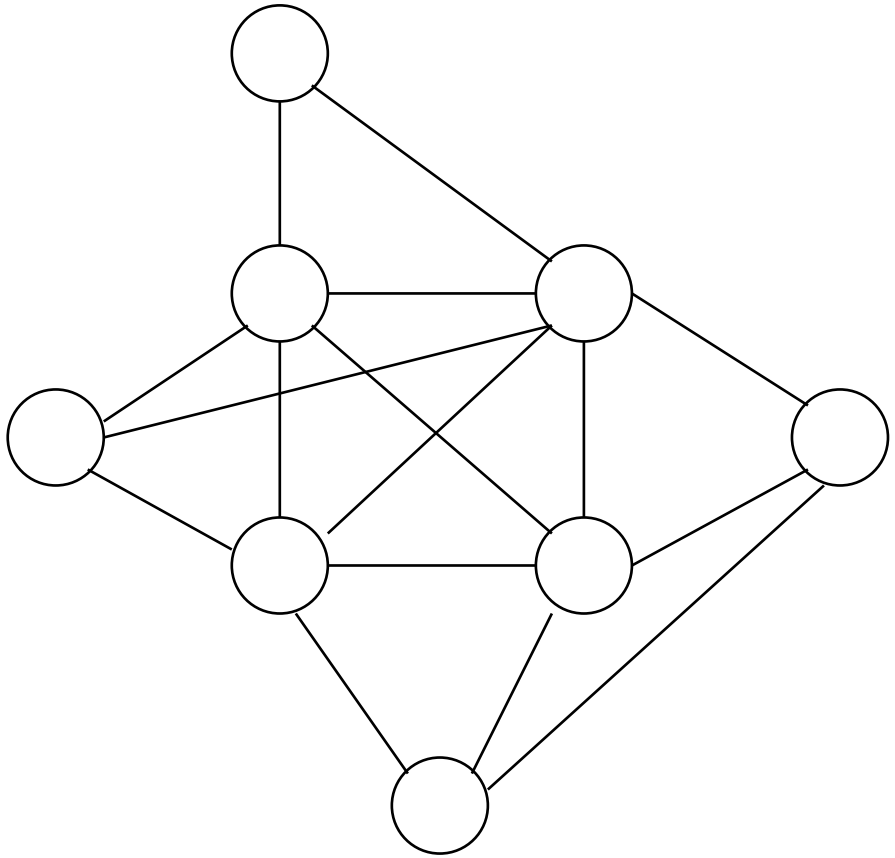
(Larrosa & Dechter, CP'02)



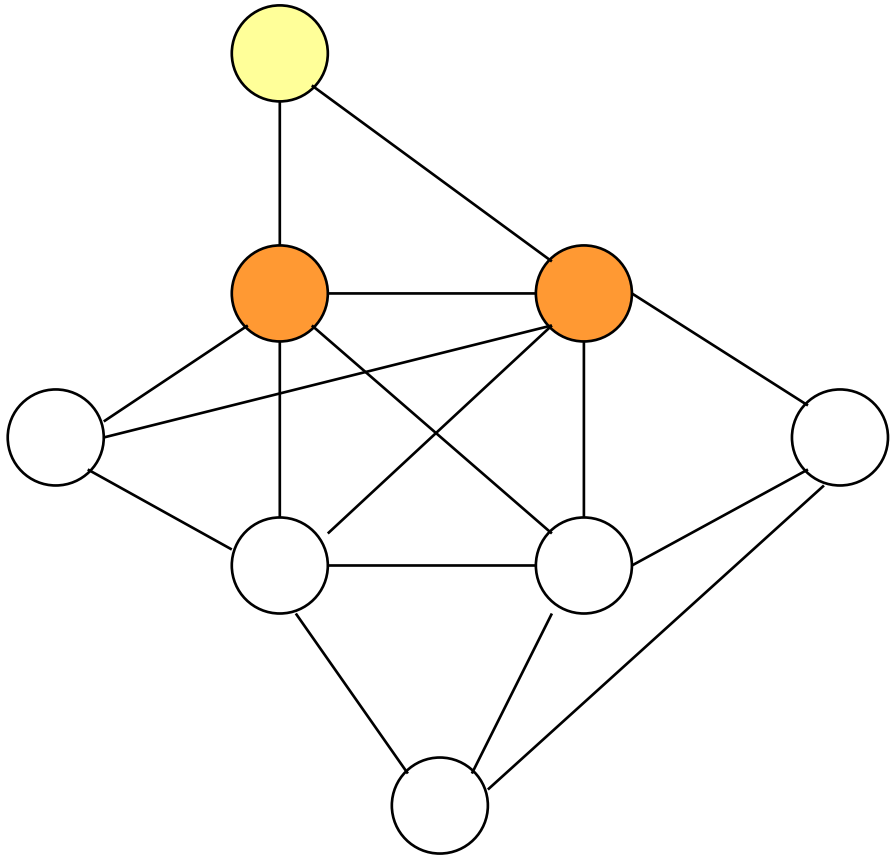
Interleaving Conditioning and Elimination



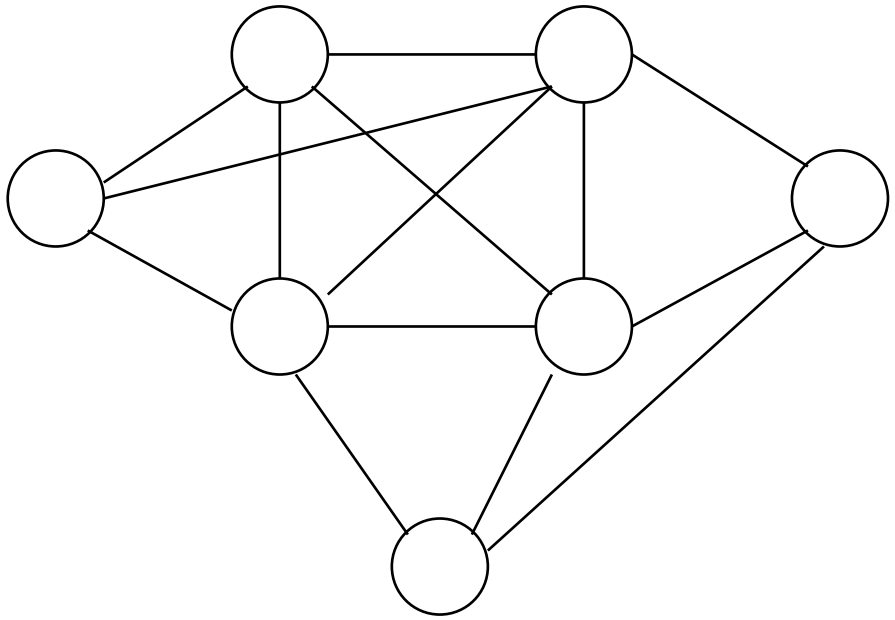
Interleaving Conditioning and Elimination



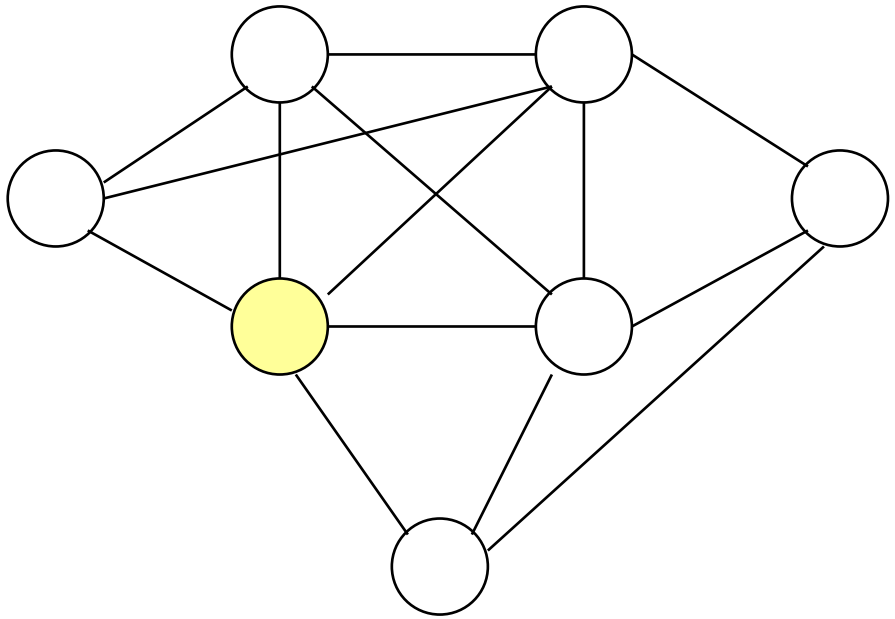
Interleaving Conditioning and Elimination



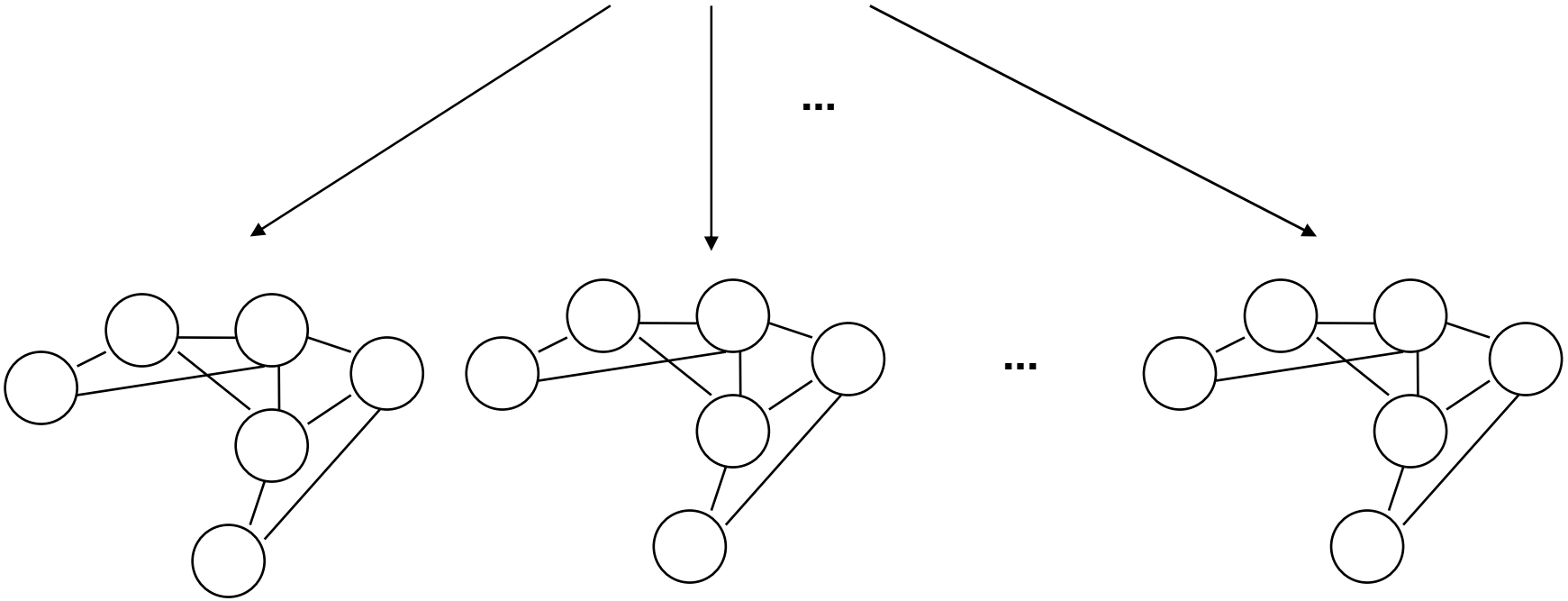
Interleaving Conditioning and Elimination

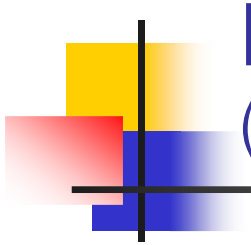


Interleaving Conditioning and Elimination



Interleaving Conditioning and Elimination





Boosting Search with Variable Elimination

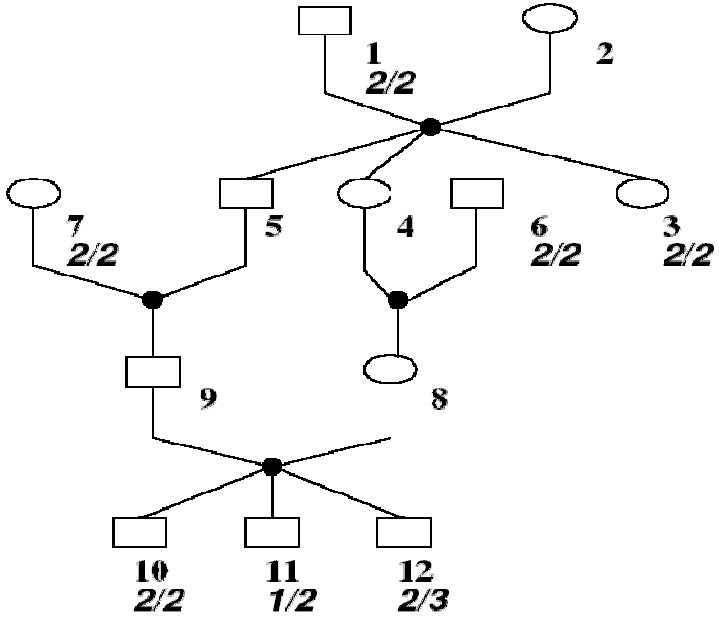
(Larrosa & Dechter, *Constraints* 2003)

- At each search node
  - Eliminate all unassigned variables with degree **≤ p**
  - Select an unassigned variable A
  - Branch on the values of A

- Properties
  - BB-VE(-1) is Depth-First Branch and Bound
  - BB-VE(w) is Variable Elimination
  - BB-VE(1) is similar to Cycle-Cutset
  - BB-VE(2) is well suited with soft local consistencies
    (add binary constraints only, independent of the elimination order)

(Sanchez et al, *Constraints* 2008)

Mendelian error detection



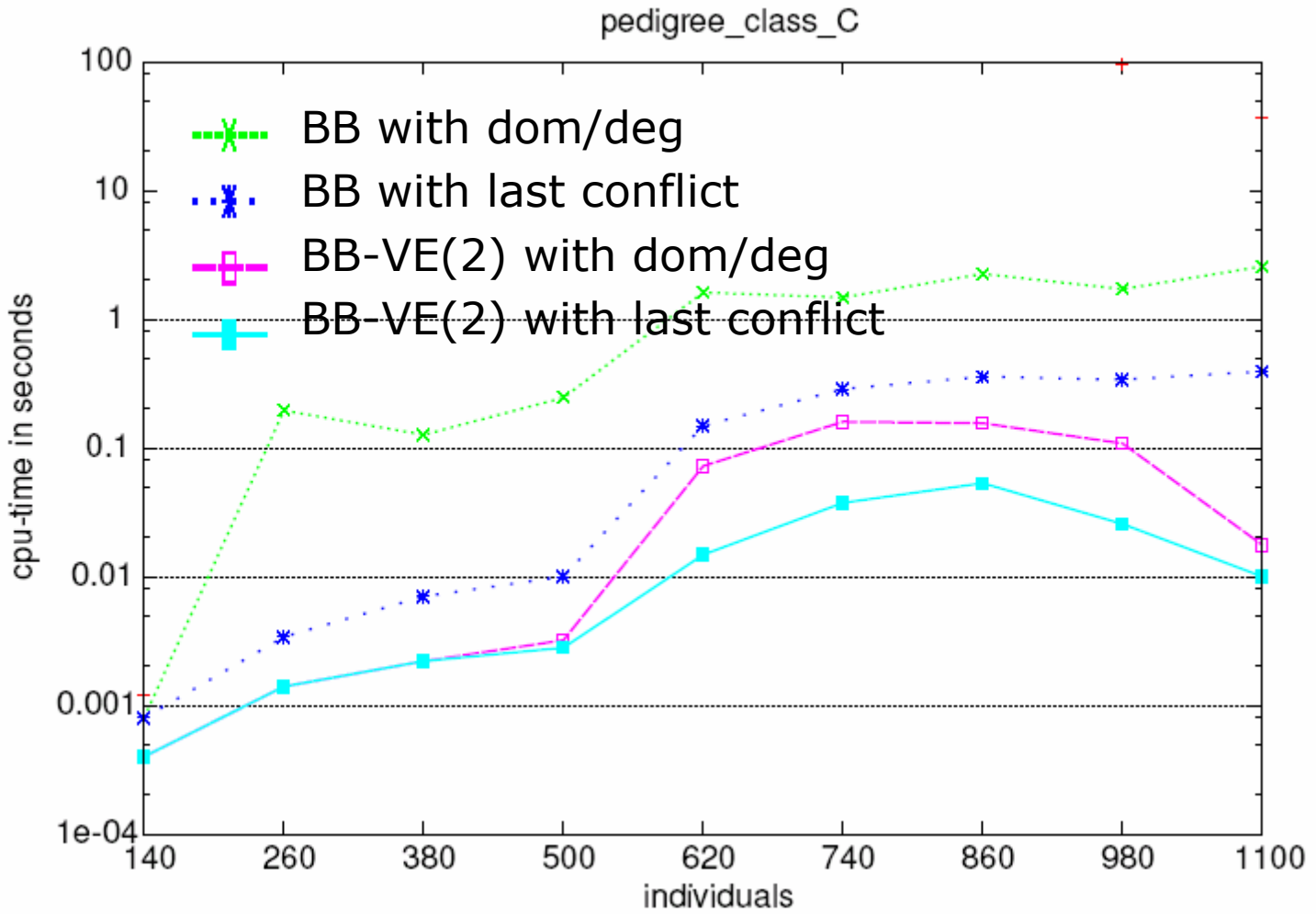
- Given a pedigree and partial observations (genotypings)
- Find the erroneous genotypings, such that their removal restores consistency

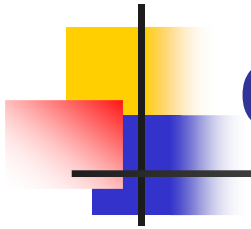
- Checking consistency is NP-complete (Aceto et al., *Comp. Sci. Tech.* 2004)
- Minimize the number of genotypings to be removed
- Maximize the joint probability of the true genotypes (MPE)

Pedigree *problem size: n≤20,000 ; d=3—66 ; e(3)≤30,000*

Pedigree

- toulbar2 v0.5 with EDAC and binary branching
- Minimize the number of genotypings to be removed
- CPU time in seconds to find and prove optimality on a 3 GHz computer with 16 GB





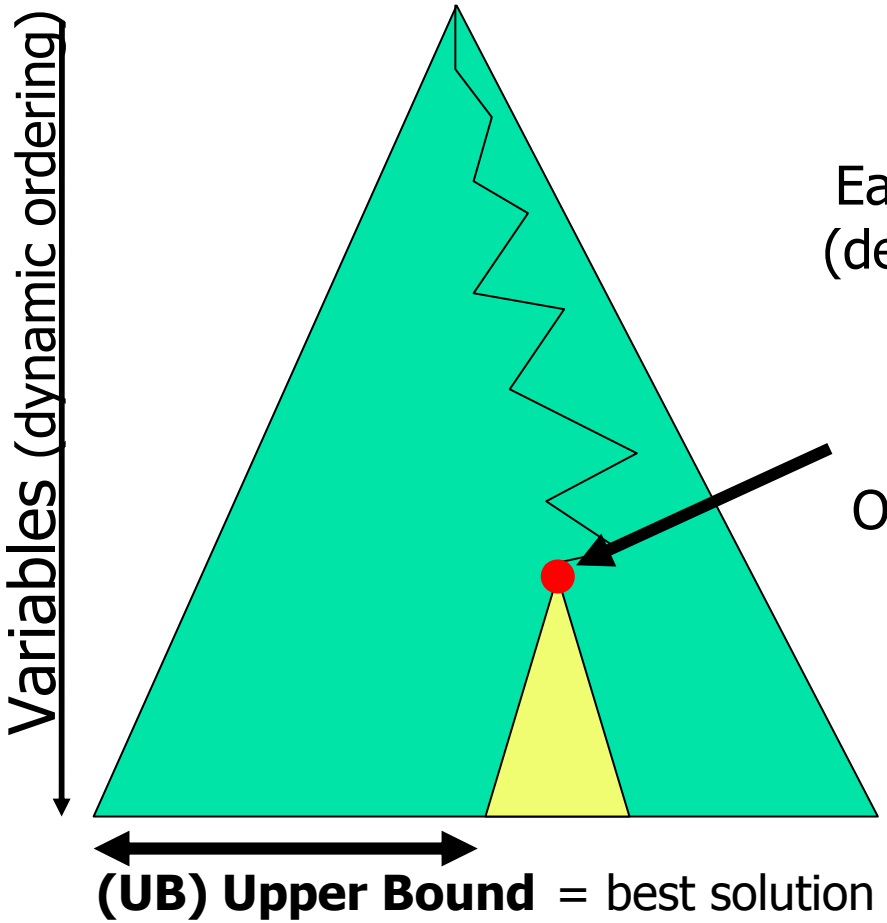
Outline

- Introduction
- Inference
- Search (OR)

- **Lower-bounds and relaxations**
  - Bounded variable elimination
  - **Local consistency**
    - Equivalence Preserving Transformations
    - Chaotic iteration of EPTs
    - Optimal set of EPTs
    - Improving sequence of EPTs

- Exploiting problem structure in search
- Software

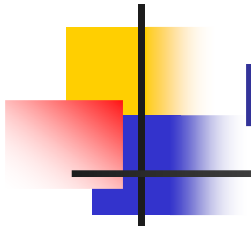
Depth-First Branch and Bound (DFBB)



Each node is a COP subproblem
(defined by current conditioning)

(LB) Lower Bound = $f_\emptyset$
under estimation of the best solution in the sub-tree
Obtained by enforcing local consistency

If $LB_\emptyset \geq k$ then prune



Local Consistency in Constraint Networks

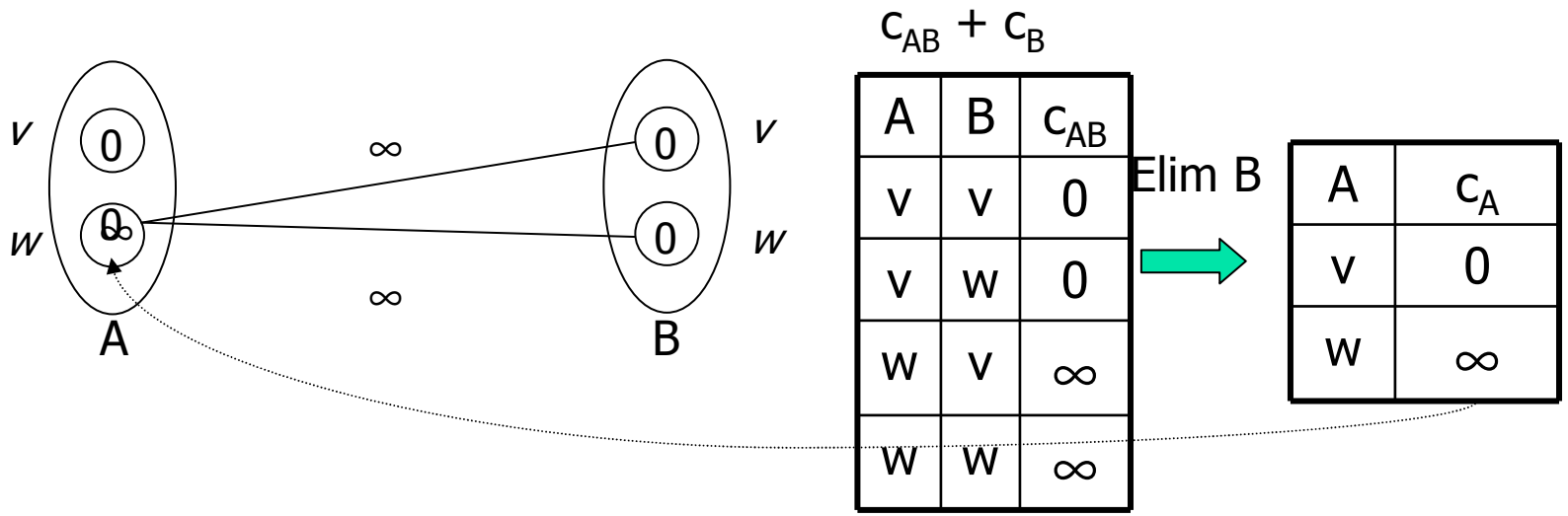
- Massive local inference
  - Time efficient (local inference, as mini buckets)
  - Infer only small constraints, added to the network
  - No variable is eliminated
  - Produces an **equivalent** more explicit problem
  - May detect **inconsistency** (prune tree search)

Arc consistency

inference in the scope of 1 constraint

Arc Consistency (binary CSP)

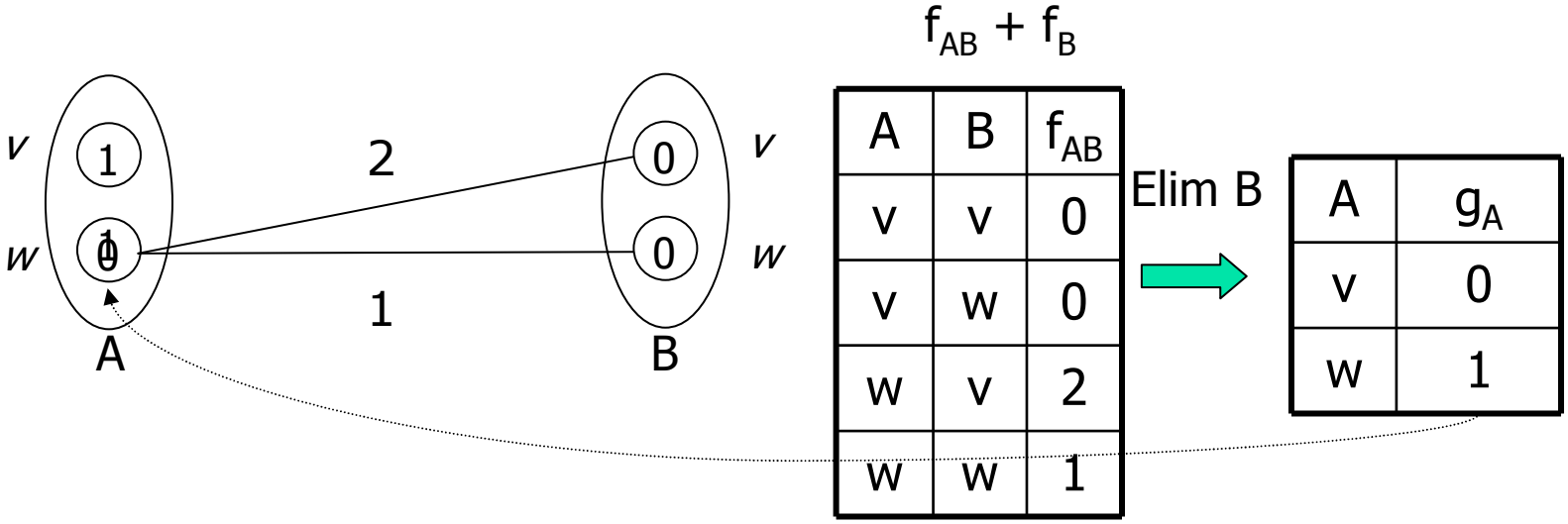
- for a constraint c_{AB} and variable A



- Applied iteratively on all constraint/variables
- Confluent, incremental, complexity in $O(md^2)$
- Empty domain => inconsistency

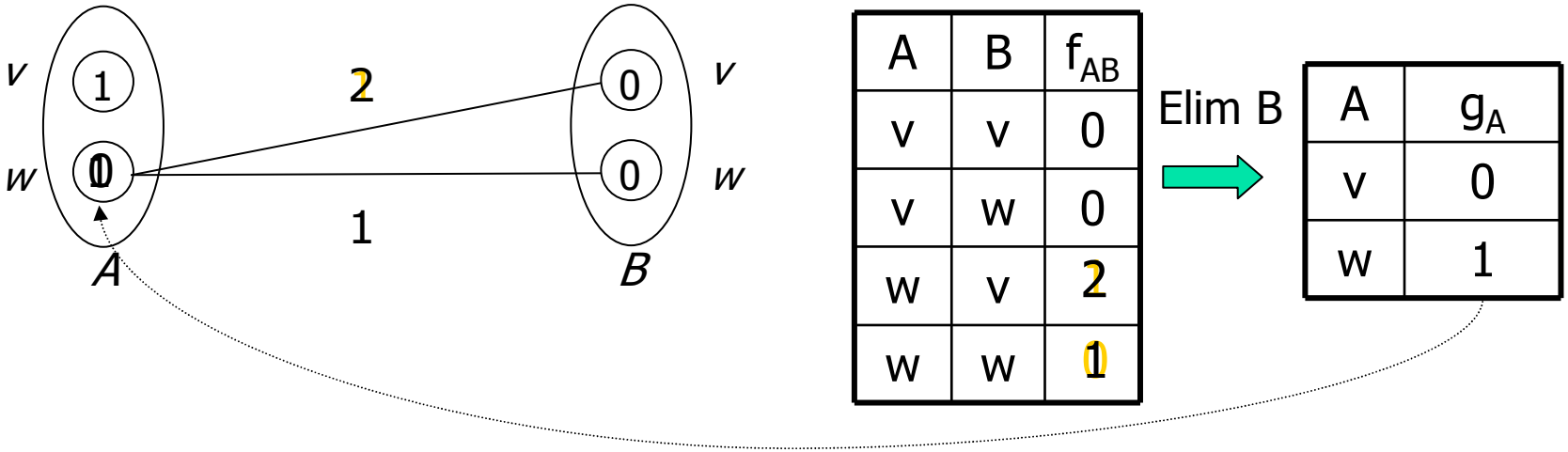
Arc Consistency and Cost Functions

- for a cost function f_{AB} and a variable A



EQUIVALENCE LOST

Shifting Costs (cost compensation)



Subtract from source in order to preserve the problem

➔ **Equivalence Preserving Transformation**

Complete Inference vs Local Inference

- Complete inference
  - Combine, eliminate, add & **forget**
  - Systematic inference
  - Exponential time/space
  - Preserves optimum
  - Provides the optimum

$f_{\varnothing}$

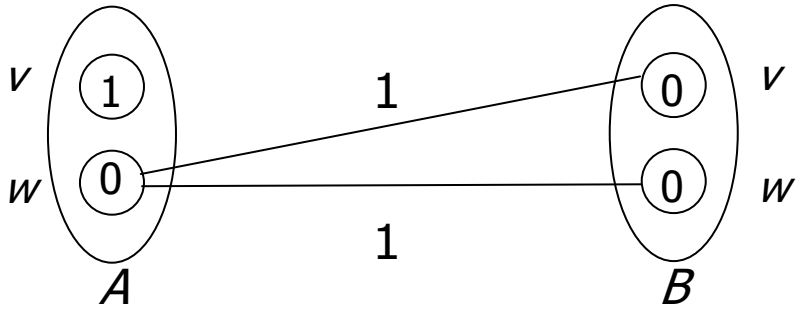
- Local consistency
  - Combine, eliminate, add & **subtract**
  - Massive local inference
  - Space/time efficient
  - Preserves equivalence
  - Provides a lb

$f_{\varnothing}$

Equivalence Preserving Transformation

- Shifting costs from f_{AB} to A

Shift(f_{AB},(A,w),1)

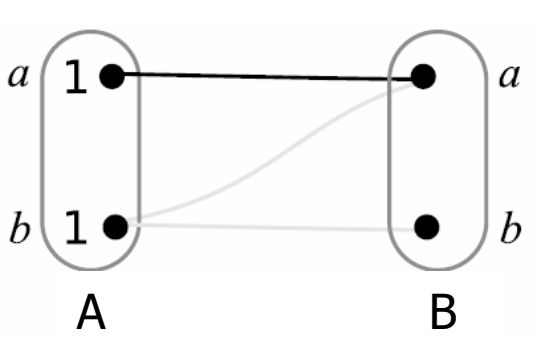


Arc EPT: shift cost in the scope of 1 cost function
Problem structure preserved

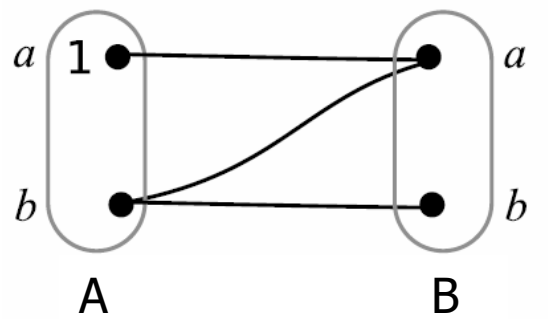
- Can be reversed (e.g. Shift(f_{AB},(A,w), -1))

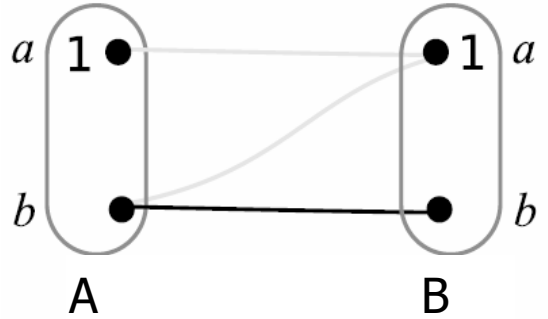
Equivalence Preserving Transformations

Shift(f_{AB},(A,b),1)

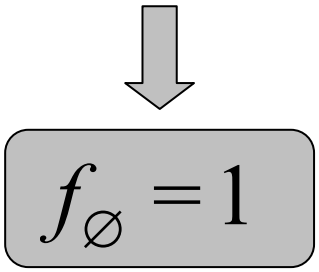


Shift(f_{AB},(B,a),1)





Shift(f_{A},(),1)

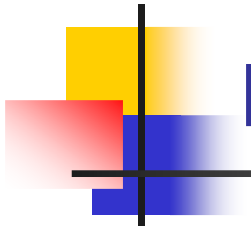


$f_{\varnothing} = 1$

Shift(f_{AB},(B,a),-1)

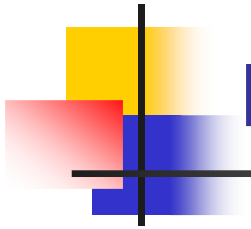
- EPTs may cycle
- EPTs may lead to different f_0

•Which EPTs should we apply?



Local Consistency

- Equivalence Preserving Transformation
- Chaotic iteration of EPTs
- Optimal set of EPTs
- Improving sequence of EPTs



Local Consistency

- Equivalence Preserving Transformation
- **Chaotic iteration of EPTs**
  - **Enforce a local property by one or two EPT(s)**
- Optimal set of EPTs
- Improving sequence of EPTs

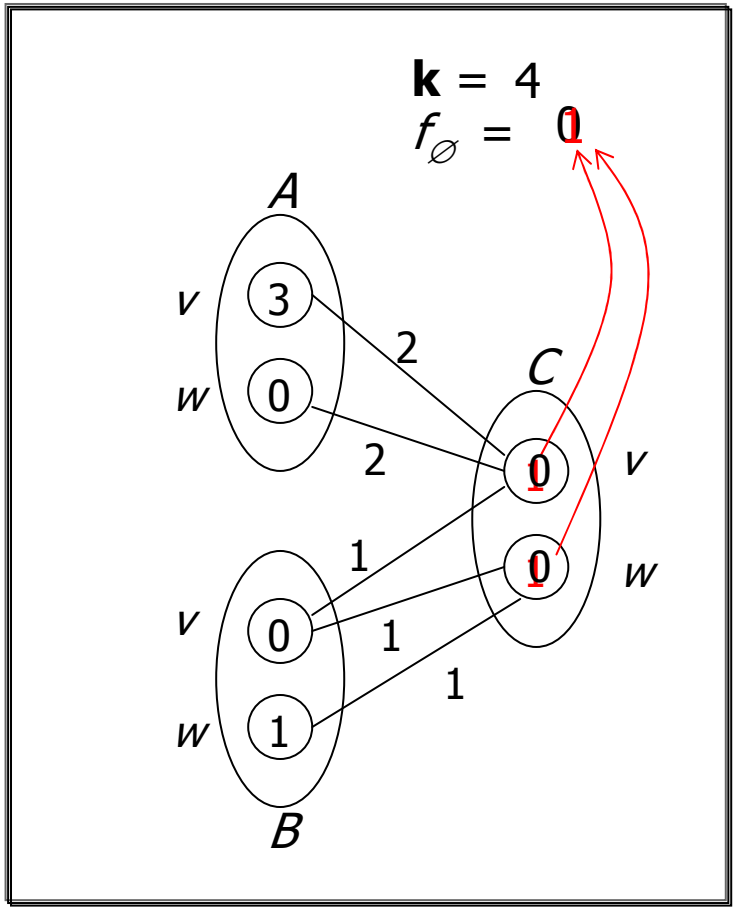
Node Consistency (NC*)

(Larrosa, AAAI 2002)

- For any variable *A*
  - $\forall a,\ f_{\varnothing} + f_A(a) < k$
  - $\exists\ a,\ f_A(a) = 0$

- Complexity:

  O(*nd*)



Shift(f_C, $\varnothing$,1) Shift(f_A, $\varnothing$,-1);Shift(f_A, $\varnothing$,1)

Arc Consistency (AC*)

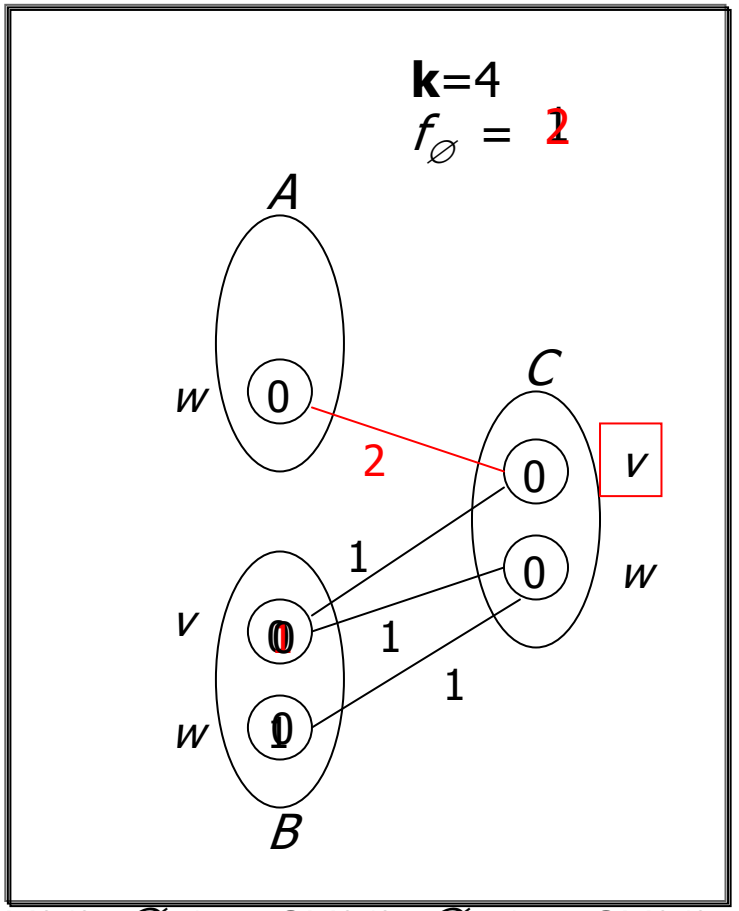
(Schiex, CP 2000)

(Larrosa, AAAI 2002)

- NC*
- For any f_{AB}
  - $\forall a \exists b$

    $f_{AB}(a,b) = 0$
- *b* is a *support*
- complexity:

  $\mathbf{O}(n^2 d^3)$



Shift(f_{AC},(C,v),2) Shift(f_{BC},(B,v),1) Shift(f_B,∅,1) Shift(f_C,∅,-2) Shift(f_C,∅,2)

Shift(f_{BC},(B,w),1)

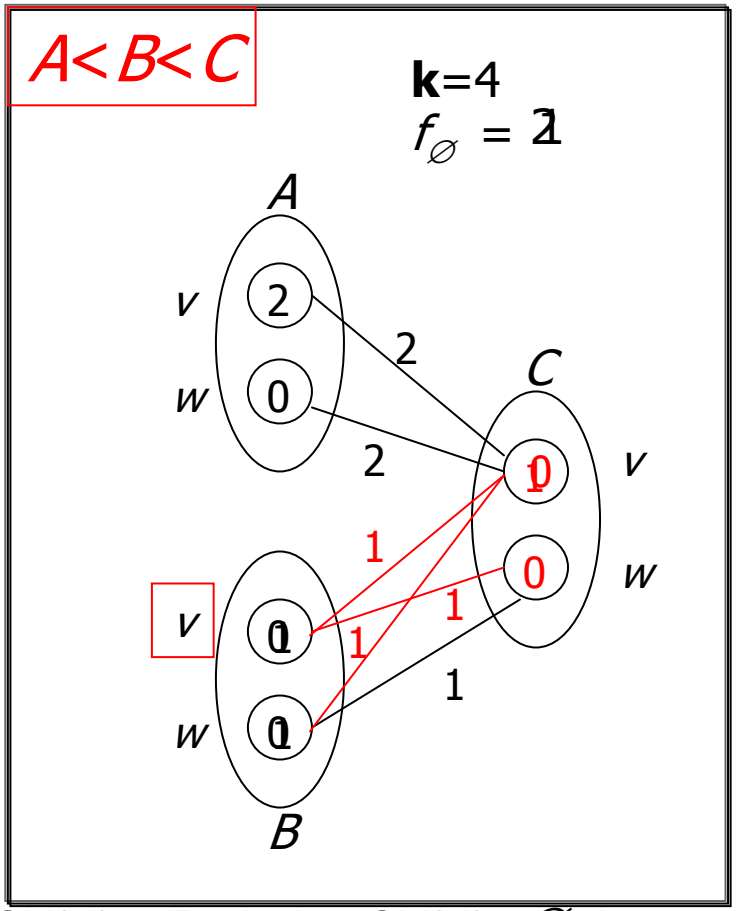
Directional AC (DAC*)

(Cooper, Fuzzy Sets and Systems 2003)

- NC*
- For all f_{AB} $(A<B)$
  - $\forall a \exists b$

    $f_{AB}(a,b) + f_B(b) = 0$
- b is a *full-support*
- complexity:

  $\mathbf{O}(ed^2)$

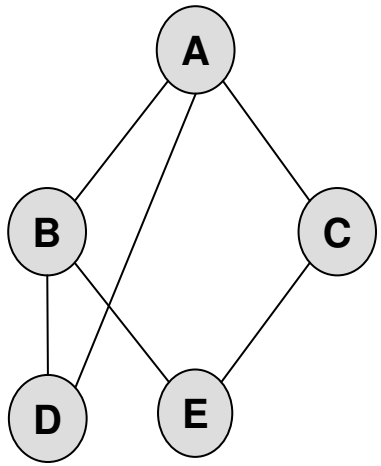


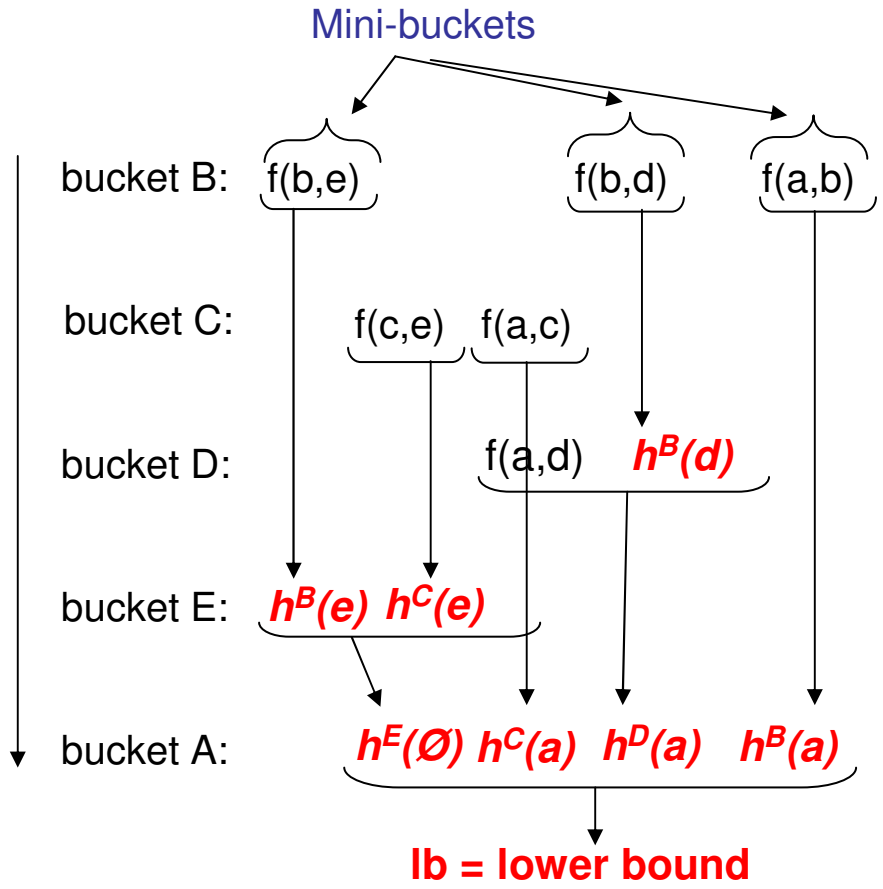
Shift(f_{BC},(C,v),-1) Shift(f_{BC},(B,v),1) Shift(f_{BC},(B,w),1) Shift(f_B, ∅,1)

Shift(f_A,∅, -2) Shift(f_A,∅, 2)

DAC lb = Mini-Bucket(2) lb

A < E < D < C < B





- **DAC provides an equivalent problem: incrementality**
- **DAC+NC (value pruning) can improve lb**

Other « Chaotic » Local Consistencies

- FDAC* = DAC+AC+NC

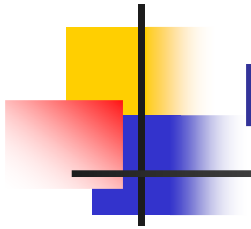
  (Cooper, Fuzzy Sets and Systems 2003)
  (Larrosa & Schiex, IJCAI 2003)
  (Larrosa & Schiex, *AI* 2004)
  (Cooper & Schiex, *AI* 2004)

  - Stronger lower bound
  - $O(end^3)$
  - Better compromise

- EDAC* = FDAC+ EAC (existential AC)

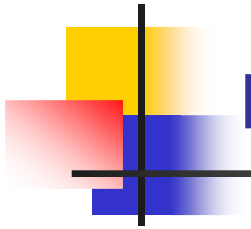
  (Heras et al., IJCAI 2005)
  (Sanchez et al, *Constraints* 2008)

  - Even stronger
  - $O(ed^2 \max\{nd, k\})$
  - Currently among the best practical choice



Local Consistency

- Equivalence Preserving Transformation
- Chaotic iteration of EPTs
- **Optimal set of simultaneously applied EPTs**
  - **Solve a linear problem in rational costs**
- Improving sequence of EPTs



Finding an EPT Sequence Maximizing the LB

Bad news

Finding a sequence of integer arc EPTs that maximizes the lower bound defines an NP-hard problem

(Cooper & Schiex, *AI* 2004)

Good news: A continuous linear formulation

- u_A : cost shifted from *A* to f_0
- p_{Aa}^{AB}: cost shifted from f_{AB} to *(A,a)*

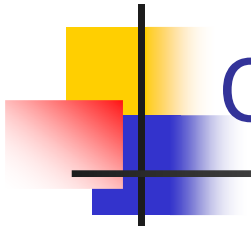
$$max \sum u_i$$

Subject to non negativity of costs

$$\forall i \in X, \forall a \in d_i, \quad c_i(a) - u_i + \sum_{(c_S \in C),(i \in S)} p_{i,a}^S \geq 0$$

$$\forall c_S \in C, |S| > 1, \forall t \in \ell(S) \quad c_S(t) - \sum_{i \in S} p_{i,t[\{i\}]}^S \geq 0$$

***n + m.r.d* variables**
***n.d + m.d^r* linear constraints**



Optimal Soft AC

(Cooper et al., IJCAI 2007)

(Schlesinger, *Kibernetika* 1976)

(Boros & Hammer, *Discrete Appl. Math.* 2002)

solved by Linear Programming

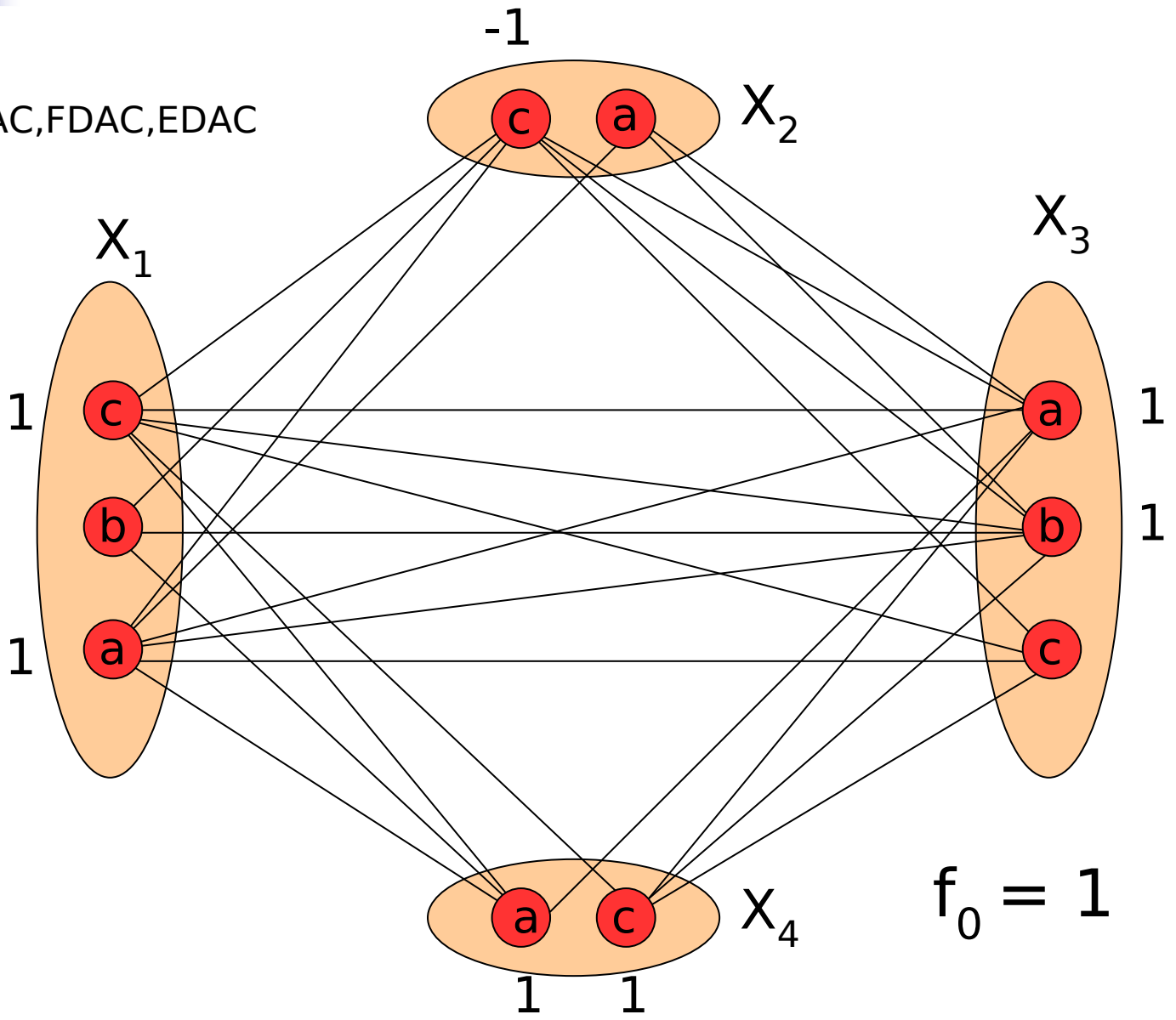
- Polynomial time, rational costs (bounded arity r)
- Computes an optimal set of EPT (u_A, p_{Aa}^{AB}) to apply simultaneously
- Stronger than AC, DAC, FDAC, EDAC...
  (or any local consistency that preserves scopes)

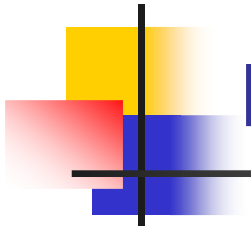
Exam

$$p^{23}_{2c} = p^{34}_{3a} = p^{31}_{3b} = p^{12}_{1a} = p^{14}_{1c} = -1$$

$$p^{23}_{3a} = p^{23}_{3b} = p^{34}_{4c} = p^{31}_{1a} = p^{31}_{1c} = p^{12}_{2c} = p^{14}_{4a} = u_4 = 1$$

AC,DAC,FDAC,EDAC





Local Consistency

- Equivalence Preserving Transformation
- Chaotic iteration of EPTs
- Optimal set of EPTs
- **Improving sequence of EPTs**
  - **Find an improving sequence using classical arc-consistency in classical CSPs**

Virtual Arc Consistency

(Cooper et al., AAAI 2008)

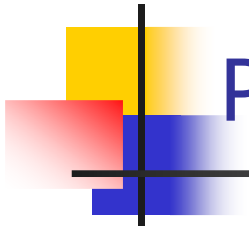
- ***Bool(P)***: a classical CSP derived from *P*
  - Forbids non zero cost assignments.

$$\forall f_S \in \mathbf{F}, S \neq \emptyset, \quad (t \in c_S) \Leftrightarrow (f_S(t) = 0)$$

  - If non empty:

    Solutions of Bool(P) = Optimal solutions of P (cost f_0)

P is Virtual AC iff AC(*Bool(P)*) is not empty



Properties

- Solves the polynomial class of submodular cost functions

$$\forall t,t',\ f(t)+f(t')\geq f\left(max(t,t')\right)+f\left(min(t,t')\right)$$

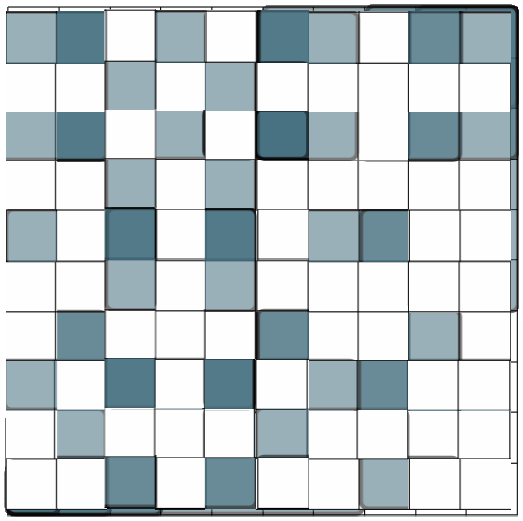
- In *Bool(P)* this means

$$\forall t,t',\quad c(t)\wedge c(t')\Rightarrow c\left(max(t,t')\right)\wedge c\left(min(t,t')\right)$$

- *Bool(P)* max-closed: AC implies consistency

Binary Submodular Cost Functions

- Decomposable in a sum of ”Generalized Intervals”

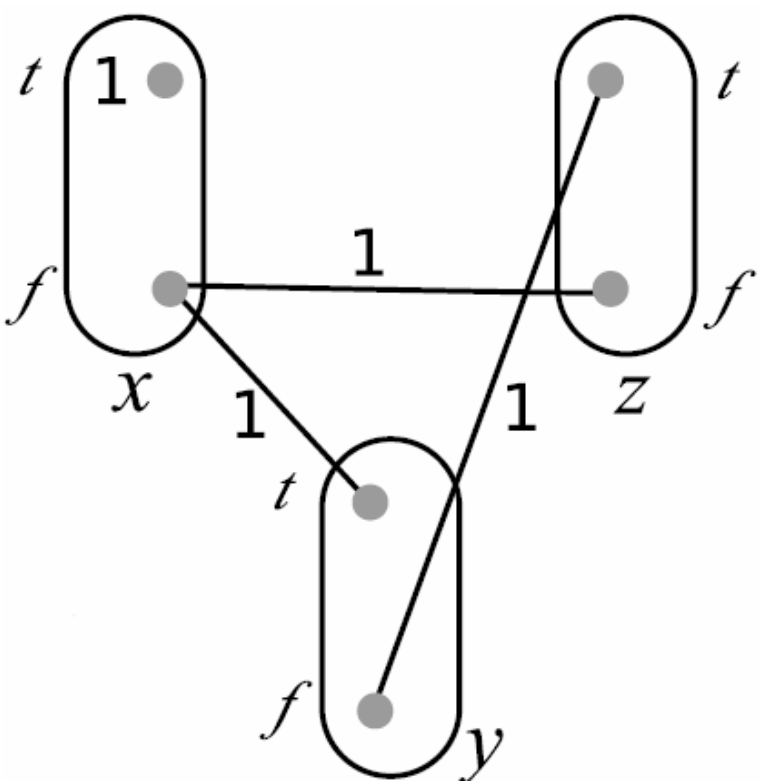


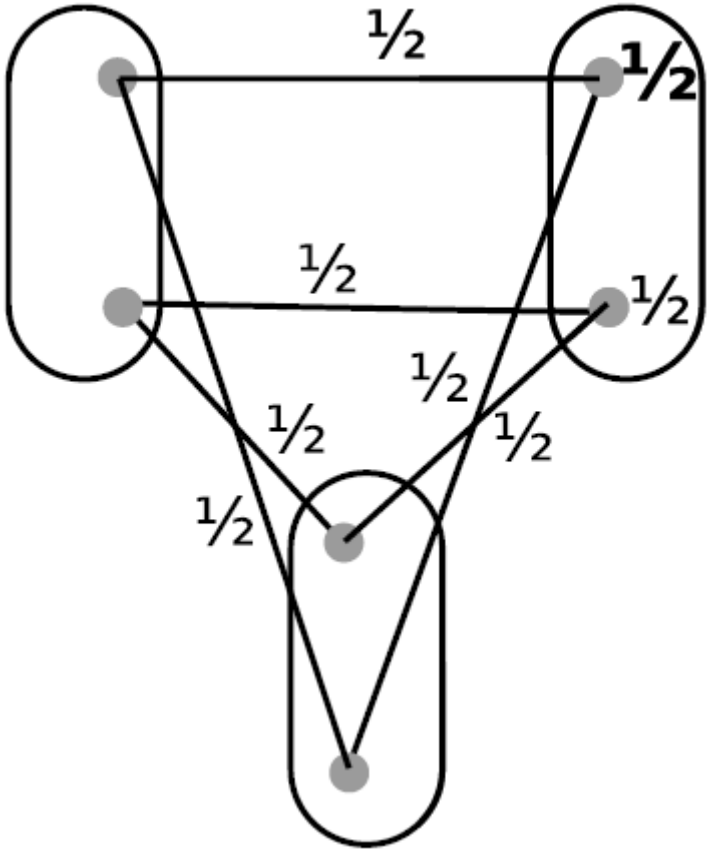
- Subsumes “*Simple Temporal CSP with strictly monotone preferences”* (Khatib et al, IJCAI 2001)

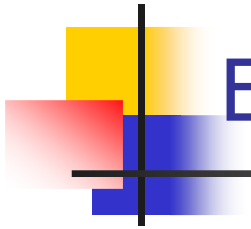
$$\delta_r^{\geq}(x,y) = \begin{cases} |x-y|^r & \text{if } x \geq y \\ \infty & \text{otherwise} \end{cases} \quad \text{where } r \in \mathbb{R},\ r \geq 1.$$

Enforcing VAC

AC,DAC,FDAC,EDAC





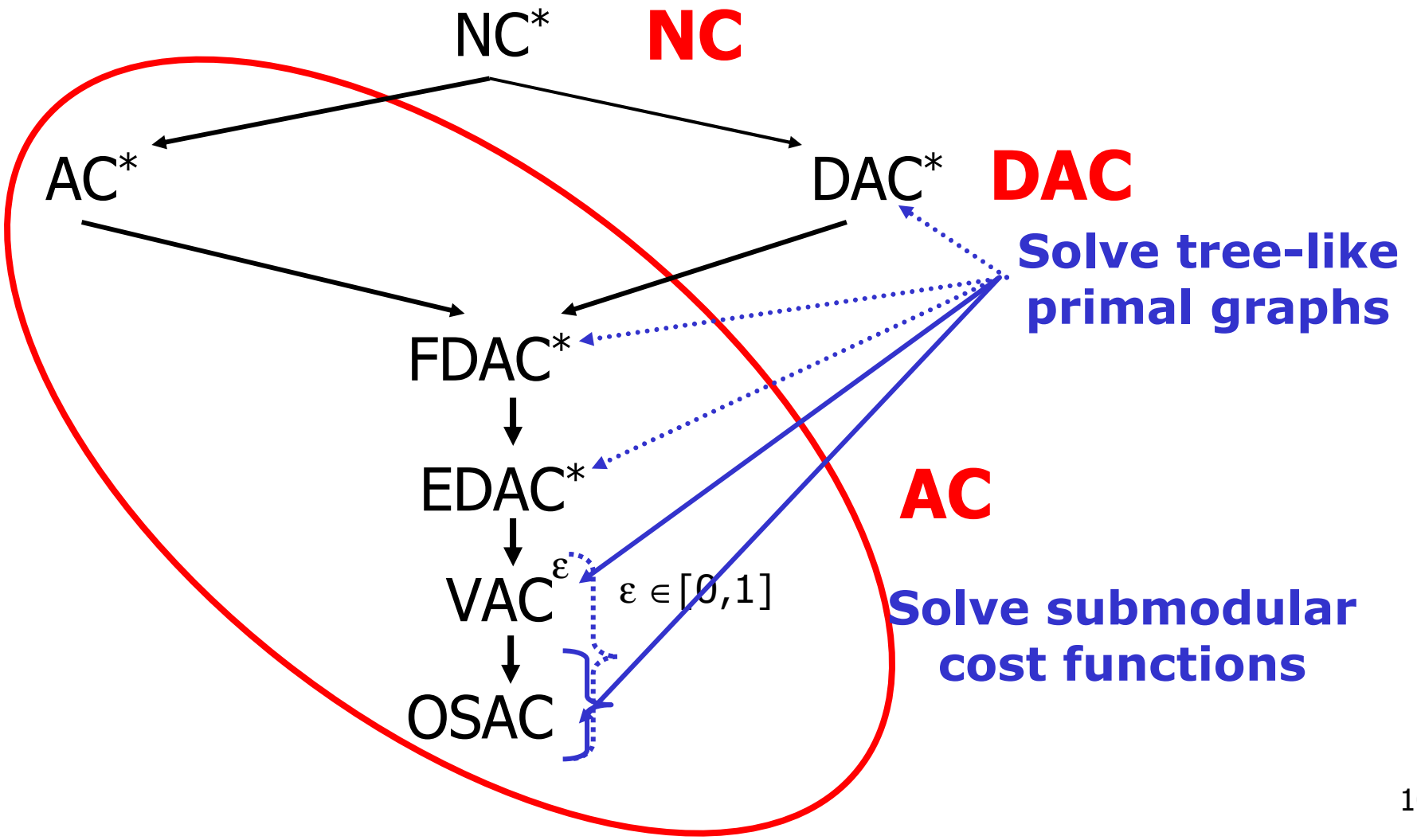


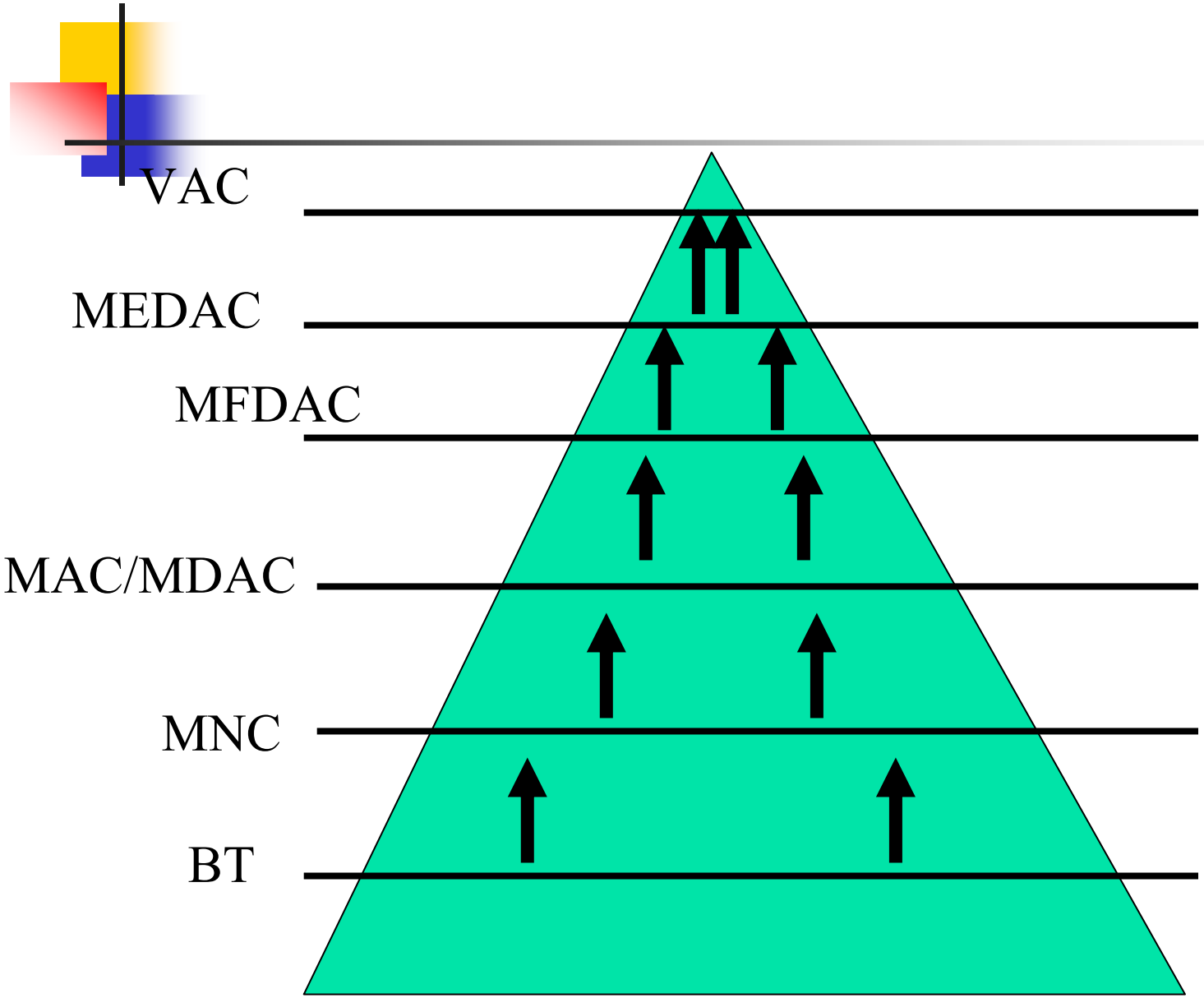
Enforcing VAC on a binary COP

- Iterative process
- One iteration $O(ed^2)$ time, $O(ed)$ space
- Number of iterations possibly unbounded: premature stop (ε threshold)

Hierarchy

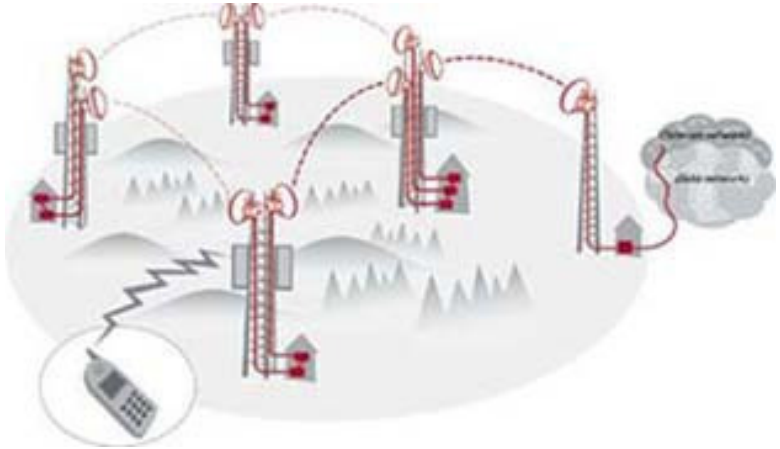
Special case: CSP (**k**=1)




VAC
MEDAC
MFDAC
MAC/MDAC
MNC
BT

Radio Link Frequency Assignment Problem

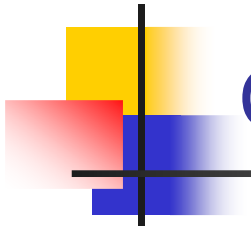
(Cabon et al., *Constraints* 1999) (Koster *et al.*, *4OR* 2003)



- Given a telecommunication network
- …find the best frequency for each communication link, avoiding interferences

- Best can be:
  - Minimize the maximum frequency, no interference (max)
  - **Minimize the global interference (sum)**
- Generalizes graph coloring problems: $|f_1 - f_2| \geq a$

***CELAR** problem size: n=100—458 ; d=44 ; m=1,000—5,000*



CELAR

toulbar2 v0.6 running on a 3 GHz computer with 16 GB

- **SCEN-06-sub1**
  n = 14, d = 44,
  m = 75, k = 2669

Solver: **toulbar2**
BB-VE(2), last conflict,dichotomic branching

	Time	Nodes
VAC	18 sec	25 10^3
EDAC	7 sec	38 10^3
FDAC	10 sec	72 10^3
AC	23 sec	410 10^3
DAC	150 sec	2.4 10^6
NC	897 sec	26 10^6

CELAR/Graph problems

(Cooper et al., AAAI 2008)

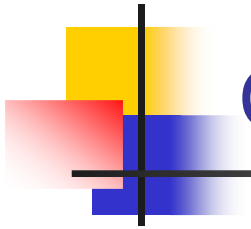
toulbar2 v0.6 running on a 3 GHz computer with 16 GB

- Maintaining VAC

  last conflict heuristics, BB-VE(2) during search, binary branching by domain splitting, premature VAC stop during search

- Closed 2 open problems by maintaining VAC

	nvars	tval	nctr	nodes	time	lb/iter	niter
graph11$_r$	232	5747	792	3272	3.7s	2.9	802
graph11	340	12820	1425	$2 \cdot 10^5$	13009s	6.63	260755
graph13$_r$	454	13153	2314	42	31s	6.7	1071
graph13	458	17588	4815	438	82s	0.5	3131



CELAR

toulbar2 v0.6 running on a 3 GHz computer with 16 GB

- SCEN-06-sub1

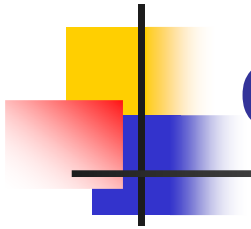
  n = 14, d = 44, m = 75, k = 2669

Variable ordering

Type of branching

toulbar2 results with EDAC	Domain / Degree	Last Conflict (Lecoutre.., 2006)
N-ary branching	234 sec. 6.1 10^6 nodes	N/A
Binary branching	197 sec. 6.7 10^6 nodes	40 sec. 247,000 nodes
Dichotomic branching	75 sec. 1.8 10^6 nodes	7 sec. 38,000 nodes

N-ary branching with dom/degree and no initial upper-bound: 265 sec., 7.2M nd.



CELAR: Local vs. Partial search with LC

toulbar2 v0.6 running on a 3 GHz computer with 16 GB

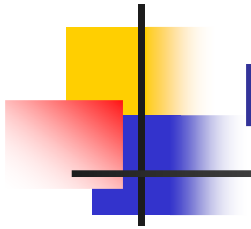
Local consistency enforcement informs variable and value ordering heuristics

Incomplete solvers

- **toulbar2** with Limited Discrepancy Search (Harvey & Ginsberg, IJCAI 1995) exploiting unary cost functions produced by EDAC
- **INCOP** (Neveu, Trombettoni and Glover, CP 2004)

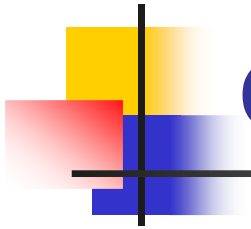
Intensification/Diversification Walk meta-heuristic

CELAR SCEN-06	Time	Solution cost
LDS(8), BB-VE(2)	35 sec	3394
INCOP	36 sec/run	3476 (best) 3750 (mean over 10 runs)



Perspectives

- Improve modeling capabilities
  - Global cost functions (Lee and Leung, IJCAI 2009) & Virtual GAC,…
- Study stronger soft local consistencies
  - Singleton arc consistency,…
- Extension to other tasks
  - Probabilistic inference,…



Outline

- Introduction
- Inference
- Search (OR)
- Lower-bounds and relaxations

- **Exploiting problem structure in search**
  - AND/OR search trees (linear space)
  - AND/OR Branch-and-Bound search
  - AND/OR search graphs (caching)
  - AND/OR search for 0-1 integer programming

- Software

Solution Techniques

AND/OR search

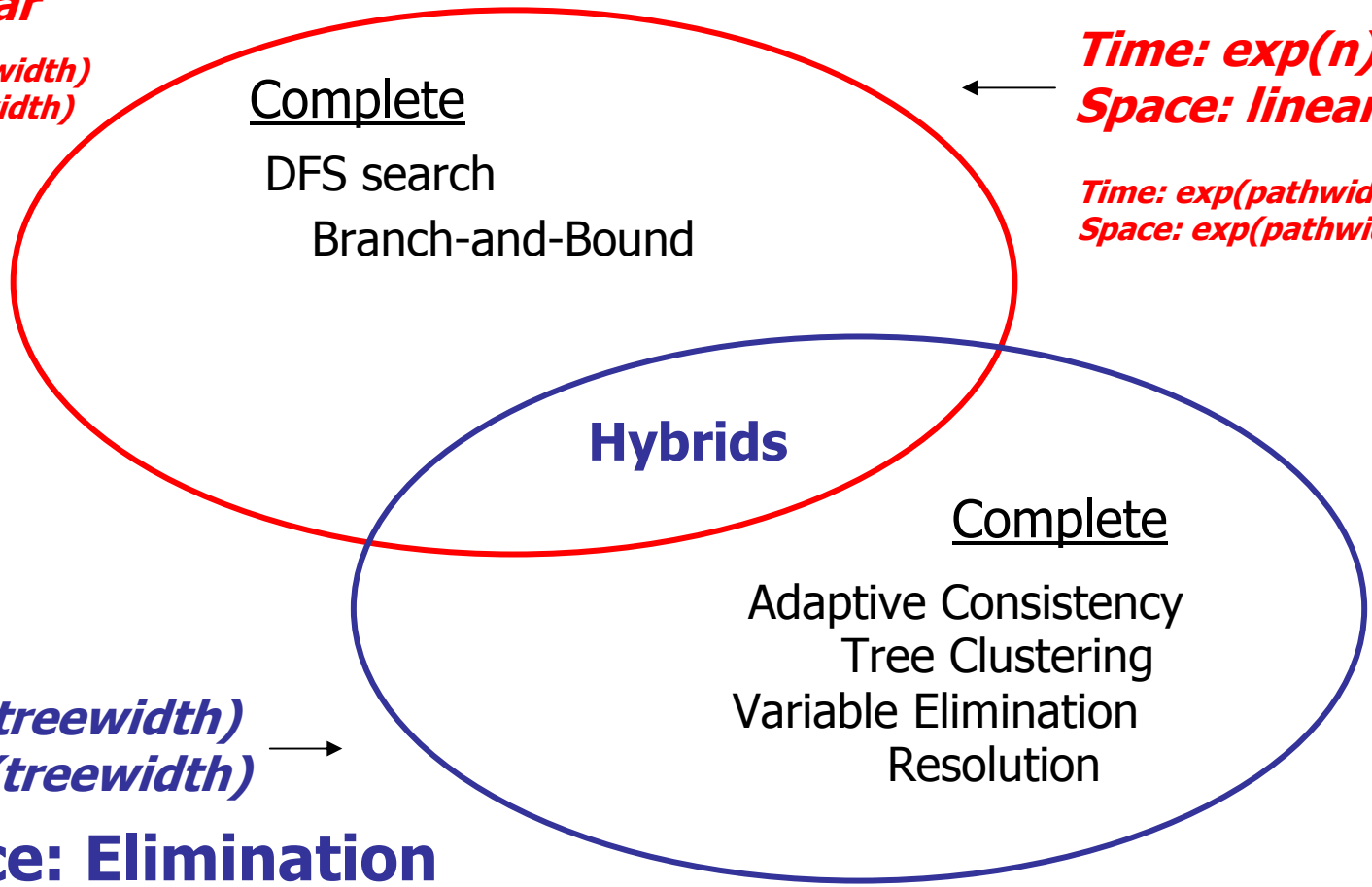
Time: exp(treewidth*log n)
Space: linear

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

Search: Conditioning

Time: exp(n)
Space: linear

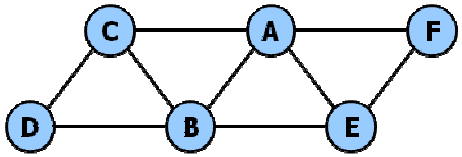
Time: exp(pathwidth)
Space: exp(pathwidth)



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

Inference: Elimination

Classic OR 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

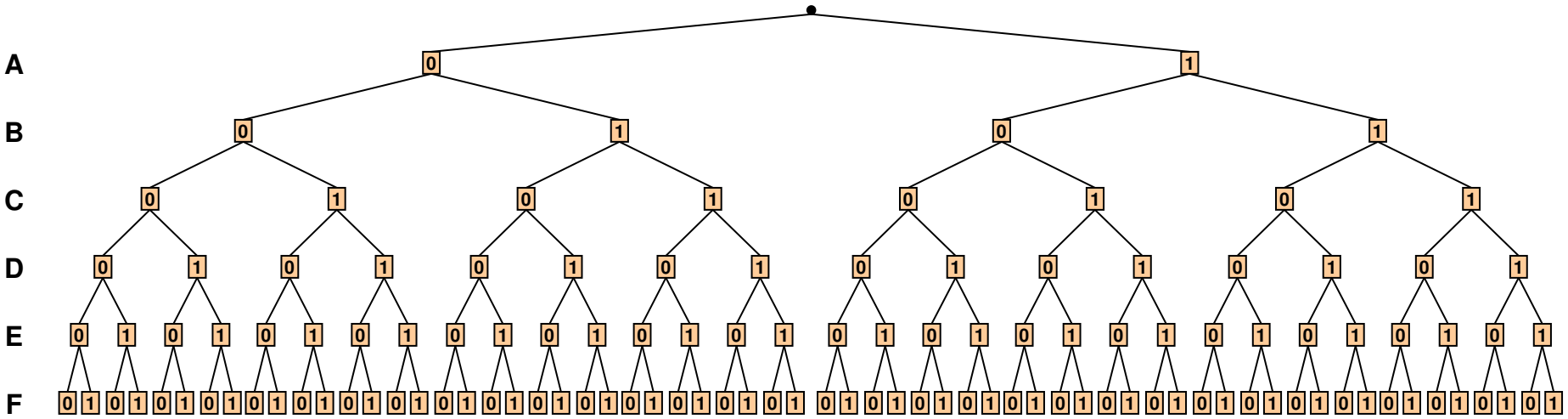
B	D	f_6
0	0	4
0	1	2
1	0	1
1	1	0

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

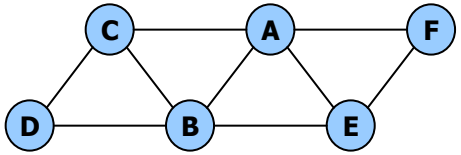
C	D	f_8
0	0	1
0	1	4
1	0	0
1	1	0

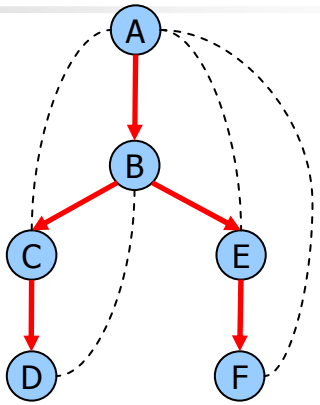
E	F	f_9
0	0	1
0	1	0
1	0	0
1	1	2

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

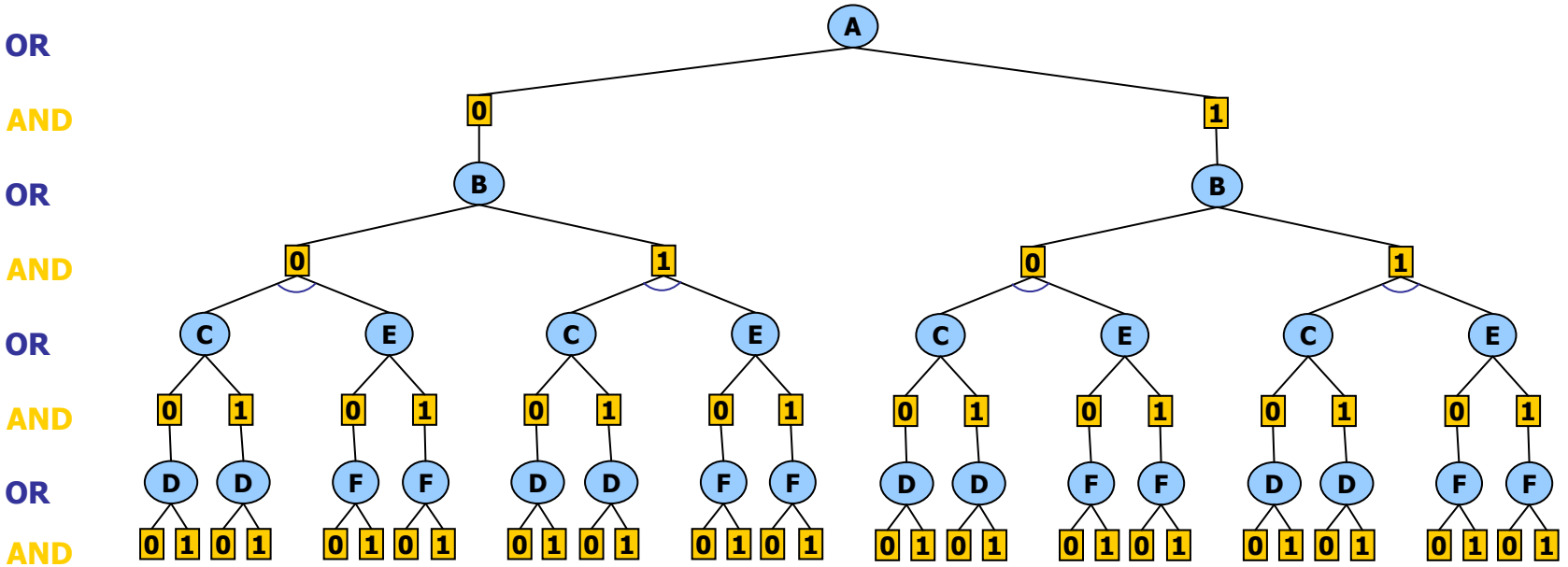


The AND/OR Search Tree

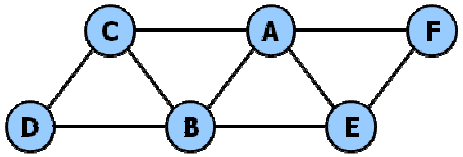


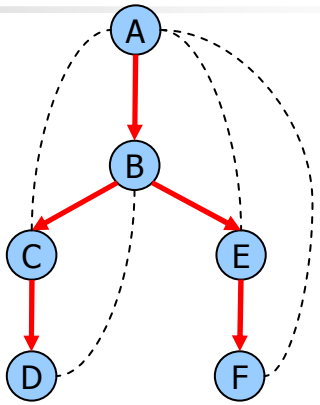


Pseudo tree (Freuder & Quinn85)

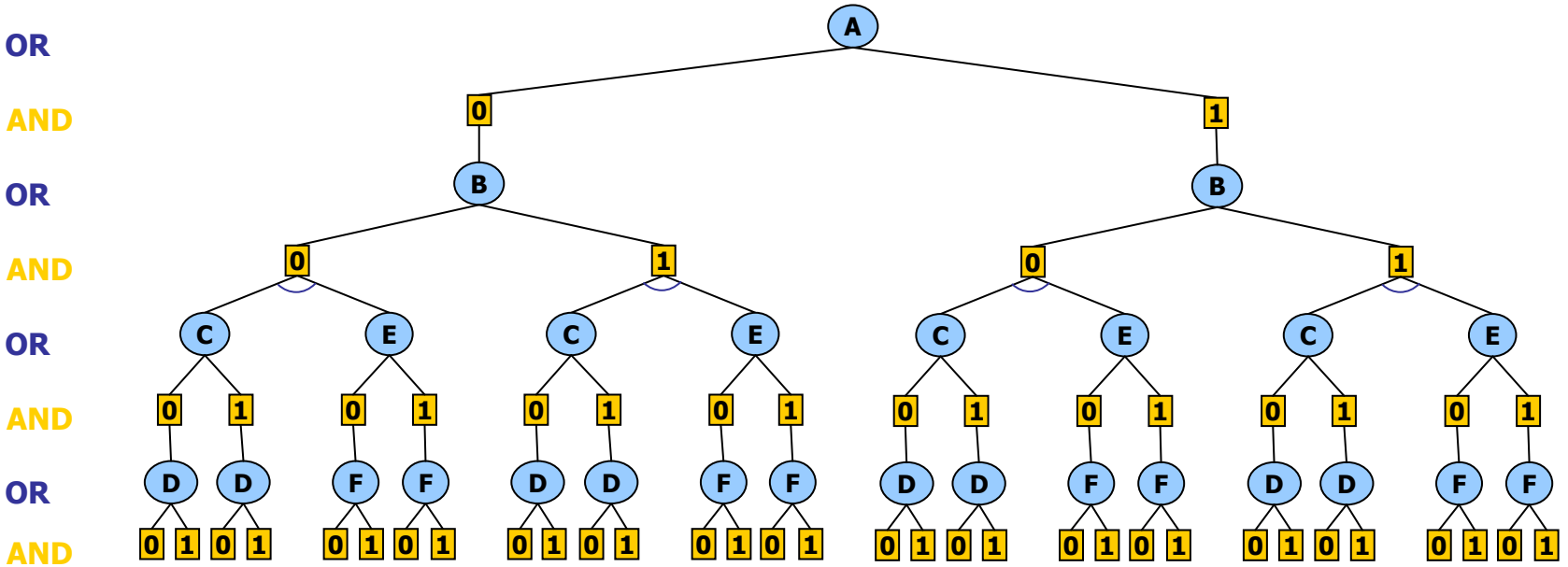


The AND/OR Search Tree



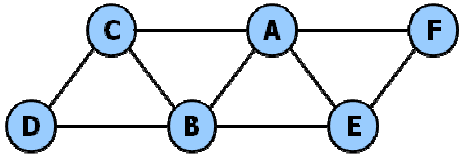


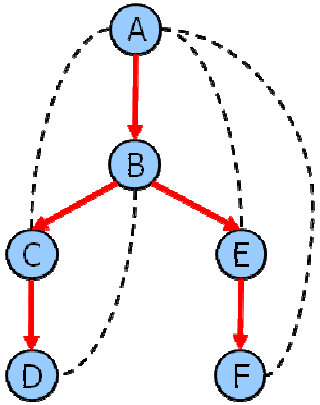
Pseudo tree



A solution subtree is (A=0, B=1, C=0, D=0, E=1, F=1)

Weighted AND/OR Search Tree





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

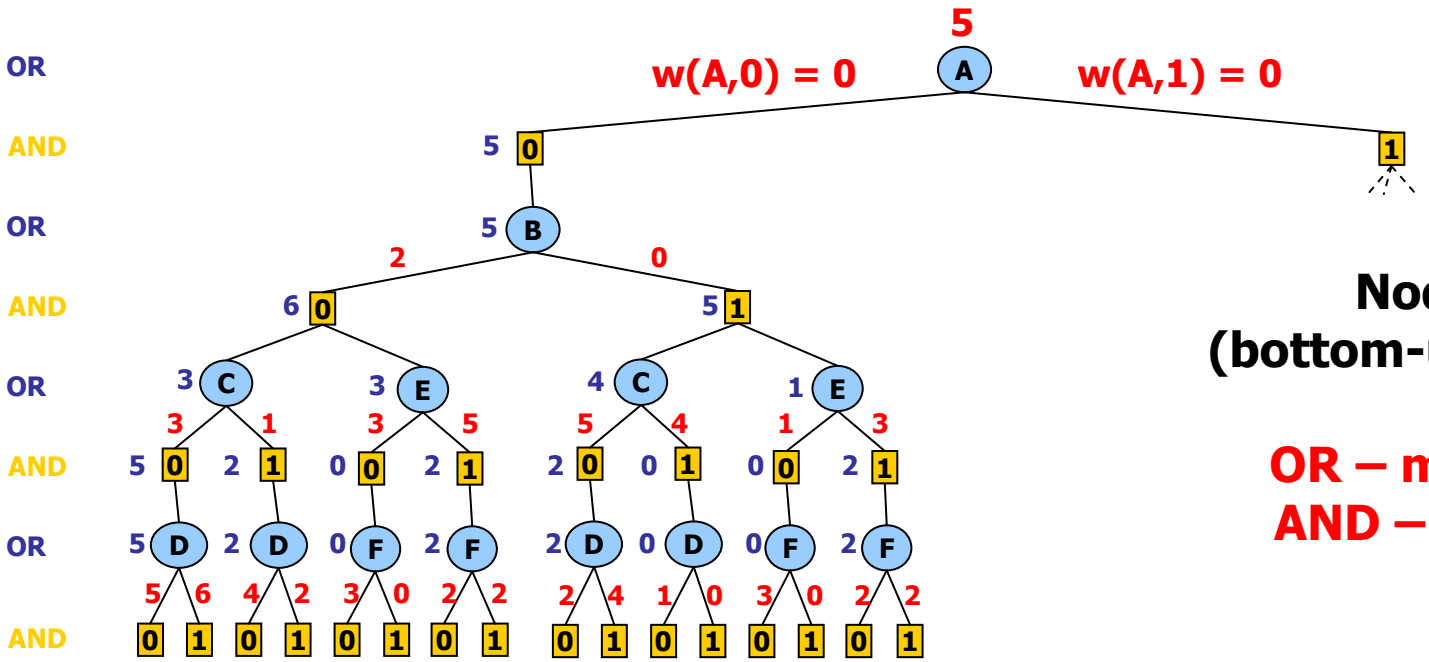
B	D	f_6
0	0	4
0	1	2
1	0	1
1	1	0

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

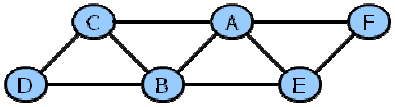
$$f(\mathrm{X}) = \min_{X} \sum_{i=1}^{9} f_i(\mathrm{X})$$

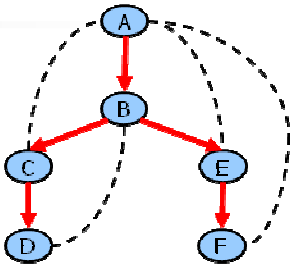


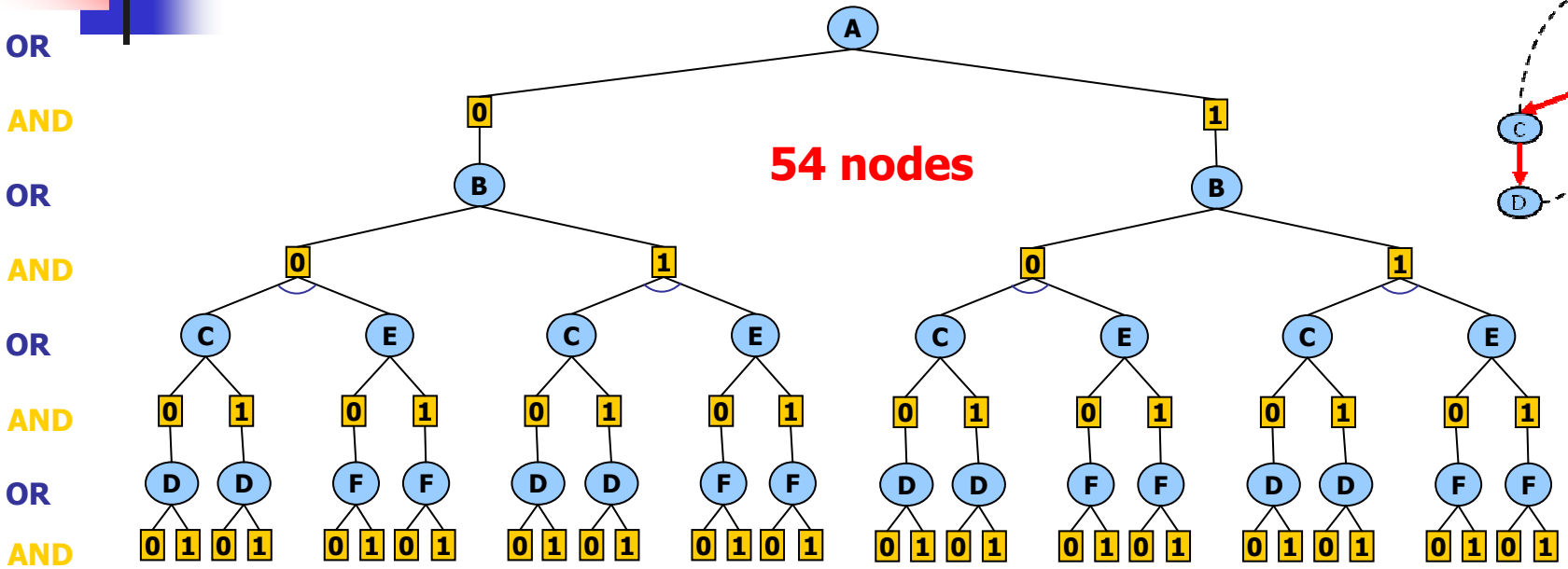
Node Value
(bottom-up evaluation)

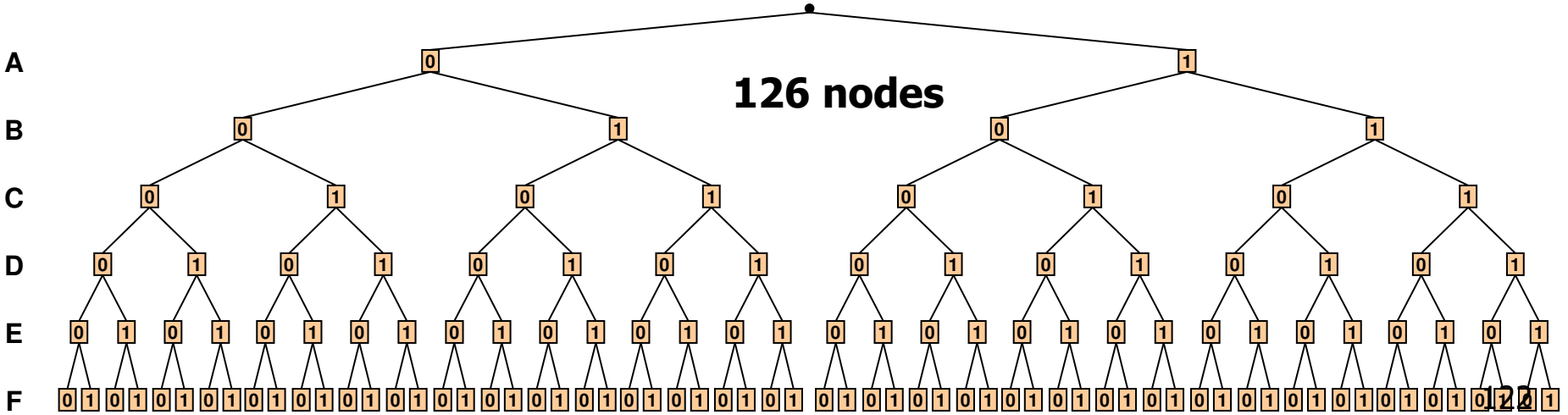
OR – minimization
AND – summation

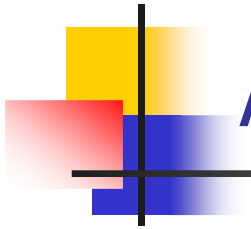
AND/OR vs. OR Spaces











AND/OR vs. OR Spaces

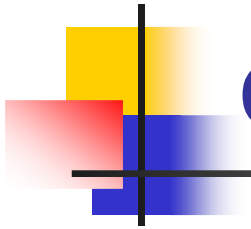
width	depth	OR space		AND/OR space		
		Time (sec.)	Nodes	Time (sec.)	AND nodes	OR nodes
5	10	3.15	2,097,150	0.03	**10,494**	5,247
4	9	3.13	2,097,150	0.01	**5,102**	2,551
5	10	3.12	2,097,150	0.03	**8,926**	4,463
4	10	3.12	2,097,150	0.02	**7,806**	3,903
5	13	3.11	2,097,150	0.10	**36,510**	18,255

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

Complexity of AND/OR Tree Search

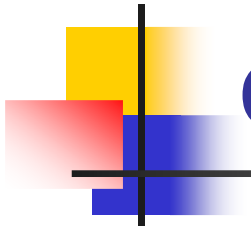
	AND/OR tree	OR tree
Space	$O(n)$	$O(n)$
Time	$O(n\ d^t)$ $O(n\ d^{w^* \log n})$ (Freuder & Quinn85), (Collin, Dechter & Katz91), (Bayardo & Miranker95), (Darwiche01)	$O(d^n)$

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



Constructing Pseudo Trees

- AND/OR search algorithms are influenced by the quality of the pseudo tree
- Finding the minimal induced width / depth pseudo tree is NP-hard
- Heuristics
  - Min-Fill (min induced width)
  - Hypergraph partitioning (min depth)



Constructing Pseudo Trees

- **Min-Fill** (Kjaerulff90)
  - Depth-first traversal of the induced graph obtained along the min-fill elimination order
  - Variables ordered according to the smallest "fill-set"

- **Hypergraph Partitioning** (Karypis &Kumar00)
  - Functions are vertices in the hypergraph and variables are hyperedges
  - Recursive decomposition of the hypergraph while minimizing the separator size at each step
  - Using state-of-the-art software package hMeTiS

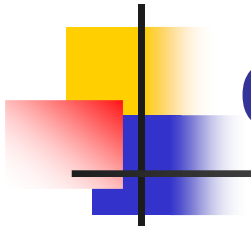
Quality of the Pseudo Trees

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



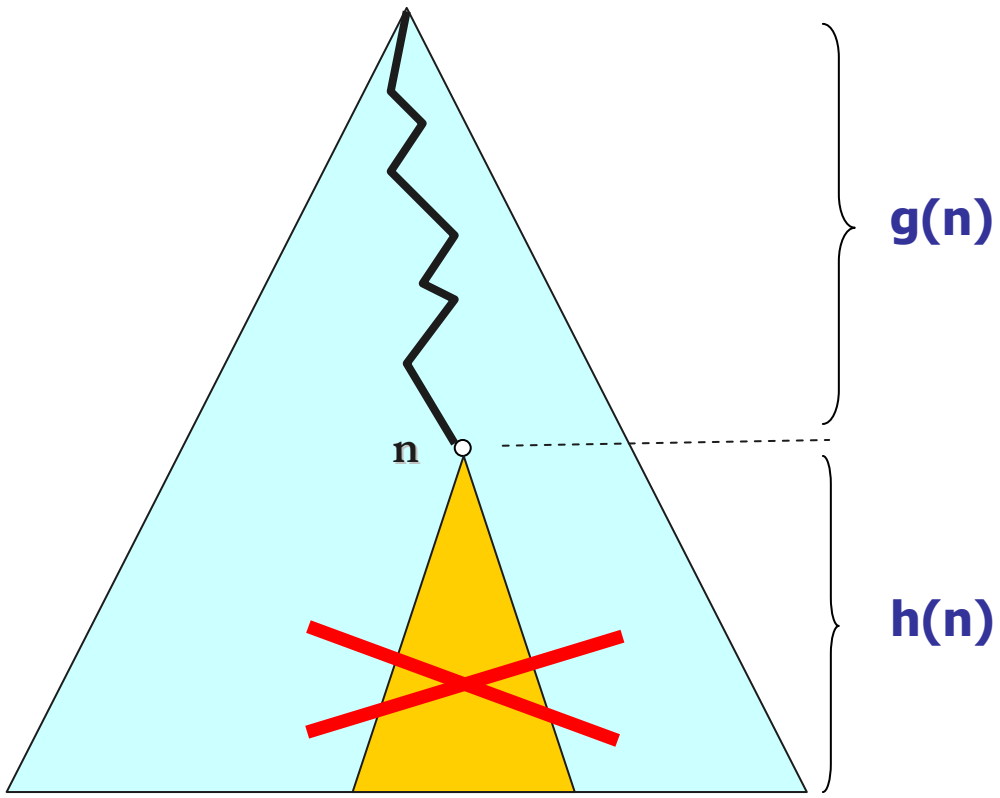
Outline

- Introduction
- Inference
- Search (OR)
- Lower-bounds and relaxations

- **Exploiting problem structure in search**
  - AND/OR search trees
  - AND/OR Branch-and-Bound search
    - Lower bounding heuristics
    - Dynamic variable orderings
  - AND/OR search graphs (caching)
  - AND/OR search for 0-1 integer programming

- Software & Applications

Classic Branch-and-Bound Search



OR Search Tree

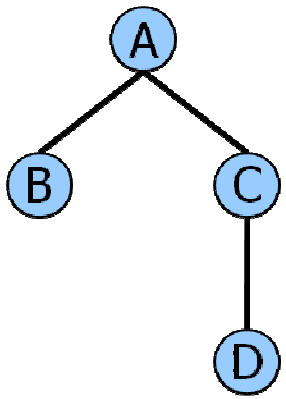
Upper Bound **UB**

Lower Bound **LB**

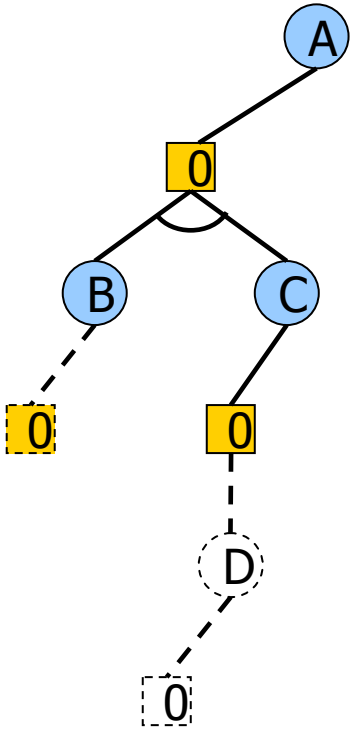
LB(n) = g(n) + h(n)

Prune if LB(n) ≥ UB

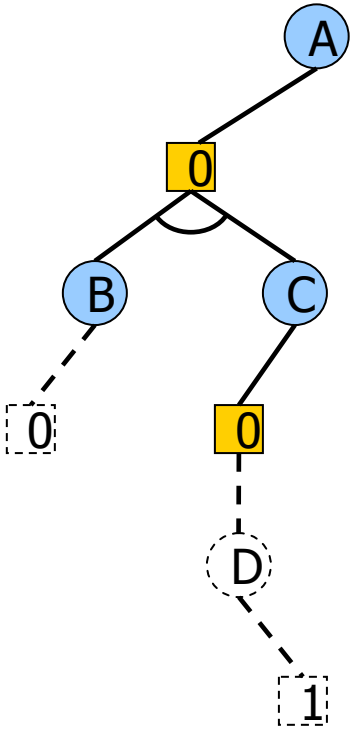
Partial Solution Tree



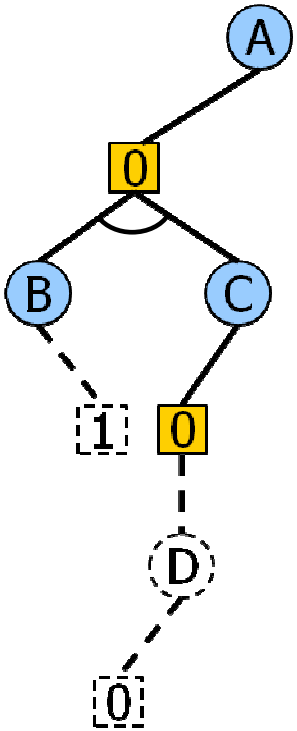
Pseudo tree



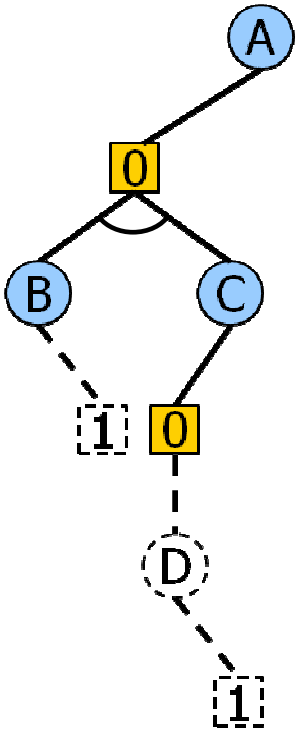
(A=0, B=0, C=0, D=0)



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



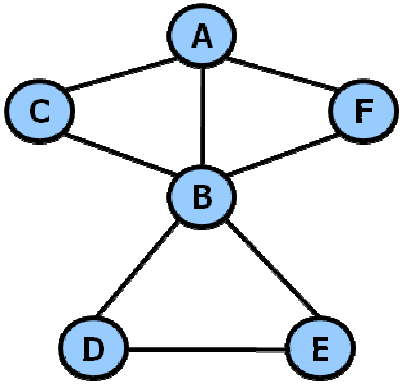
(A=0, B=1, C=0, D=0)



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

Extension(T') – solution trees that extend T'

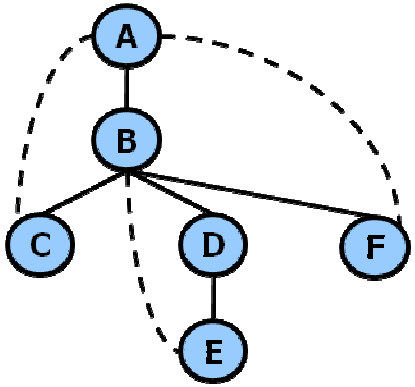
Exact Evaluation Function

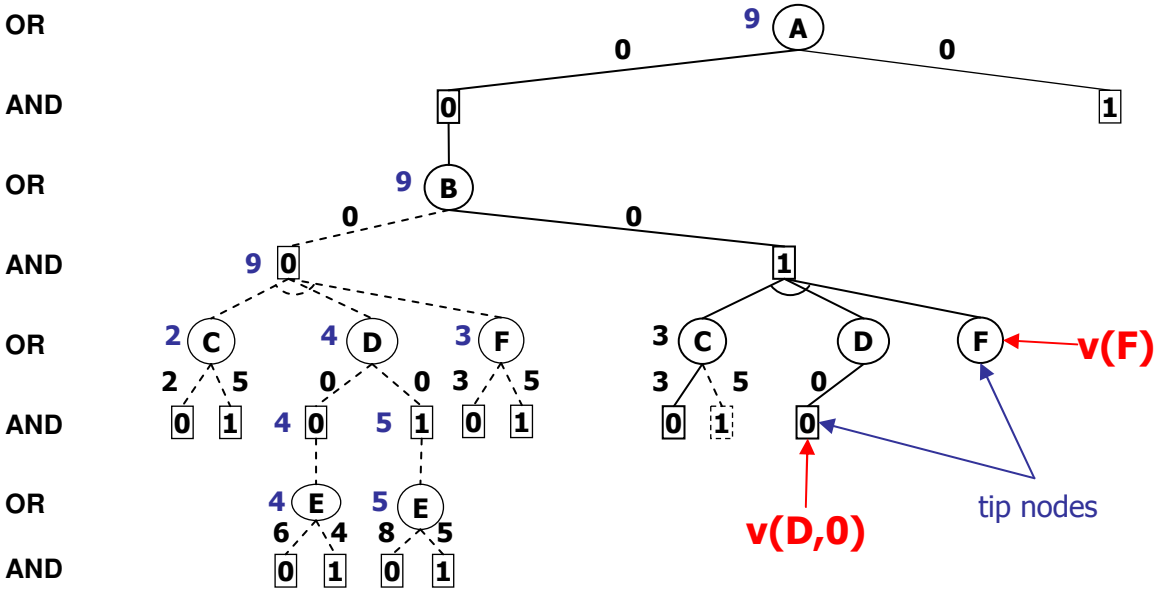


A	B	C	f_1(ABC)
0	0	0	2
0	0	1	5
0	1	0	3
0	1	1	5
1	0	0	9
1	0	1	3
1	1	0	7
1	1	1	2

A	B	F	f_2(ABF)
0	0	0	3
0	0	1	5
0	1	0	1
0	1	1	4
1	0	0	6
1	0	1	5
1	1	0	6
1	1	1	5

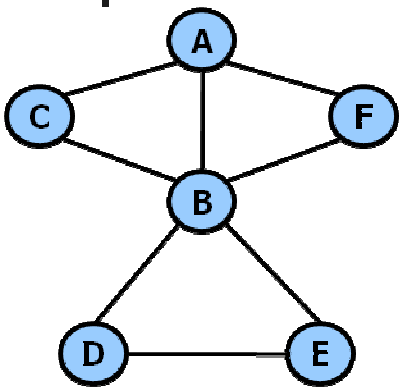
B	D	E	f_3(BDE)
0	0	0	6
0	0	1	4
0	1	0	8
0	1	1	5
1	0	0	9
1	0	1	3
1	1	0	7
1	1	1	4





f*(T') = w(A,0) + w(B,1) + w(C,0) + w(D,0) + v(D,0) + v(F)

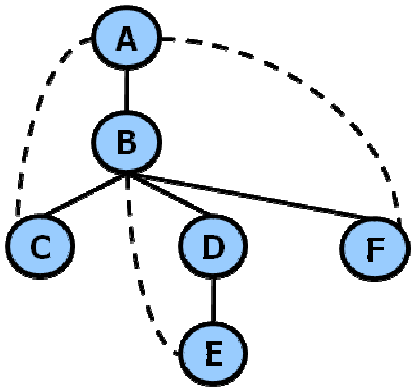
Heuristic Evaluation Function

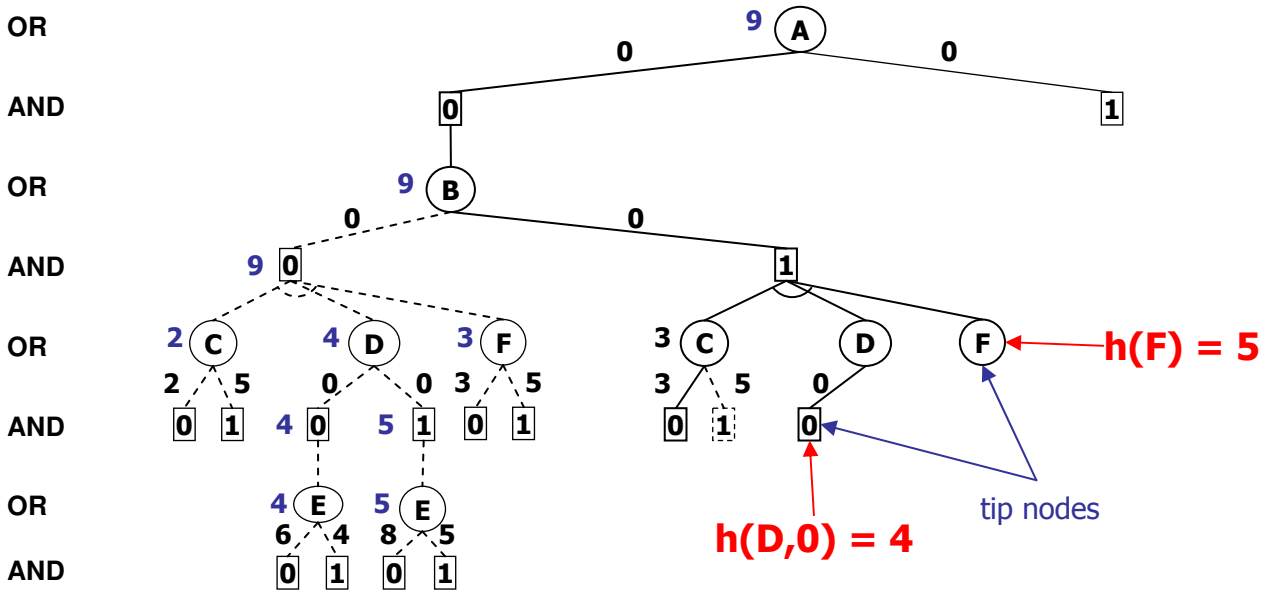


A	B	C	f_1(ABC)
0	0	0	2
0	0	1	5
0	1	0	3
0	1	1	5
1	0	0	9
1	0	1	3
1	1	0	7
1	1	1	2

A	B	F	f_2(ABF)
0	0	0	3
0	0	1	5
0	1	0	1
0	1	1	4
1	0	0	6
1	0	1	5
1	1	0	6
1	1	1	5

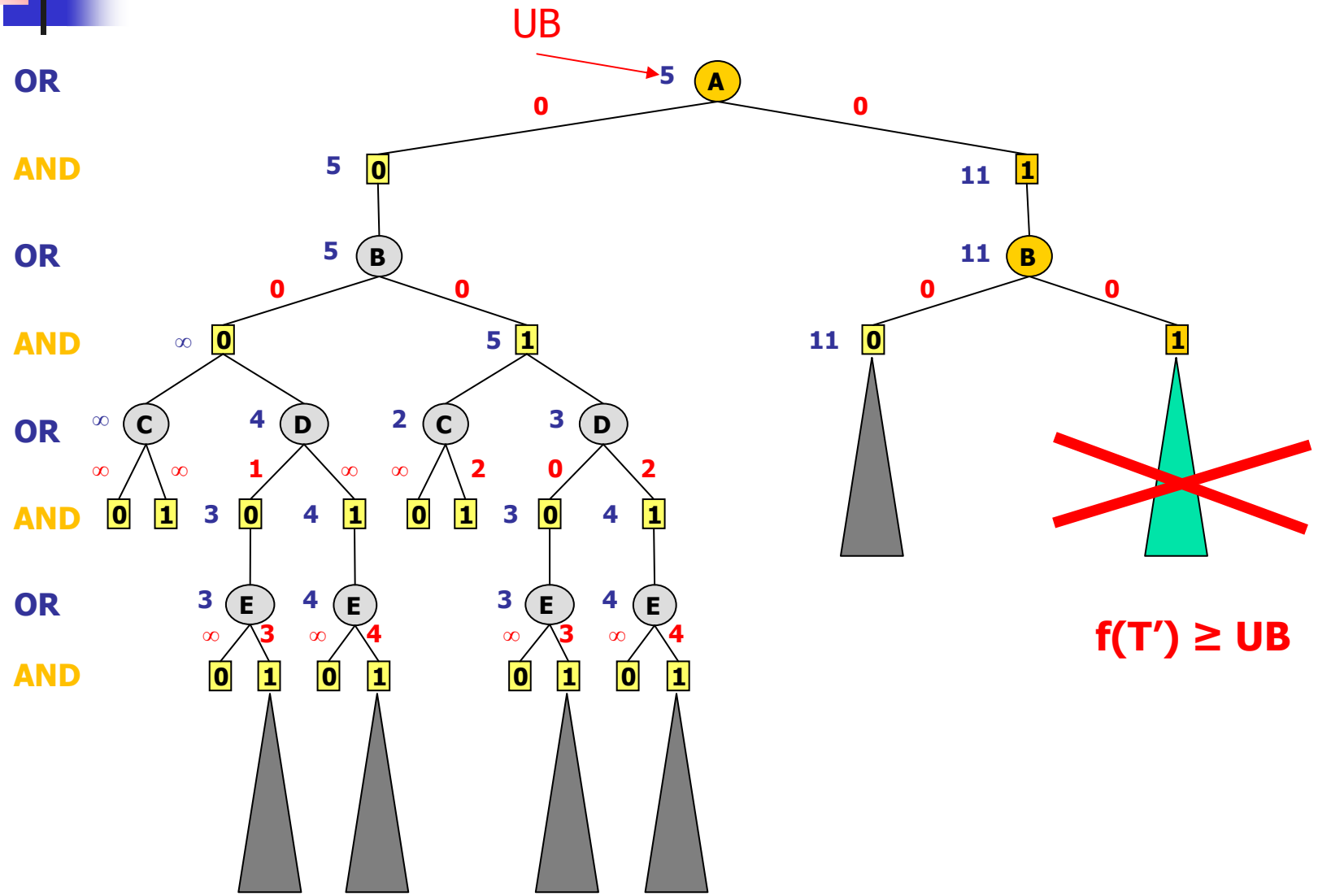
B	D	E	f_3(BDE)
0	0	0	6
0	0	1	4
0	1	0	8
0	1	1	5
1	0	0	9
1	0	1	3
1	1	0	7
1	1	1	4

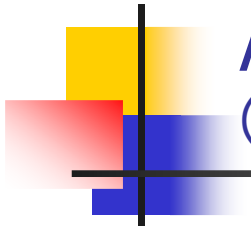




f(T') = w(A,0) + w(B,1) + w(C,0) + w(D,0) + h(D,0) + h(F) = 12 ≤ f*(T')

AND/OR Branch-and-Bound Search





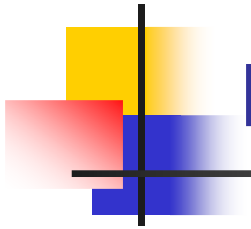
AND/OR Branch-and-Bound Search (AOBB)

(Marinescu & Dechter, IJCAI'05)

- Associate each node n with a heuristic lower bound h(n) on v(n)

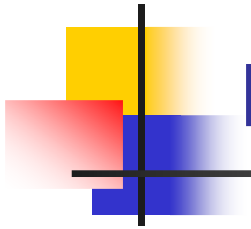
- EXPAND (top-down)
  - Evaluate f(T') and prune search if f(T') ≥ UB
  - Expand the tip node n

- PROPAGATE (bottom-up)
  - Update value of the parent p of n
    - OR nodes: minimization
    - AND nodes: summation



Heuristics for AND/OR Branch-and-Bound

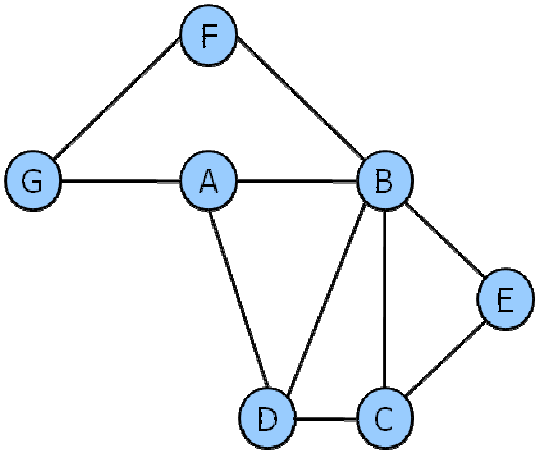
- In the AND/OR search space **h(n)** can be computed using any heuristic. We used:
  - Static Mini-Bucket heuristics
    (Kask & Dechter, AIJ'01), (Marinescu & Dechter, IJCAI'05)
  - Dynamic Mini-Bucket heuristics
    (Marinescu & Dechter, IJCAI'05)
  - Maintaining local consistency
    (Larrosa & Schiex, AAAI'03), (de Givry et al., IJCAI'05)
  - LP relaxations
    (Nemhauser & Wosley, 1998)

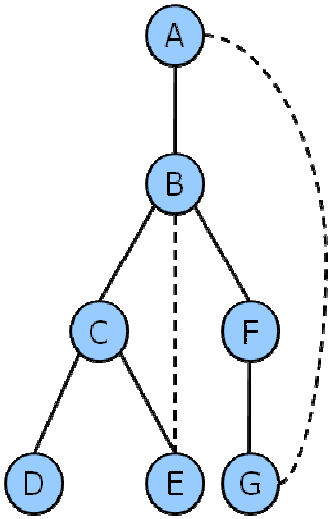


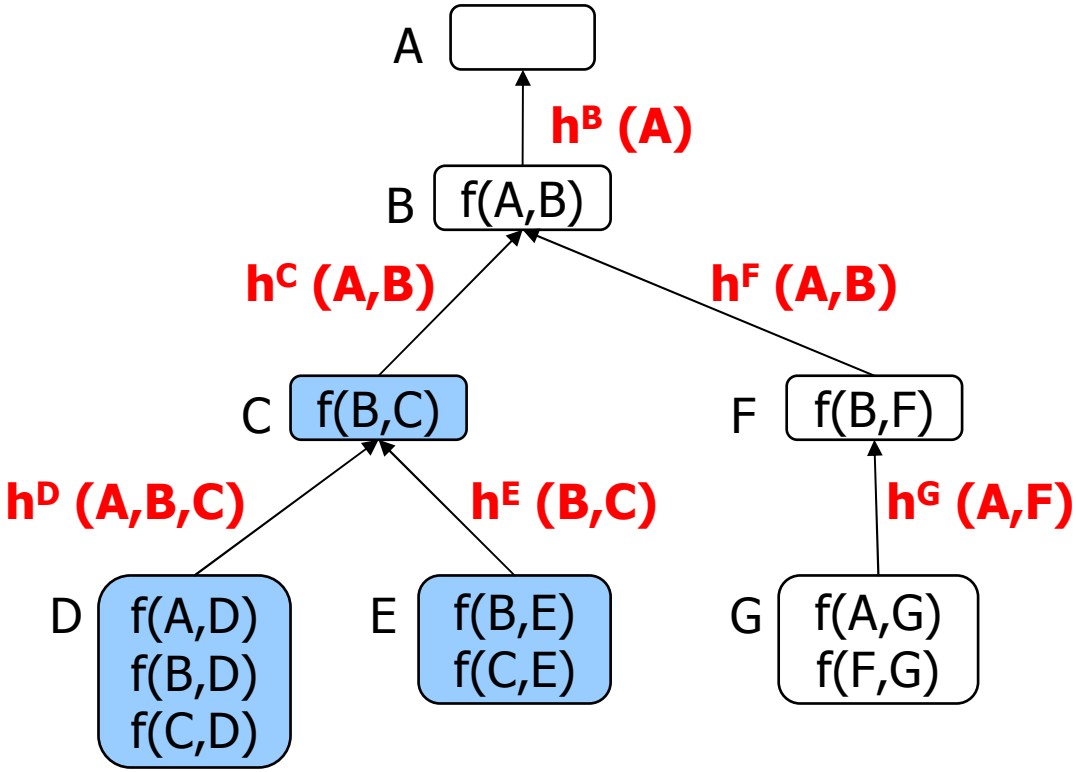
Mini-Bucket Heuristics

- Static Mini-Buckets
  - Pre-compiled
  - Reduced overhead
  - Less accurate
  - Static variable ordering
- Dynamic Mini-Buckets
  - Computed dynamically
  - Higher overhead
  - High accuracy
  - Dynamic variable ordering

Bucket Elimination



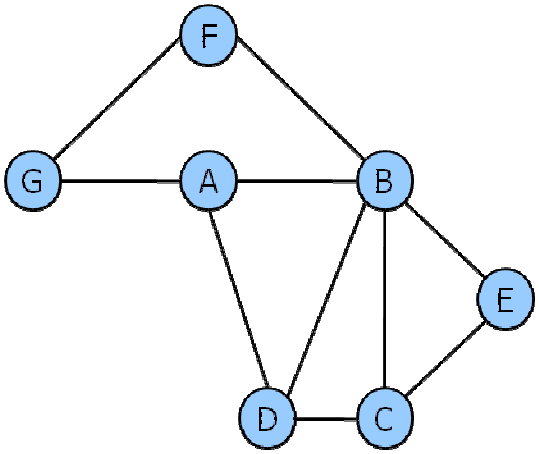


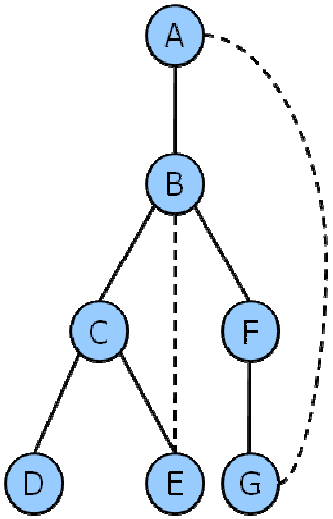


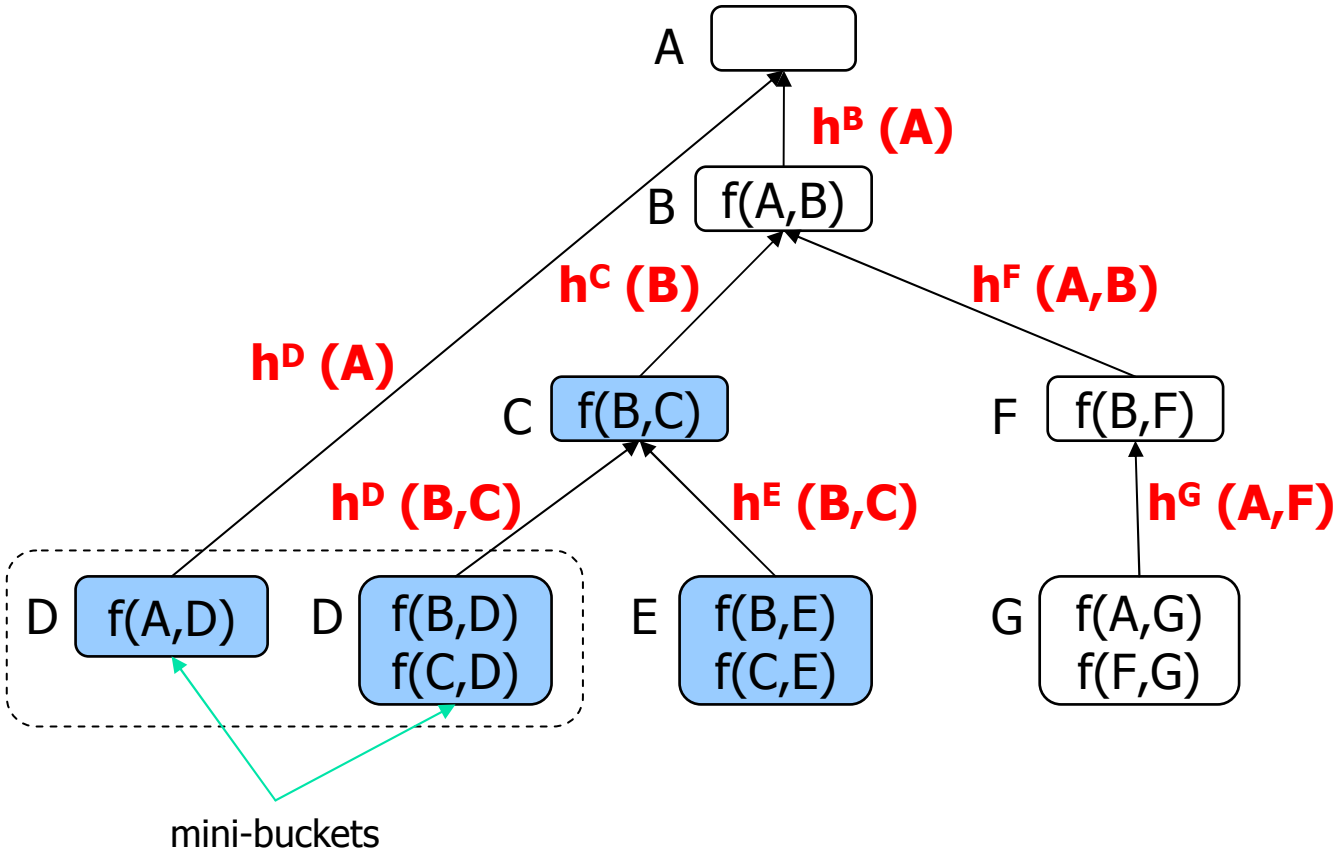
$$h^*(a, b, c) = h^D(a, b, c) + h^E(b, c)$$

Ordering: (A, B, C, D, E, F, G)

Static Mini-Bucket Heuristics



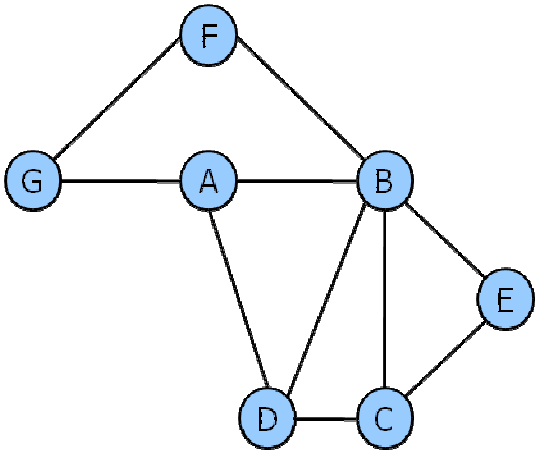


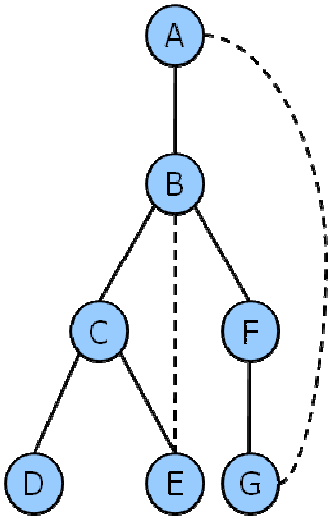


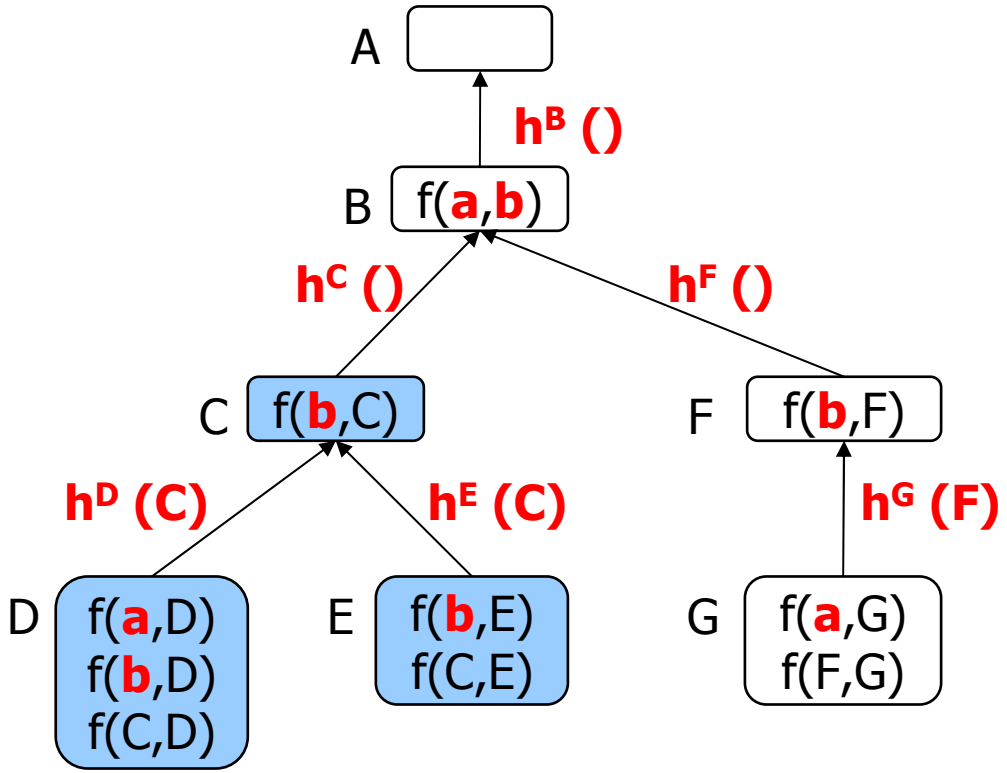
$$h(a, b, c) = h^D(a) + h^D(b, c) + h^E(b, c) \leq h^*(a, b, c)$$

Ordering: (A, B, C, D, E, F, G)

Dynamic Mini-Bucket Heuristics

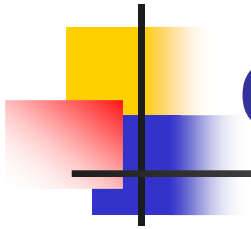






$h(a, b, c) = h^D(c) + h^E(c)$
$= h^*(a, b, c)$

Ordering: (A, B, C, D, E, F, G)

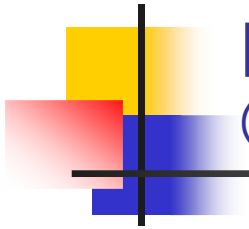


Outline

- Introduction
- Inference
- Search (OR)
- Lower-bounds and relaxations

- **Exploiting problem structure in search**
  - AND/OR search trees
  - AND/OR Branch-and-Bound search
    - Lower bounding heuristics
    - Dynamic variable orderings
  - AND/OR search graphs (caching)
  - AND/OR search for 0-1 integer programming

- Software & Applications



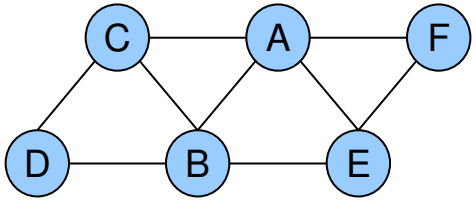
Dynamic Variable Orderings

(Marinescu & Dechter, ECAI'06)

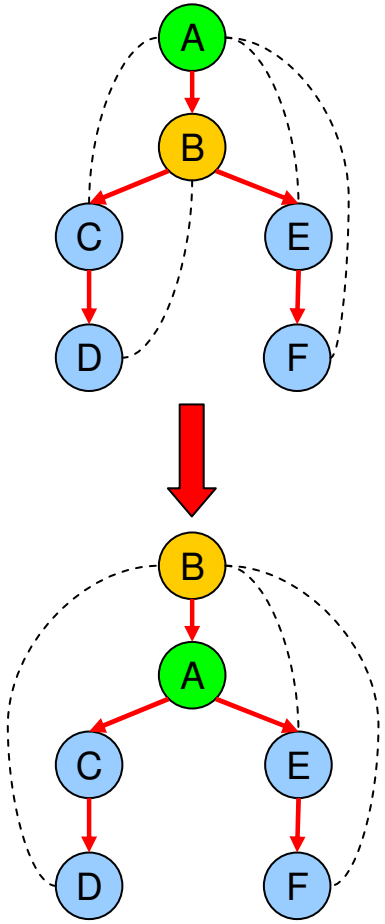
- Variable ordering heuristics:
  - **Semantic-based**
    - Aim at shrinking the size of the search space based on context and current value assignments
      - e.g. min-domain, min-dom/deg, min reduced cost
  - **Graph-based**
    - Aim at maximizing the problem decomposition
      - e.g. pseudo-tree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

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

Partial Variable Ordering



Primal graph



Variable Groups/Chains:
- {A,B}
- {C,D}
- {E,F}

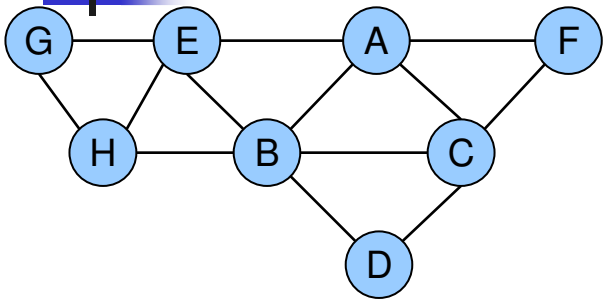
Instantiate {A,B}
before {C,D} and {E,F}

*{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 & Terrioux04)]

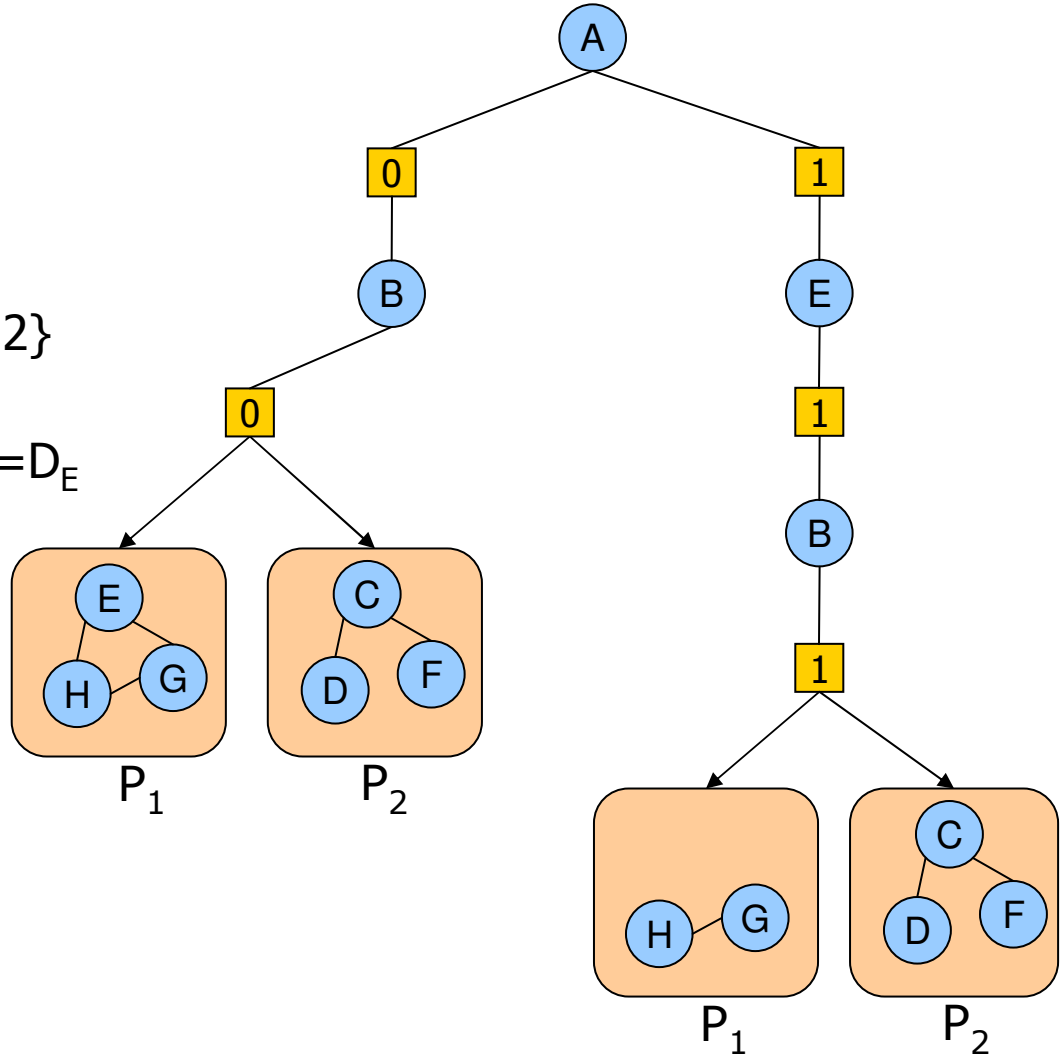
Full Dynamic Variable Ordering



domains $D_A=\{0,1\}$ $D_B=\{0,1,2\}$
$D_E=\{0,1,2,3\}$
$D_C=D_D=D_F=D_G=D_H=D_E$

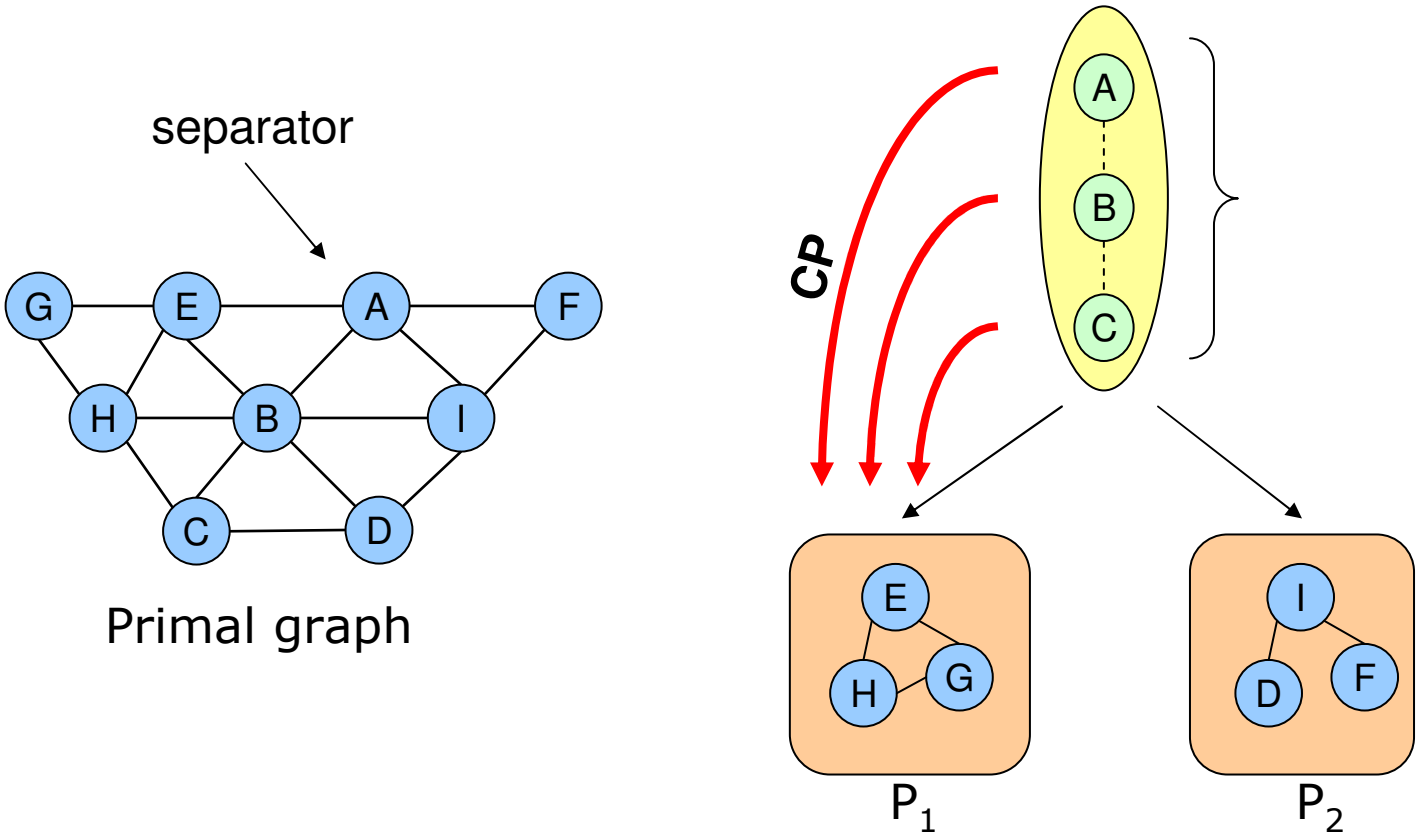
A	B	f(AB)
0	0	**3**
0	1	**8**
0	2	**8**
1	0	**4**
1	1	**0**
1	2	**6**

A	E	f(AE)
0	0	**0**
0	1	**5**
0	2	**1**
0	3	**4**
1	0	**8**
1	1	**8**
1	2	**0**
1	3	**5**



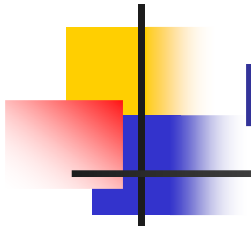
[similar idea exploited in #SAT (Bayardo & Pehoushek00)]

Dynamic Separator Ordering



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

[similar idea exploited in SAT (Li & van Beek04)]



Experiments

- Benchmarks
  - Belief Networks (BN)
  - Weighted CSPs (WCSP)
- Algorithms
  - AOBB
  - SamIam (BN)
  - Superlink (Genetic linkage analysis)
  - Toolbar (ie, DFBB+EDAC)
- Heuristics
  - Mini-Bucket heuristics (BN, WCSP)
  - EDAC heuristics (WCSP)

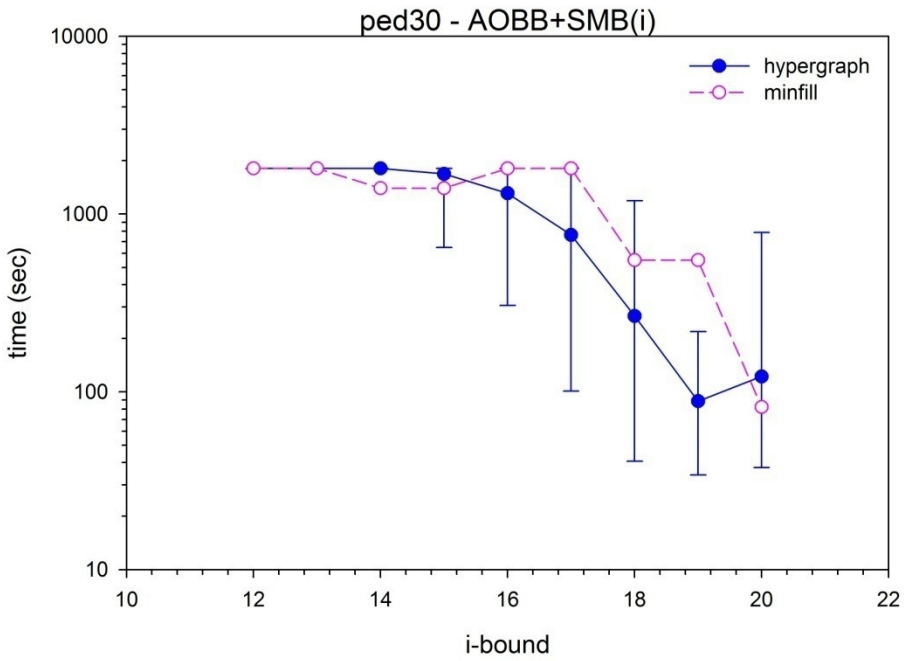
Genetic Linkage Analysis

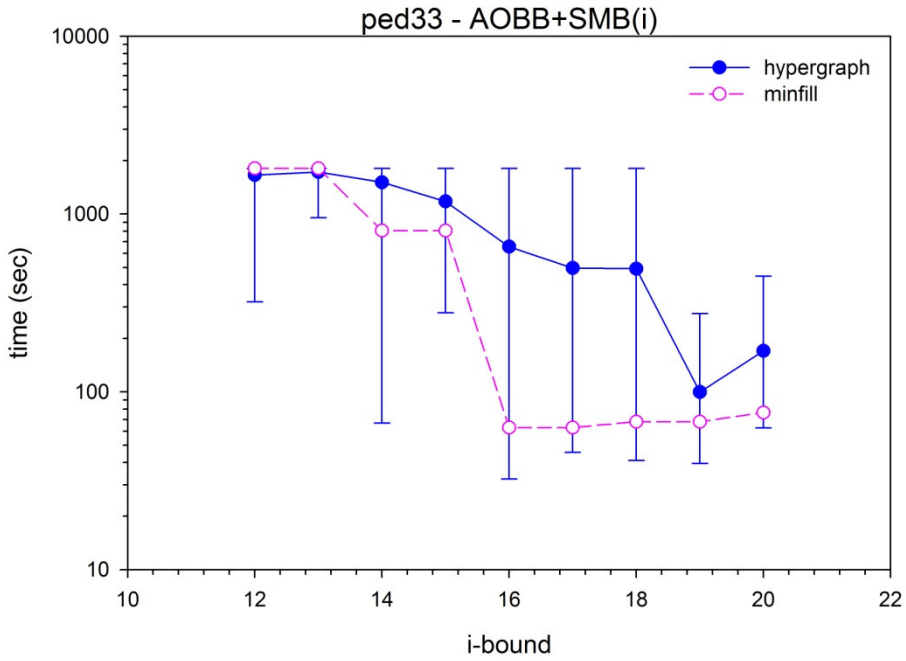
(Fishelson&Geiger02)

pedigree (n, d) (w*, h)	**Superlink** v. 1.6 time	**SamIam** v. 2.3.2 time	**MBE(i) BB+SMB(i) AOBB+SMB(i)** i=12 time	i=12 nodes	**MBE(i) BB+SMB(i) AOBB+SMB(i)** i=16 time	i=16 nodes	**MBE(i) BB+SMB(i) AOBB+SMB(i)** i=20 time	i=20 nodes
ped18 (1184, 5) (21, 119)	139.06	157.05	0.51 - -	- -	4.59 - 270.96	- 2,555,078	19.30 - **20.27**	- 7,689
ped25 (994, 5) (29, 53)	-	out	0.34 - -	- -	3.20 - -	- -	33.42 - **1894.17**	- 11,709,153
ped30 (1016, 5) (25, 51)	13095.83	out	0.31 - 5563.22	- 63,068,960	2.66 - 1811.34	- 20,275,620	24.88 - **82.25**	- 588,558
ped33 (581, 5) (26, 48)	-	out	0.41 - 2335.28	- 32,444,818	5.28 - **62.91**	- 807,071	51.24 - 76.47	- 320,279
ped39 (1272, 5) (23, 94)	322.14	out	0.52 - -	- -	8.41 - 4041.56	- 52,804,044	81.27 - 141.23	- 407,280

Min-fill pseudo tree. Time limit 3 hours.

Impact of the Pseudo Tree





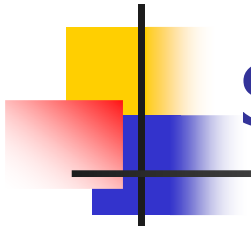
Runtime distribution for hypergraph pseudo trees over 20 independent runs.
ped30 and **ped33** linkage networks.

Dynamic Variable Orderings

(Bensana et al.99)

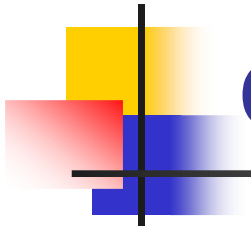
spot5	n	w*		toolbar	BBEDAC	AOEDAC	AOEDAC+PVO	DVO+AOEDAC	AOEDAC+DSO
	c	h							
29	16	7	time	4.56	109.66	613.79	545.43	**0.83**	11.36
	57	8	nodes	218,846	710,122	8,997,894	7,837,447	8,698	92,970
42b	14	9	time	-	-	-	-	-	**6825.4**
	75	9	nodes	-	-	-	-	-	27,698,614
54	14	9	time	0.31	0.97	31.34	9.11	**0.06**	0.75
	75	9	nodes	21,939	8,270	823,326	90,495	688	6,614
404	16	10	time	151.11	2232.89	255.83	152.81	12.09	**1.74**
	89	12	nodes	6,215,135	7,598,995	3,260,610	1,984,747	88,079	14,844
408b	18	10	time	-	-	-	-	-	**747.71**
	106	13	nodes	-	-	-	-	-	2,134,472
503	22	11	time	-	-	-	-	-	**53.72**
	131	15	nodes	-	-	-	-	-	231,480

SPOT5 benchmark. Time limit 2 hours.



Summary

- New generation of depth-first AND/OR Branch-and-Bound search
- Heuristics based on
  - Mini-Bucket approximation (static, dynamic)
  - Local consistency (EDAC)
- Dynamic variable orderings
- Superior to state-of-the-art solvers traversing the classic OR search space



Outline

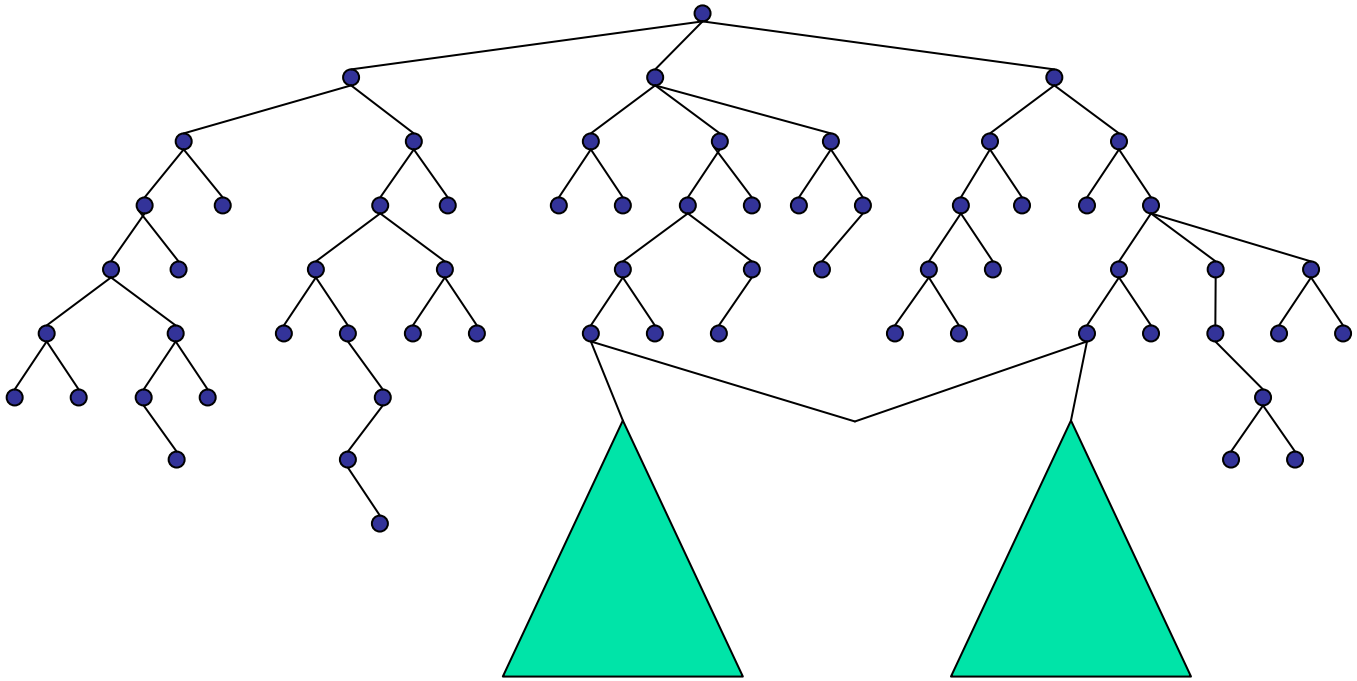
- Introduction
- Inference
- Search (OR)
- Lower-bounds and relaxations

- **Exploiting problem structure in search**
  - AND/OR search trees
  - AND/OR Branch-and-Bound search
  - AND/OR search graphs (caching)
    - AND/OR Branch-and-Bound with caching
    - Best-First AND/OR search
  - AND/OR search for 0-1 integer programming

- Software

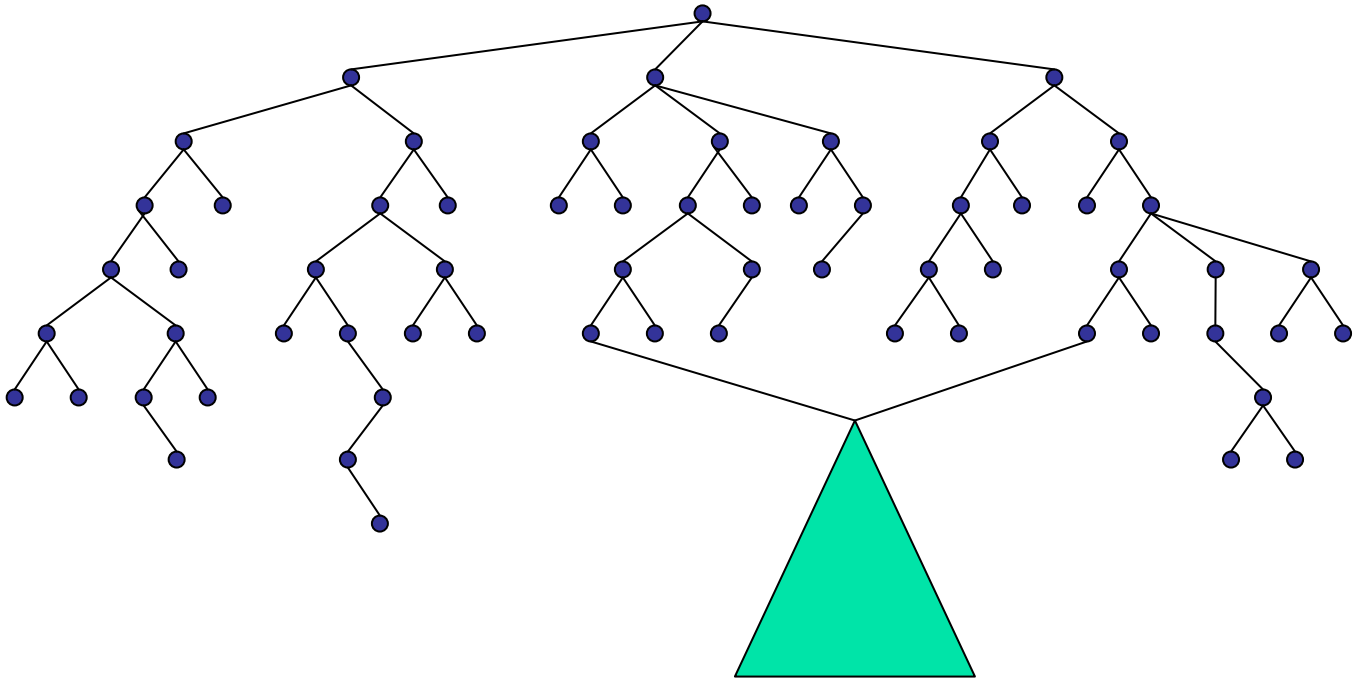
From Search Trees to Search Graphs

- Any two nodes that root identical sub-trees or sub-graphs can be merged



From Search Trees to Search Graphs

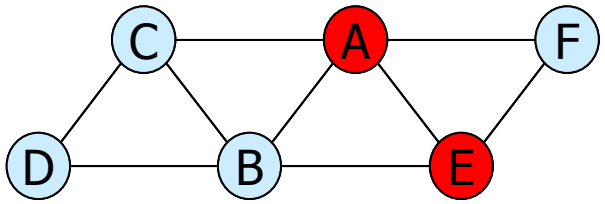
- Any two nodes that root identical sub-trees or sub-graphs can be merged

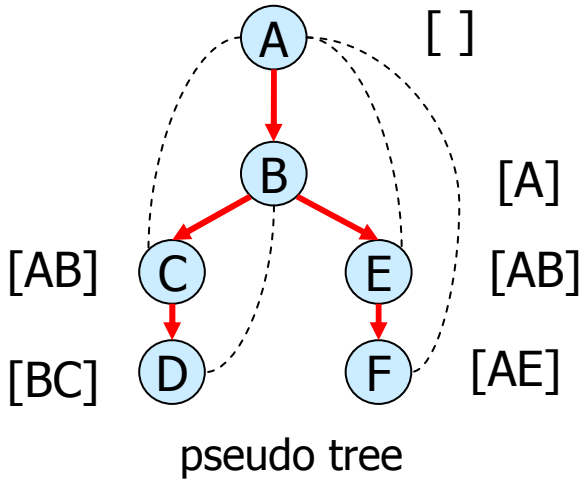


Merging Based on Context

- One way of recognizing nodes that can be merged (based on graph structure)

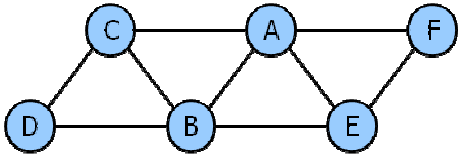
  context(X) = ancestors of X in the pseudo tree that are connected to X, or to descendants of X

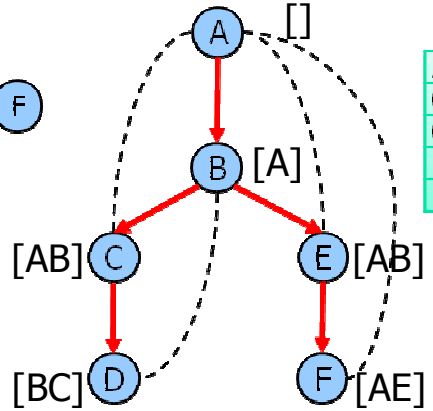




pseudo tree

AND/OR Search Graph





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

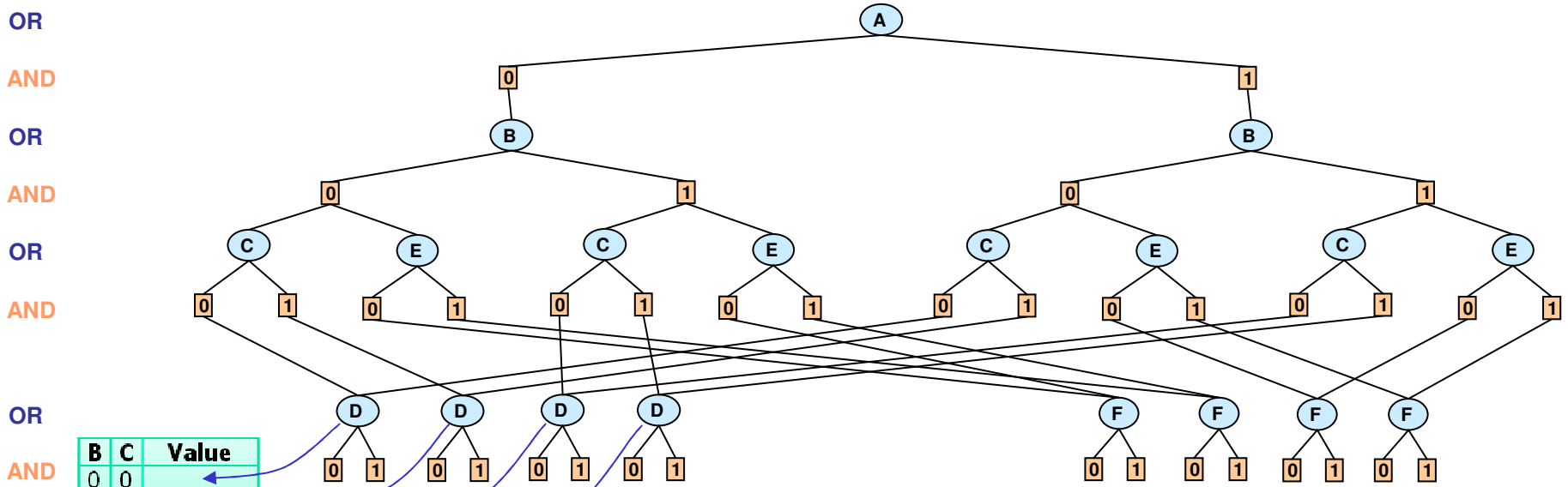
B	D	f_6
0	0	4
0	1	2
1	0	1
1	1	0

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

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



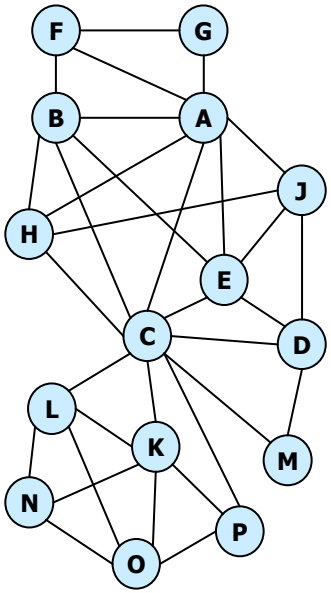
B	C	Value
0	0	
0	1	
1	0	
1	1	

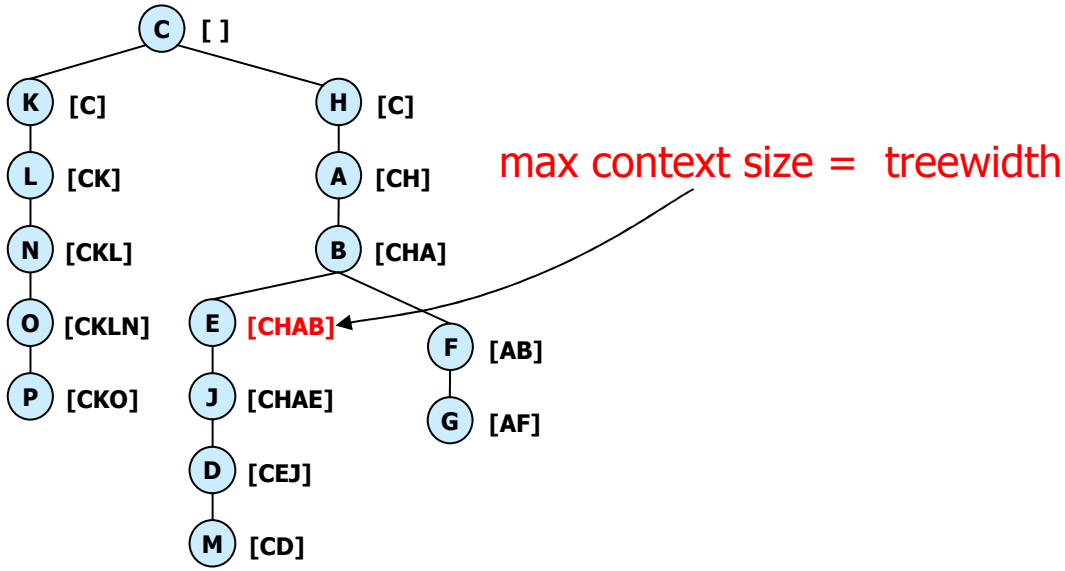
Cache table for D

context minimal graph

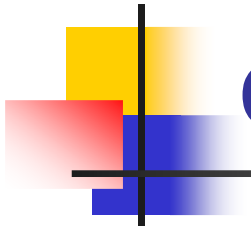
How Big Is The Context?

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





(C K H A B E J L N O D P M F G)



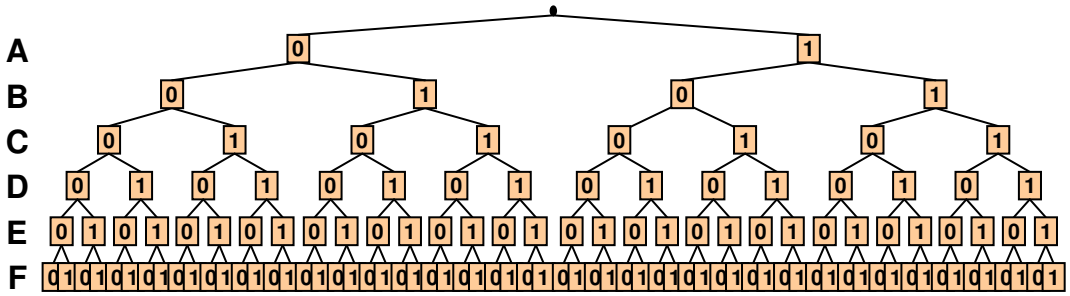
Complexity of AND/OR Graph Search

	AND/OR graph	OR graph
Space	$O(n\ d^{w^*})$	$O(n\ d^{pw^*})$
Time	$O(n\ d^{w^*})$	$O(n\ d^{pw^*})$

d = domain size
n = number of variables
w*= treewidth
pw*= pathwidth

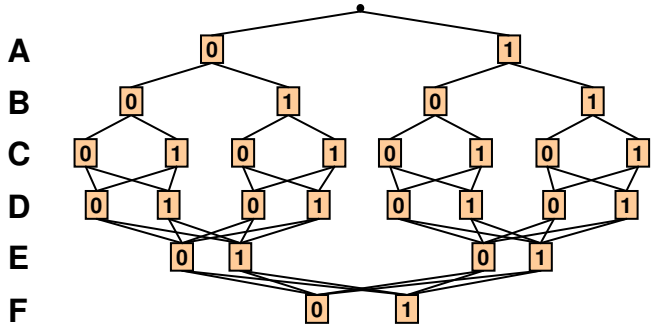
$w^* \leq pw^* \leq w^* \log n$

All Four Search Spaces



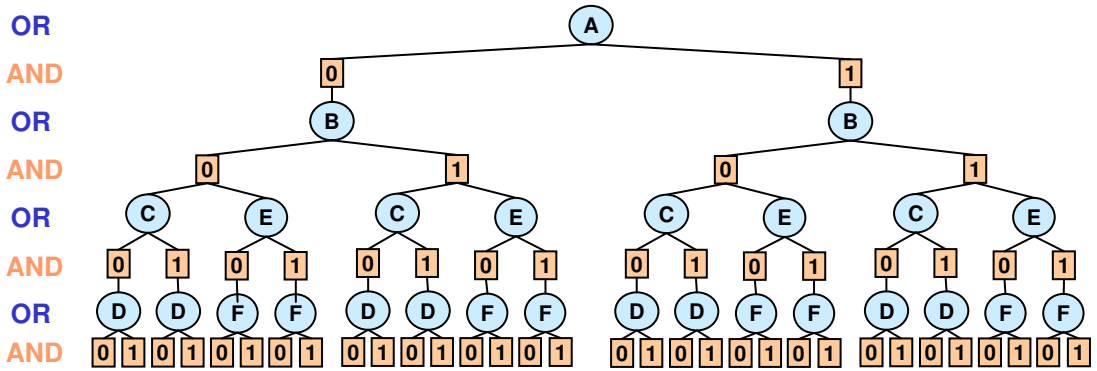
Full OR search tree

126 nodes



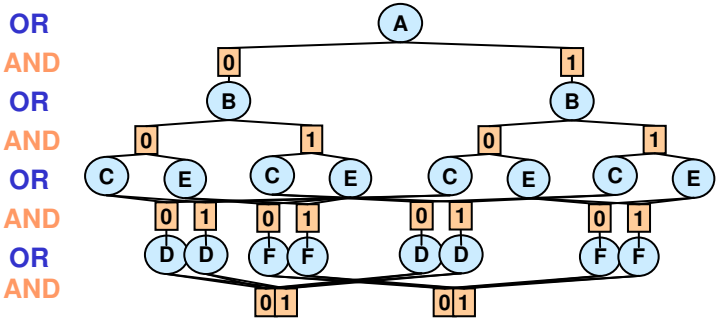
Context minimal OR search graph

28 nodes



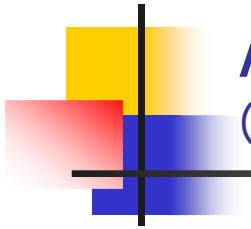
Full AND/OR search tree

54 AND nodes



Context minimal AND/OR search graph

18 AND nodes



AND/OR Branch-and-Bound with Caching

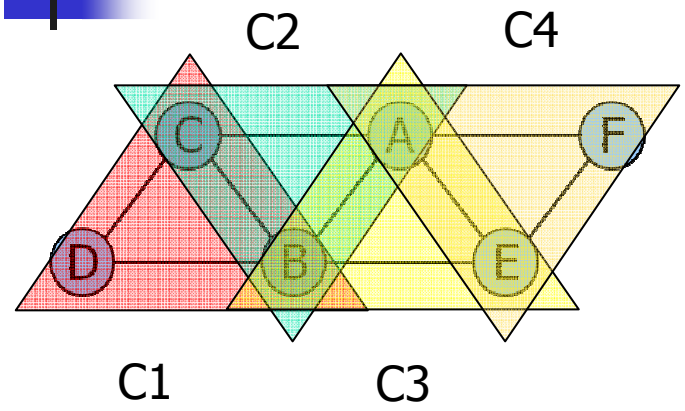
(Marinescu & Dechter, AAAI'06)

- Associate each node n with a heuristic lower bound h(n) on v(n)

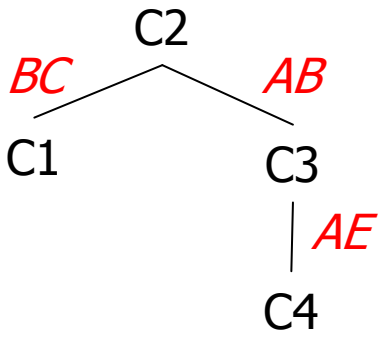
- EXPAND (top-down)
  - Evaluate f(T') and prune search if f(T') ≥ UB
  - If not in cache, expand the tip node n

- PROPAGATE (bottom-up)
  - Update value of the parent p of n
    - OR nodes: minimization
    - AND nodes: summation
  - Cache value of n, based on context

Backtrack with Tree Decomposition



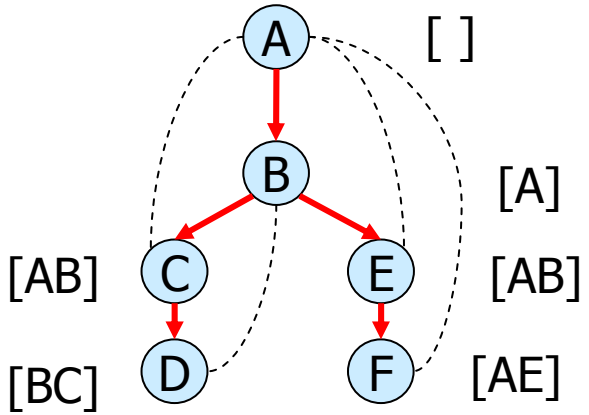
(Jegou & Terrioux, ECAI 2004)



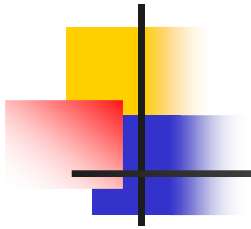
tree decomposition (w=2)

BTD:

- AND/OR graph search (caching on separators)
- Partial variable ordering (dynamic inside clusters)
- Maintaining local consistency



pseudo tree (w=2)



Backtrack with Tree Decomposition

- Before the search
  - Merge clusters with a separator size **> p**
  - Time O(k exp(w')), Space O(exp(p))
  - More freedom for variable ordering heuristics

- Properties
  - BTD(-1) is Depth-First Branch and Bound
  - BTD(0) solves connected components independently
  - BTD(1) exploits bi-connected components
  - BTD(s) is Backtrack with Tree Decomposition

s: largest separator size

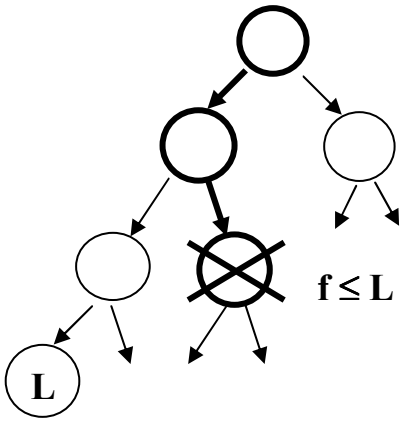
Basic Heuristic Search Schemes

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. DF Branch-and-Bound

Use heuristic function $f(x^p)$ to prune the depth-first search tree

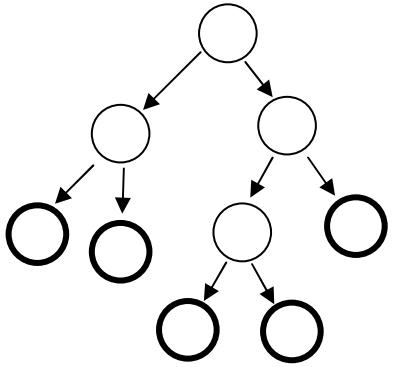
Linear space

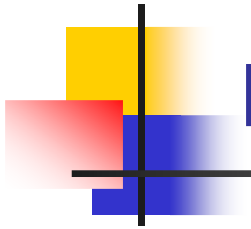


2. Best-First Search

Always expand the node with the highest heuristic value $f(x^p)$

Needs lots of memory





Best-First Principle

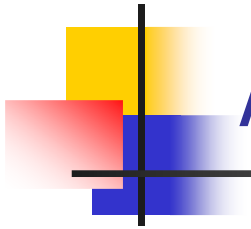
- Best-first search expands first the node with the best heuristic evaluation function among all node encountered so far
- It **never** expands nodes whose cost is beyond the optimal one, unlike depth-first search algorithms (Dechter & Pearl, 1985)
- Superior among memory intensive algorithms employing the **same heuristic function**

Best-First AND/OR Search (AOBF)

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

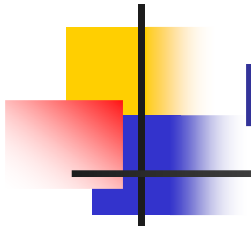
- **Maintains the set of best partial solution trees**
- **Top-down Step (EXPAND)**
  - Traces down marked connectors from root
    - i.e., best partial solution tree
  - Expands a tip node **n** by generating its successors **n'**
  - Associate each successor with heuristic estimate **h(n')**
    - 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)



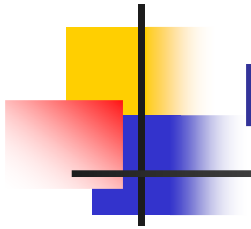
AOBF versus AOBB

- **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



Lower Bounding 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

- Benchmarks
  - Belief Networks (BN)
  - Weighted CSPs (WCSP)
- Algorithms
  - AOBB-C – AND/OR Branch-and-Bound w/ caching
  - AOBF-C – Best-first AND/OR Search
  - SamIam
  - Superlink
  - Toolbar (DFBB+EDAC), Toolbar-BTD (BTD+EDAC)
- Heuristics
  - Mini-Bucket heuristics

Genetic Linkage Analysis

(Fishelson & Geiger02)

pedigree (w*, h) (n, d)	**SamIam Superlink**	**MBE(i) BB-C+SMB(i) AOBB+SMB(i) AOBB-C+SMB(i) AOBF-C+SMB(i)** i=12		**MBE(i) BB-C+SMB(i) AOBB+SMB(i) AOBB-C+SMB(i) AOBF-C+SMB(i)** i=14		**MBE(i) BB-C+SMB(i) AOBB+SMB(i) AOBB-C+SMB(i) AOBF-C+SMB(i)** i=16		**MBE(i) BB-C+SMB(i) AOBB+SMB(i) AOBB-C+SMB(i) AOBF-C+SMB(i)** i=18	
		time	nodes	time	nodes	time	nodes	time	nodes
		0.42		0.83		1.78		5.75	
ped30		-	-	-	-	-	-	-	-
(23, 118)	out	-	-	-	-	-	-	214.10	1,379,131
(1016, 5)	13095.83	10212.70	93,233,570	8858.22	82,552,957	-	-	34.19	193,436
		out		out		out		**30.39**	72,798
		0.58		2.31		7.84		33.44	
ped33		-	-	-	-	-	-	-	-
(37, 165)	out	2804.61	34,229,495	737.96	9,114,411	3896.98	50,072,988	159.50	1,647,488
(581, 5)	-	1426.99	11,349,475	307.39	2,504,020	1823.43	14,925,943	86.17	453,987
		out		140.61	407,387	out		**74.86**	134,068
		4.20		31.33		96.28		out	
ped42		-	-	-	-	-	-		
(25, 76)	out	-	-	-	-	-	-		
(448, 5)	561.31	-	-	-	-	2364.67	22,595,247		
		out		out		**133.19**	93,831		

Min-fill pseudo tree. Time limit 3 hours.

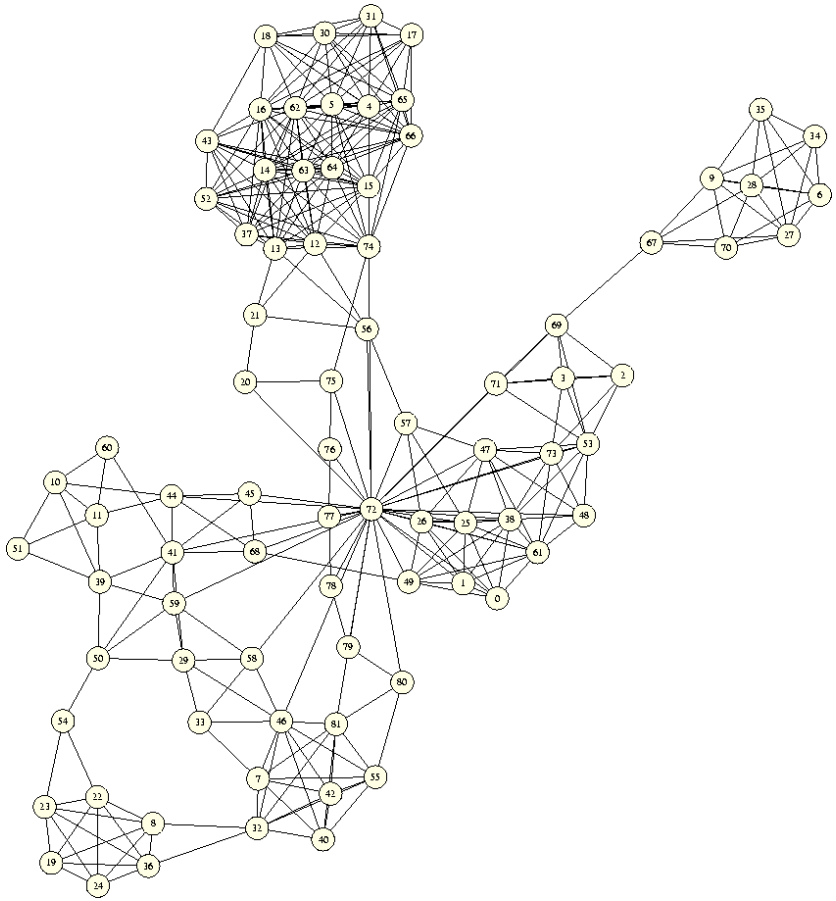
Mastermind Games

mastermind (w*, h) (n, r, k)	MBE(i) BB-C+SMB(i) AOBB+SMB(i) AOBB-C+SMB(i) AOBF-C+SMB(i) i=12		MBE(i) BB-C+SMB(i) AOBB+SMB(i) AOBB-C+SMB(i) AOBF-C+SMB(i) i=14		MBE(i) BB-C+SMB(i) AOBB+SMB(i) AOBB-C+SMB(i) AOBF-C+SMB(i) i=16		MBE(i) BB-C+SMB(i) AOBB+SMB(i) AOBB-C+SMB(i) AOBF-C+SMB(i) i=18	
	time	nodes	time	nodes	time	nodes	time	nodes
mm-04-08-04 (39, 103) (2616, 3, 2)	1.36 - 494.50 114.02 38.55	 - 744,993 82,070 33,069	2.08 - 270.60 66.84 29.19	 - 447,464 61,328 26,729	4.86 - 506.74 93.50 44.95	 - 798,507 79,555 38,989	16.53 - 80.86 30.80 **20.64**	 - 107,463 13,924 3,957
mm-03-08-05 (41, 111) (3692, 3, 2)	2.34 - - - out	 - - - 	8.52 - - - out	 - - - 	8.31 - - - 473.07	 - - - 199,725	24.94 - 1084.48 117.39 **36.99**	 - 1,122,008 55,033 8,297
mm-10-08-03 (51, 132) (2606, 3, 2)	1.64 - 161.35 19.86 **4.80**	 - 290,594 14,518 3,705	3.09 - 99.09 19.47 8.16	 - 326,662 14,739 4,501	7.55 - 89.06 22.34 11.17	 - 151,128 13,557 3,622	21.08 - 84.16 29.80 24.67	 - 127,130 9,388 3,619

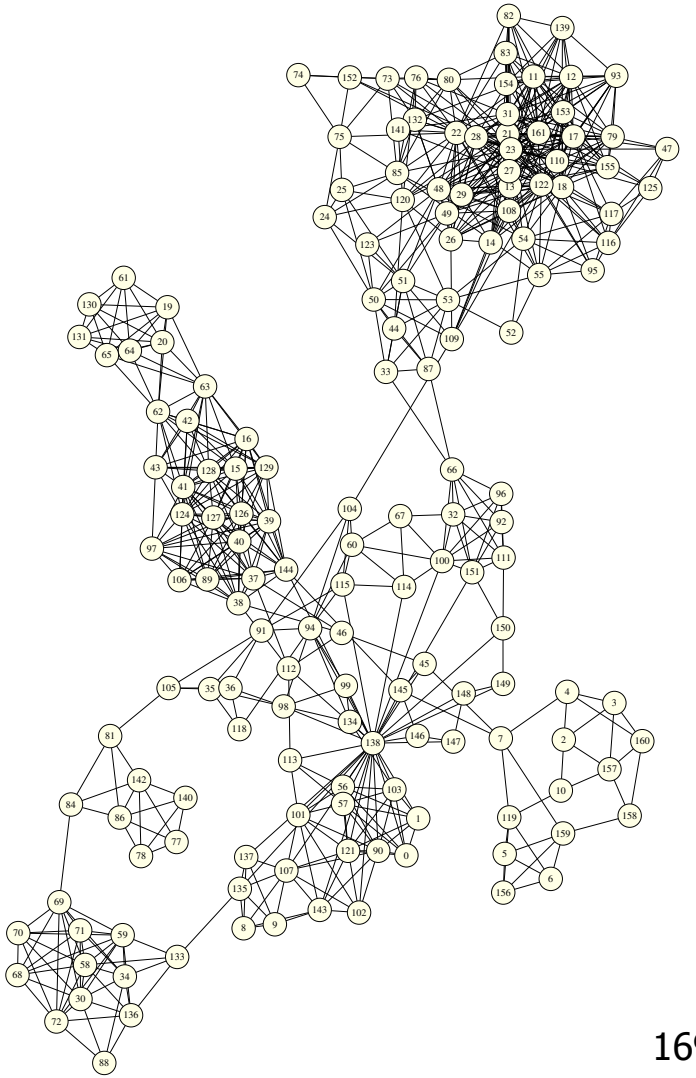
Min-fill pseudo trees. Time limit 1 hour.

toolbar, toolbar-BTD were not able to solve any instance.

- CELAR SCEN-06
  n=100, d=44,
  m=350, optimum=3389



- CELAR SCEN-07r
  n=162, d=44,
  m=764, optimum=343592



toulbar2 v0.8 running on a 2.6 GHz computer with 32 GB

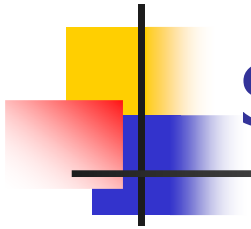
CELAR

(Sanchez et al., IJCAI 2009)

- Maximum Cardinality Search tree decomposition heuristic
- Root selection: largest (SCEN-06) / most costly (SCEN-07) cluster
- Last-conflict variable ordering and dichotomic branching

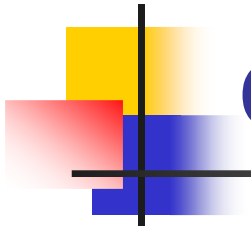
- Closed 1 open problem by exploiting tree decompostion and EDAC

CELAR	n	d	m	k	p	w	DFBB	BTD	RDS-BTD
SCEN-06	100	44	350	∞	∞	11	2588 sec.	221 sec.	316 sec.
SCEN-07r	162	44	764	354008	3	53	- > 50days	6 days	4.5 days



Summary

- New memory intensive AND/OR search algorithms for optimization in graphical models
- Depth-first and best-first control strategies
- Superior to state-of-the-art OR and AND/OR Branch-and-Bound tree search algorithms



Outline

- Introduction
- Inference
- Search (OR)
- Lower-bounds and relaxations

- **Exploiting problem structure in search**
  - AND/OR search spaces (tree, graph)
  - Searching the AND/OR space
  - AND/OR search for 0-1 integer programming

- Software

0-1 Integer Linear Programming

minimize : $z = c_1x_1 + c_2x_2 + \ldots + c_nx_n$

subject to :

$$a_1^1x_1 + a_2^1x_2 + \ldots + a_n^1x_n \le b^1$$
$$a_1^2x_1 + a_2^2x_2 + \ldots + a_n^2x_n \le b^2$$
$$\cdots$$
$$a_1^mx_1 + a_2^mx_2 + \ldots + a_n^mx_n \le b^m$$
$$x_1, x_2, \ldots, x_n \in \{0,1\}$$

:

- VLSI circuit design
- Scheduling
- Routing
- Combinatorial auctions
- Facility location
- …

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

subject to :

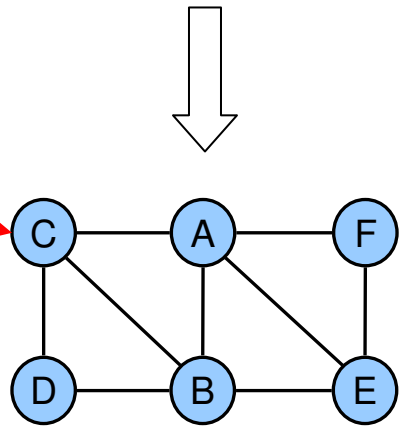
$3A - 12B + C \le 3$

$-2B + 5C - 3D \le -2$

$2A + B - 4E \le 2$

$A - 3E + F \le 1$

$A, B, C, D, E, F \in \{0,1\}$



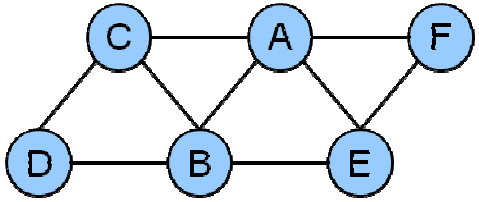
primal graph

AND/OR Search Tree

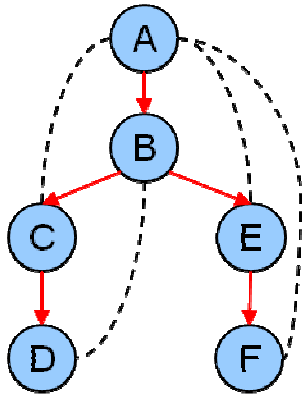
minimize : $z = 7A + 3B - 2C + 5D - 6E + 8F$

subject to :

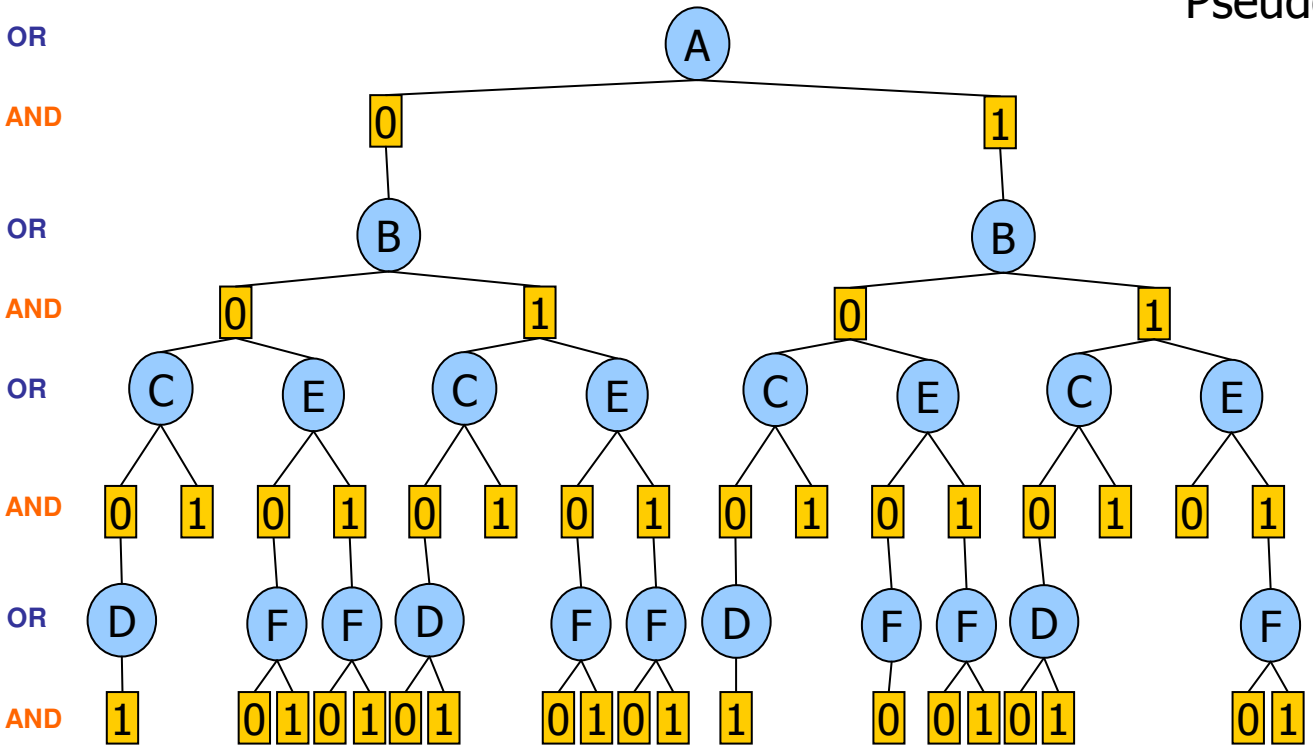
$$3A - 12B + C \le 3$$
$$-2B + 5C - 3D \le -2$$
$$2A + B - 4E \le 2$$
$$A - 3E + F \le 1$$
$$A, B, C, D, E, F \in \{0,1\}$$



Primal graph



Pseudo tree

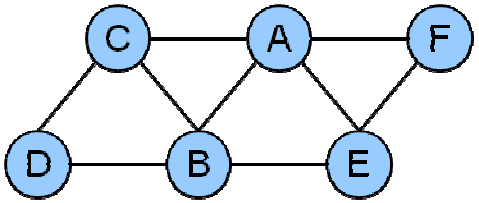


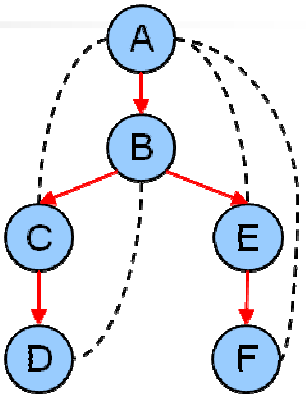
Weighted AND/OR Search Tree

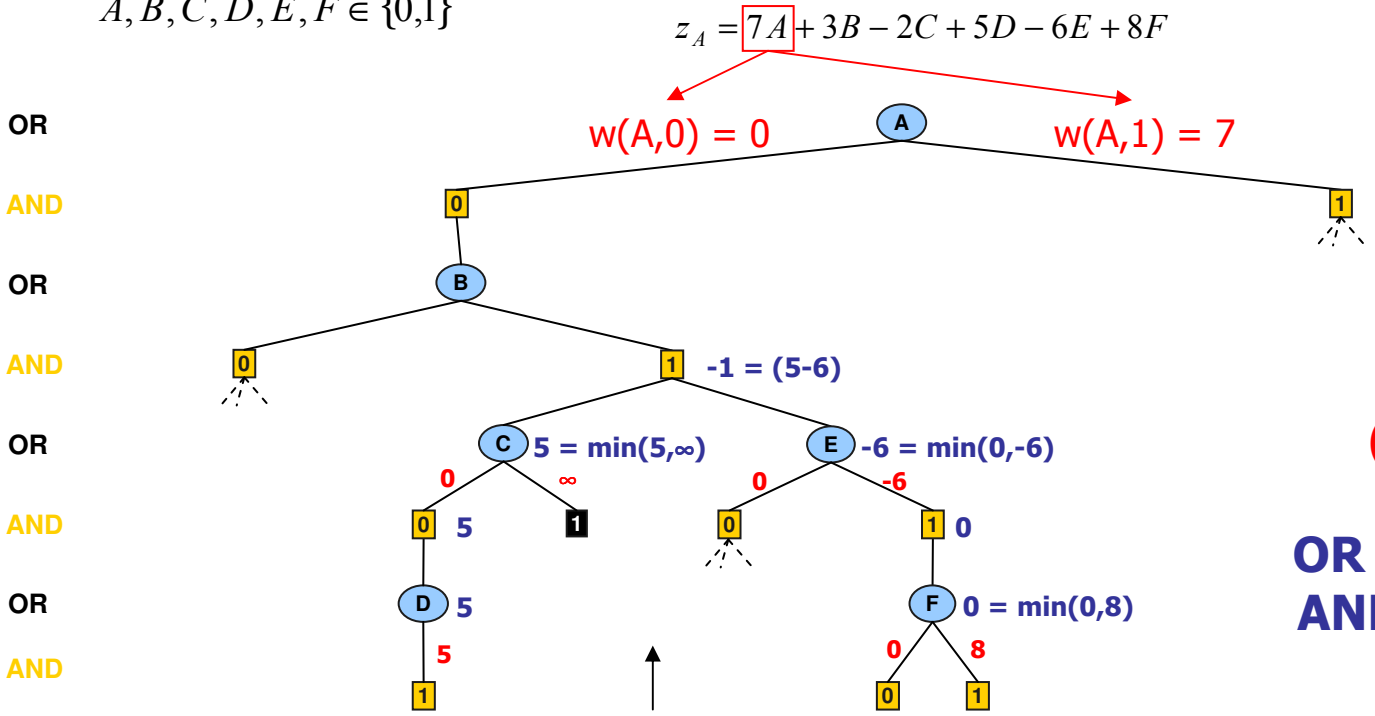
minimize : $z = 7A + 3B - 2C + 5D - 6E + 8F$

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)

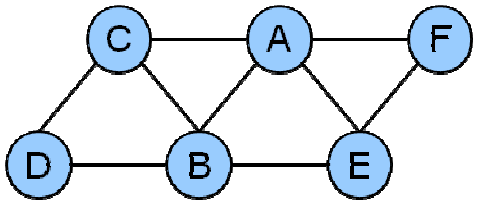
OR – minimization
AND – summation

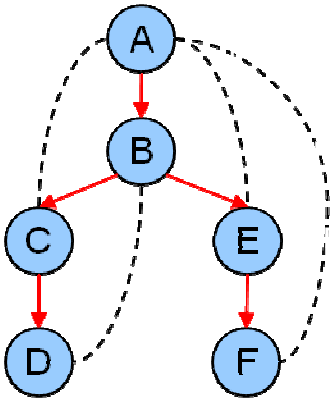
AND/OR Search Graph

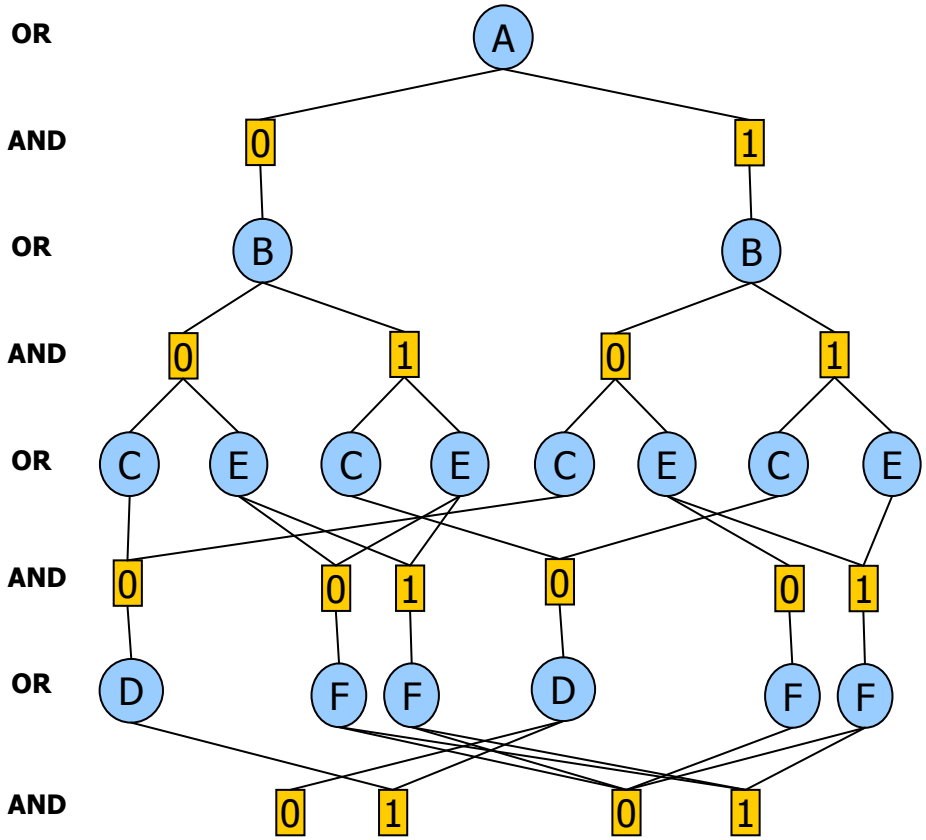
minimize : $z = 7A + 3B - 2C + 5D - 6E + 8F$

subject to :

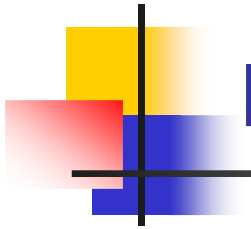
$$3A - 12B + C \le 3$$
$$-2B + 5C - 3D \le -2$$
$$2A + B - 4E \le 2$$
$$A - 3E + F \le 1$$
$$A, B, C, D, E, F \in \{0,1\}$$







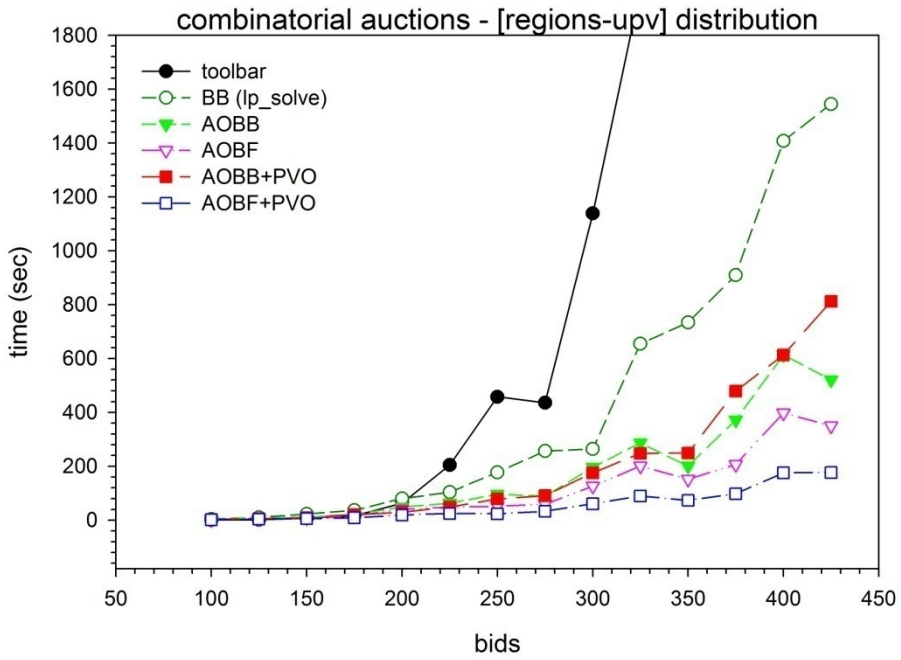
16 nodes (graph) vs. 54 nodes (tree)

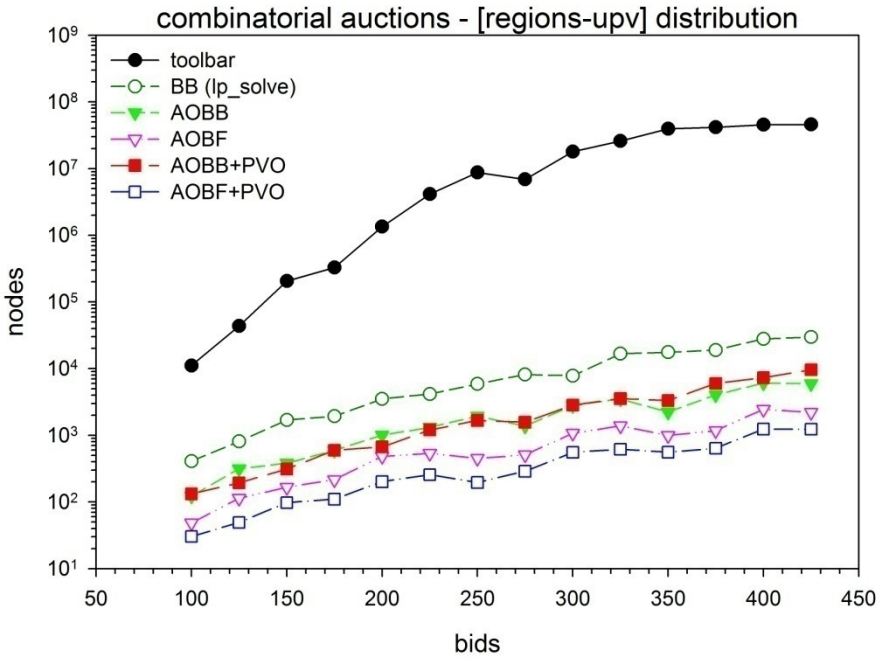


Experiments

- Algorithms
  - AOBB, AOBF – tree search
  - AOBB+PVO, AOBF+PVO – tree search
  - AOBB-C, AOBF-C – graph search
  - lp_solve 5.5, CPLEX 11.0, toolbar (DFBB+EDAC)
- Benchmarks
  - Combinatorial auctions
  - MAX-SAT instances
- Implementation
  - LP relaxation solved by lp_solve 5.5 library
  - BB (lp_solve) baseline solver

Combinatorial Auctions

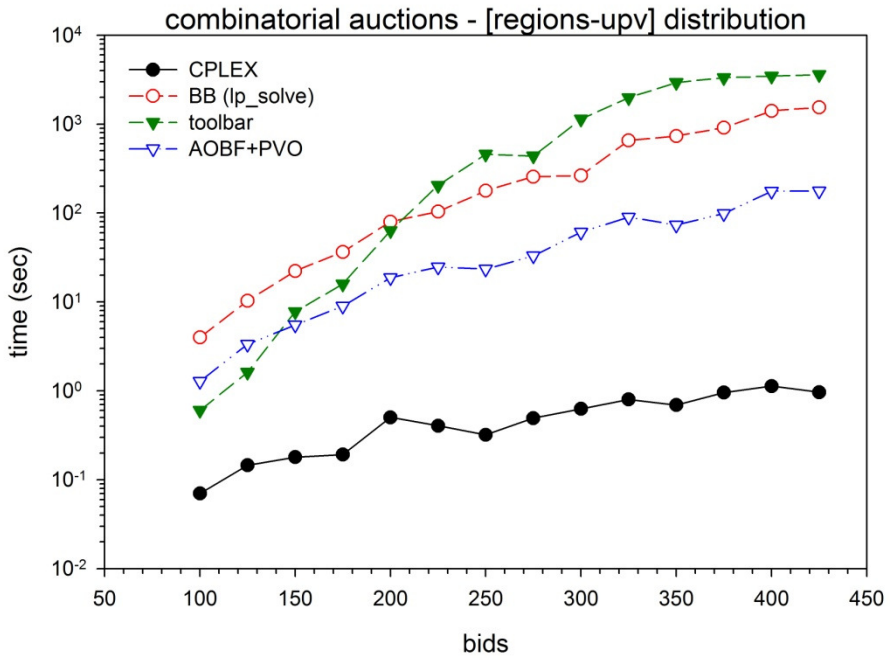


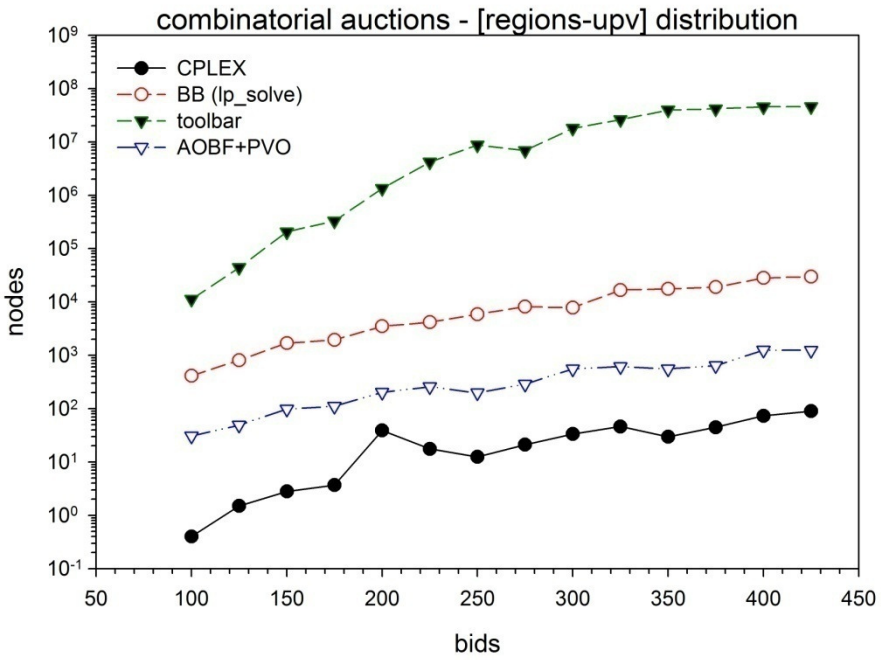


Combinatorial auctions from **`regions-upv`** distribution with 100 goods and increasing number of bids. Time limit 1 hour.

Very large treewidth $\in$ [68, 184]

Combinatorial Auctions





Combinatorial auctions from **`regions-upv`** distribution with 100 goods and increasing number of bids. Time limit 1 hour.

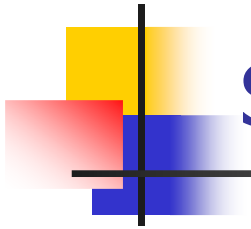
Very large treewidth $\in$ [68, 184]

MAX-SAT Instances (pret)

pret (w*, h)	BB CPLEX		AOBB AOBF		AOBB+PVO AOBF+PVO		AOBB-C AOBF-C	
	Tree search		Tree search		Tree search		Graph search	
	time	nodes	time	nodes	time	nodes	time	nodes
pret60-40 (6, 13)	- 676.94	- 3,926,422	7.88 7.56	1,255 1,202	8.41 8.70	1,216 1,326	7.38 **3.58**	1,216 568
pret60-60 (6, 13)	- 535.05	- 2,963,435	8.56 8.08	1,259 1,184	8.70 8.31	1,247 1,206	7.30 **3.56**	1,140 538
pret60-75 (6, 13)	- 402.53	- 2,005,738	6.97 7.38	1,124 1,145	6.80 8.42	1,089 1,149	6.34 **3.08**	1,067 506
pret150-40 (6, 15)	- out	-	95.11 101.78	6,625 6,535	108.84 101.97	7,152 6,246	75.19 **19.70**	5,625 1,379
pret150-60 (6, 15)	- out	-	98.88 106.36	6,851 6,723	112.64 102.28	7,347 6,375	78.25 **19.75**	5,813 1,393
pret150-75 (6, 15)	- out	-	108.14 98.95	7,311 6,282	115.16 103.03	7,452 6,394	84.97 **20.95**	6,114 1,430

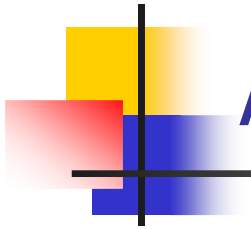
pret MAX-SAT instances. Time limit 10 hours.

BB solver could not solve any instance.



Summary

- New AND/OR search algorithms for 0-1 Integer Programming
- Dynamic variable orderings
- Superior to baseline OR Branch-and-Bound from the lp_solve library
- Outperform CPLEX on selected MAX-SAT instances (e.g., pret, dubois)



Algorithms for AND/OR Space

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

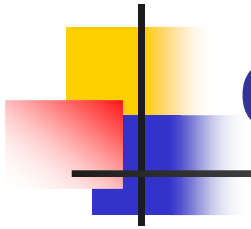
- **Pseudo-search re-arrangement**, for any CSP task
  (Freuder & Quinn 1985)

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

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

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

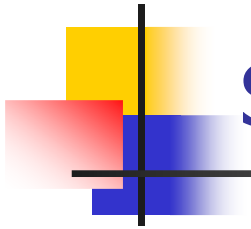
- **Value Elimination**
  (Bacchus, Dalmao & Pittasi, 2003)



Outline

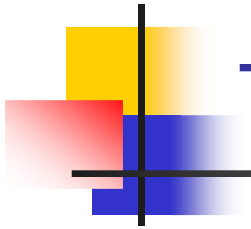
- Introduction
- Inference
- Search (OR)
- Lower-bounds and relaxations
- Exploiting problem structure in search

- **Software**
  - aolib and toulbar2 software packages
  - Results from UAI-06, CP-06 and UAI-08 solver competitions



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
  - CP'08 Competition
    - 534 2-ary MAX-CSP instances
    - 278 n-ary MAX-CSP instances
  - UAI'08 Competition
    - 480 MPE instances



Toulbar2 and aolib

- toulbar2

  http://mulcyber.toulouse.inra.fr/gf/project/toulbar2

  (Open source WCSP, MPE solver in C++)

- aolib

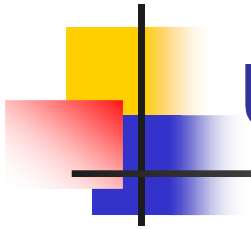
  http://graphmod.ics.uci.edu/group/Software

  (WCSP, MPE, ILP solver in C++, inference and counting)

- Large set of benchmarks

  http://carlit.toulouse.inra.fr/cgi-bin/awki.cgi/SoftCSP

  http://graphmod.ics.uci.edu/group/Repository

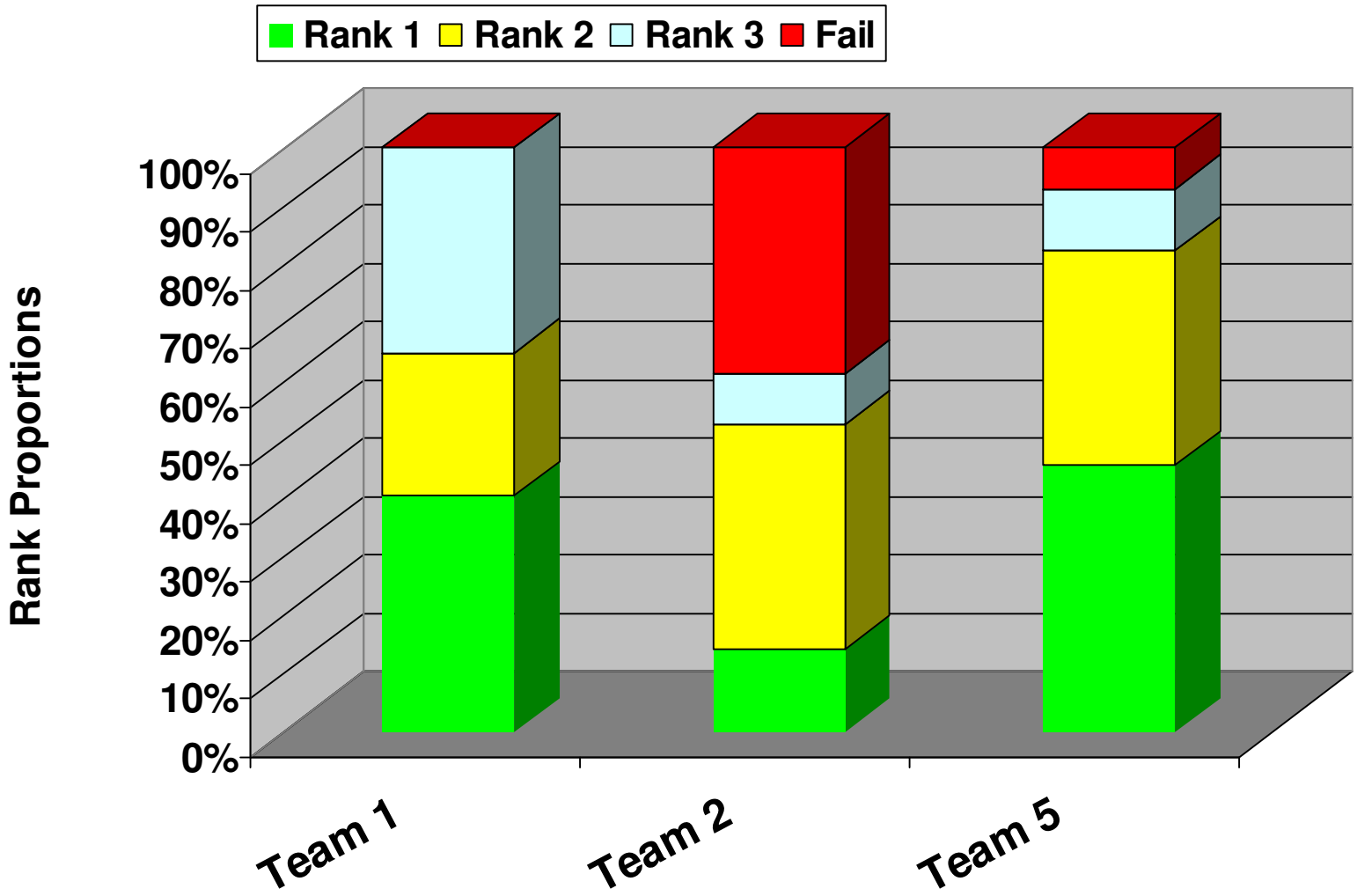


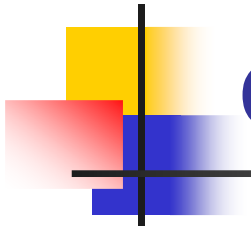
UAI'06 Competitors

- **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
    - Used **AOBB-C+SMB(i)** solver for MPE

UAI'06 Results

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



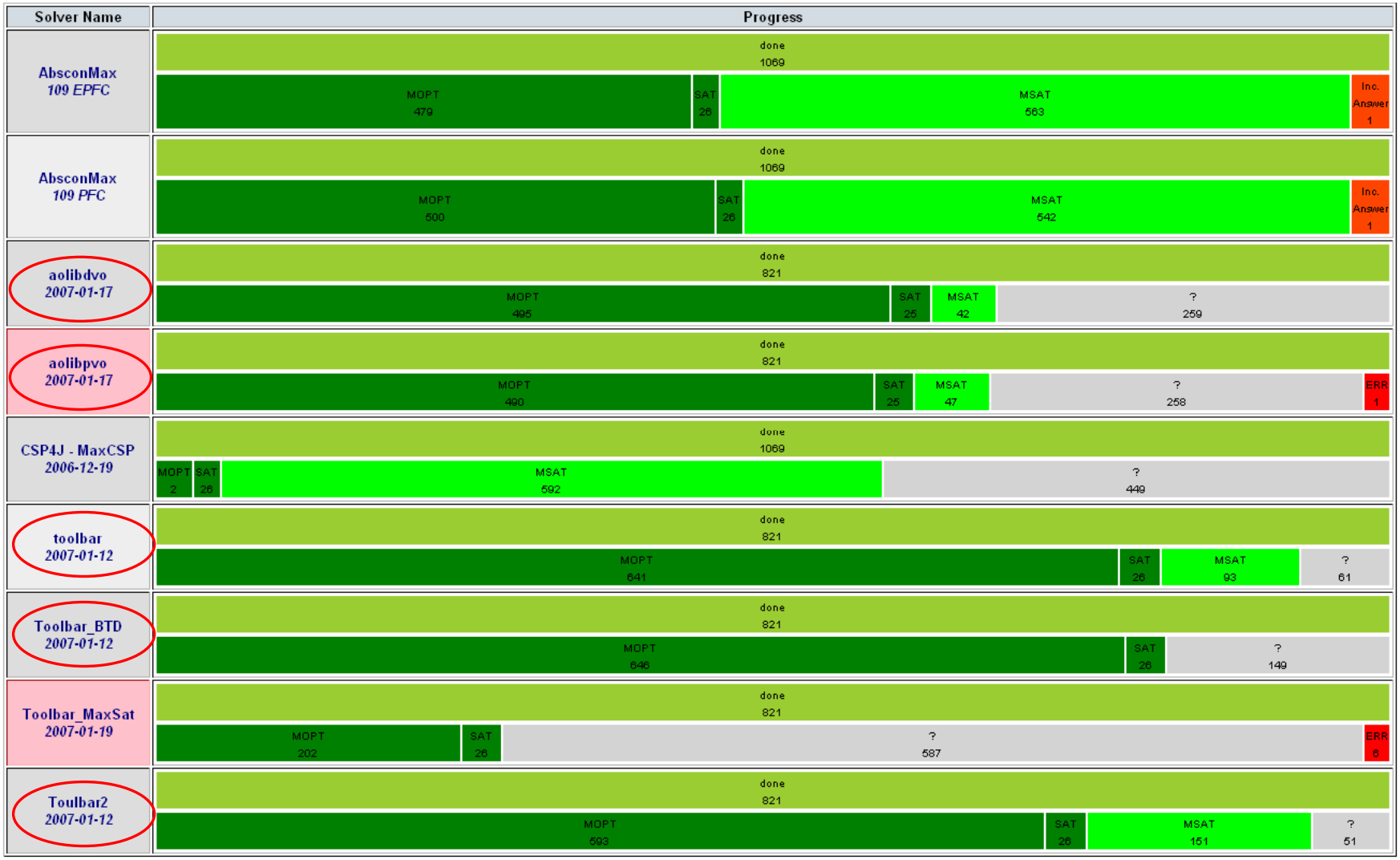


CP'06 Competitors

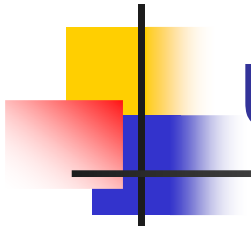
- Solvers
  - AbsconMax (ie, DFBB+MRDAC)
  - **aolibdvo** (ie, AOBB+EDAC+DVO solver)
  - **aolibpvo** (ie, AOBB+EDAC+PVO solver)
  - CSP4J-MaxCSP
  - **Toolbar** (ie, DFBB+EDAC)
  - **Toolbar_BTD** (ie, BTD+EDAC+VE)
  - **Toolbar_MaxSAT** (ie, DPLL+specific EPT rules)
  - **Toulbar2** (ie, DFBB+EDAC+VE+LDS)

CP'06 Results

Overall ranking on all selected competition benchmarks



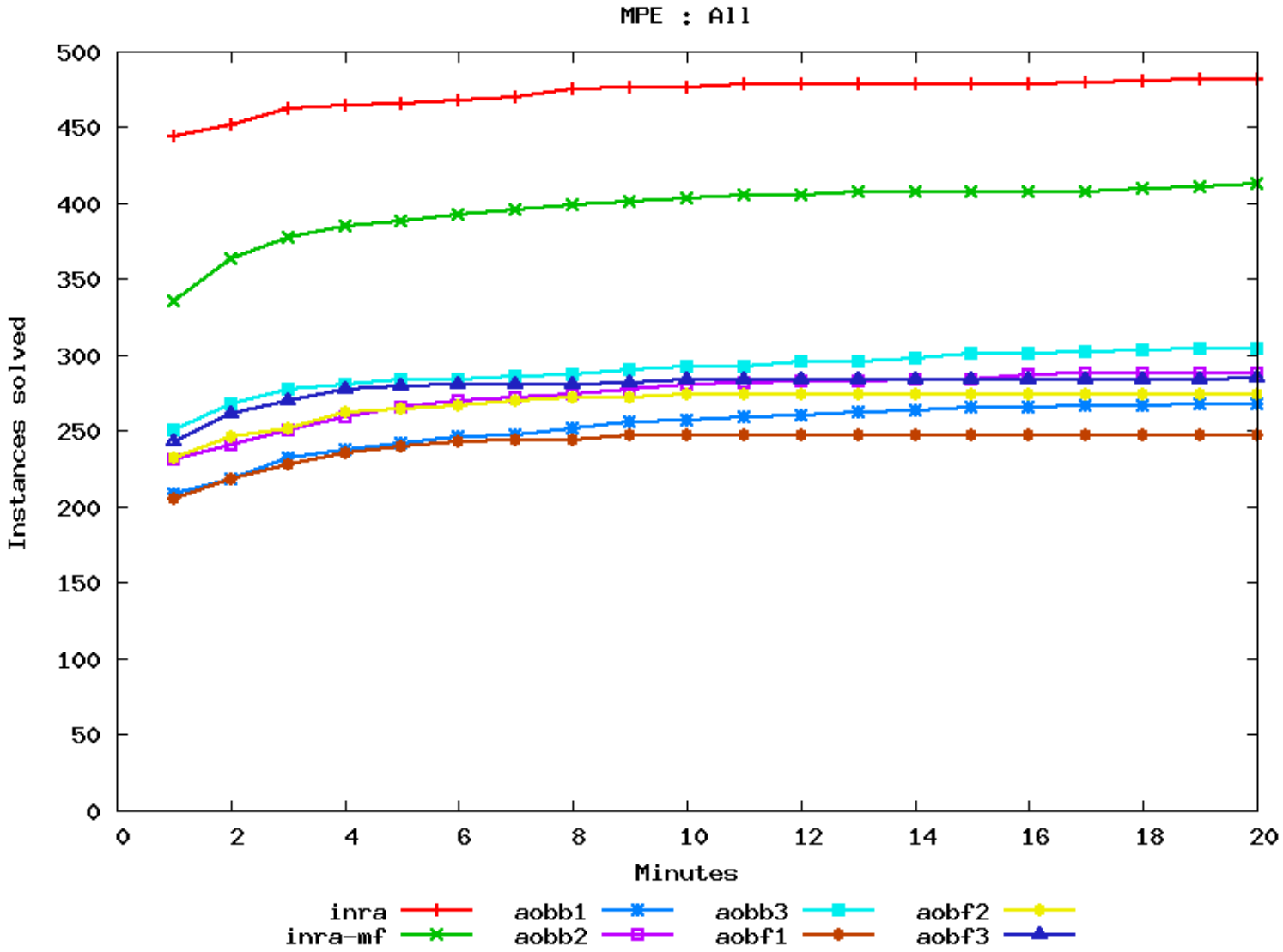
The longest dark green bar wins



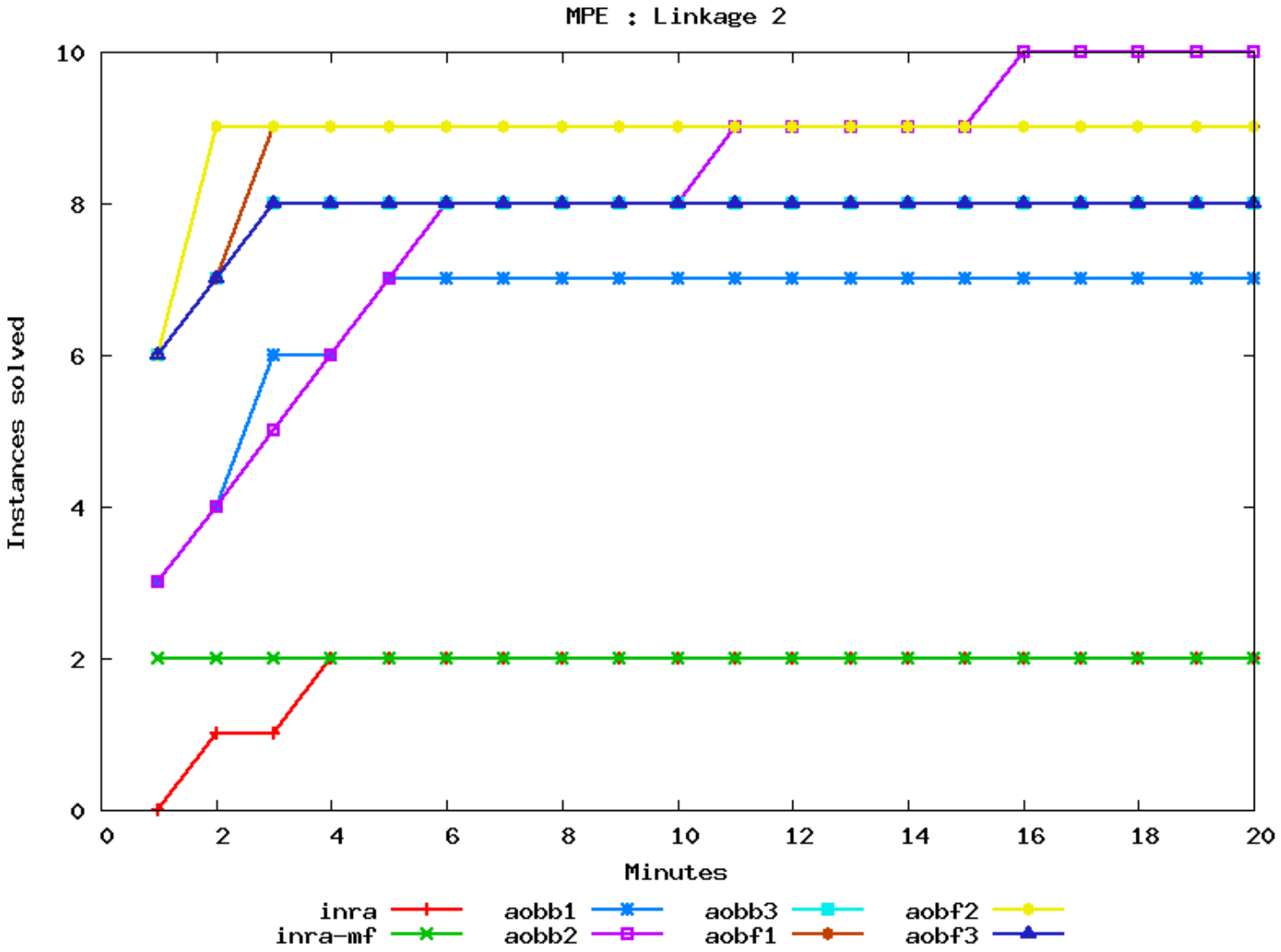
UAI'08 Competition

- **AOBB-C+SMB(i) – (i = 18, 20, 22)**
  - AND/OR Branch-and-Bound with pre-compiled mini-bucket heuristics (i-bound), full caching, static pseudo-trees, constraint propagation
- **AOBF-C+SMB(i) – (i = 18, 20, 22)**
  - AND/OR Best-First search with pre-compiled mini-bucket heuristics (i-bound), full caching, static pseudo-trees, no constraint propagation
- **Toulbar2**
  - OR Branch-and-Bound, dynamic variable/value orderings, EDAC consistency for binary and ternary cost functions, variable elimination of small degree (2) during search
- **Toulbar2/BTD**
  - DFBB exploiting a tree decomposition (AND/OR), same search inside clusters as toulbar2, full caching (no cluster merging), combines RDS and EDAC, and caching lower bounds

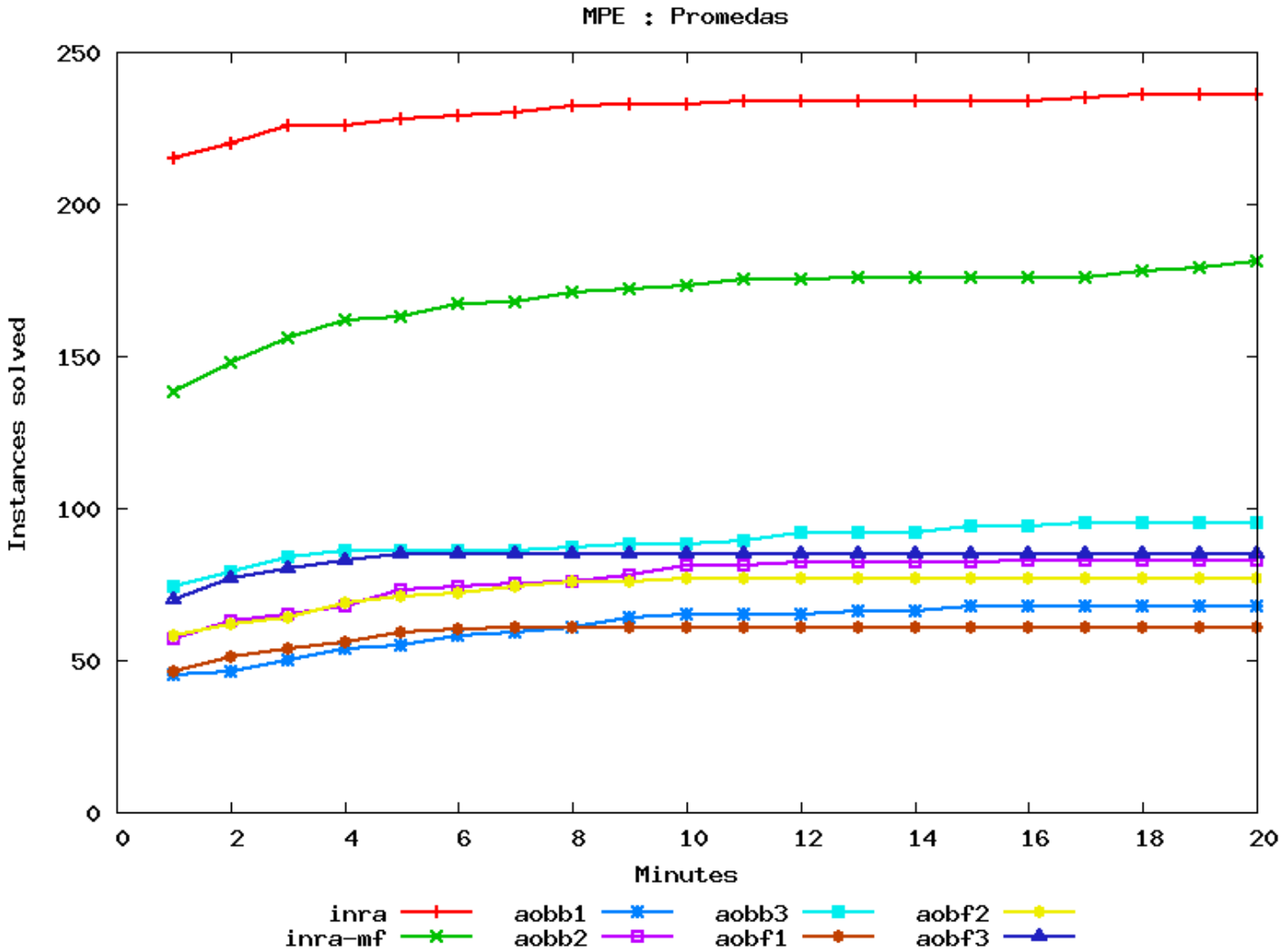
UAI'08 Competition Results

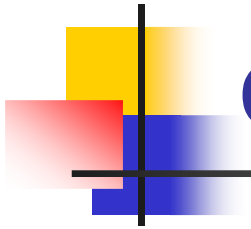


UAI'08 Competition Results (II)



UAI'08 Competition Results (III)

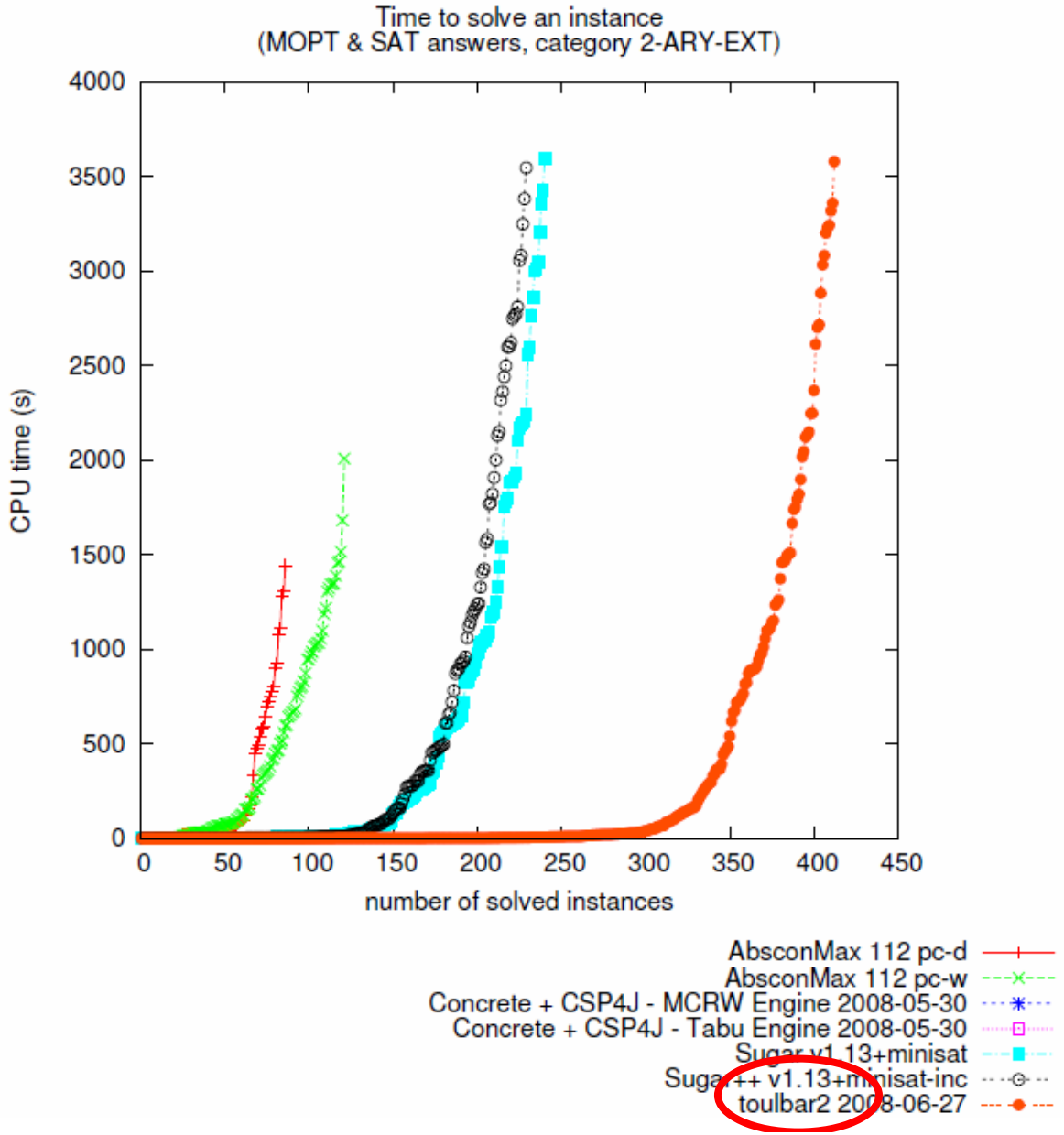




CP'08 Competitors

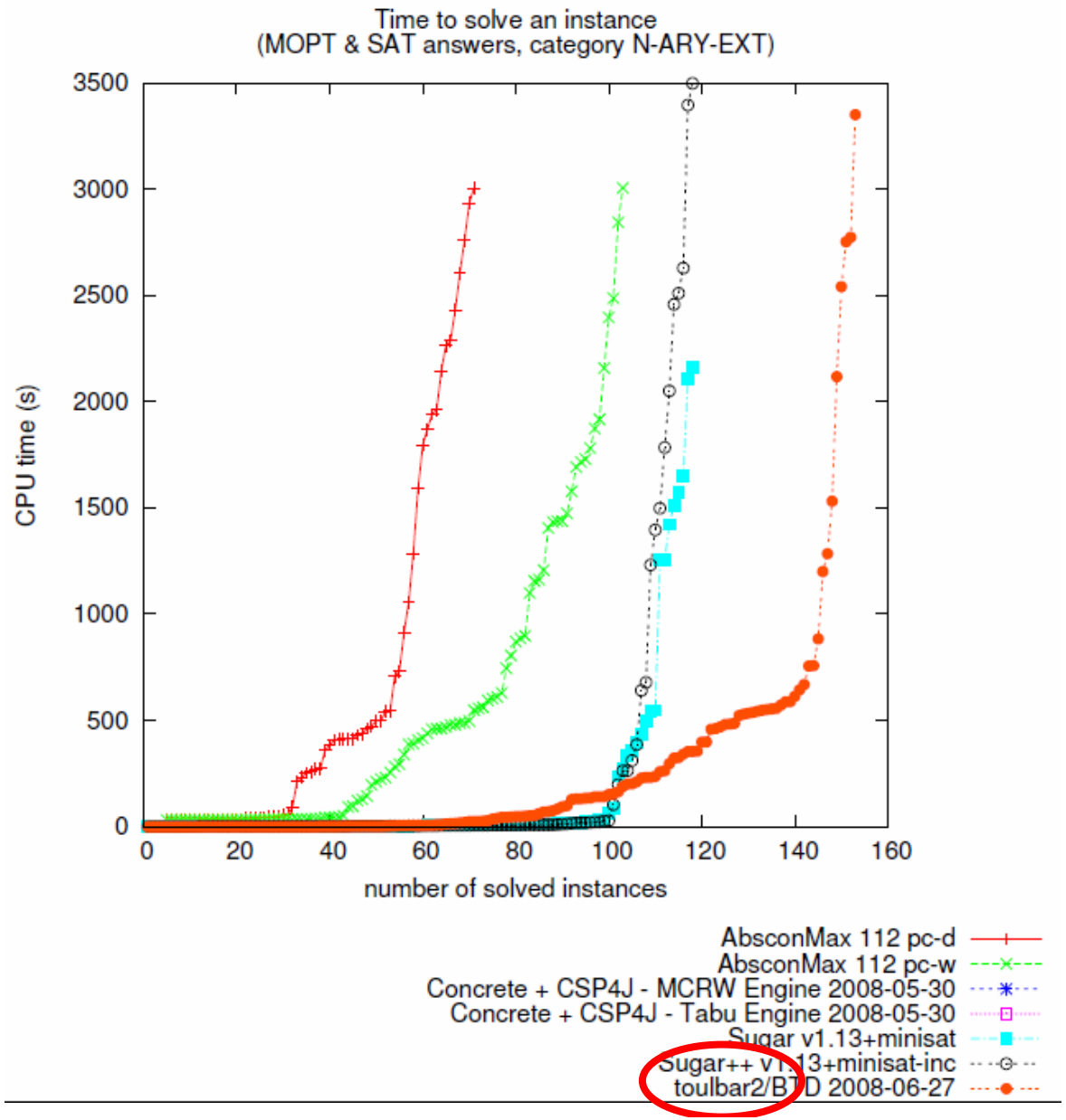
- Solvers
  - AbsconMax (ie, DFBB+MRDAC)
  - CSP4J-MaxCSP
  - Sugar (SAT-based solver)
  - **Toulbar2** (ie, BTD+EDAC+VE)

CP'08 Results

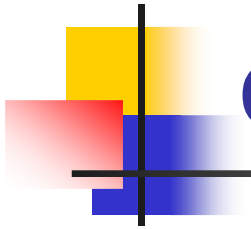


1

CP'08 Results



1



Conclusions

- **Only a few principles:**

1. Inference and search should be combined → time-space trade-off
2. AND/OR search should be used
3. Caching in search should be used