Chapter 13

Constraint Optimization
And counting, and enumeration
275 class

Outline

- **Introduction**
  - Optimization tasks for graphical models
  - Solving optimization problems with inference and search
- **Inference**
  - Bucket elimination, dynamic programming
  - Mini-bucket elimination
- **Search**
  - Branch and bound and best-first
  - Lower-bounding heuristics
  - AND/OR search spaces
- **Hybrids of search and inference**
  - Cutset decomposition
  - Super-bucket scheme

Constraint Satisfaction

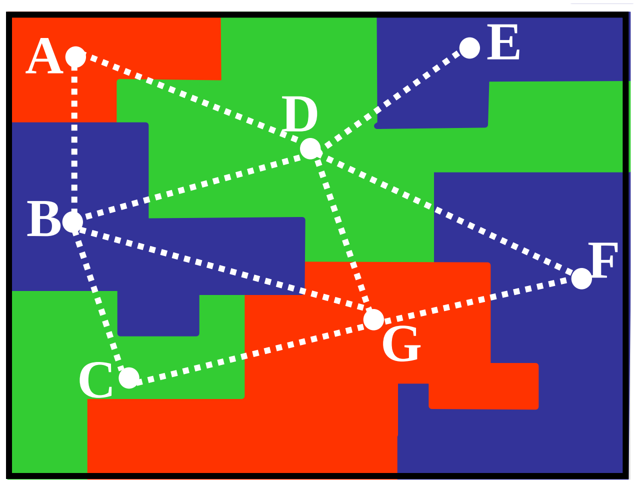
Example: map coloring

Variables - countries (A,B,C,etc.)

Values - colors (e.g., red, green, yellow)

Constraints: $A \neq B$, $A \neq D$, $D \neq E$, *etc.*

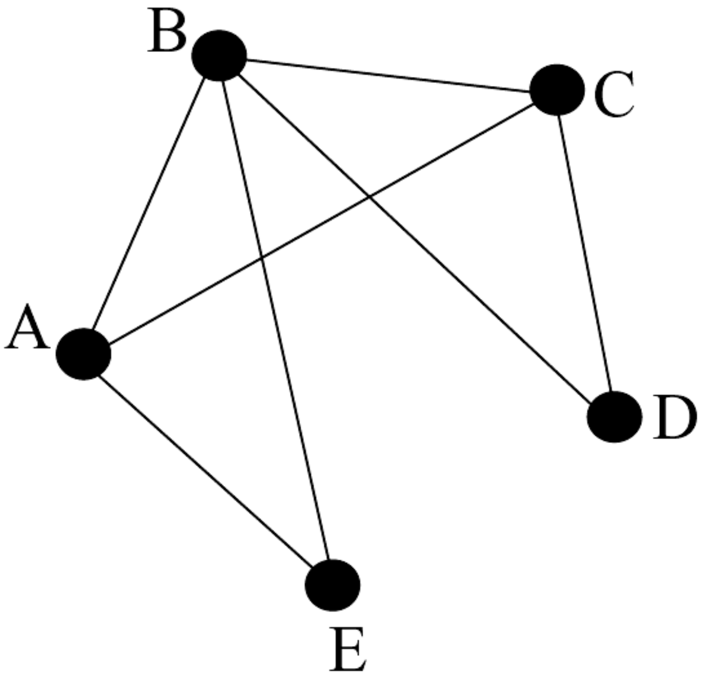
A	B
red	green
red	yellow
green	red
green	yellow
yellow	green
yellow	red



Task: consistency? Find a solution, all solutions, counting

Propositional Satisfiability

$\varphi = \{(\neg C), (A \vee B \vee C), (\neg A \vee B \vee E), (\neg B \vee C \vee D)\}$.



Constraint Optimization Problems
for Graphical Models

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

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

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

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

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

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

Global Cost Function

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

Constraint Optimization Problems
for Graphical Models

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

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

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

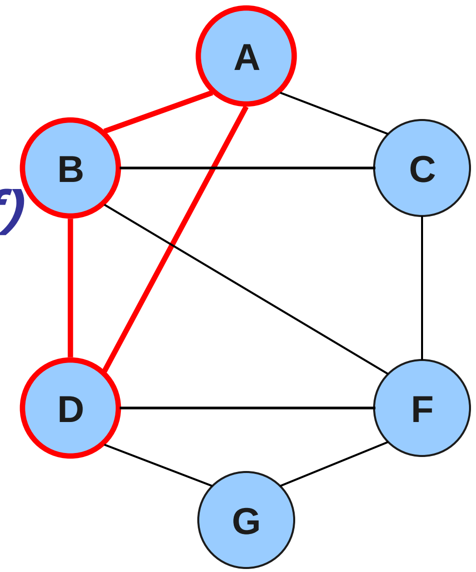
$F = \{f_1, ..., f_m\}$ - cost functions

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

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

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

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



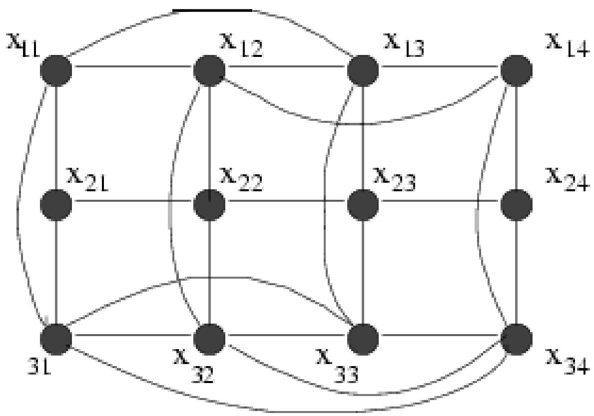
f1(A,B,D)
f2(D,F,G)
f3(B,C,F)

Global Cost Function

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

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 TotalFuelCost($X_1,..., X_N$)

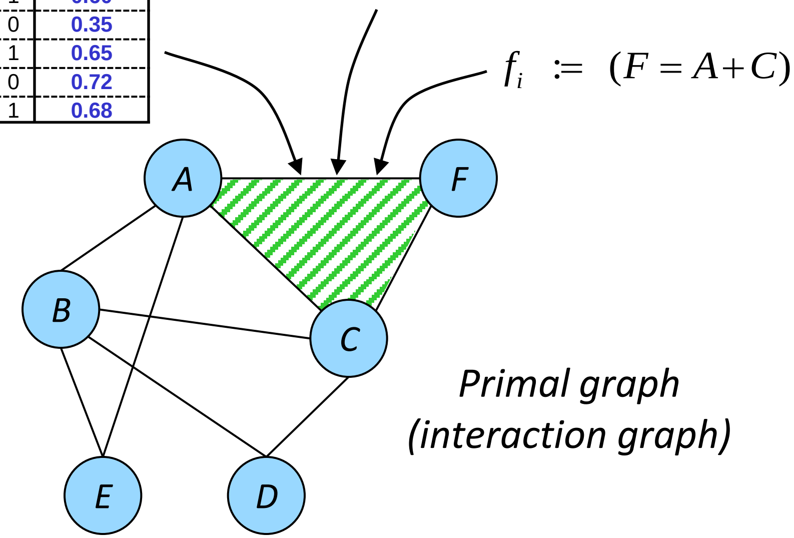
Graphical Models

- 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
- Operators:
  - combination
  - elimination (projection)
- Tasks:
  - **Belief updating**: $\Sigma_{X\text{-}y} \prod_j P_i$
  - **MPE**: $\max_X \prod_j P_j$
  - **CSP**: $\prod_X \times_j C_j$
  - **Max-CSP**: $\min_X \Sigma_j f_j$

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

Relation

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



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

Combination of Cost Functions

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

+

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

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

= 0 + 6

Elimination in a Cost Function

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

Min

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

A	g(A)
b	*1*
g	*2*
r	*1*

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

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

Conditioning 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

Assign($\boldsymbol{f}_{AB}$, *A*, *b*) →

g(*B*)

3

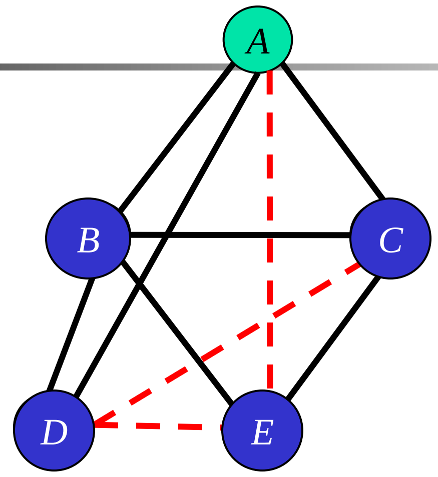
Assign($\boldsymbol{g}$, *B*, *r*) →

$h_{\varnothing}$

Outline

- **Introduction**
  - Optimization tasks for graphical models
  - Solving by inference and search
- **Inference**
  - **Bucket elimination, dynamic programming, tree-clustering, bucket-elimination**
  - Mini-bucket elimination, belief propagation
- **Search**
  - Branch and bound and best-first
  - Lower-bounding heuristics
  - AND/OR search spaces
- **Hybrids of search and inference**
  - Cutset decomposition
  - Super-bucket scheme

Computing the Optimal Cost Solution



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

Combination

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

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

Variable Elimination

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 and Briochi, 1973)

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

$\min_b \sum$ ← *Elimination operator*

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

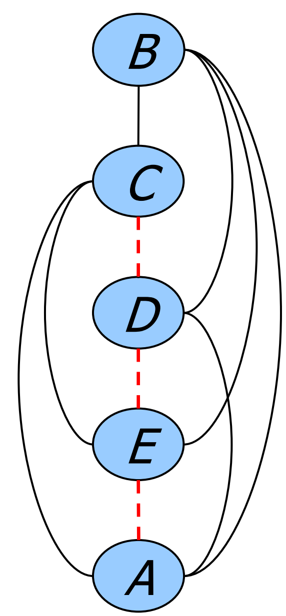
bucket C: $f(c,a)\ f(c,e)$ $h^B(a,d,c,e)$

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

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

bucket A: $h^E(a)$

OPT



Generating the Optimal Assignment

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

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: $f(c,a)\ f(c,e)$ $\quad h^B(a,d,c,e)$

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

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

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

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

Complexity

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

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

$\min_b \sum$ ← *Elimination operator*

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

bucket C: $f(c,a)\ f(c,e)$ $\quad h^B(a,d,c,e)$

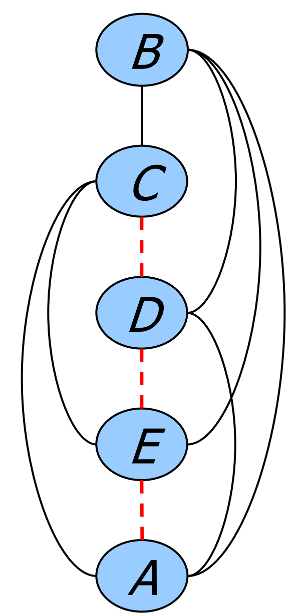
bucket D: $f(a,d)$ $\quad h^C(a,d,e)$

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

bucket A: $h^E(a)$

OPT

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



Complexity of bucket elimination

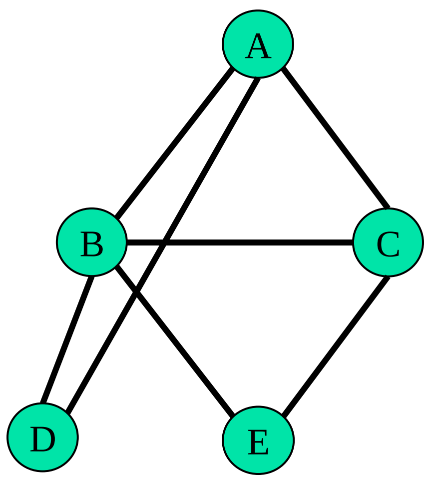
Bucket-elimination is **time** and **space**

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

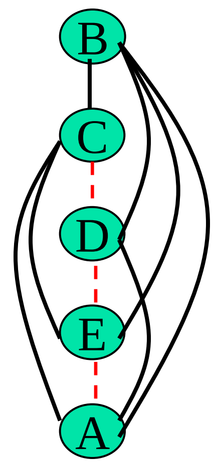
$w^*(d)$ – the induced width of the primal graph along ordering d

r = number of functions

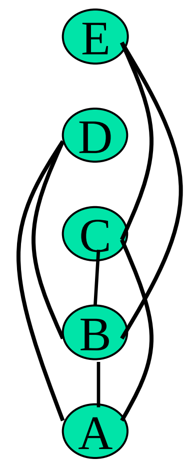
The effect of the ordering:



constraint graph



$w^*(d_1) = 4$



$w^*(d_2) = 2$

Finding smallest induced-width is hard

Complexity

*Algorithm **elim-opt** (Dechter, 1996)*

Non-serial Dynamic Programming (Bertele and Briochi, 1973)

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

$\min_b \sum$ ← *Elimination operator*

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

bucket C: $f(c,a)\ f(c,e)$ $\quad h^B(a,d,c,e)$

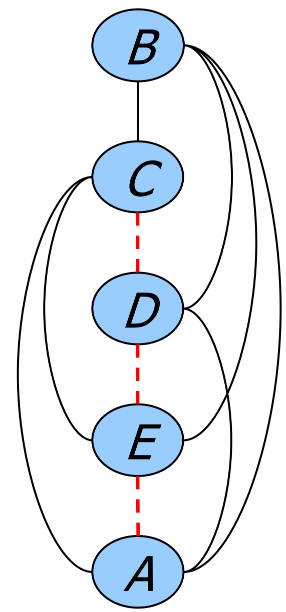
bucket D: $f(a,d)$ $\quad h^C(a,d,e)$

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

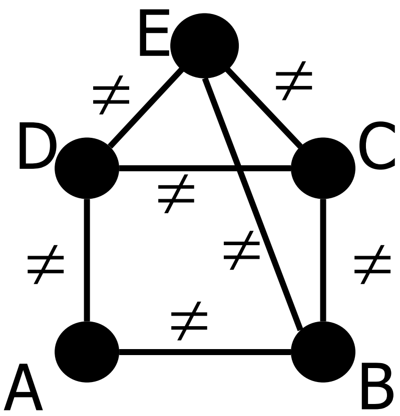
bucket A: $h^E(a)$

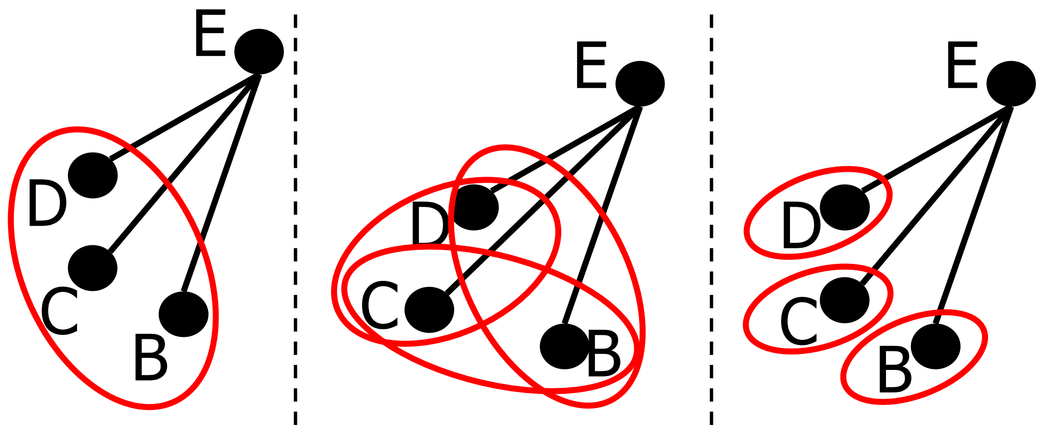
OPT

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



Directional i-consistency





Adaptive | d-path | d-arc

E : E ≠ D, E ≠ C, E ≠ B → R_{DCB}

D : D ≠ C, D ≠ A

C : C ≠ B

B : A ≠ B

A :

Adaptive: R_{DCB}

d-path: R_{DC}, R_{DB}, R_{CB}

d-arc: R_D, R_C, R_D

Mini-bucket approximation: MPE task

Split a bucket into mini-buckets =>bound complexity

$$\text{bucket}(X) = \{ h_1, \ldots, h_r, h_{r+1}, \ldots, h_n \}$$

$$h^X = \max_X \prod_{i=1}^{n} h_i$$

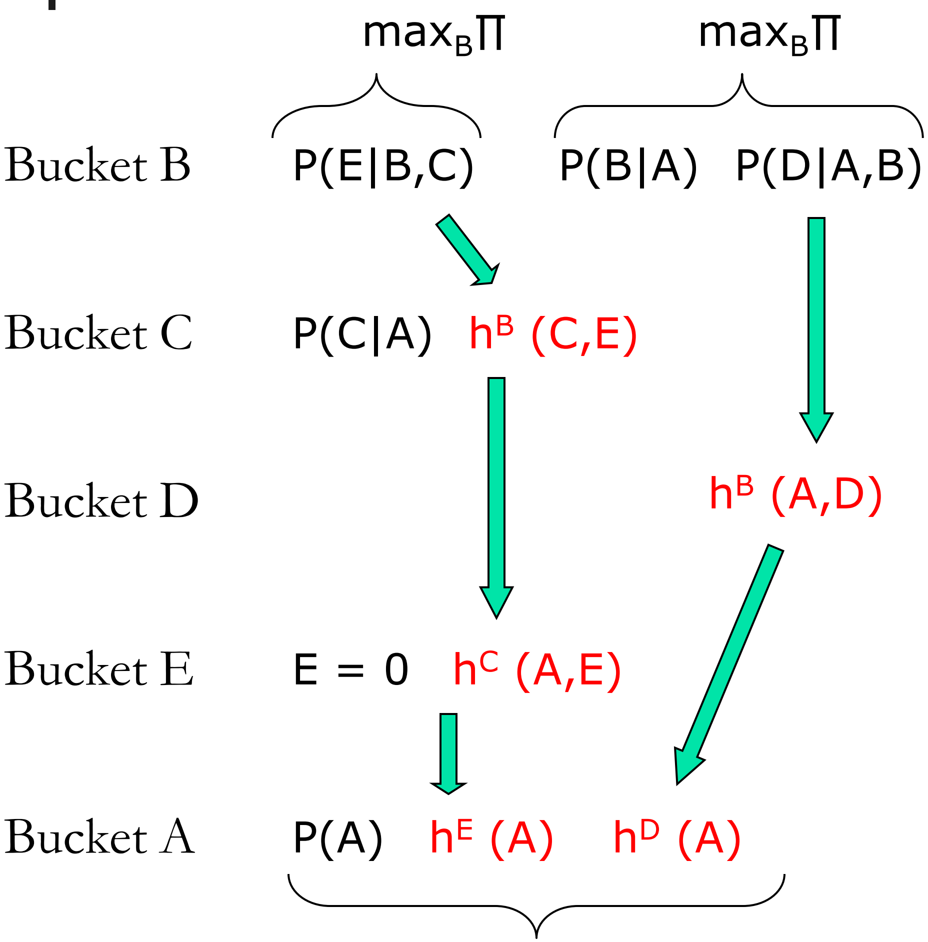
$$\{ h_1, \ldots, h_r \} \qquad \{ h_{r+1}, \ldots, h_n \}$$

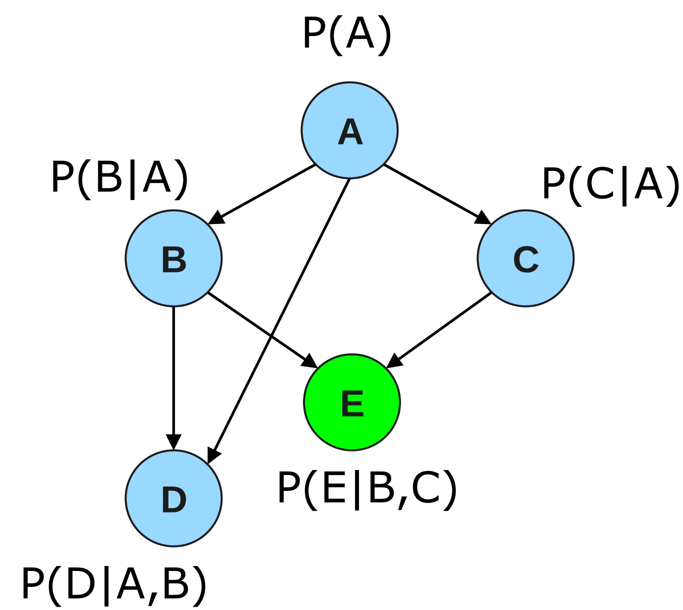
$$g^X = (\max_X \prod_{i=1}^{r} h_i) \cdot (\max_X \prod_{i=r+1}^{n} h_i)$$

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

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

Mini-Bucket Elimination





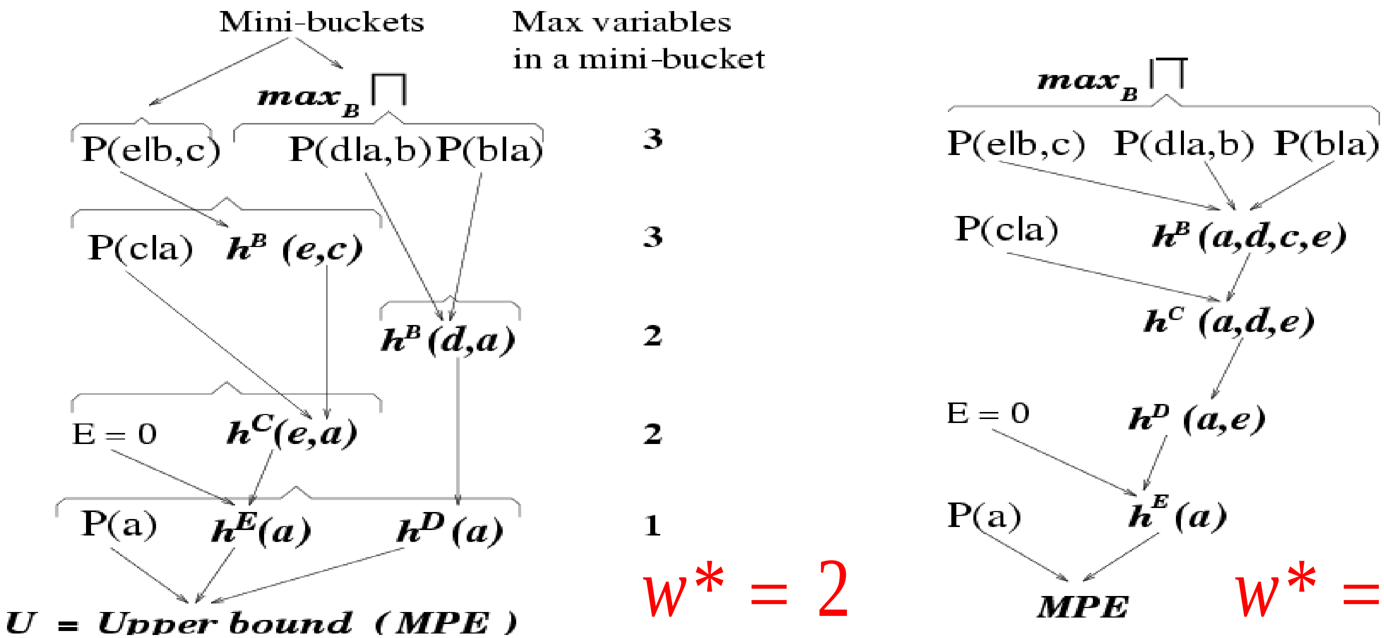
MPE* is an upper bound on MPE --U
Generating a solution yields a lower bound--L

MBE-MPE(i)

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

- Input: i – max number of variables allowed in a mini-bucket
- Output: [lower bound (cost 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.

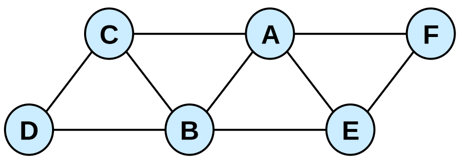
- Possible use of mini-bucket approximations:
  - As anytime algorithms
  - As heuristics in search

- Other tasks: similar mini-bucket approximations for: belief updating, MAP and MEU (Dechter and Rish, 1997)

Outline

- **Introduction**
  - Optimization tasks for graphical models
  - Solving by inference and search
- **Inference**
  - Bucket elimination, dynamic programming
  - Mini-bucket elimination
- **Search**
  - **Branch and bound and best-first**
  - **Lower-bounding heuristics**
  - AND/OR search spaces
- **Hybrids of search and inference**
  - Cutset decomposition
  - Super-bucket scheme

The Search Space



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

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

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

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

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

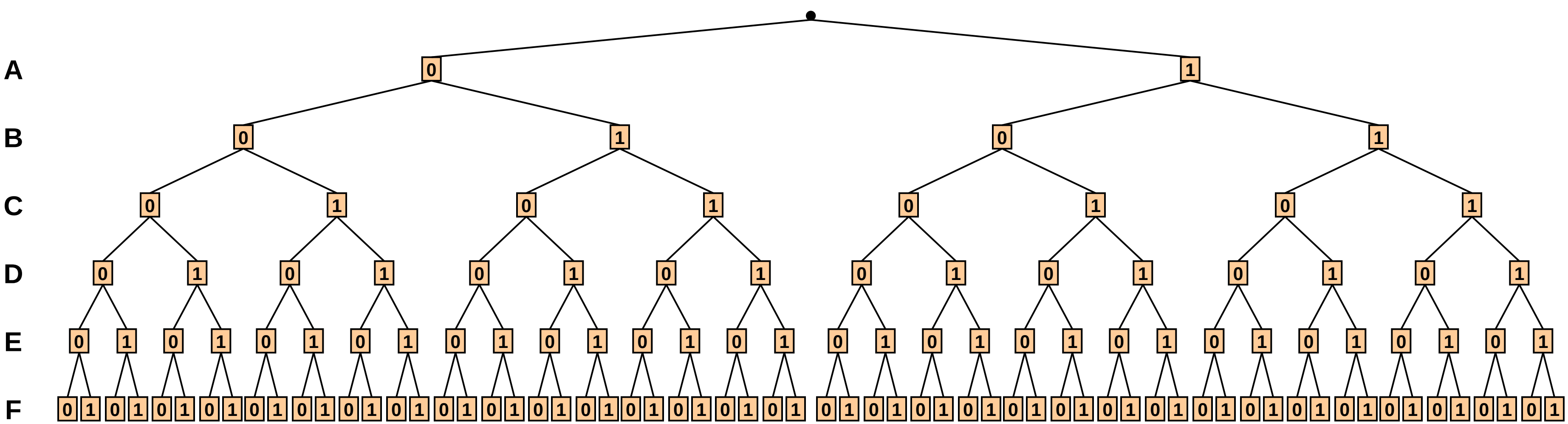
B	D	f_6
0	0	**4**
0	1	**2**
1	0	**1**
1	1	**0**

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

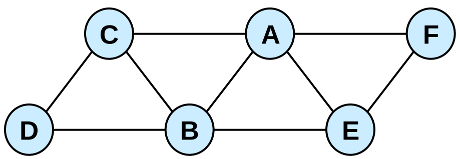
C	D	f_8
0	0	**1**
0	1	**4**
1	0	**0**
1	1	**0**

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

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



The Search Space



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

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

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

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

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

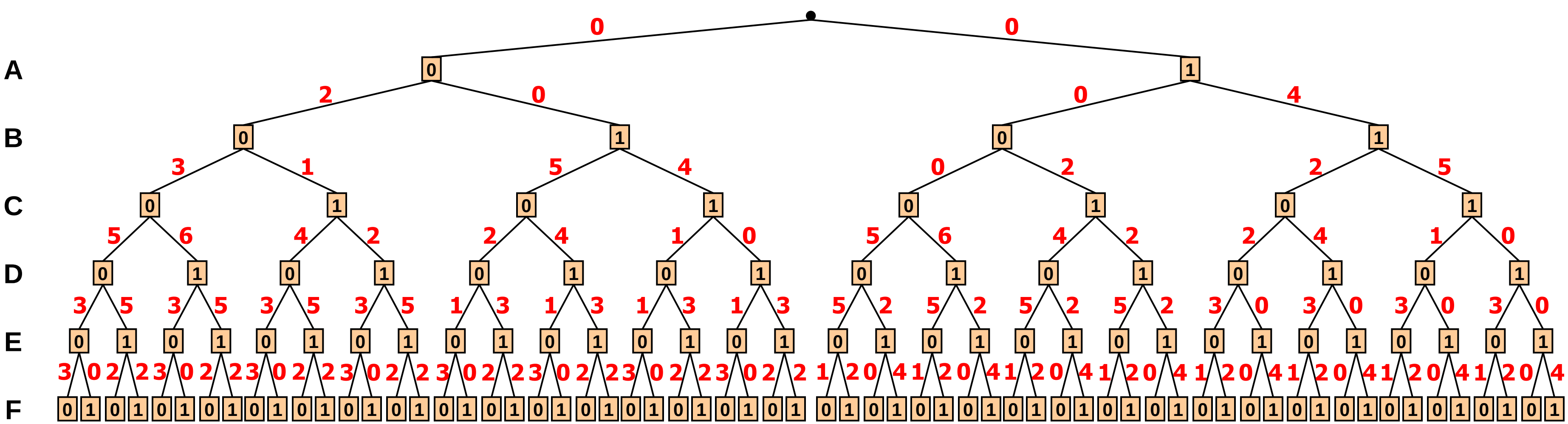
B	D	f_6
0	0	**4**
0	1	**2**
1	0	**1**
1	1	**0**

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

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

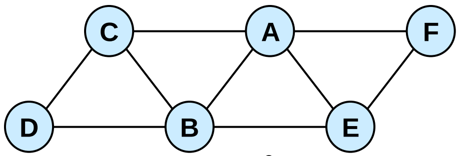
E	F	f_9
0	0	**1**
0	1	**0**
1	0	**0**
1	1	**2**

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



Arc-cost is calculated based on cost components.

The Value Function



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

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**

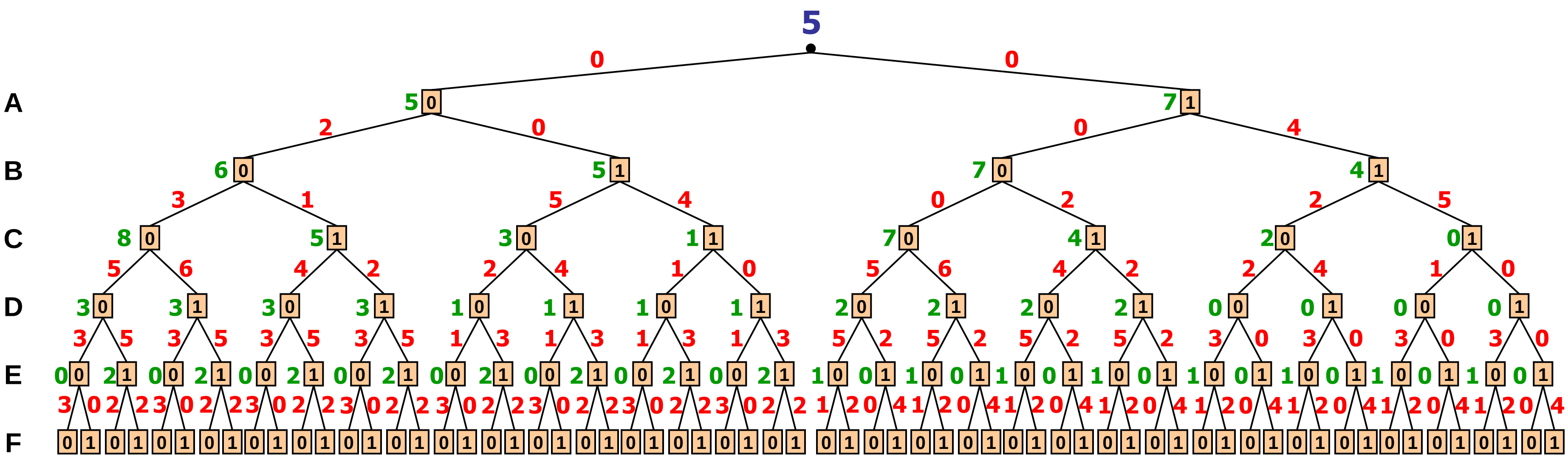
B	C	f_5
0	0	**0**
0	1	**1**
1	0	**2**
1	1	**4**

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**

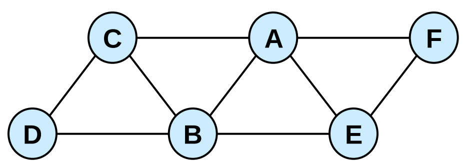
C	D	f_8
0	0	**1**
0	1	**4**
1	0	**0**
1	1	**0**

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



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**

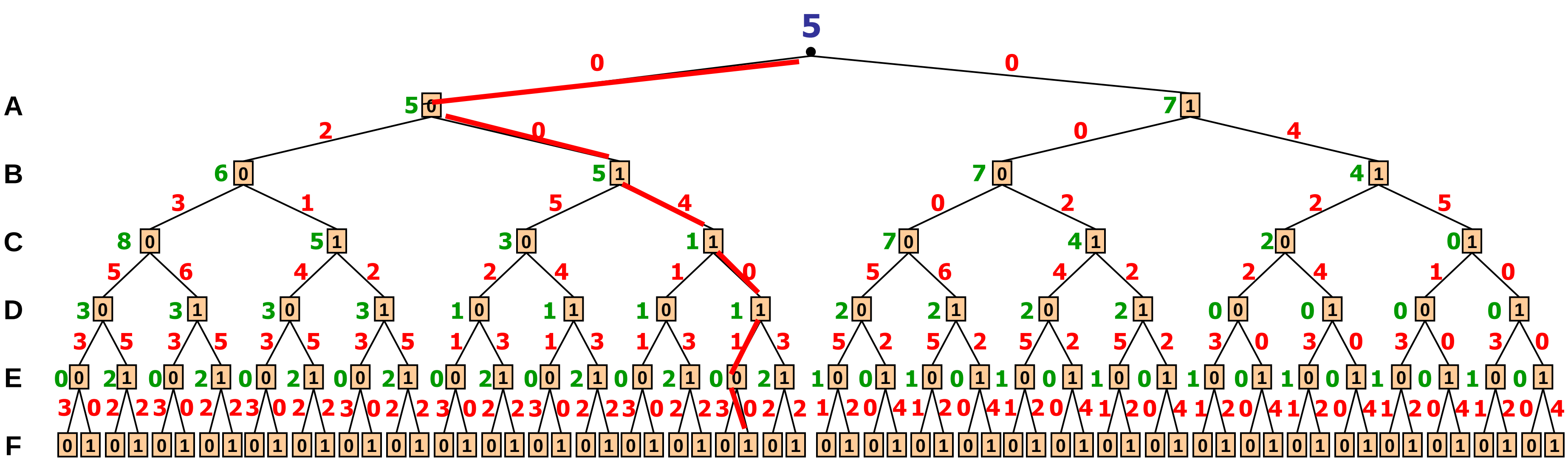
B	C	f_5
0	0	**0**
0	1	**1**
1	0	**2**
1	1	**4**

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**

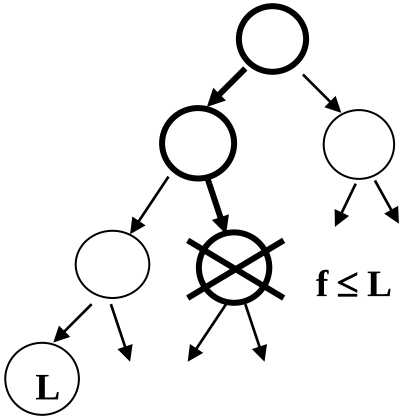


Value of node = minimal cost solution below it

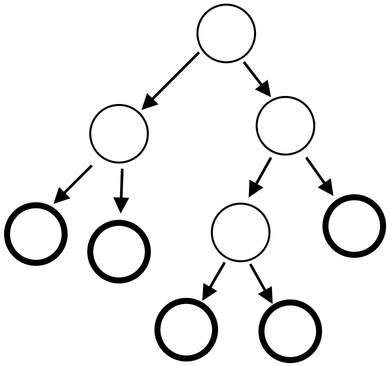
Basic Heuristic Search Schemes

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

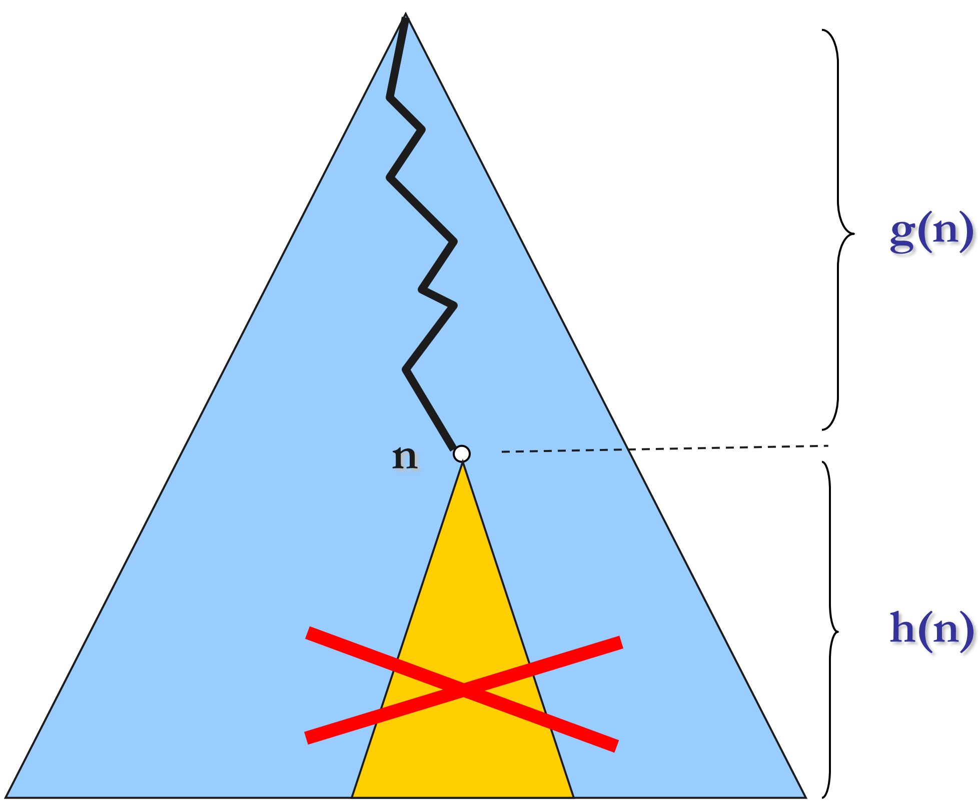
1.Branch and Bound
Use heuristic function $f(x^p)$ to prune the depth-first search tree.
Linear space



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



Classic Branch-and-Bound



OR Search Tree

Upper Bound **UB**

Lower Bound **LB**

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

Prune if $LB(n) \geq UB$

How to Generate Heuristics

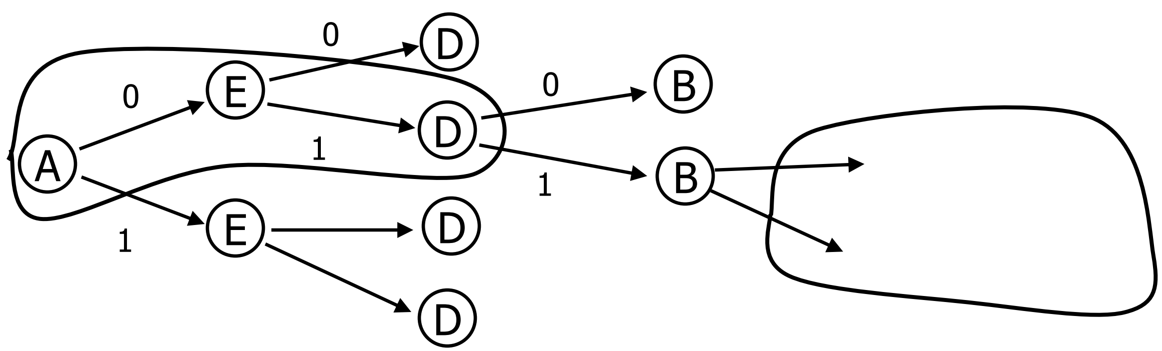
- The principle of relaxed models
  - Linear optimization for integer programs
  - Mini-bucket elimination
  - Bounded directional consistency ideas

Generating Heuristic for graphical models
(Kask and Dechter, 1999)

Given a cost function

$$C(a,b,c,d,e) = f(a) \bullet f(b,a) \bullet f(c,a) \bullet f(e,b,c) \bullet P(d,b,a)$$

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



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

$$= f(a) \bullet \min_{b,c} f(b,a) \bullet P(c,a) \bullet P(e,b,c) \bullet P(d,a,b)$$

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

Generating Heuristics (cont.)

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

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

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

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

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

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

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

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

Generating Heuristics (cont.)

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

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

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

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

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

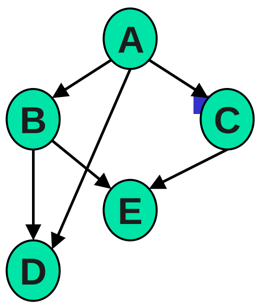
$$= H(a,e,d)$$

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

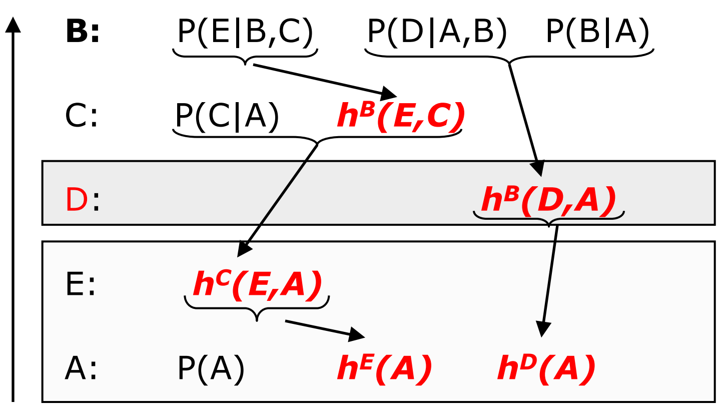
The heuristic function H is what is compiled during the preprocessing stage of the Mini-Bucket algorithm.

Static MBE Heuristics

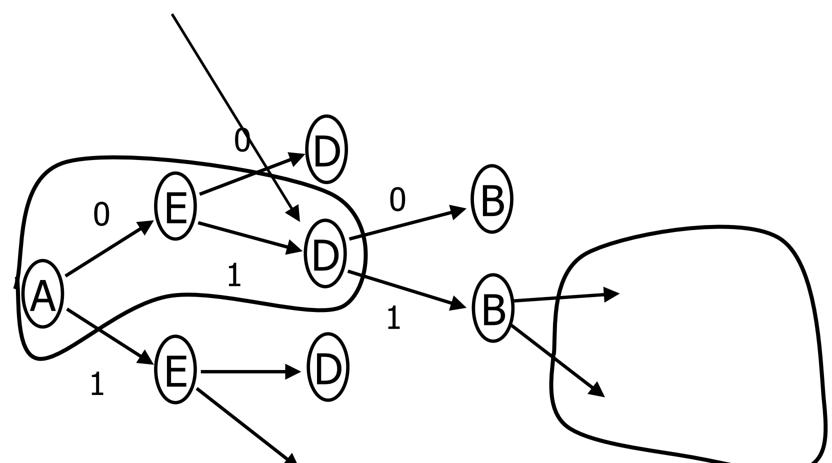
- Given a partial assignment x^p, estimate the cost of the best extension to a full solution
- The evaluation function $f(x^p)$ can be computed using function recorded by the Mini-Bucket scheme



Belief Network



f(a,e,D))=g(a,e) · H(a,e,D)



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

Heuristics Properties

- MB Heuristic is monotone, admissible
- Retrieved 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)

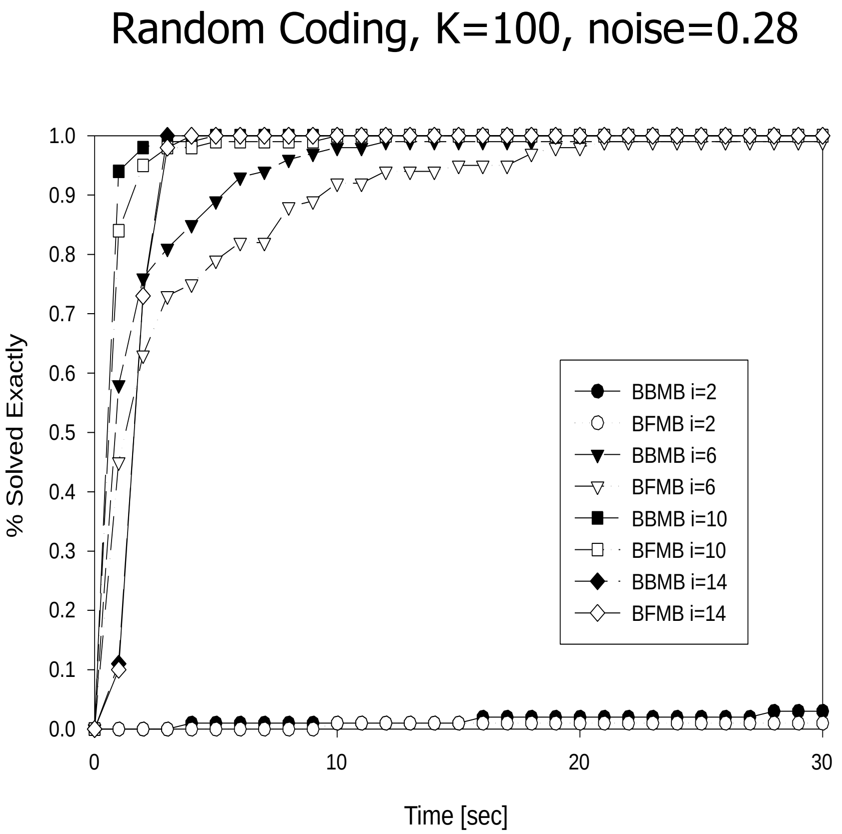
Test networks:

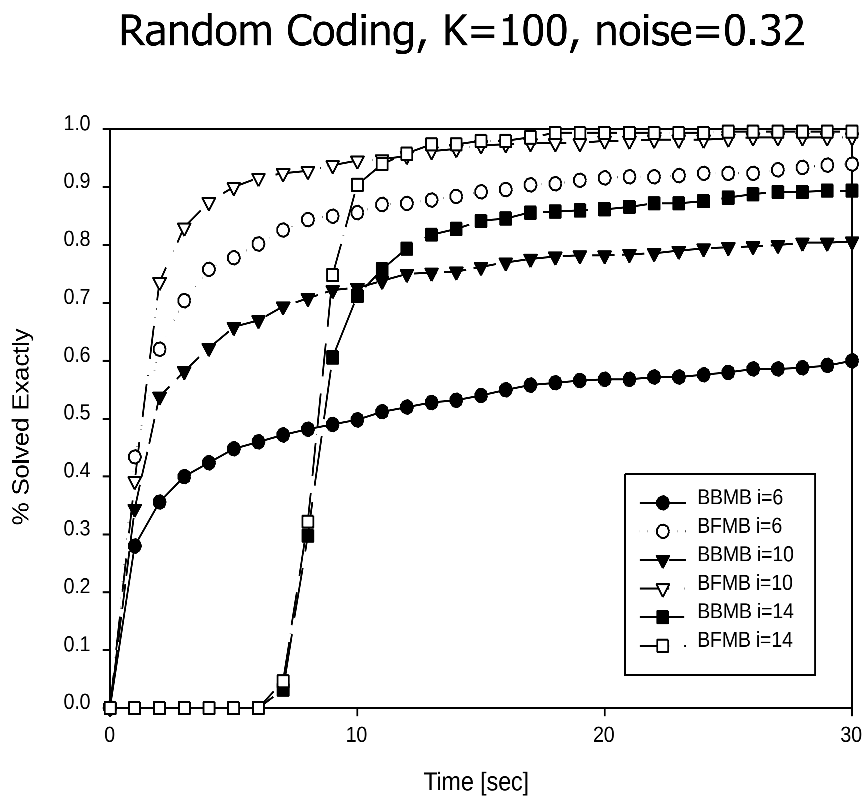
- Random Coding (Bayesian)
- CPCS (Bayesian)
- Random (CSP)

Measures of performance

- Compare accuracy given a fixed amount of time - 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, Bayesian networks, coding





Max-CSP experiments

(Kask and Dechter, 2000)

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

Dynamic MB Heuristics

- Rather than pre-compiling, the mini-bucket heuristics can be generated during search
- D*ynamic mini-bucket heuristics* use the Mini-Bucket algorithm to produce a bound for any node in the search space

(a partial assignment, along the given variable ordering)

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

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

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

Empirical Results: Max-CSP

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

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

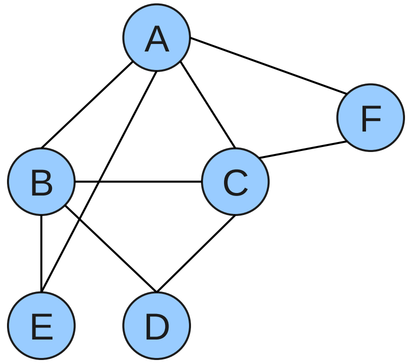
$N = 100$, $K = 5$, $C = 300$. $w^* = 33.9$. 10 instances. time = 600sec.								
T	BBMB i=2	i=3	i=4	i=5	i=6	i=7	BBBT i=2	PFC-MPRDAC
	# solved time backtracks	# solved time backtracks	# solved time backtracks	# solved time backtracks	# solved time backtracks	# solved time backtracks	# solved time backtracks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

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

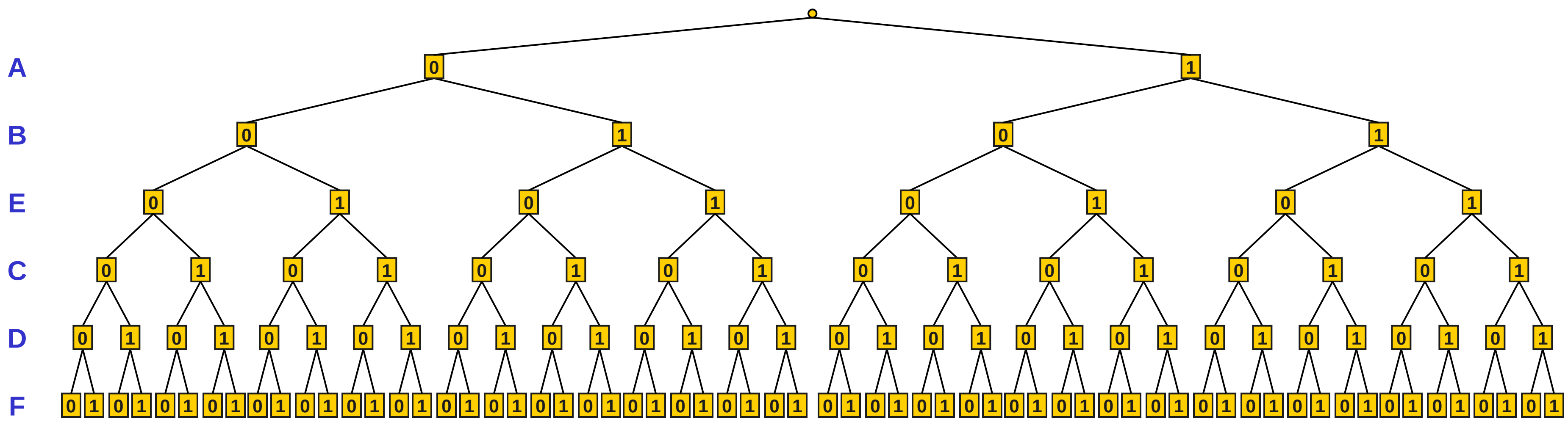
Outline

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

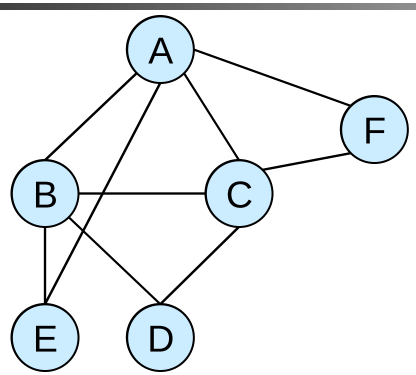
Classic OR Search Space



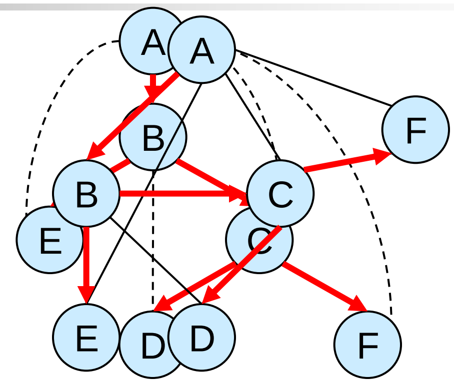
Ordering: A B E C D F



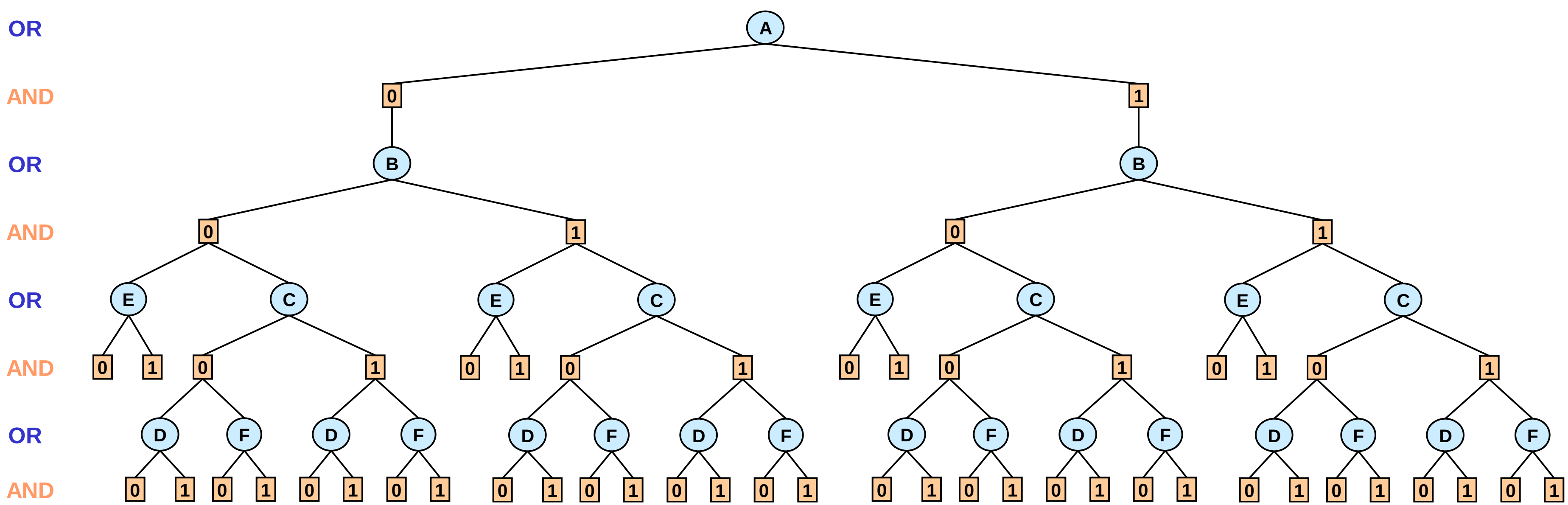
AND/OR Search Space



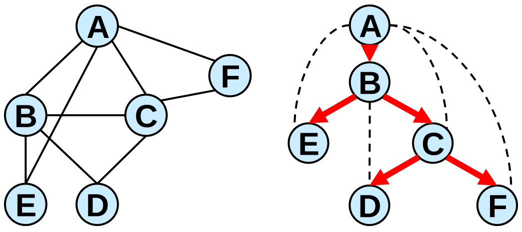
Primal graph

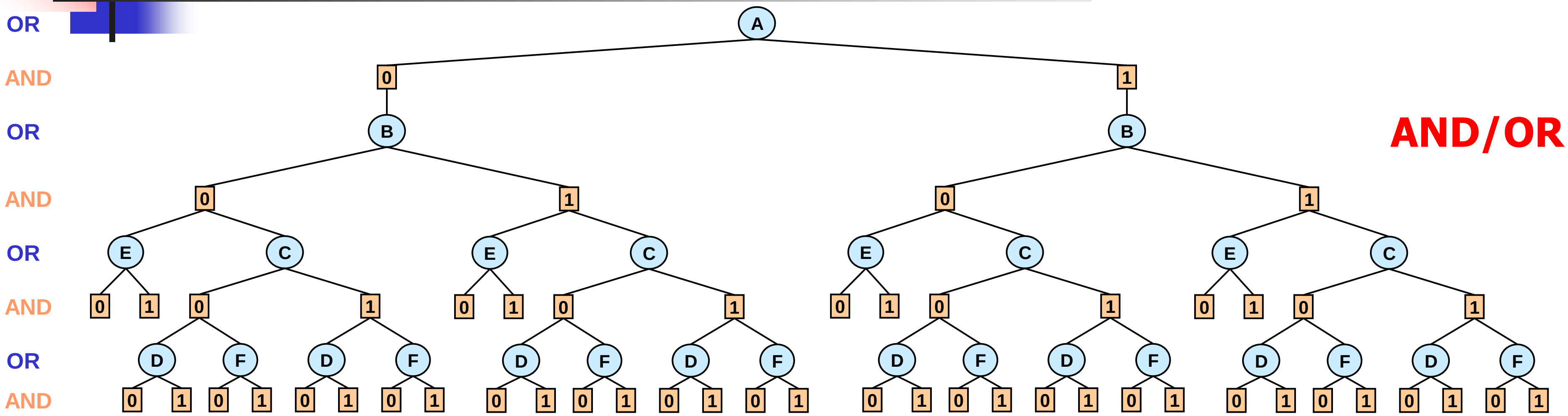


DFS tree

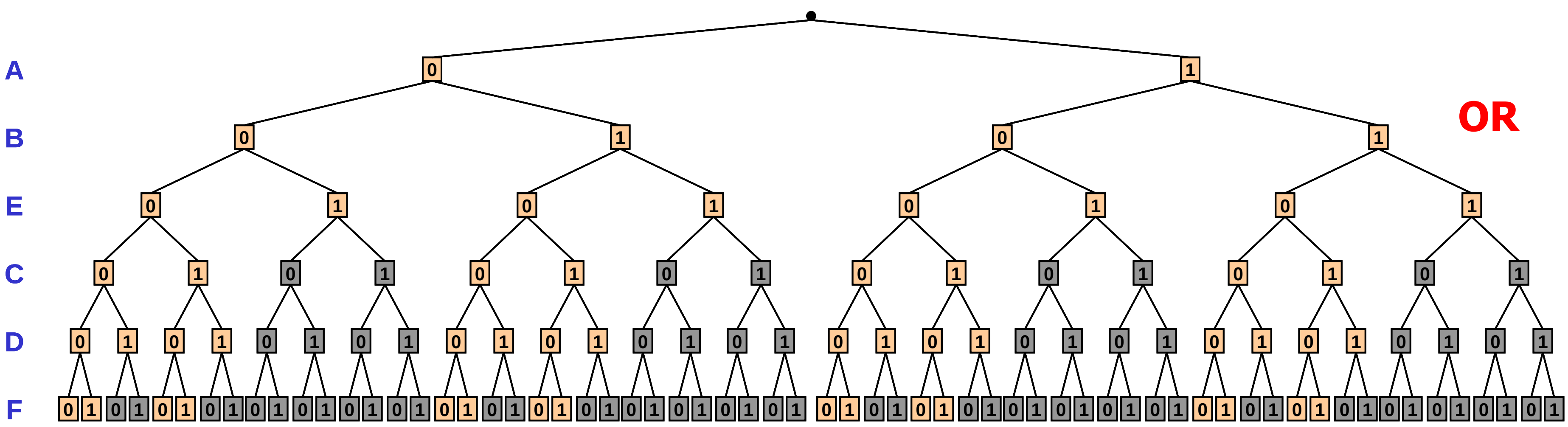


AND/OR vs. OR





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

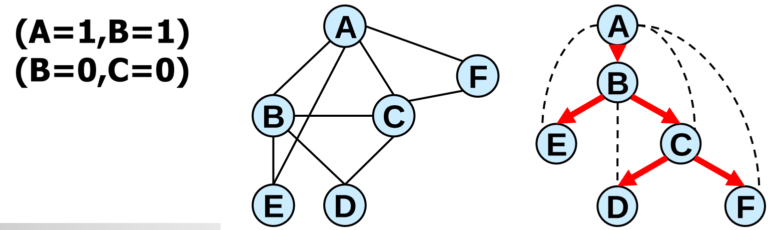


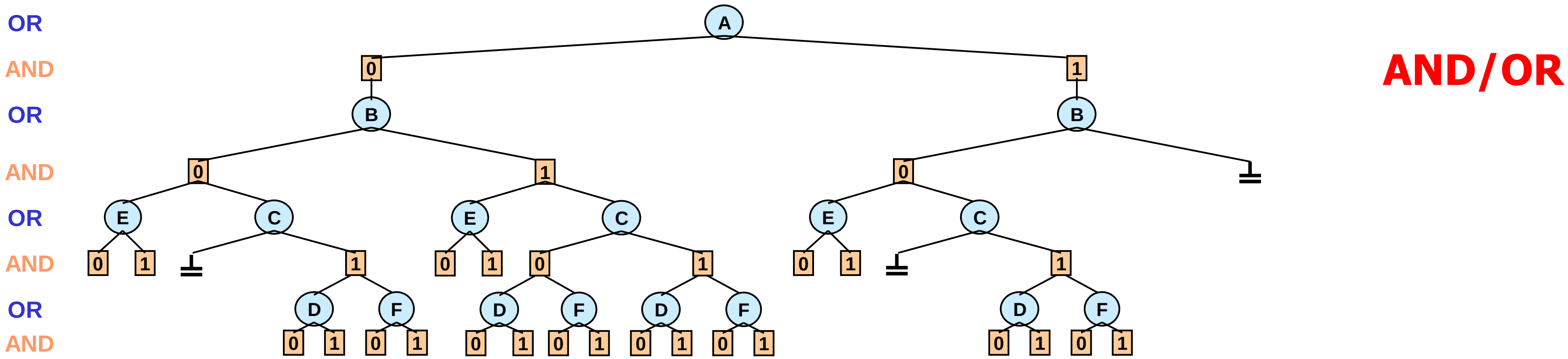
OR space vs. AND/OR space

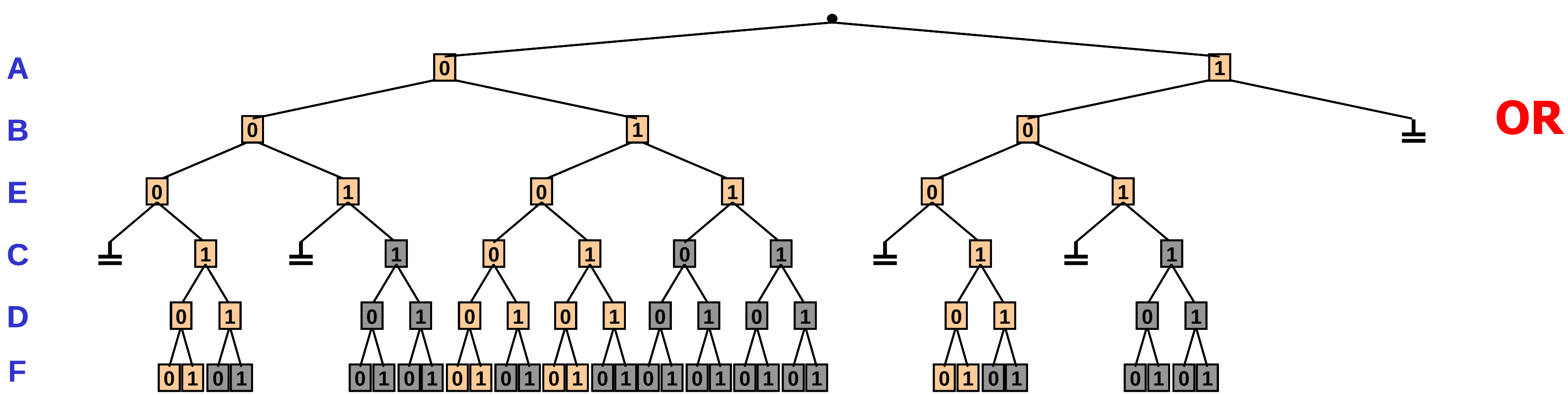
width	height	OR space			AND/OR space		
		Time (sec.)	Nodes	Backtracks	Time (sec.)	AND nodes	OR nodes
5	10	3.154	2,097,150	1,048,575	0.03	10,494	5,247
4	9	3.135	2,097,150	1,048,575	0.01	5,102	2,551
5	10	3.124	2,097,150	1,048,575	0.03	8,926	4,463
4	10	3.125	2,097,150	1,048,575	0.02	7,806	3,903
5	13	3.104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

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

AND/OR vs. OR

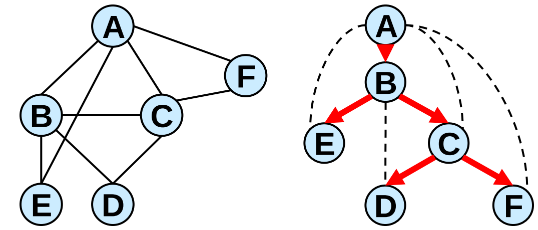


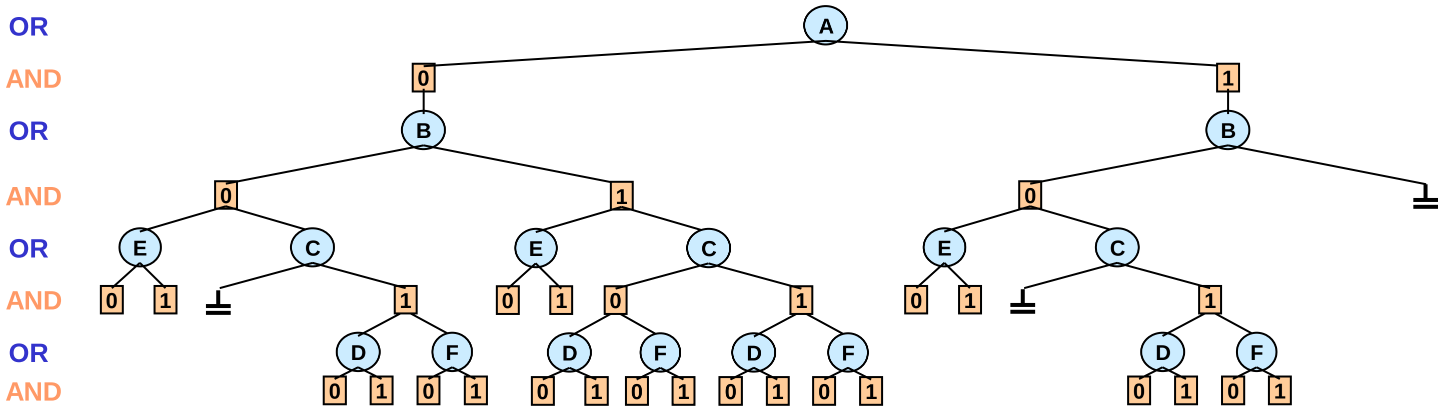




AND/OR vs. OR

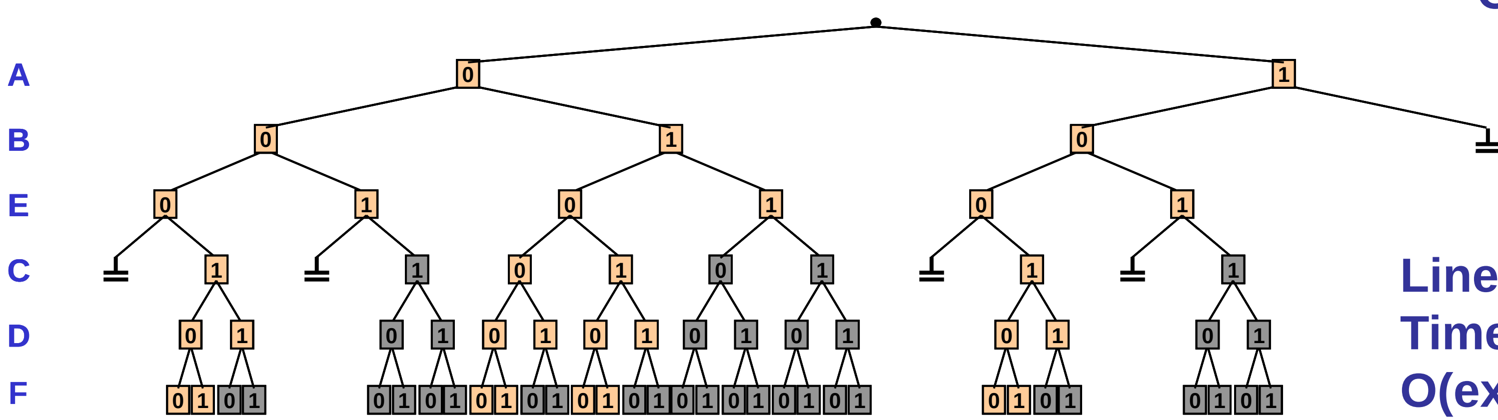
(A=1,B=1)
(B=0,C=0)





AND/OR

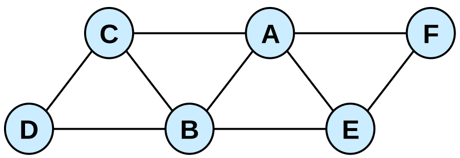
Space: linear
Time:
O(exp(m))
O(w* log n)



OR

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

#CSP – AND/OR Search Tree

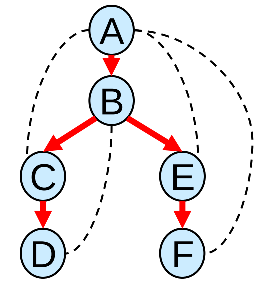


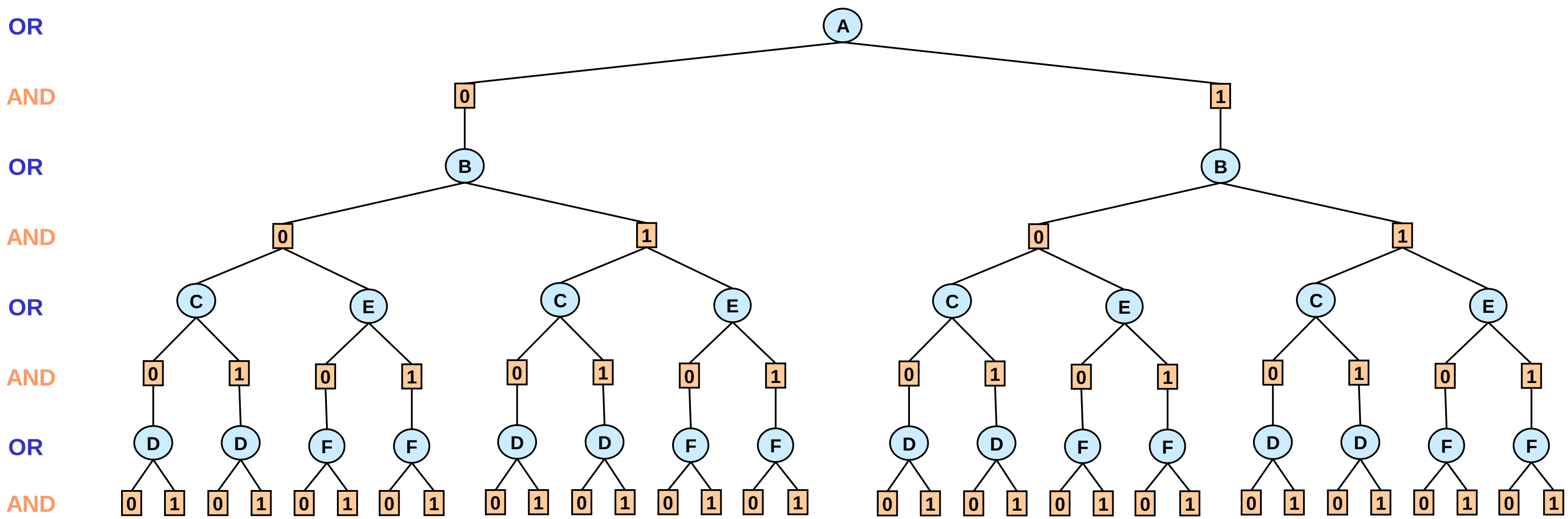
A	B	C	R_{ABC}
0	0	0	**1**
0	0	1	**1**
0	1	0	**0**
0	1	1	**1**
1	0	0	**1**
1	0	1	**1**
1	1	0	**1**
1	1	1	**0**

B	C	D	R_{BCD}
0	0	0	**1**
0	0	1	**1**
0	1	0	**1**
0	1	1	**0**
1	0	0	**1**
1	0	1	**0**
1	1	0	**1**
1	1	1	**1**

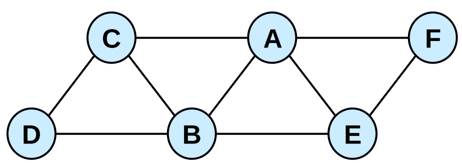
A	B	E	R_{ABE}
0	0	0	**1**
0	0	1	**0**
0	1	0	**1**
0	1	1	**1**
1	0	0	**0**
1	0	1	**1**
1	1	0	**1**
1	1	1	**0**

A	E	F	R_{AEF}
0	0	0	**0**
0	0	1	**1**
0	1	0	**1**
0	1	1	**1**
1	0	0	**1**
1	0	1	**1**
1	1	0	**1**
1	1	1	**0**





#CSP – AND/OR Search Tree

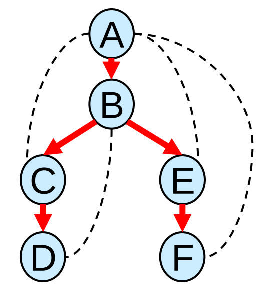


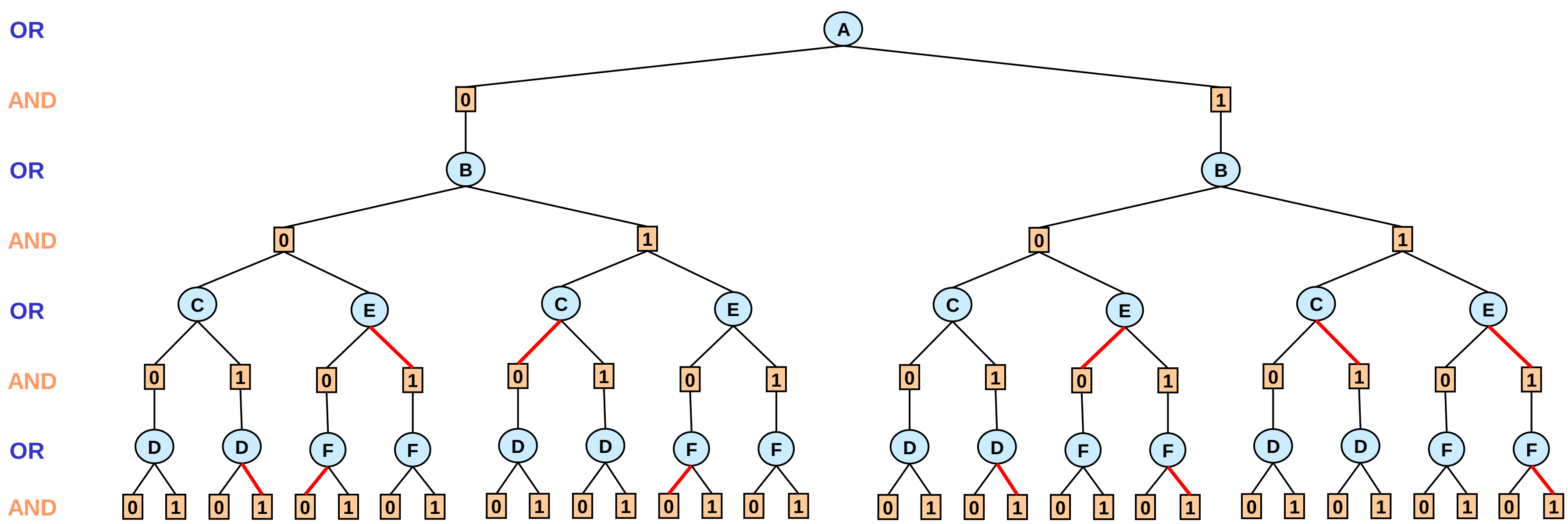
A	B	C	R_{ABC}
0	0	0	**1**
0	0	1	**1**
0	1	0	**0**
0	1	1	**1**
1	0	0	**1**
1	0	1	**1**
1	1	0	**1**
1	1	1	**0**

B	C	D	R_{BCD}
0	0	0	**1**
0	0	1	**1**
0	1	0	**1**
0	1	1	**0**
1	0	0	**1**
1	0	1	**0**
1	1	0	**1**
1	1	1	**1**

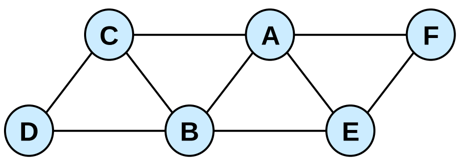
A	B	E	R_{ABE}
0	0	0	**1**
0	0	1	**0**
0	1	0	**1**
0	1	1	**1**
1	0	0	**0**
1	0	1	**1**
1	1	0	**1**
1	1	1	**0**

A	E	F	R_{AEF}
0	0	0	**0**
0	0	1	**1**
0	1	0	**1**
0	1	1	**1**
1	0	0	**1**
1	0	1	**1**
1	1	0	**1**
1	1	1	**0**





#CSP – AND/OR Tree DFS

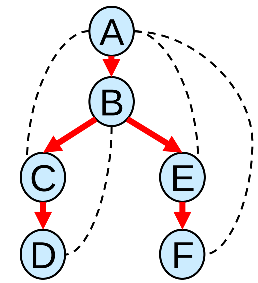


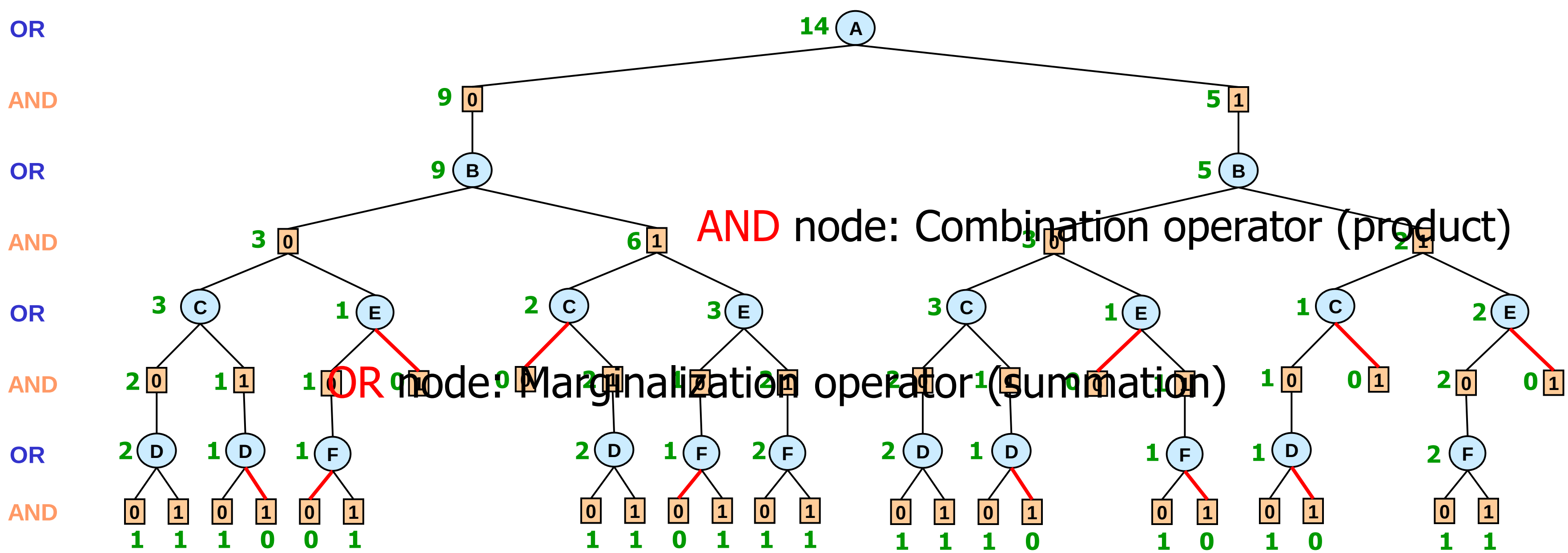
A	B	C	R_{ABC}
0	0	0	**1**
0	0	1	**1**
0	1	0	**0**
0	1	1	**1**
1	0	0	**1**
1	0	1	**1**
1	1	0	**1**
1	1	1	**0**

B	C	D	R_{BCD}
0	0	0	**1**
0	0	1	**1**
0	1	0	**1**
0	1	1	**0**
1	0	0	**1**
1	0	1	**0**
1	1	0	**1**
1	1	1	**1**

A	B	E	R_{ABE}
0	0	0	**1**
0	0	1	**0**
0	1	0	**1**
0	1	1	**1**
1	0	0	**0**
1	0	1	**1**
1	1	0	**1**
1	1	1	**0**

A	E	F	R_{AEF}
0	0	0	**0**
0	0	1	**1**
0	1	0	**1**
0	1	1	**1**
1	0	0	**1**
1	0	1	**1**
1	1	0	**1**
1	1	1	**0**

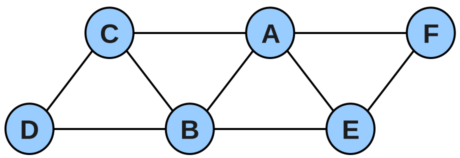


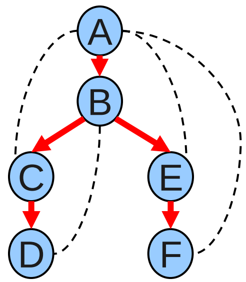


AND node: Combination operator (product)

OR node: Marginalization operator (summation)

AND/OR Tree Search for COP





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

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

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

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

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

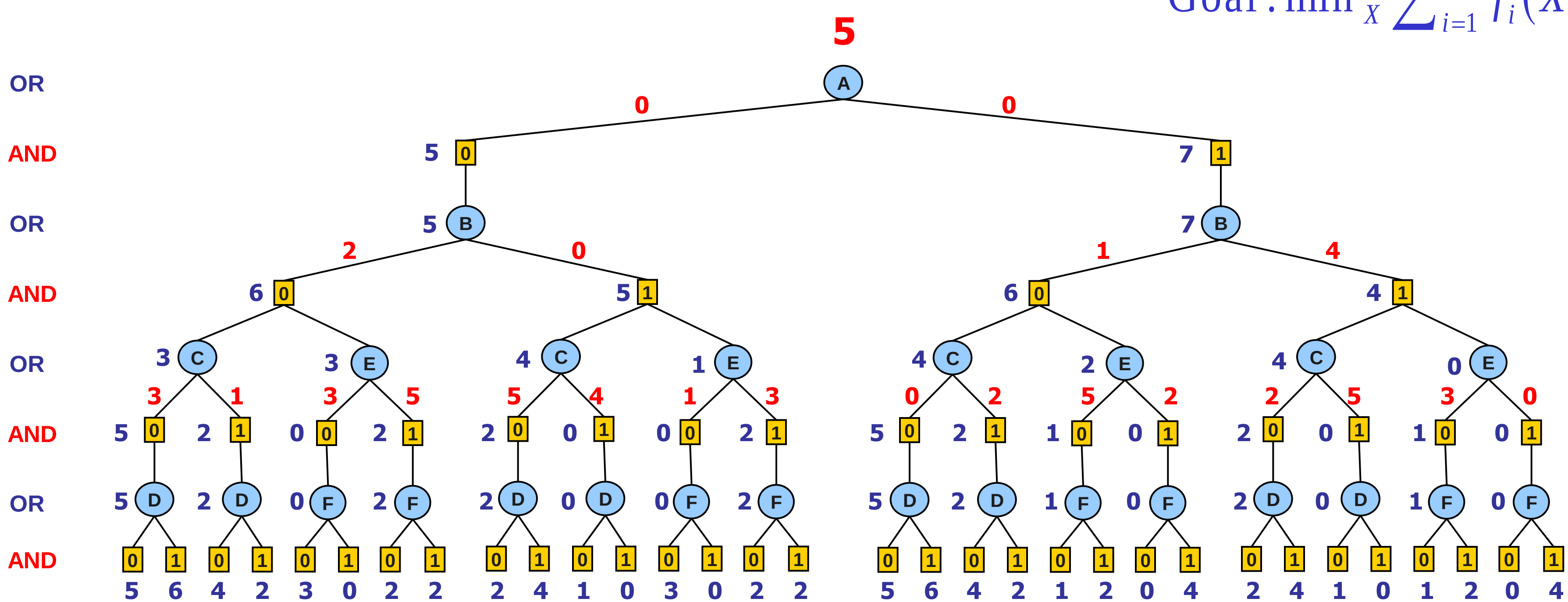
B	D	f_6
0	0	4
0	1	2
1	0	1
1	1	0

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

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

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

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



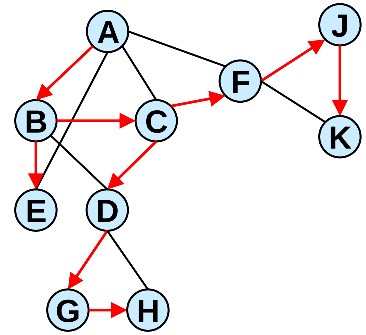
OR node = Marginalization operator (minimization)

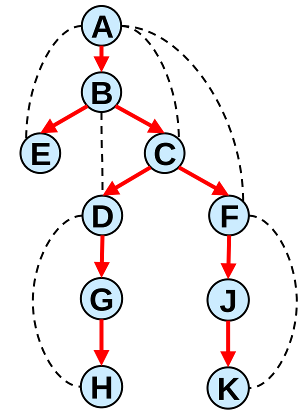
AND node = Combination operator (summation)

Summary of AND/OR Search Trees

- Based on a backbone pseudo-tree
- A solution is a subtree
- Each node has a **value** – cost of the optimal solution to the subproblem (computed recursively based on the values of the descendants)
- **Solving a task = finding the value of the root node**
- AND/OR search tree and algorithms are
  ([Freuder & Quinn85], [Collin, Dechter & Katz91], [Bayardo & Miranker95])
  - Space: O(n)
  - Time: O(exp(m)), where m is the depth of the pseudo-tree
  - Time: O(exp(w* log n))
  - BFS is time and space O(exp(w* log n)

Caching



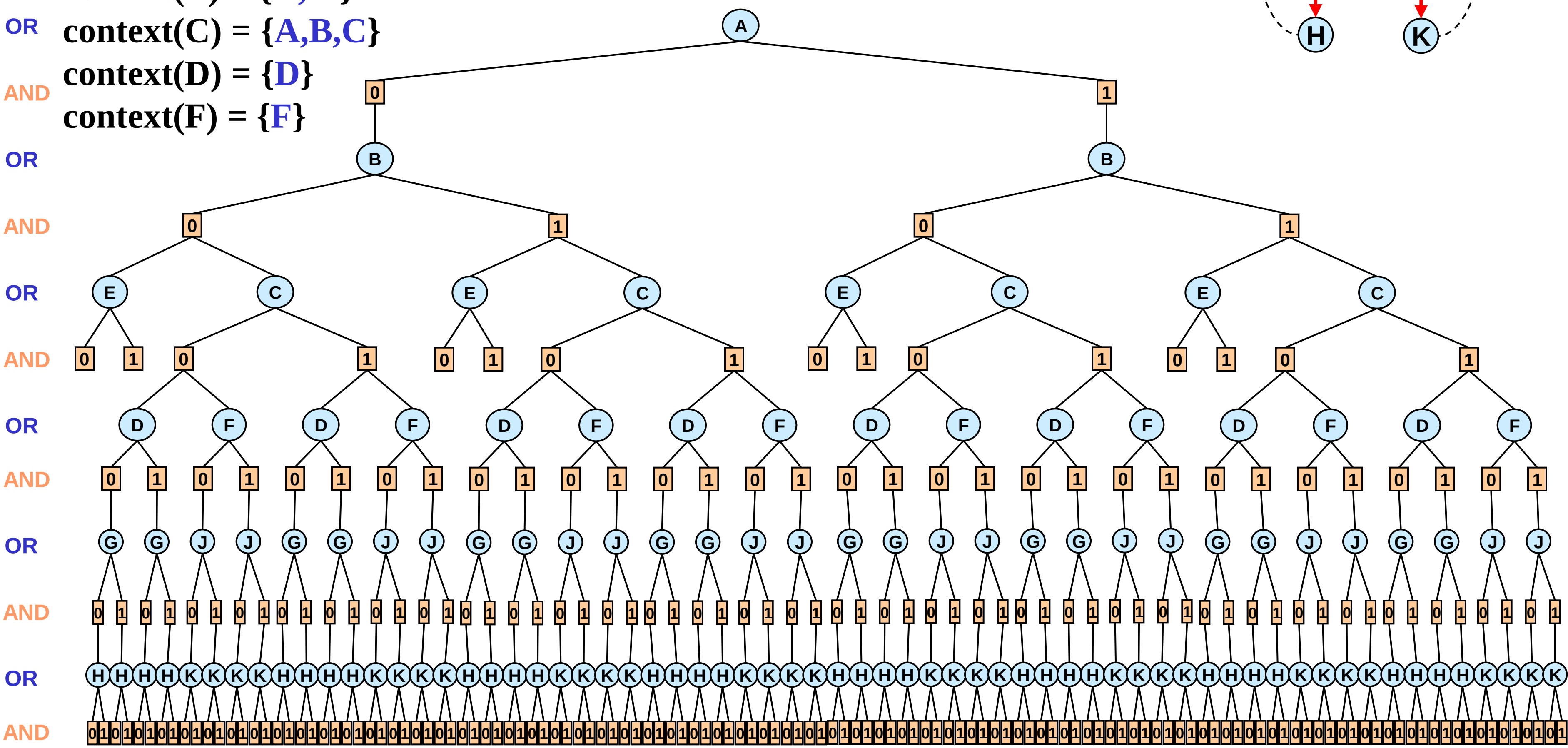


context(B) = {A, B}

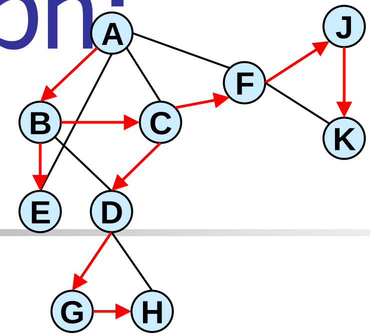
context(C) = {A,B,C}

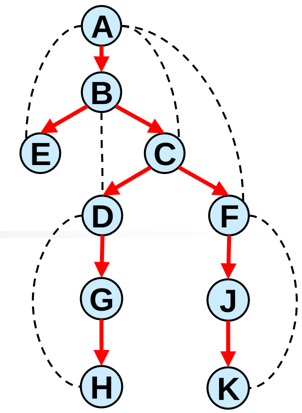
context(D) = {D}

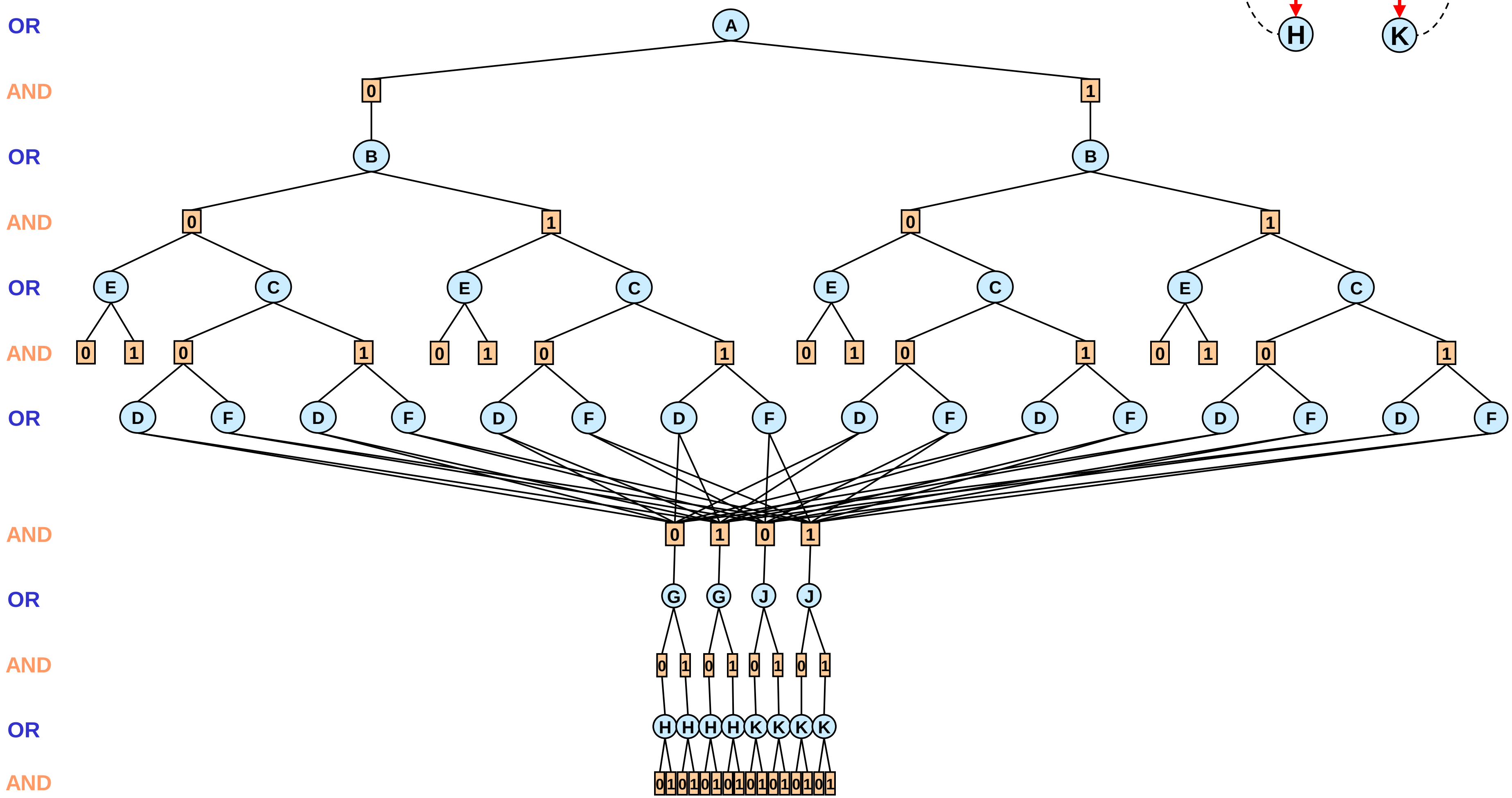
context(F) = {F}



An AND/OR Graph: Caching Goods



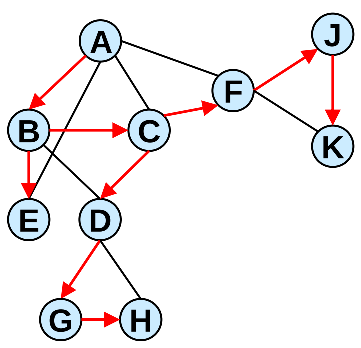


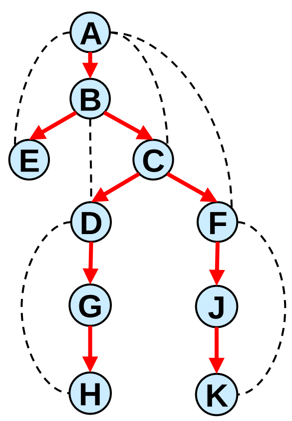


Context-based Caching

- Caching is possible when context is the same

- context = parent-separator set in induced pseudo-graph
  = current variable + parents connected to subtree below





context(B) = {A, B}

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

context(D) = {D}

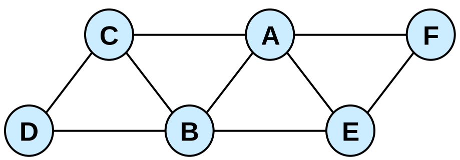
context(F) = {F}

Complexity of AND/OR Graph

- **Theorem:** Traversing the AND/OR search graph is time and space exponential in the induced width/tree-width.

- If applied to the OR graph complexity is time and space exponential in the path-width.

#CSP – AND/OR Search Tree

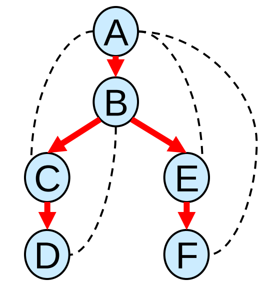


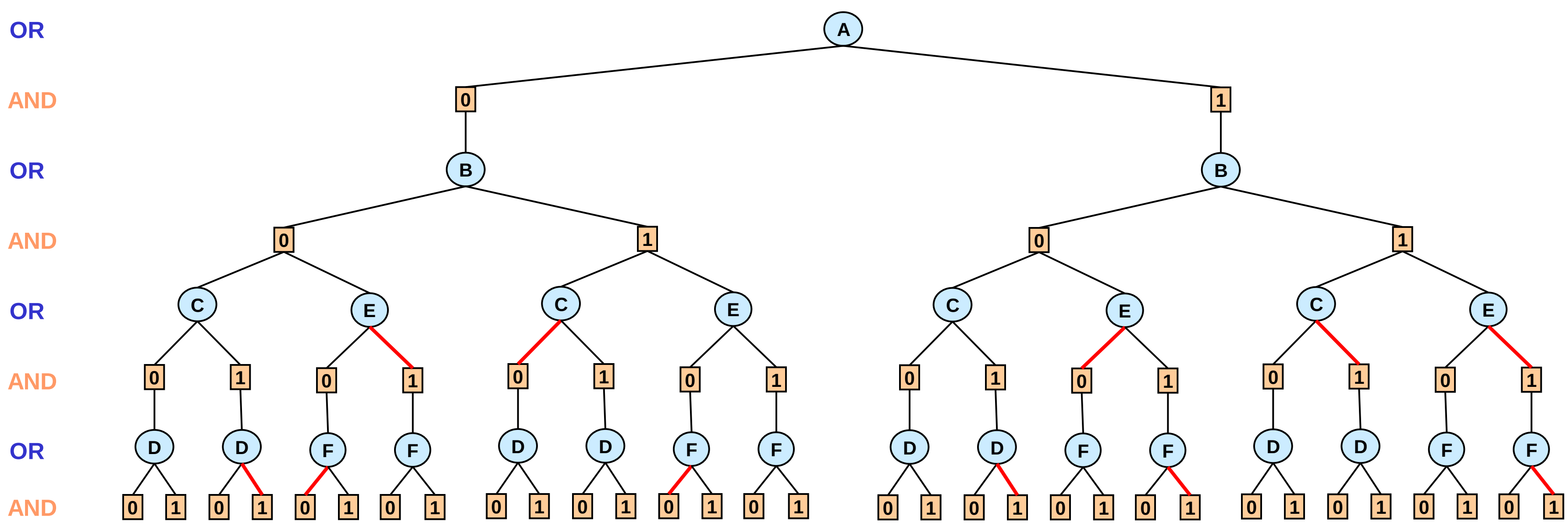
A	B	C	R_{ABC}
0	0	0	**1**
0	0	1	**1**
0	1	0	**0**
0	1	1	**1**
1	0	0	**1**
1	0	1	**1**
1	1	0	**1**
1	1	1	**0**

B	C	D	R_{BCD}
0	0	0	**1**
0	0	1	**1**
0	1	0	**1**
0	1	1	**0**
1	0	0	**1**
1	0	1	**0**
1	1	0	**1**
1	1	1	**1**

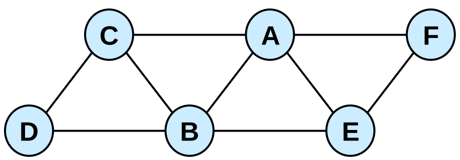
A	B	E	R_{ABE}
0	0	0	**1**
0	0	1	**0**
0	1	0	**1**
0	1	1	**1**
1	0	0	**0**
1	0	1	**1**
1	1	0	**1**
1	1	1	**0**

A	E	F	R_{AEF}
0	0	0	**0**
0	0	1	**1**
0	1	0	**1**
0	1	1	**1**
1	0	0	**1**
1	0	1	**1**
1	1	0	**1**
1	1	1	**0**





#CSP – AND/OR Tree DFS

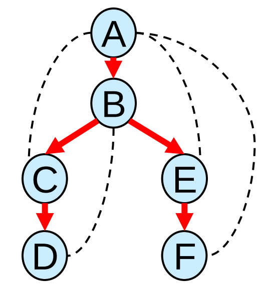


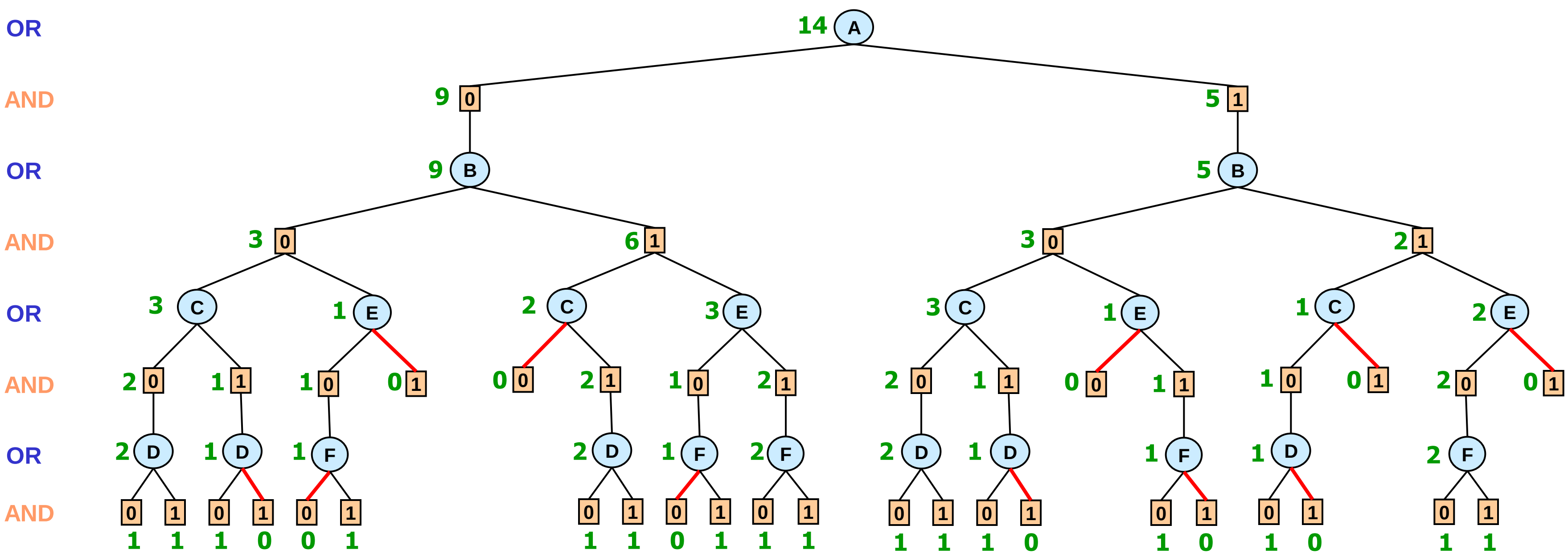
A	B	C	R_{ABC}
0	0	0	**1**
0	0	1	**1**
0	1	0	**0**
0	1	1	**1**
1	0	0	**1**
1	0	1	**1**
1	1	0	**1**
1	1	1	**0**

B	C	D	R_{BCD}
0	0	0	**1**
0	0	1	**1**
0	1	0	**1**
0	1	1	**0**
1	0	0	**1**
1	0	1	**0**
1	1	0	**1**
1	1	1	**1**

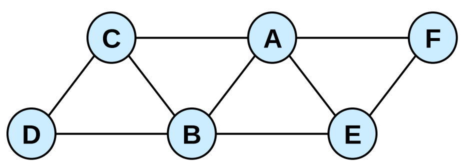
A	B	E	R_{ABE}
0	0	0	**1**
0	0	1	**0**
0	1	0	**1**
0	1	1	**1**
1	0	0	**0**
1	0	1	**1**
1	1	0	**1**
1	1	1	**0**

A	E	F	R_{AEF}
0	0	0	**0**
0	0	1	**1**
0	1	0	**1**
0	1	1	**1**
1	0	0	**1**
1	0	1	**1**
1	1	0	**1**
1	1	1	**0**





#CSP – AND/OR Search Graph (Caching Goods)

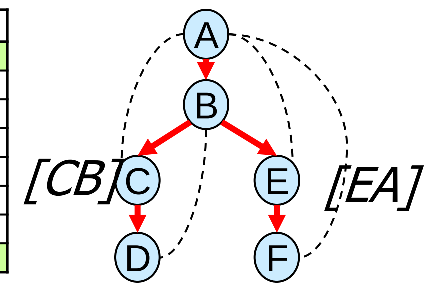


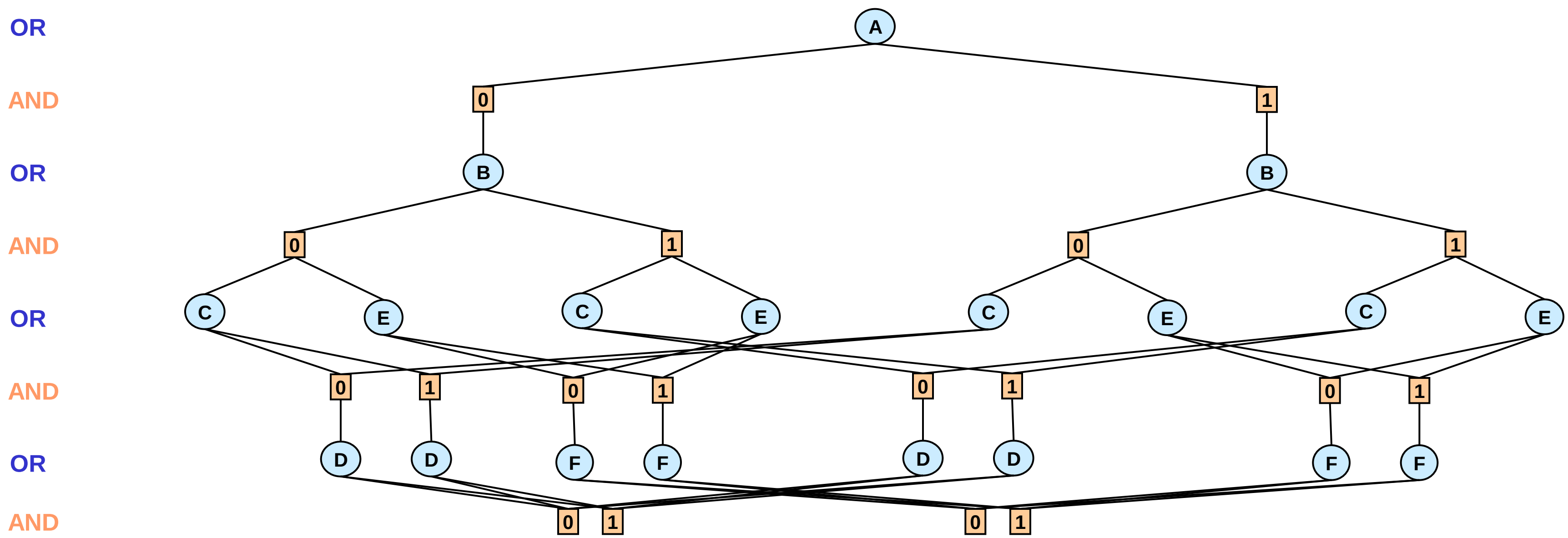
A	B	C	R_{ABC}
0	0	0	**1**
0	0	1	**1**
0	1	0	**0**
0	1	1	**1**
1	0	0	**1**
1	0	1	**1**
1	1	0	**1**
1	1	1	**0**

B	C	D	R_{BCD}
0	0	0	**1**
0	0	1	**1**
0	1	0	**1**
0	1	1	**0**
1	0	0	**1**
1	0	1	**0**
1	1	0	**1**
1	1	1	**1**

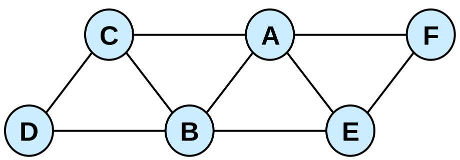
A	B	E	R_{ABE}
0	0	0	**1**
0	0	1	**0**
0	1	0	**1**
0	1	1	**1**
1	0	0	**0**
1	0	1	**1**
1	1	0	**1**
1	1	1	**0**

A	E	F	R_{AEF}
0	0	0	**0**
0	0	1	**1**
0	1	0	**1**
0	1	1	**1**
1	0	0	**1**
1	0	1	**1**
1	1	0	**1**
1	1	1	**0**





#CSP – AND/OR Search Graph (Caching Goods)

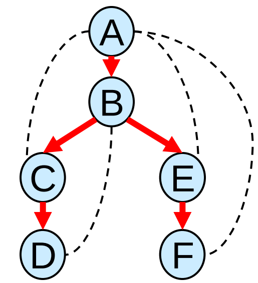


A	B	C	R_{ABC}
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

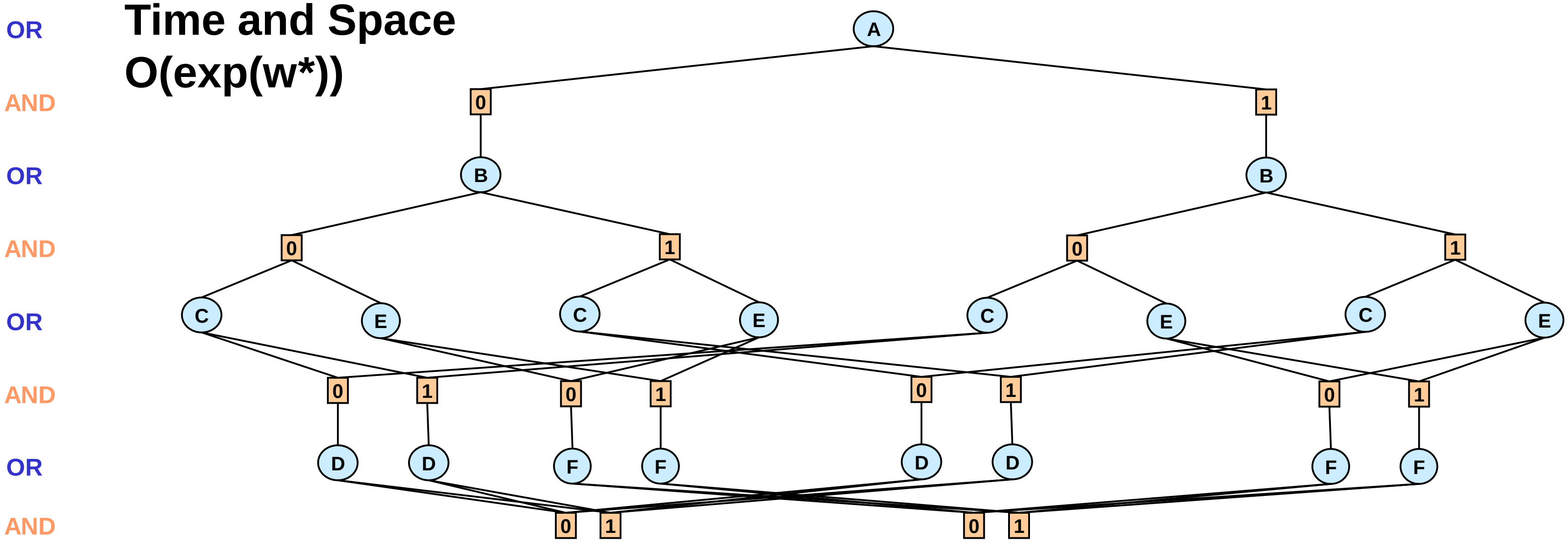
B	C	D	R_{BCD}
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

A	B	E	R_{ABE}
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

A	E	F	R_{AEF}
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

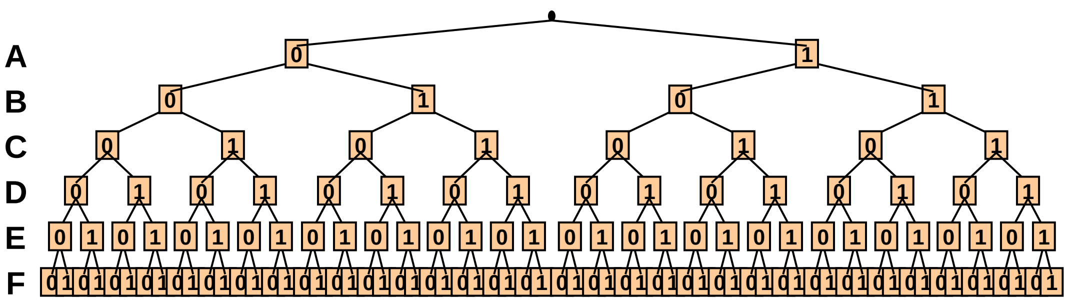


Time and Space O(exp(w*))



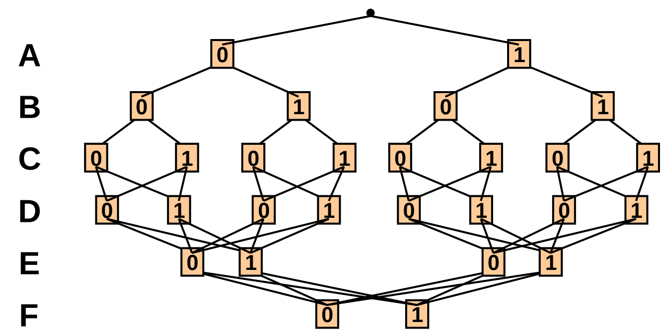
Space O(exp(sep-w*))

All Four Search Spaces



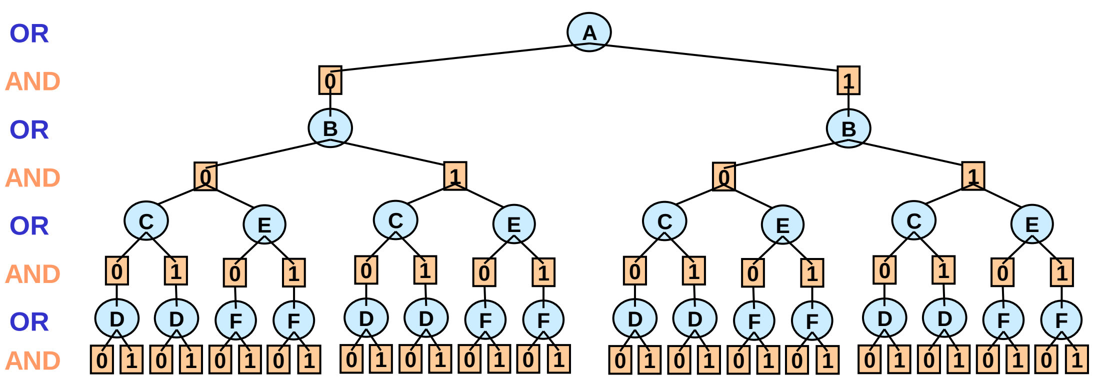
Full OR search tree

126 nodes



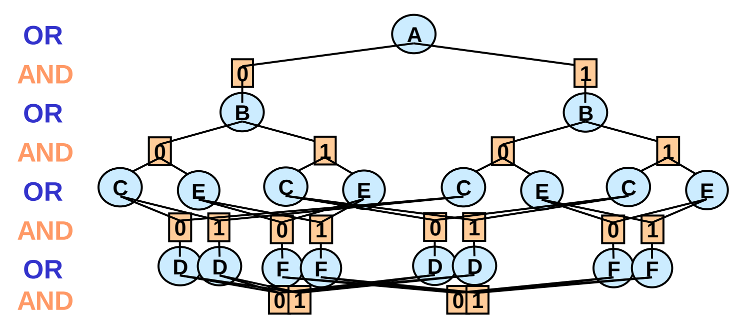
Context minimal OR search graph

28 nodes



Full AND/OR search tree

54 AND nodes



Context minimal AND/OR search graph

18 AND nodes

AND/OR vs. OR DFS algorithms

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

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

  (Freuder85; Bayardo95; Darwiche01)

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

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

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

Searching AND/OR Graphs

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

i=0 → i=w*

Space: O(n)

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

Space: O(exp w*)

Time: O(exp w*)

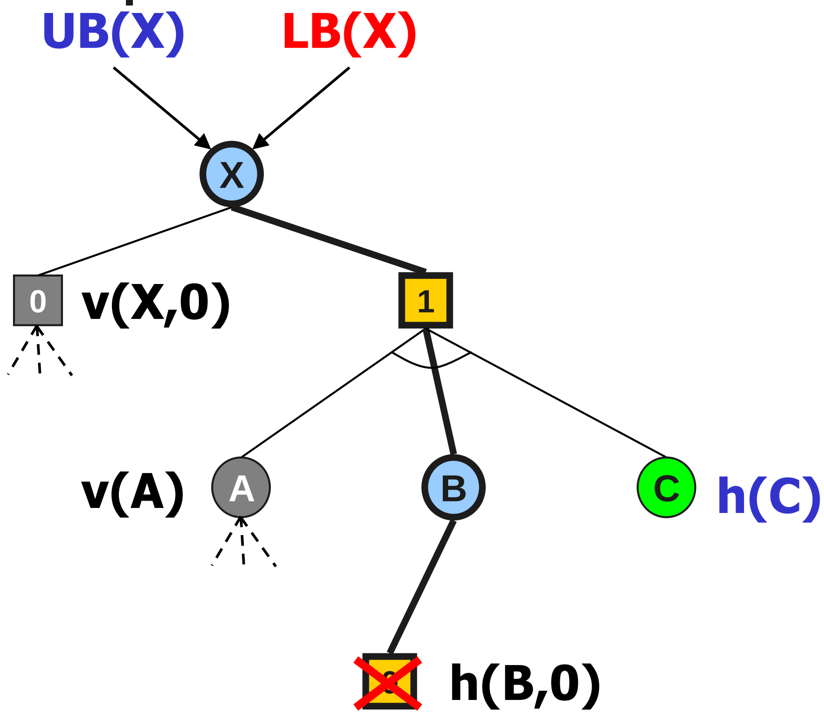
AO(i) time complexity?

AND/OR Branch-and-Bound (AOBB)

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

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

Lower/Upper Bounds



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

LB(X) = LB(X,1)

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

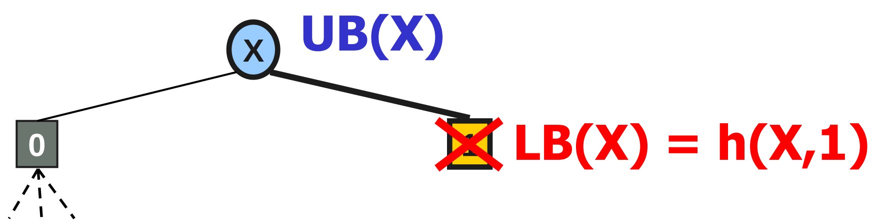
LB(B) = LB(B,0)

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

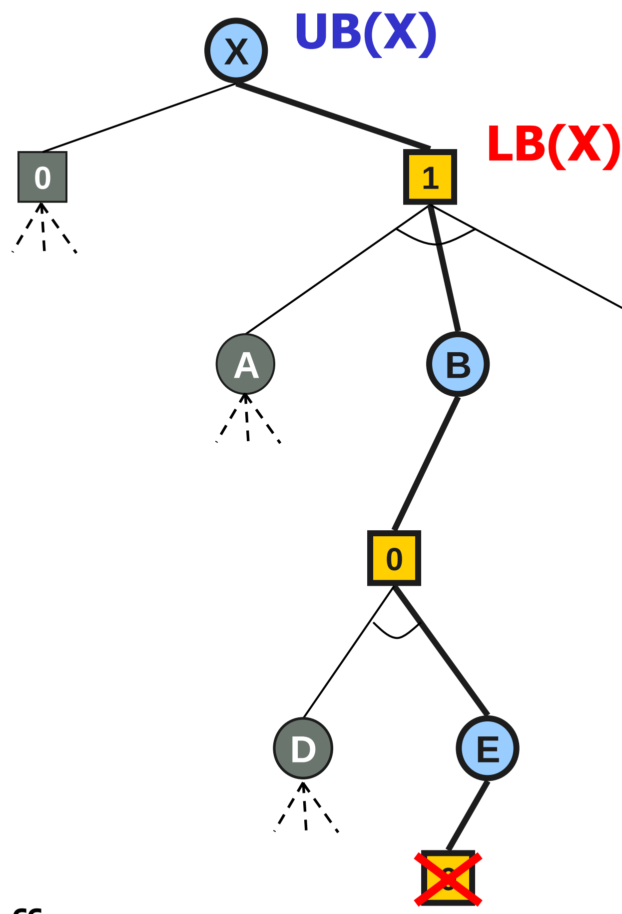
Prune below AND node (B,0) if LB(X) ≥ UB(X)

Shallow/Deep Cutoffs

Prune if LB(X) ≥ UB(X)



Shallow cutoff



Deep cutoff

Reminiscent of **Minimax** shallow/deep cutoffs

Summary of AOBB

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

Heuristics for AND/OR

- In the AND/OR search space **h(n)** can be computed using any heuristic. We used:
  - Static Mini-Bucket heuristics
  - Dynamic Mini-Bucket heuristics
  - Maintaining FDAC [Larrosa & Schiex03]
    (full directional soft arc-consistency)

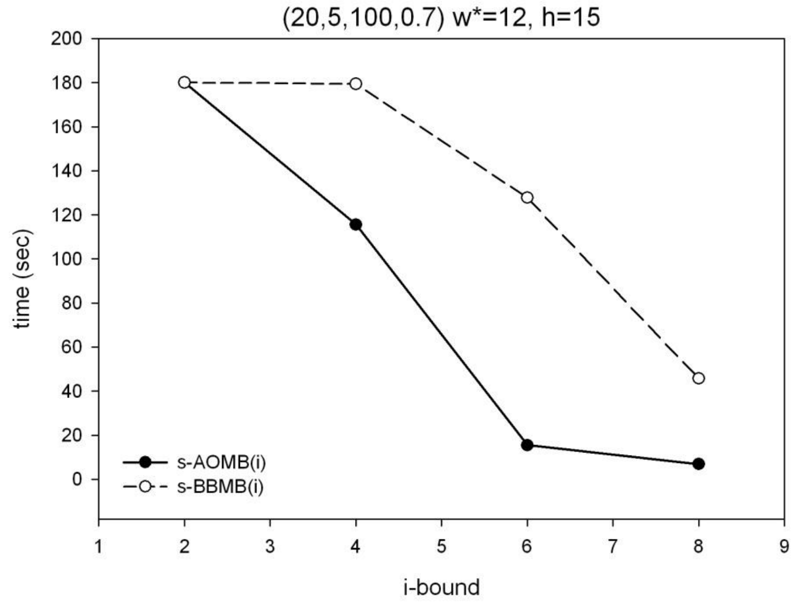
Empirical Evaluation

- Tasks
  - Solving WCSPs
  - Finding the MPE in belief networks
- Benchmarks (WCSP)
  - Random binary WCSPs
  - RLFAP networks (CELAR6)
  - Bayesian Networks Repository
- Algorithms
  - s-AOMB(i), d-AOMB(i), AOMFDAC
  - s-BBMB(i), d-BBMB(i), BBMFDAC
  - Static variable ordering (dfs traversal of the pseudo-tree)

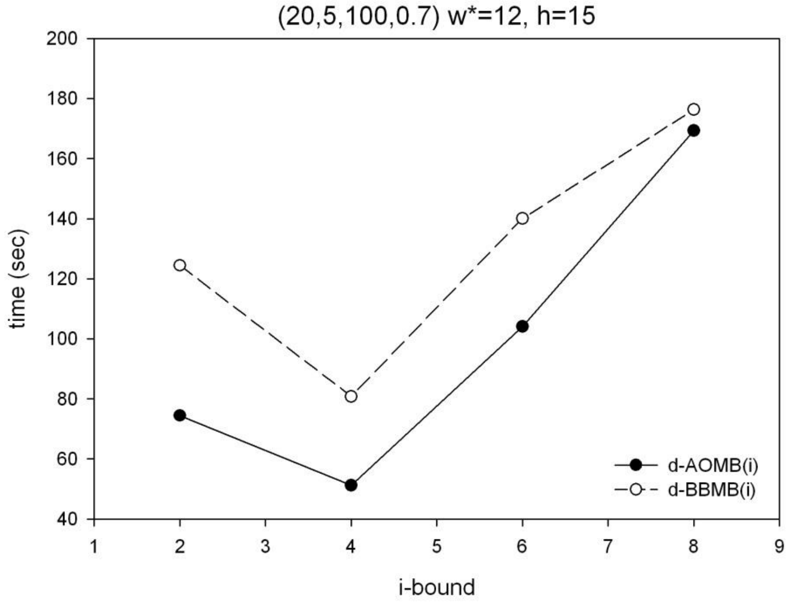
Random Binary WCSPs

(Marinescu and Dechter, 2005)

S-AOMB vs S-BBMB



D-AOMB vs D-BBMB

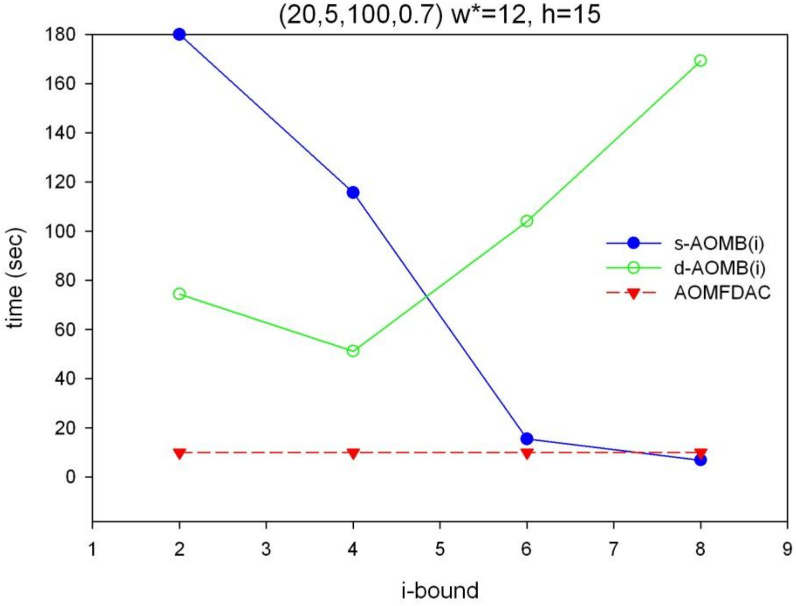


Random networks with n=20 (number of variables), d=5 (domain size), c=100 (number of constraints), t=70% (tightness). Time limit 180 seconds.

AO search is superior to OR search

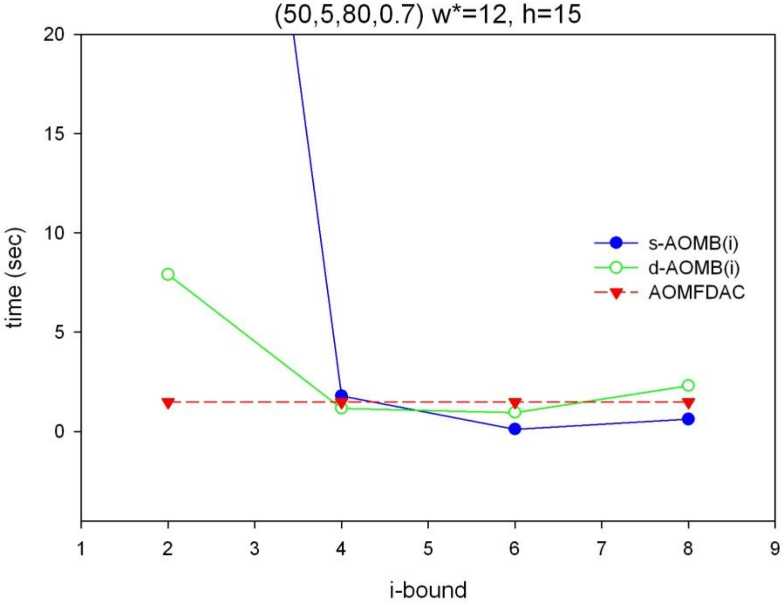
Random Binary WCSPs (contd.)

dense



n=20 (variables), d=5 (domain size), c=100 (constraints), t=70% (tightness)

sparse



n=50 (variables), d=5 (domain size), c=80 (constraints), t=70% (tightness)

AOMB for large i is competitive with AOMFDAC

Resource Allocation

Radio Link Frequency Assignment Problem (RLFAP)

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

CELAR6 sub-instances

AOMFDAC is superior to **OR**MFDAC

Bayesian Networks Repository

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

Time limit 600 seconds

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

Static AO is better with accurate heuristic (large i)

Outline

- **Introduction**
  - Optimization tasks for graphical models
  - Solving by inference and search
- **Inference**
  - Bucket elimination, dynamic programming
  - Mini-bucket elimination, belief propagation
- **Search**
  - Branch and bound and best-first
  - Lower-bounding heuristics
  - **AND/OR search spaces**
    - **Searching trees**
    - **Searching graphs**
- **Hybrids of search and inference**
  - Cutset decomposition
  - Super-bucket scheme

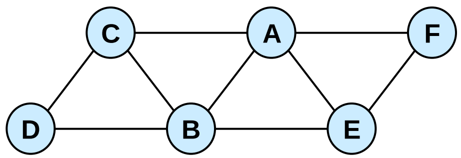
From Searching Trees to Searching Graphs

- Any two nodes that root identical subtrees/subgraphs can be merged

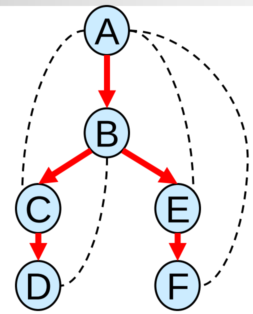
- Minimal AND/OR search graph:
closure under merge of the AND/OR search tree

    - Inconsistent sub-trees can be pruned too.
    - Some portions can be collapsed or reduced.

AND/OR Search Graph

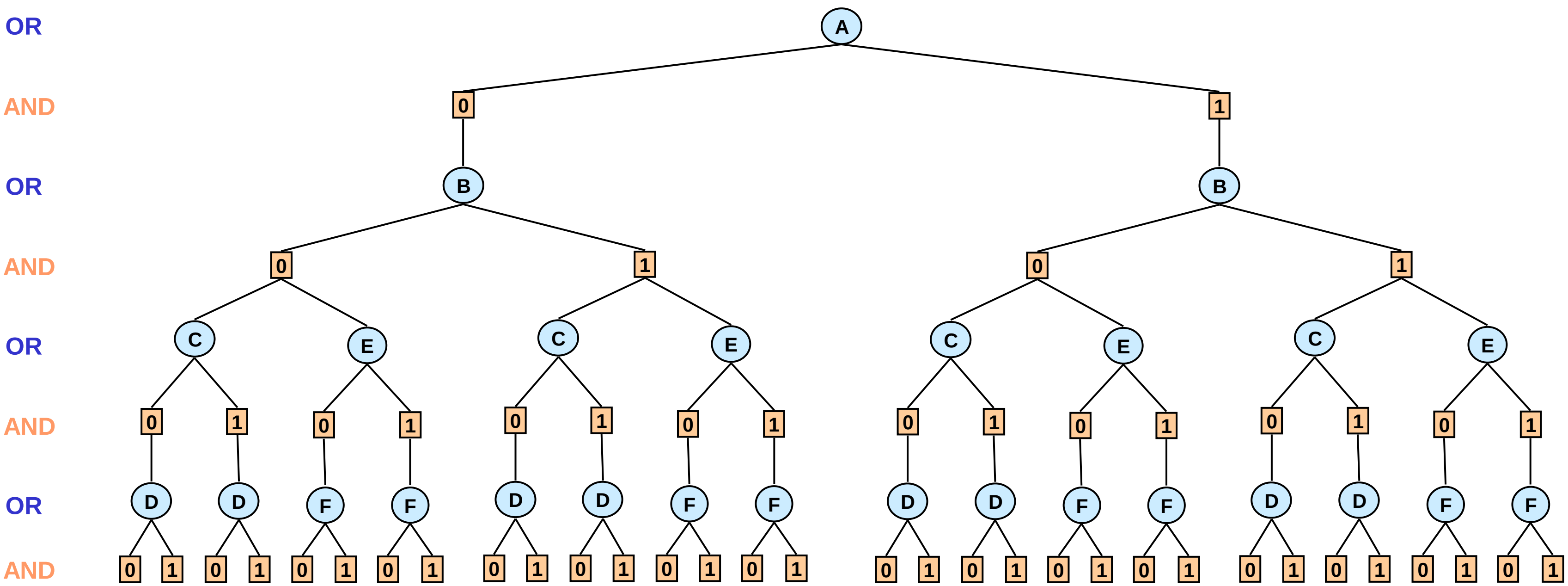


Primal graph

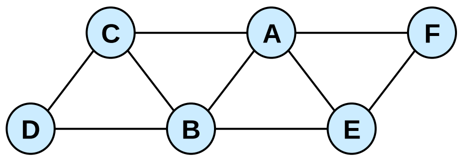


Pseudo-tree

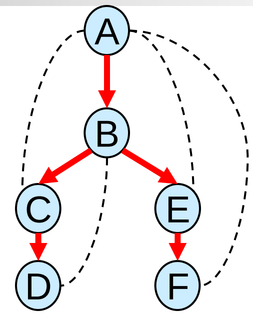
context(A) = {**A**}
context(B) = {**B,A**}
context(C) = {**C,B**}
context(D) = {**D**}
context(E) = {**E,A**}
context(F) = {**F**}



AND/OR Search Graph

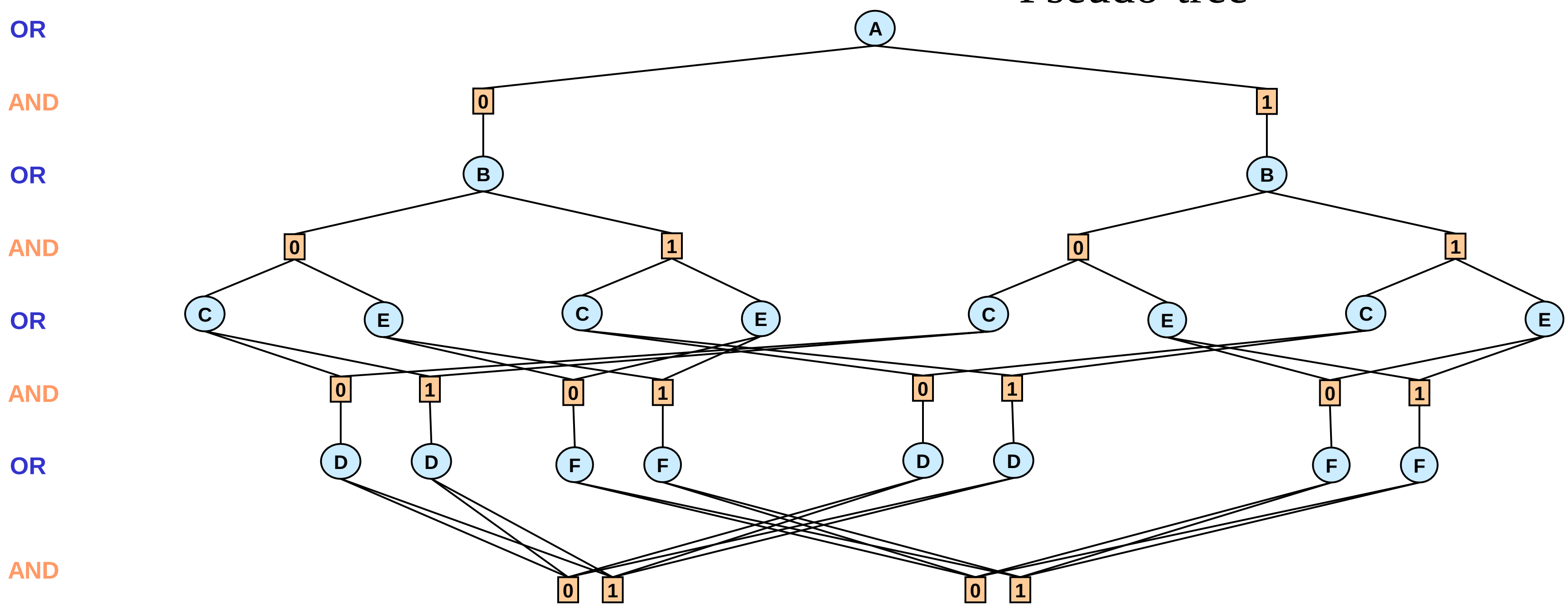


Primal graph

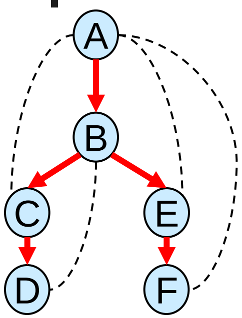


Pseudo-tree

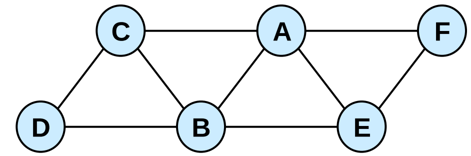
context(A) = {**A**}
context(B) = {**B,A**}
context(C) = {**C,B**}
context(D) = {**D**}
context(E) = {**E,A**}
context(F) = {**F**}



Context-based caching



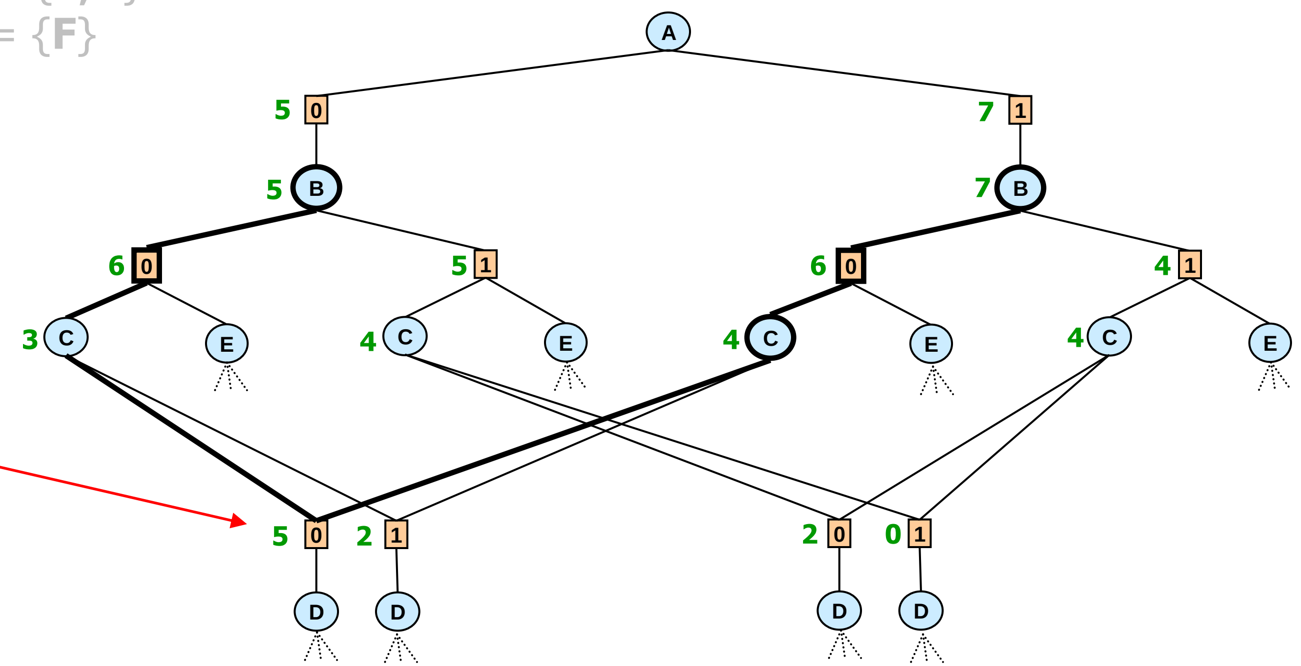
context(A) = {**A**}
context(B) = {**B,A**}
context(C) = {C,B}
context(D) = {**D**}
context(E) = {**E,A**}
context(F) = {**F**}



Primal graph

Cache Table (C)

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



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

Searching AND/OR Graphs

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

j=0 → j=w*

Space: O(n)

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

Space: O(exp w*)

Time: O(exp w*)

AO(j) time complexity?