



# Chapter 13

---

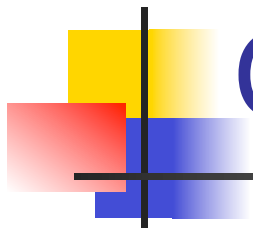
Constraint Optimization  
And counting, and enumeration  
275 class



# 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

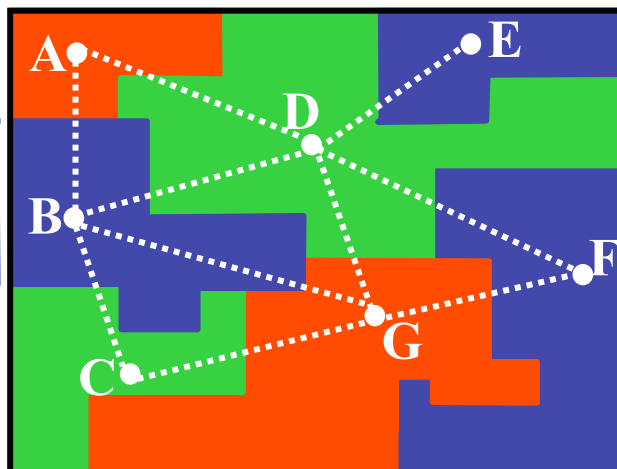
## Example: map coloring

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

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

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

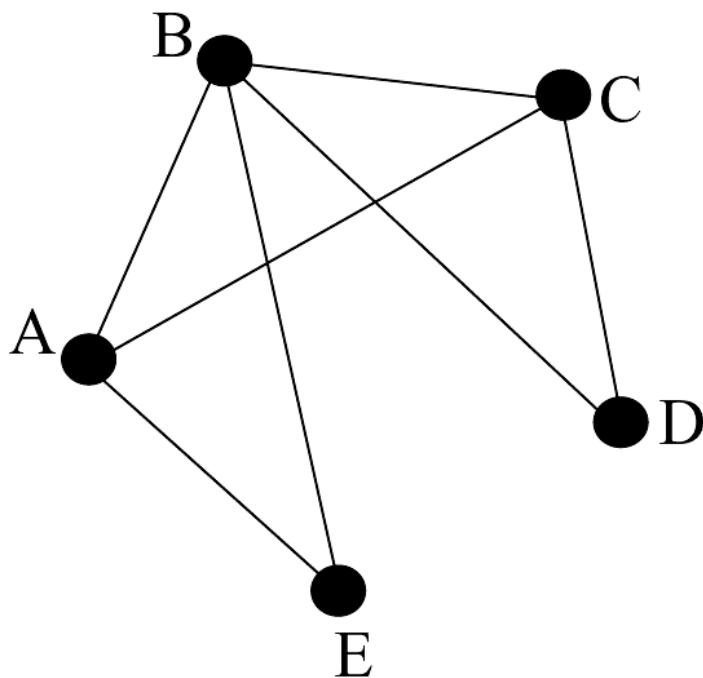
| A      | B      |
|--------|--------|
| red    | green  |
| red    | yellow |
| green  | red    |
| green  | yellow |
| yellow | green  |
| yellow | red    |



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

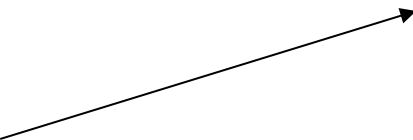
A *finite COP* is a triple  $R = \langle X, D, F \rangle$  where:

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

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

$F = \{f_1, \dots, f_m\}$  - 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:

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

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

$F = \{f_1, \dots, f_m\}$  - 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**

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

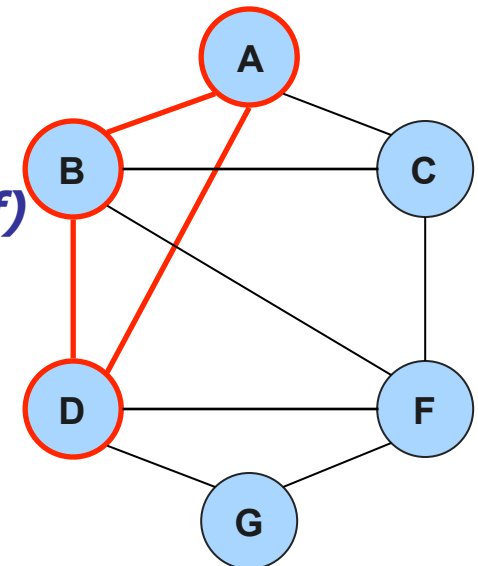
$f1(A,B,D)$

$f2(D,F,G)$

$f3(B,C,F)$

Global Cost Function

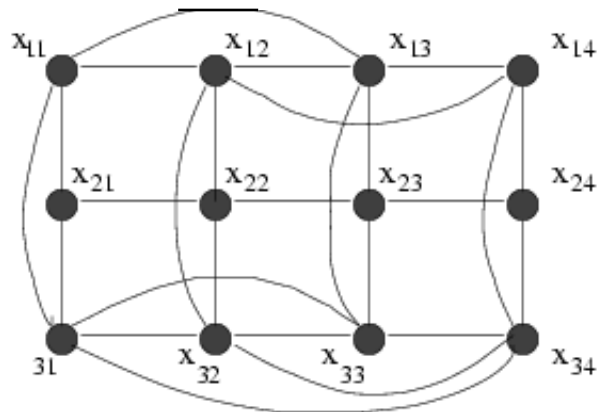
$$F(X) = \sum_{i=1}^m f_i(X)$$





# Constrained Optimization

## Example: power plant scheduling



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

Variables =  $\{X_1, \dots, X_n\}$ , domain =  $\{ON, OFF\}$ .

Constraints :  $X_1 \vee X_2, \neg X_3 \vee X_4$ , min - up and min - down time,

power demand :  $\sum \text{Power}(X_i) \geq \text{Demand}$

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

# Graphical Models

- A graphical model  $(\mathbf{X}, \mathbf{D}, \mathbf{F})$ :

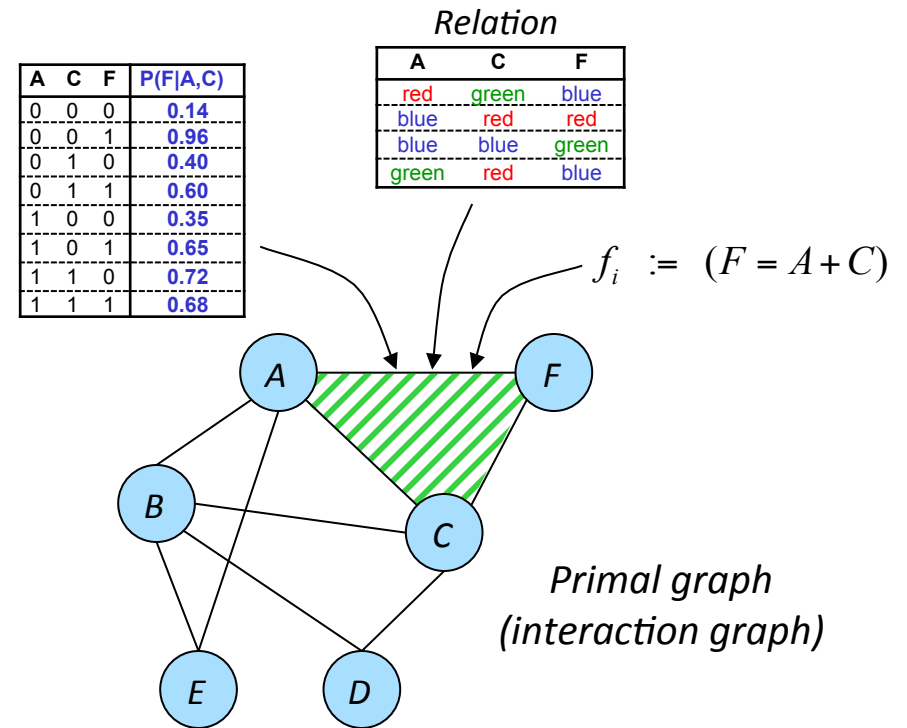
- $\mathbf{X} = \{X_1, \dots, X_n\}$  variables
- $\mathbf{D} = \{D_1, \dots, D_n\}$  domains
- $\mathbf{F} = \{f_1, \dots, f_m\}$  functions

- Operators:

- combination
- elimination (projection)

- Tasks:

- **Belief updating:**  $\sum_{x-y} \prod_j P_i$
- **MPE:**  $\max_x \prod_j P_j$
- **CSP:**  $\prod_x \times_j C_j$
- **Max-CSP:**  $\min_x \sum_j f_j$



- 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      |

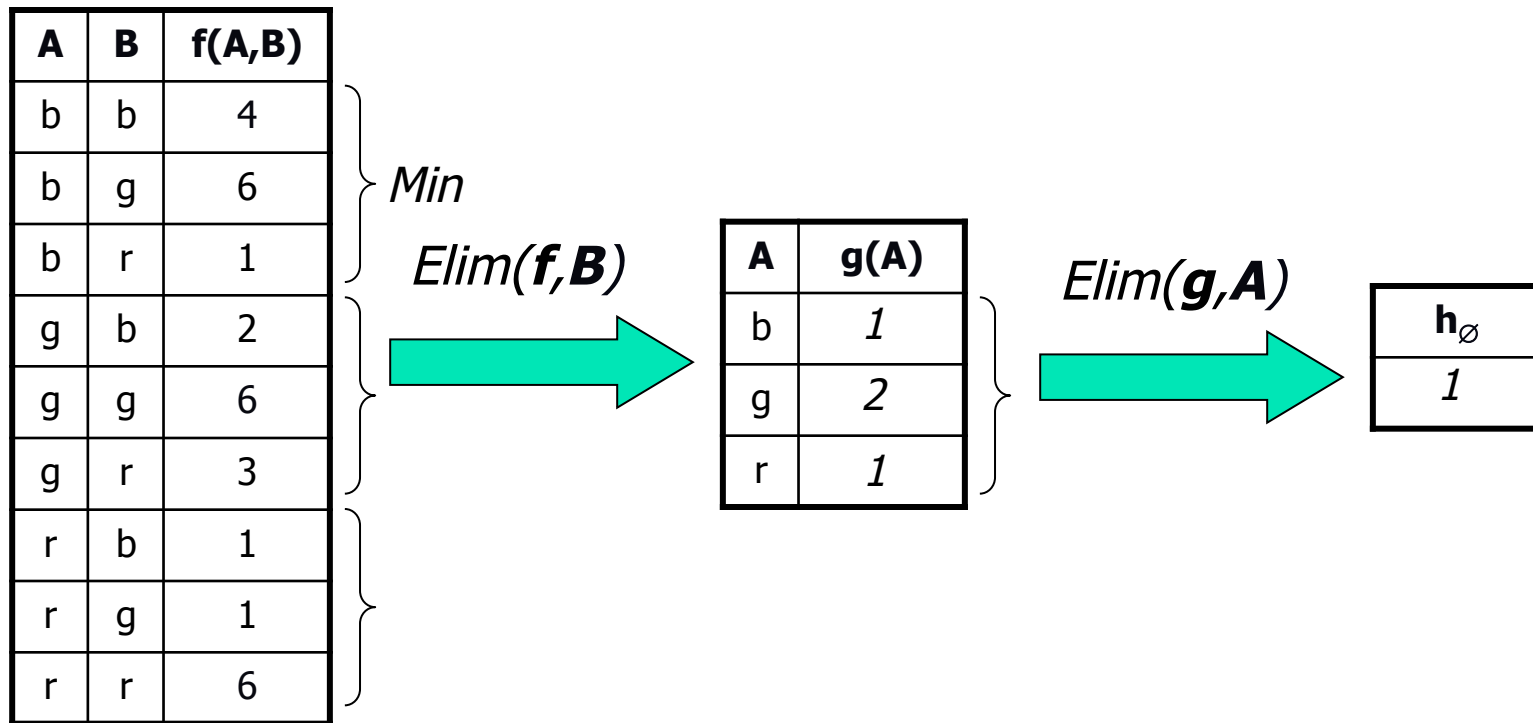
+

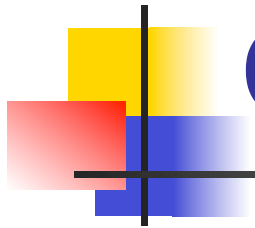
| B | C | f(B,C) |
|---|---|--------|
| b | b | 6      |
| b | g | 0      |
| g | b | 0      |
| g | g | 6      |

| 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

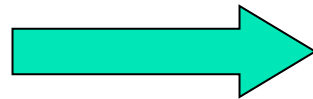




# Conditioning a Cost Function

| 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      |

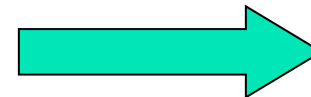
$Assign(\mathbf{f}_{AB}, A, b)$



$g(B)$

|   |
|---|
|   |
| 3 |

$Assign(\mathbf{g}, B, r)$



$h_{\emptyset}$

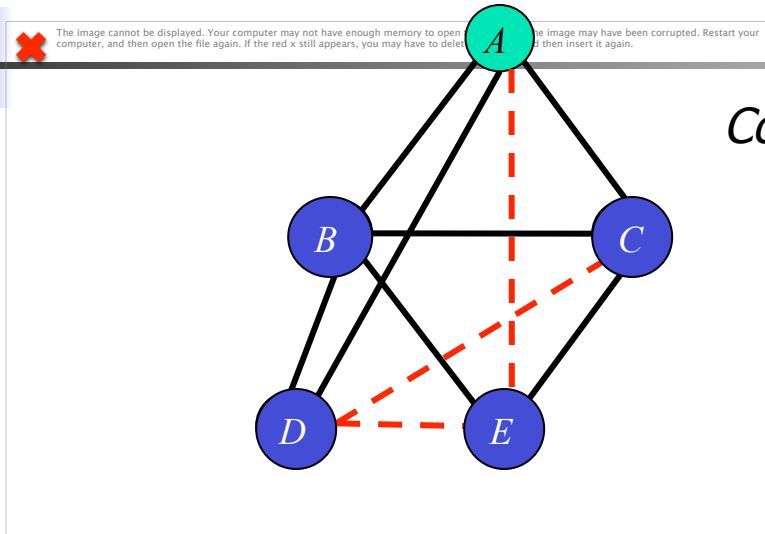


# Outline

---

- **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



*Constraint graph*

$$OPT = \min_{e=0,d,c,b} \underbrace{f(a,b) + f(a,c) + f(a,d)}_{\text{solid lines}} + \underbrace{f(b,c) + f(b,d) + f(b,e) + f(c,e)}_{\text{dashed lines}}$$

*Combination*

$$\min_{e=0} \min_d f(a,d) + \min_c f(a,c) + f(c,e) + \min_b \underbrace{f(a,b) + f(b,c) + f(b,d) + f(b,e)}_{\text{dashed lines}}$$

*Variable Elimination*

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

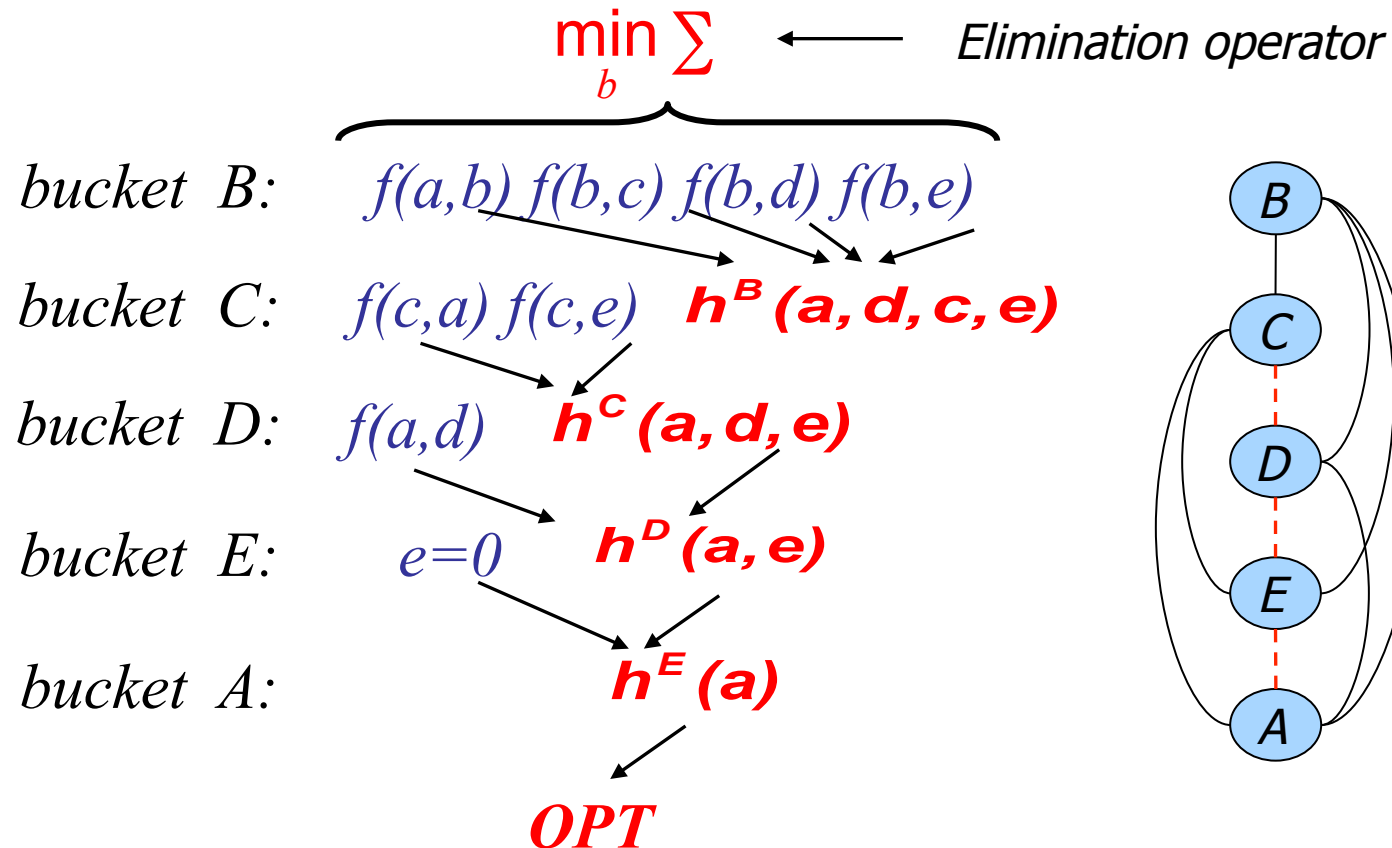
# Finding

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

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

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

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





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

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

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

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

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

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

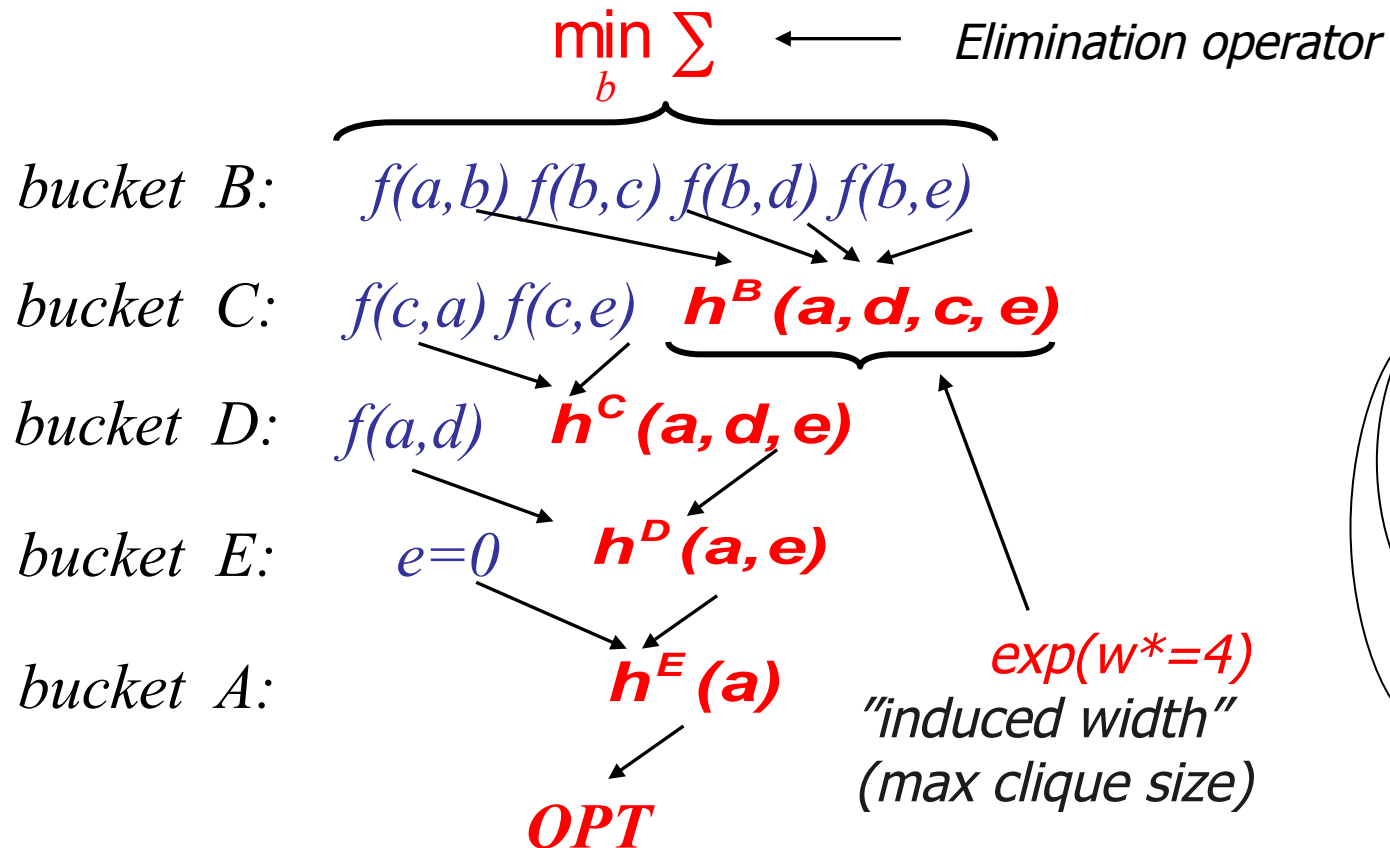
**Return  $(a', b', c', d', e')$**

# Complexity

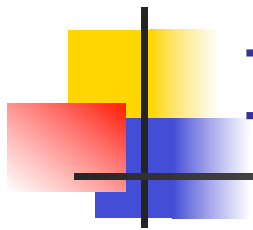
Algorithm **elim-opt** (Dechter, 1996)

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

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







# Induced Width

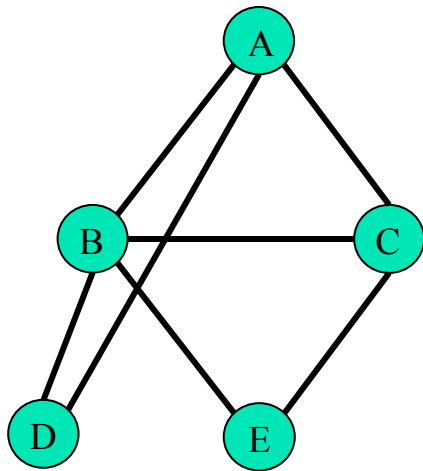
Bucket-elimination is **time** and **space**

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

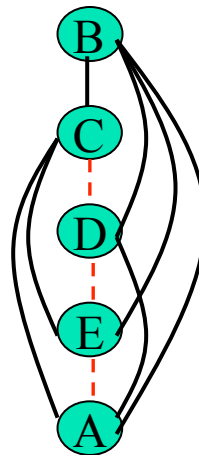
$w^*(d)$  – the induced width of the primal graph along ordering  $d$

$r$  = number of functions

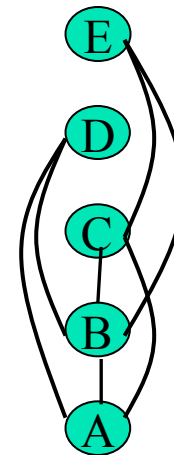
The effect of the ordering:



constraint graph



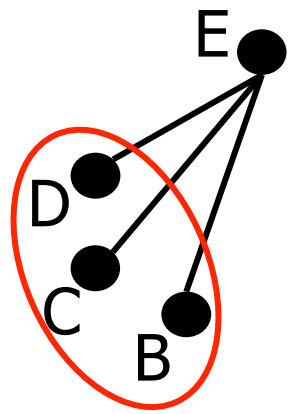
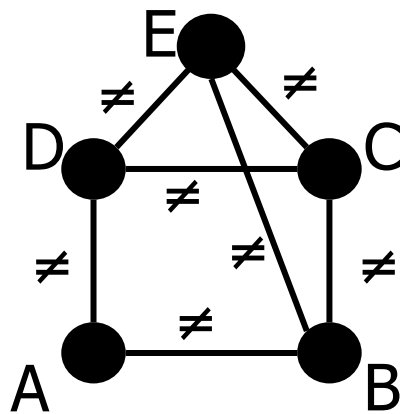
$$w^*(d_1) = 4$$



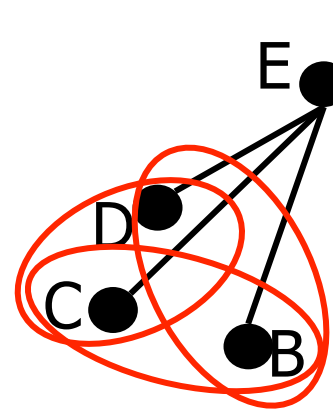
$$w^*(d_2) = 2$$

Finding smallest induced-width is hard

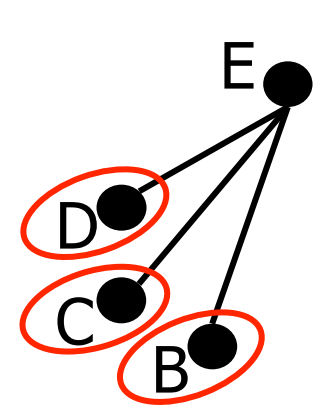
# Directional i-consistency



Adaptive



d-path



d-arc

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

D:  $D \neq C, D \neq A$

C:  $C \neq B$

B:  $A \neq B$

A:

$R_{DCB}$

$R_{DC}, R_{DB}$   
 $R_{CB}$

$R_D$   
 $R_C$   
 $R_B$



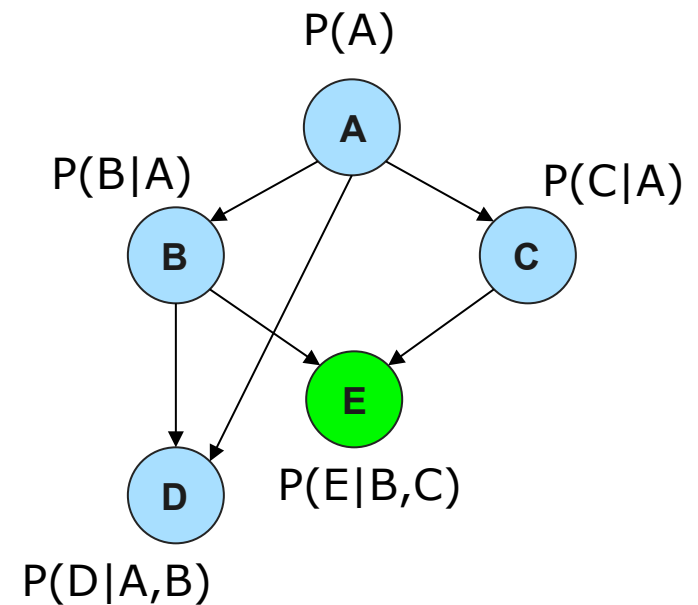
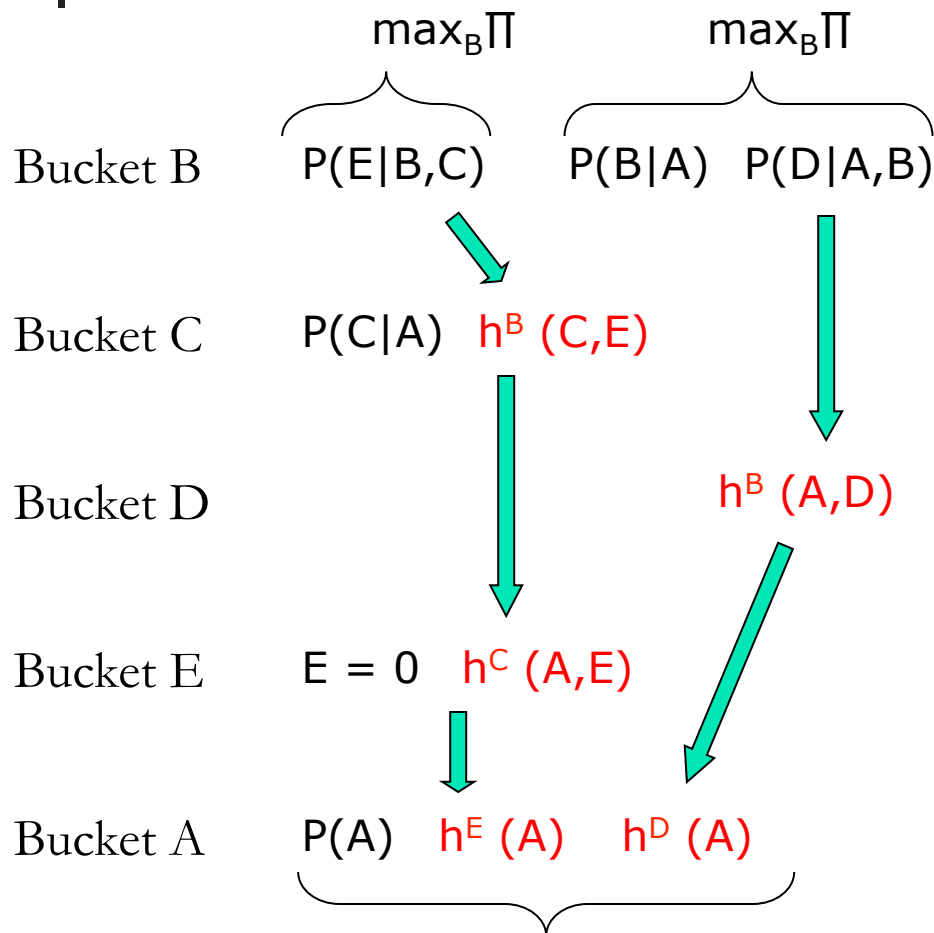
# Mini-bucket approximation

Split a bucket into mini-buckets => bound complexity

$$\begin{aligned} \text{bucket } (X) &= \{ \mathbf{h}_1, \dots, \mathbf{h}_r, \mathbf{h}_{r+1}, \dots, \mathbf{h}_n \} \\ &\quad \downarrow \quad \quad \quad \downarrow \\ &\quad \mathbf{h}^X = \max_X \prod_{i=1}^n h_i \\ &\quad \downarrow \quad \quad \quad \downarrow \\ &\quad \{ \mathbf{h}_1, \dots, \mathbf{h}_r \} \quad \quad \quad \{ \mathbf{h}_{r+1}, \dots, \mathbf{h}_n \} \\ &\quad \downarrow \quad \quad \quad \downarrow \\ &\quad \mathbf{g}^X = \left( \max_X \prod_{i=1}^r h_i \right) \cdot \left( \max_X \prod_{i=r+1}^n h_i \right) \\ &\quad \downarrow \\ &\quad \mathbf{h}^X \leq \mathbf{g}^X \end{aligned}$$

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

# Mini-Bucket Elimination



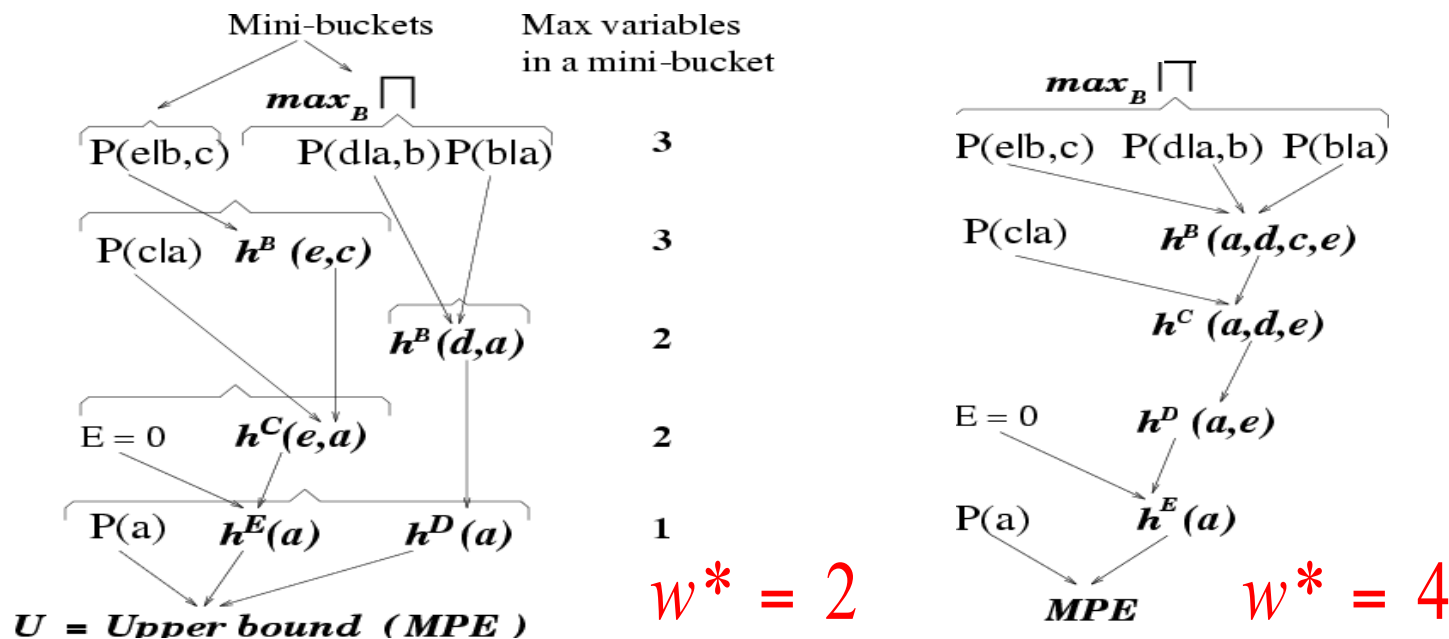
**MPE\* is an upper bound on MPE --U**  
**Generating a solution yields a lower bound--L**

# 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*





# 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



# Branch and Bound

**procedure** BRANCH-AND-BOUND

**Input:** A cost network  $\mathcal{C} = (X, D, C_h, C_s)$ ,  $L$  current upper-bound, An upper-bound function  $f$  defined for every partial solution.

**Output:** Either an optimal (maximal) solution, or notification that the network is inconsistent.

$i \leftarrow 1$  (initialize variable counter)

$D'_i \leftarrow D_i$  (copy domain)

While  $1 \leq i \leq n$

    instantiate  $x_i \leftarrow \text{SELECTVALUE}$

    If  $x_i$  is null (no value was returned)

$i \leftarrow i - 1$  (backtrack)

    Else

$i \leftarrow i + 1$  (step forward)

$D'_i \leftarrow D_i$

    Endwhile

    If  $i = 0$

        Return "inconsistent"

    Else

        Compute  $C = C(x_1, \dots, x_n)$ ,  $U \leftarrow \max\{C, L\}$

$i \leftarrow n - 1$

    end procedure

**procedure** SELECTVALUE

    If  $i = 0$  return  $U$  as the solution value and the most recent assignment as solution.

    While  $D'_i$  is not empty

        select an element  $a \in D'_i$  having max  $f(\vec{a}_{i-1}, a)$

        and remove  $a$  from  $D'_i$

        If  $\langle x_i, a \rangle$  is consistent with  $\vec{a}_{i-1}$  and  $f > L$ , then

        Return  $a$  (else prune  $a$ )

    Endwhile

    Return null (no consistent value)

end procedure

Figure 13.9: The Branch and Bound algorithm





# A Combinatorial Auction Example

**Example 13.1.2** Consider the combinatorial auction problem. A simple approach for modeling this problem as a *COP* is to associate each bid  $b_i$  with a variable having two values  $\{0, 1\}$ , where  $b_i = 1$  means that the bid is selected by the auctioneer. Otherwise, it is assigned  $b_i = 0$ . For every two bids that share an item there is a binary constraint prohibiting the assignment of 1 to both bid variables. Therefore, the variables are:  $b_1, \dots, b_r$  having domains  $\{0, 1\}$  and the constraints are:  $\forall i, j, \text{ if } b_i \text{ and } b_j \text{ share an item, there is a constraint } R_{ij}$ , such that,  $(b_i = 1, b_j = 1) \notin R_{ij}$ . The cost functions are: for every  $b_i$   $F(b_i) = r_i$  if  $b_i = 1$  and otherwise "0". The global cost function is  $C = \sum_i F_i(b_i)$ . The task is to find a consistent assignment to  $b_1, \dots, b_5$  having  $\max_{b_1, \dots, b_5} \sum_i F_i(b_i)$ .

Consider a problem instance given by the following bids:  $b_1 = \{1, 2, 3, 4\}$ ,  $b_2 = \{2, 3, 6\}$ ,  $b_3 = \{1, 5, 4\}$ ,  $b_4 = \{2, 8\}$ ,  $b_5 = \{5, 6\}$  and the costs  $r_1 = 8$ ,  $r_2 = 6$ ,  $r_3 = 5$ ,  $r_4 = 2$ ,  $r_5 = 2$ . In this case the variables are  $b_1, b_2, b_3, b_4, b_5$ , their domains are  $\{0, 1\}$  and the constraints are  $R_{12}, R_{13}, R_{14}, R_{24}, R_{25}, R_{35}$ . The cost network for this problem formulation is identical to its constraint network, since all the cost components are unary. The reader can verify that an optimal solution is given by  $b_1 = 0, b_2 = 1, b_3 = 1, b_4 = 0, b_5 = 0$ . Namely, selecting bids  $b_2$  and  $b_3$  is an optimal choice with total cost of 11.  $\square$

# Search tree for first-choice heuristic

$$f_{fc}(\vec{a}_i) = \sum_{F_j \in C} \max_{a_{i+1}, \dots, a_n} F_j(\vec{a}_n)$$

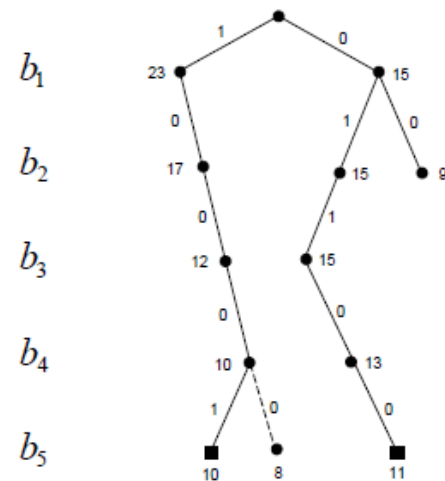


Figure 13.3: Branch and bound search space for the auction problem

**Example 13.2.1** Consider the auction problem described in Example 13.1.2. Searching for a solution in the order  $d = b_1, b_2, b_3, b_4, b_5$  yields the search space in Figure 13.3, traversed from left to right. The search space is highly constrained in this case. The evaluation bounding function at each node should *overestimate* its best extension for a solution. The first-choice bounding function for the root node (the empty assignment) is 23. The first solution encountered (selecting  $\langle b_1, 1 \rangle$  and  $\langle b_5, 1 \rangle$ ) has a cost of 10, which becomes the current global *lower bound*. The next solution encountered has the cost of 11 (when  $\langle b_2, 1 \rangle$  and  $\langle b_3, 1 \rangle$ ), while the rest of the variables are assigned 0). Subsequently, the partial assignment  $\langle b_1, 0 \rangle, \langle b_2, 0 \rangle$  is explored. Since  $f_{fc}(\langle b_1, 0 \rangle, \langle b_2, 0 \rangle) = 9$ , this upper bound is lower than 11, and therefore search can be pruned.  $\square$

# Bucket-Elimination for the Auction Example

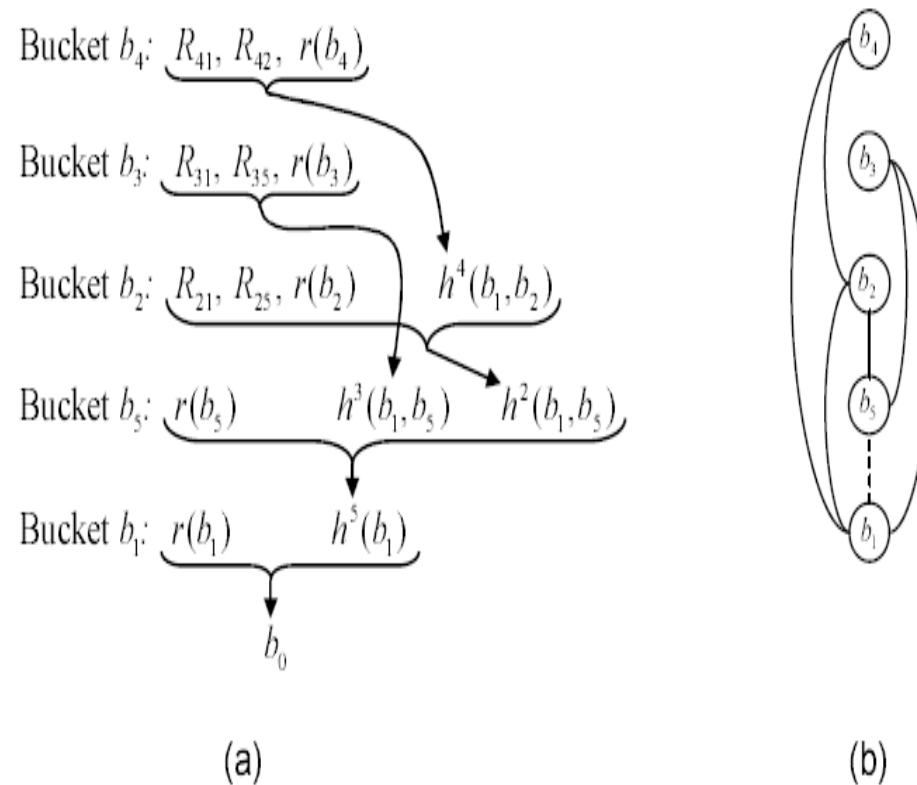
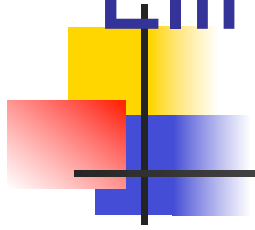


Figure 13.8: Schematic execution of elim-opt on the auction problem

# Elim-OPT for auction problem



Bucket  $b_4$ :  $\{R_{41}, R_{42}, r(b_4)\}$

Bucket  $b_3$ :  $\{R_{31}, R_{35}, r(b_3)\}$

Bucket  $b_2$ :  $\{R_{21}, R_{25}, r(b_2)\}$

Bucket  $b_5$ :  $\{r(b_5)\}$

Bucket  $b_1$ :  $\{r(b_1)\}$

$b_0$

(a)

Bucket  $b_4$ . In this bucket we compute  $h^4(b_1, b_2) = \max_{\{(b_4 | (b_1, b_2, b_4) \in R_{41} \bowtie R_{42}\}} r(b_4)$ ,

$$h^4(b_1, b_2) = \begin{cases} 0 & \text{if } b_1 = 1, \text{ or } b_2 = 1 \\ 2 & \text{if } b_1 = 0, b_2 = 0 \end{cases}$$

Bucket  $b_3$  we compute  $h^3(b_1, b_5) = \max_{\{(b_3 | (b_1, b_3, b_5) \in R_{31} \bowtie R_{35}\}} r(b_3)$ , yielding:

$$h^3(b_1, b_5) = \begin{cases} 0 & \text{if } b_1 = 1, \text{ or } b_5 = 1 \\ 5 & \text{if } b_1 = 0, b_5 = 0 \end{cases}$$

Bucket  $b_2$ , which now includes a new function, gives us:  
 $\max_{\{(b_2 | (b_1, b_2, b_5) \in R_{21} \bowtie R_{25}\}} (r(b_2) + h^4(b_1, b_2))$ , yielding:

$$h^2(b_1, b_5) = \begin{cases} 0 & \text{if } b_1 = 1, b_5 = 1 \\ 0 & \text{if } b_1 = 1, b_5 = 0 \\ 2 & \text{if } b_1 = 0, b_5 = 1 \\ 6 & \text{if } b_1 = 0, b_5 = 0 \end{cases}$$



# Example MBE-opt

**Example 13.4.3** Let us apply algorithm mbe-opt(2) to the auction problem along the ordering  $d = b_1, b_5, b_2, b_3, b_4$ . Figure 13.12 shows the resulting mini-buckets; square brackets denote the choice for partitioning.

We start with processing bucket  $b_4$ . A possible partitioning places the constraint  $R_{41}$  in one mini-bucket and the rest in the other. We compute a constraint  $R^4(b_1) = \pi_{b_1} R_{41}$ , which is the universal constraint so it need not be recorded. In the second mini-bucket, we also compute  $h^4(b_2) = \max_{\{(b_4|(b_4, b_2) \in R_{42}\}} r(b_4)$ , yielding:

$$h^4(b_2) = \begin{cases} 0 & \text{if } b_2 = 1 \\ 2 & \text{if } b_2 = 0 \end{cases}$$

Processing the first mini-bucket of  $b_3$ , which includes only hard constraints, will also not effect the domain of  $b_1$ . Processing the second mini-bucket of  $b_3$  by  $h^3(b_5) = \max_{\{(b_3|(b_3, b_5) \in R_{35}\}} r(b_3)$ , yields:

$$h^3(b_5) = \begin{cases} 0 & \text{if } b_5 = 1 \\ 5 & \text{if } b_5 = 0 \end{cases}$$

Processing the second mini-bucket of  $b_2$  (the first mini-bucket includes a constraint whose projection is a universal constraint) by  $h^2(b_5) = \max_{\{(b_2|(b_2, b_5) \in R_{25}\}} (r(b_2) + h^4(b_2))$ , gives us:

$$h^2(b_5) = \begin{cases} 2 & \text{if } b_5 = 1 \\ 6 & \text{if } b_5 = 0 \end{cases}$$

Processing bucket  $b_5$  (with full buckets now) the algorithm computes  $h^5 = \max_{b_5} (r(b_5) + h^2(b_5) + h^3(b_5))$ , yielding:

$$h^5 = 11$$

Finally, in the bucket of  $b_1$  the algorithm computes  $h^1 = \max_{b_1} (r(b_1) + h^5)$ , yielding:  $h^1 = \max\{0 + 11, 8 + 11\} = 19$ .

The maximal upper-bound cost is therefore 19. To compute a maximizing tuple we select in bucket  $b_1$  the value  $b_1 = 1$ , which maximizes  $r(b_1) + h^5$ . Given  $b_1 = 1$ , we choose  $b_5 = 0$ , which maximizes the functions in bucket  $b_5$ . Then, in bucket  $b_2$ , we can choose only  $b_2 = 0$ , due to the constraint  $R_{12}$ . Likewise, the only subsequent choices are  $b_3 = 0$  and  $b_4 = 0$ . Therefore, the cost of the generated solution is 8, yielding the interval  $[8, 19]$  bounding the maximal solution.  $\square$

Bucket  $b_4$  :  $[R_{41}], [R_{42}, r(b_4)]$

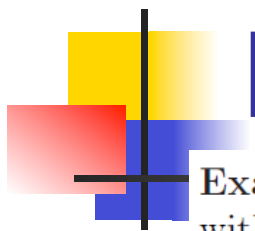
Bucket  $b_3$  :  $[R_{31}], [R_{35}, r(b_3)]$

Bucket  $b_2$  :  $[R_{21}], [R_{25}, r(b_2) \parallel h^4(b_2)]$

Bucket  $b_5$  :  $[r(b_5) \parallel h^3(b_5), h^2(b_5)]$

Bucket  $b_1$  :  $r(b_1) \parallel h^5$

Yielding: opt:  $h^1 = M'$



# BnB with first-cut heuristic

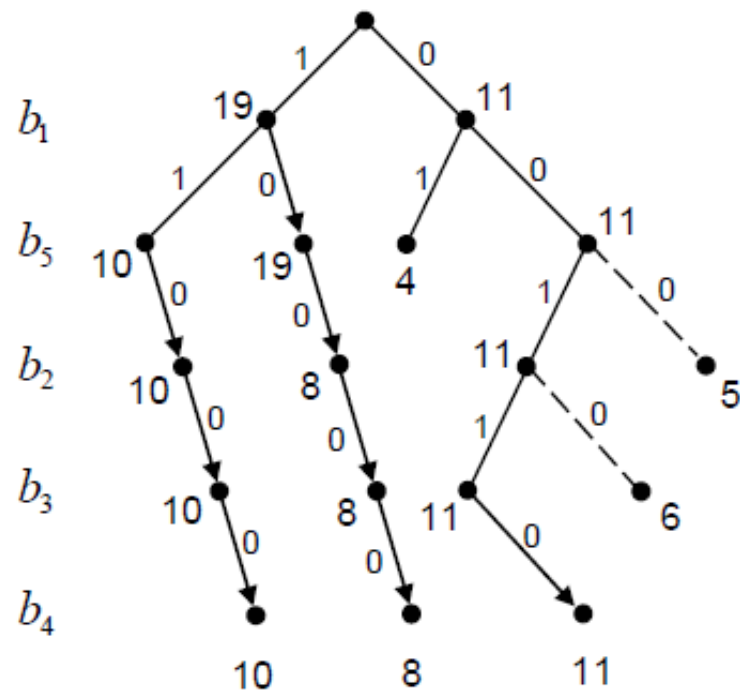
**Example 13.5.4** We next demonstrate branch and bound search on the auction problem with two bounding functions. The first bounding function is first-cut  $f_{fc}$ , and the second is computed by  $\text{mbe-opt}(2)$ , denoted  $f_{mb}$ . Figure 13.14 shows the search space with the two bounding functions searching along ordering  $b_1, b_5, b_2, b_3, b_4$ . We start with  $f_{fc}$ .

Starting with  $b_1$  we get  $f_{fc}(b_1) = r(b_1) + 15$ . Thus,  $f_{fc}(b_1 = 0) = 15$  and  $f_{fc}(b_1 = 1) = 8 + 15 = 23$ . Therefore  $b_1 = 1$  is selected first. Next, the algorithm decides a value for  $b_5$ . We get  $f_{fc}(b_1 = 1, b_5 = 0) = 8 + 0 + 13 = 21$ , while  $f_{fc}(b_1 = 1, b_5 = 1) = 8 + 2 + 13 = 23$ , consequently  $b_5 = 1$  is selected. For the rest of the variables the constraints dictates  $b_2 = 0, b_3 = 0, b_4 = 0$ , yielding a solution with cost of 10 and a current lower bound of

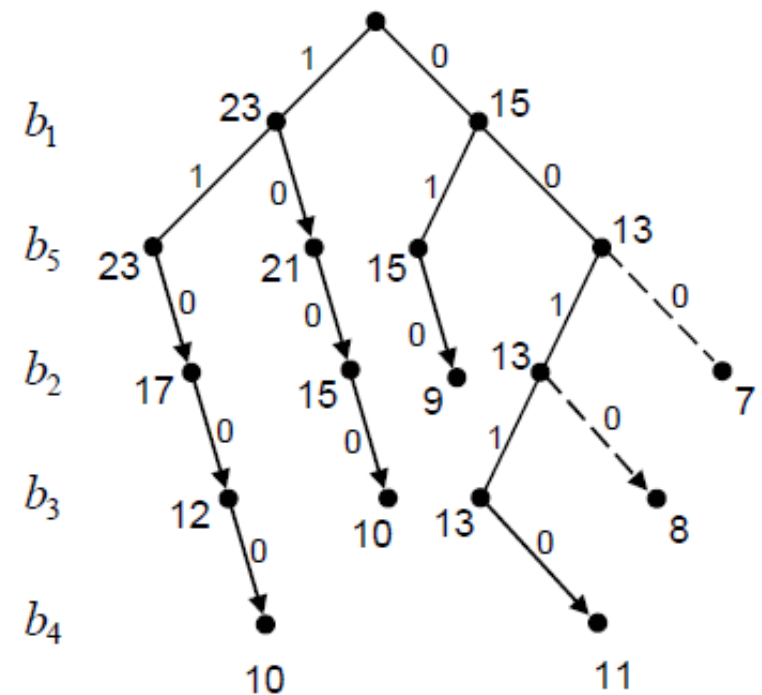
$L = 10$ . The  $f_{fc}$  values are denoted next to each node. We backtrack to the closest choice node and choose the only alternative  $b_5 = 0$ , whose value is 21. Then,  $b_2 = 0$  is the only choice due to the constraint with  $b_1 = 1$ . Since  $f_{fc}(b_1 = 1, b_5 = 0, b_2 = 0) = 8 + 7 = 15$ , pruning will not occur. The only choice for  $b_3$  is 0, with the value  $8 + 2 = 10$ . Since the upper bound for this node equals the lower-bound, we prune. BnB backtracks to the next choice point of  $b_1 = 0$ , whose bound value is  $(0 + 15 = 15)$ , and so on. The next variable is  $b_5$ . Since  $f_{fc}(b_1 = 0, b_5 = 1) = 15$ , and  $f_{fc}(b_1 = 0, b_5 = 0) = 0 + 13 = 13$ ,  $b_5 = 1$  is selected. Then, the only value possible for  $b_2$  is “0” with :  $f_{fc}(b_1 = 0, b_5 = 1, b_2 = 0) = 9$ . Since this upper bound is lower than the global lower bound, 10, pruning occurs and the algorithm backtracks to  $b_5 = 0$ . The next variable is  $b_2$ . We get  $f_{fc}(b_1 = 0, b_5 = 0, b_2 = 1) = 13$  while  $f_{fc}(b_1 = 0, b_5 = 0, b_2 = 0) = 7$ . We select  $b_2 = 1$ . Subsequently, for  $b_3$  we have the choice of  $f_{fc}(b_1 = 0, b_5 = 0, b_2 = 1, b_3 = 1) = 13$ , or  $f_{fc}(b_1 = 0, b_5 = 0, b_2 = 1, b_3 = 0) = 8$ . We select  $b_3 = 1$  and are left with  $b_4 = 0$ , with cost of  $f(b_1 = 0, b_5 = 0, b_2 = 1, b_3 = 1, b_4 = 0) = 11$ . We update the global lower bound to  $L = 11$ . Backtracking to the next choice point, the algorithm tries  $b_3 = 0$  but its function evaluates to “8,” causing another backtracking to  $b_2 = 0$  with bounding value 7, causing backtracking to the root.



# BnB with first-cut (b) and mini-bucket (a) heuristics



(a)



(b)



# Search with MB heuristic

Let us now apply BnB with  $f_{mb}$ , which is the bounding function extracted from `mbe-opt(2)`. Based on the functions in the augmented bucket of  $b_1$  produced by `mbe-opt(2)`,  $f_{mb}(b_1 = 0) = r(b_1) + h^5 = 8 + 11 = 19$  while  $f_{mb}(b_1 = 1) = 0 + 11 = 11$ . Consequently,  $b_1 = 1$  is chosen. We next evaluate  $f_{mb}(b_1 = 1, b_5) = 8 + h^3(b_5) + h^2(b_5)$ , yielding  $f_{mb}(b_5 = 0) = 19$  and  $f_{mb}(b_5 = 1) = 10$ . Consequently,  $b_5 = 0$  is chosen. The path is now deterministic, allowing the only choices:  $b_2 = b_3 = b_4 = 0$ . We end up with a solution having a cost of 8, which becomes the first global lower bound  $L = 8$ . *BnB* backtracks. The path is deterministic, dictating the choices ( $b_2 = b_3 = b_4 = 0$ ) whose bounding cost equals 10, yielding a solution with cost 10 ( $b_1 = 1, b_5 = 1, b_2 = 0, b_3 = 0, b_4 = 0$ ). The global lower bound  $L$  is updated to 10. The algorithm backtracks to the next choice point, which is ( $b_1 = 0$ ), whose bounding cost is 11. Next, for  $b_5$ ,  $f_{mb}(b_1 = 0, b_5) = 0 + r(b_5) + h^2(b_5) + h^3(b_5)$ , yielding  $f_{mb}(b_5 = 1) = 4$ , which can be immediately pruned (less than 10), and  $f_{mb}(b_5 = 1) = 11$ . We select  $b_5 = 0$ . Subsequently, choosing a value for  $b_2$  we get:  $f_{mb}(b_1 = 0, b_5 = 0, b_2) = r(b_2) + h^4(b_2) + h^3(b_5)$  (note that  $h^2(b_5)$  is not included since it is created in bucket  $b_2$ ). This yields  $f_{mb}(b_2 = 1) = 11$ , while  $f_{mb}(b_2 = 0) = 5$ , which is pruned. The next choice for  $b_3$  is determined using the bounding function  $f_{mb}(b_1 = 0, b_5 = 0, b_2 = 1, b_3) = 6 + r(b_3) + h^4(b_2 = 1)$ , yielding for  $b_3 = 1$  the bound 11, while for  $b_3 = 0$  the bound 6, which will be pruned.  $b_3 = 1$  is selected. Subsequently, only  $b_4 = 0$  is feasible and we get the best cost solution of value 11. The global lower bound,  $L$  is updated to 11 and *BnB* will lead to only pruned choices.

We see in this example that the performance of *BnB* using these two bounding func-





# Bucket-elimination for counting

## Algorithm elim-count

**Input:** A constraint network  $\mathcal{R} = (X, D, C)$ , ordering  $d$ .

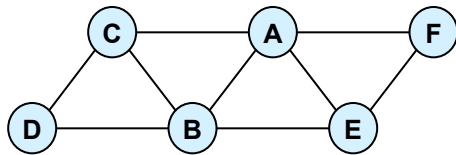
**Output:** Augmented output buckets including the intermediate count functions and The number of solutions.

1. **Initialize:** Partition  $C$  (0-1 cost functions) into ordered buckets  $bucket_1, \dots, bucket_n$ ,  
We denote a function in a bucket  $N_i$ , and its scope  $S_i$ .)
2. **Backward:** For  $p \leftarrow n$  downto 1, do  
Generate the function  $N^p$ :  $N^p = \sum_{X_p} \prod_{N_i \in bucket_p} N_i$ .  
Add  $N^p$  to the bucket of the latest variable in  $\bigcup_{i=1}^p S_i - \{X_p\}$ .
3. **Return** the number of solutions,  $N^1$  and the set of output buckets with the original and computed functions.

Figure 13.9: Algorithm *elim-count*

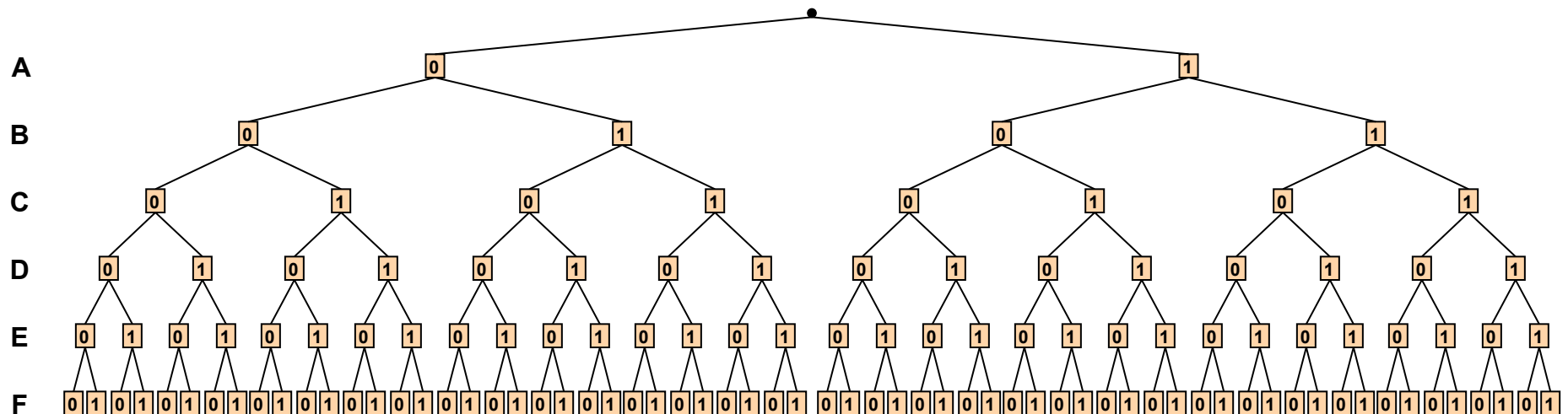


# The Search Space

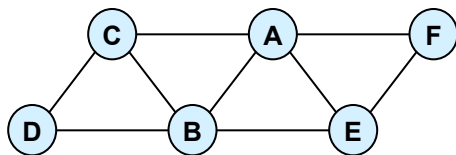


| A B $f_1$ | A C $f_2$ | A E $f_3$ | A F $f_4$ | B C $f_5$ | B D $f_6$ | B E $f_7$ | C D $f_8$ | E F $f_9$ |
|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| 0 0 2     | 0 0 3     | 0 0 0     | 0 0 2     | 0 0 0     | 0 0 4     | 0 0 3     | 0 0 1     | 0 0 1     |
| 0 1 0     | 0 1 0     | 0 1 3     | 0 1 0     | 0 1 1     | 0 1 2     | 0 1 2     | 0 1 4     | 0 1 0     |
| 1 0 1     | 1 0 0     | 1 0 2     | 1 0 0     | 1 0 2     | 1 0 1     | 1 0 1     | 1 0 0     | 1 0 0     |
| 1 1 4     | 1 1 1     | 1 1 0     | 1 1 2     | 1 1 4     | 1 1 0     | 1 1 0     | 1 1 0     | 1 1 2     |

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

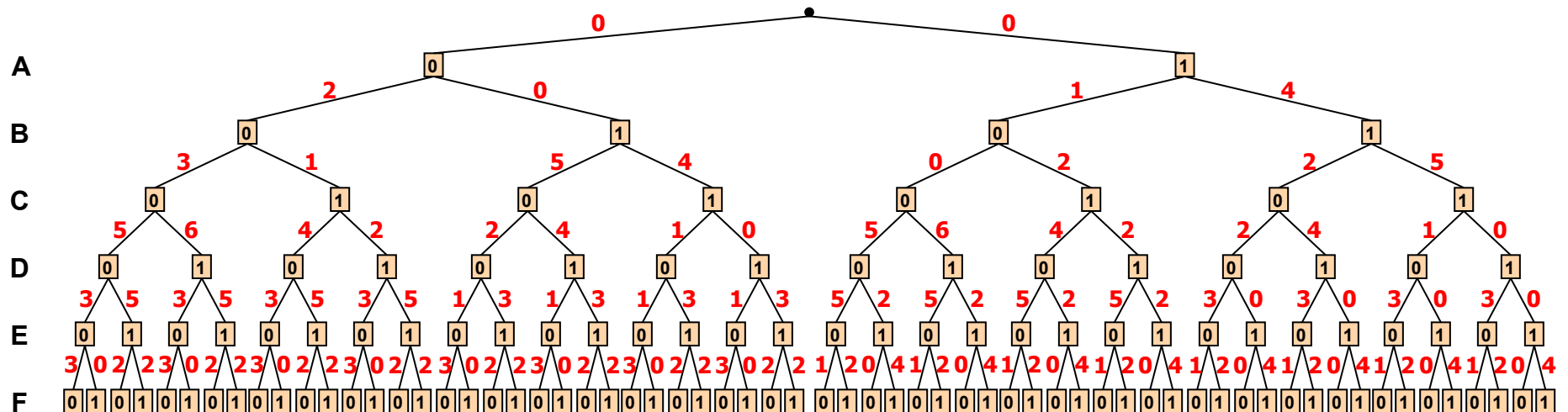


# The Search Space



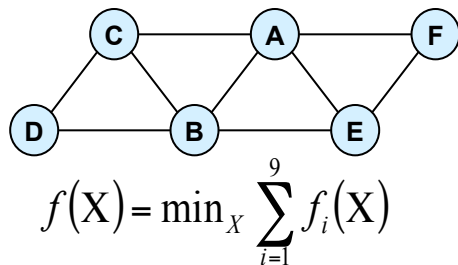
| A | B | $f_1$ | A | C | $f_2$ | A | E | $f_3$ | A | F | $f_4$ | B | C | $f_5$ | B | D | $f_6$ | B | E | $f_7$ | C | D | $f_8$ | E | F | $f_9$ |
|---|---|-------|---|---|-------|---|---|-------|---|---|-------|---|---|-------|---|---|-------|---|---|-------|---|---|-------|---|---|-------|
| 0 | 0 | 2     | 0 | 0 | 3     | 0 | 0 | 0     | 0 | 0 | 2     | 0 | 0 | 0     | 0 | 0 | 4     | 0 | 0 | 3     | 0 | 0 | 1     | 0 | 0 | 1     |
| 0 | 1 | 0     | 0 | 1 | 0     | 0 | 1 | 3     | 0 | 1 | 0     | 0 | 1 | 1     | 0 | 1 | 2     | 0 | 1 | 2     | 0 | 1 | 4     | 0 | 1 | 0     |
| 1 | 0 | 1     | 1 | 0 | 0     | 1 | 0 | 2     | 1 | 0 | 0     | 1 | 0 | 2     | 1 | 0 | 1     | 1 | 0 | 1     | 1 | 0 | 0     | 1 | 0 | 0     |
| 1 | 1 | 4     | 1 | 1 | 1     | 1 | 1 | 0     | 1 | 1 | 2     | 1 | 1 | 4     | 1 | 1 | 0     | 1 | 1 | 0     | 1 | 1 | 0     | 1 | 1 | 2     |

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

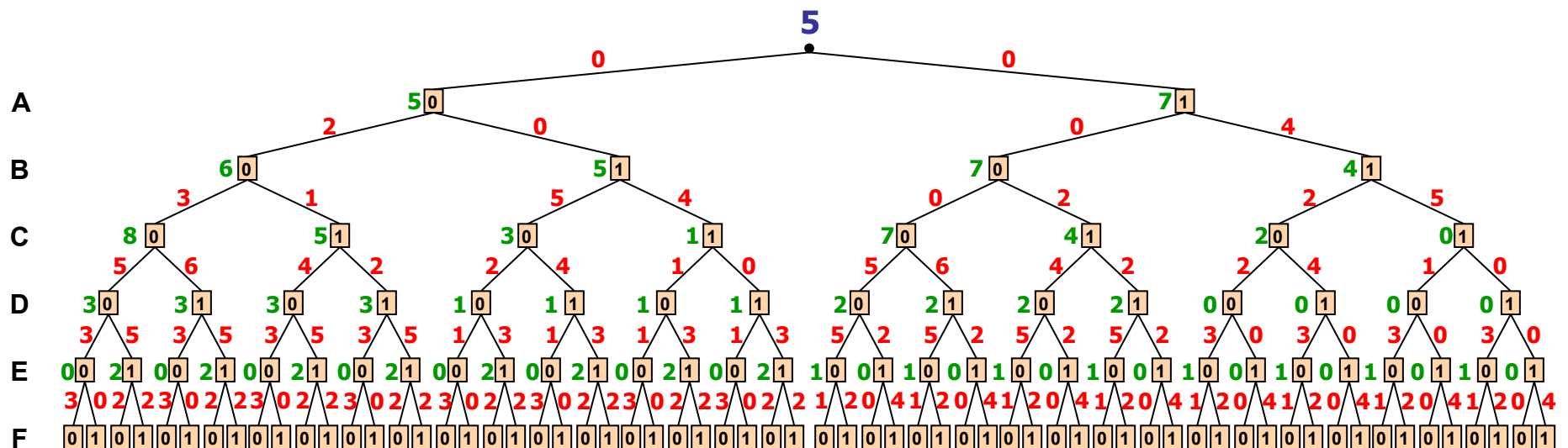


**Arc-cost is calculated based on cost components.**

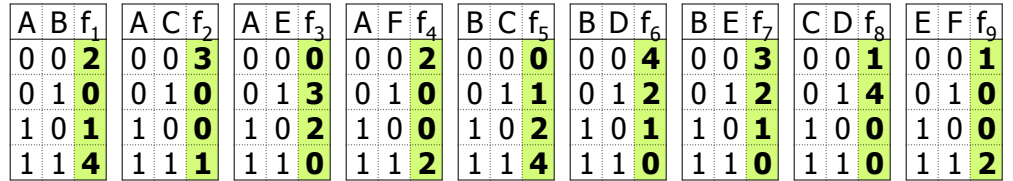
# The Value Function

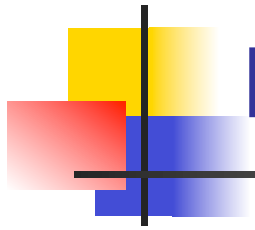


| A | B | $f_1$ | A | C | $f_2$ | A | E | $f_3$ | A | F | $f_4$ | B | C | $f_5$ | B | D | $f_6$ | B | E | $f_7$ | C | D | $f_8$ | E | F | $f_9$ |
|---|---|-------|---|---|-------|---|---|-------|---|---|-------|---|---|-------|---|---|-------|---|---|-------|---|---|-------|---|---|-------|
| 0 | 0 | 2     | 0 | 0 | 3     | 0 | 0 | 0     | 0 | 0 | 2     | 0 | 0 | 0     | 0 | 0 | 4     | 0 | 0 | 3     | 0 | 0 | 1     | 0 | 0 | 1     |
| 0 | 1 | 0     | 0 | 1 | 0     | 0 | 1 | 3     | 0 | 1 | 0     | 0 | 1 | 1     | 0 | 1 | 2     | 0 | 1 | 2     | 0 | 1 | 4     | 0 | 1 | 0     |
| 1 | 0 | 1     | 1 | 0 | 0     | 1 | 0 | 2     | 1 | 0 | 0     | 1 | 0 | 2     | 1 | 0 | 1     | 1 | 0 | 1     | 1 | 0 | 0     | 1 | 0 | 0     |
| 1 | 1 | 4     | 1 | 1 | 1     | 1 | 1 | 0     | 1 | 1 | 2     | 1 | 1 | 4     | 1 | 1 | 0     | 1 | 1 | 0     | 1 | 1 | 0     | 1 | 1 | 2     |



**Value** of node = minimal cost solution below it





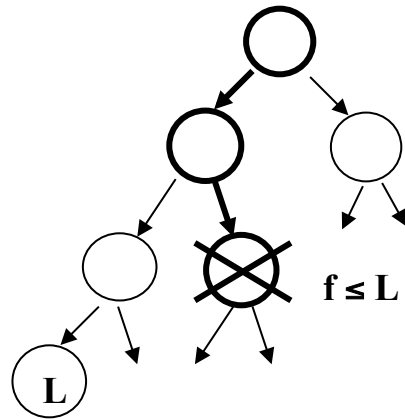
# Basic Heuristic Search Schemes

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.

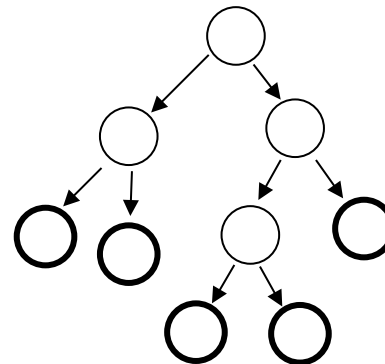
Linear space



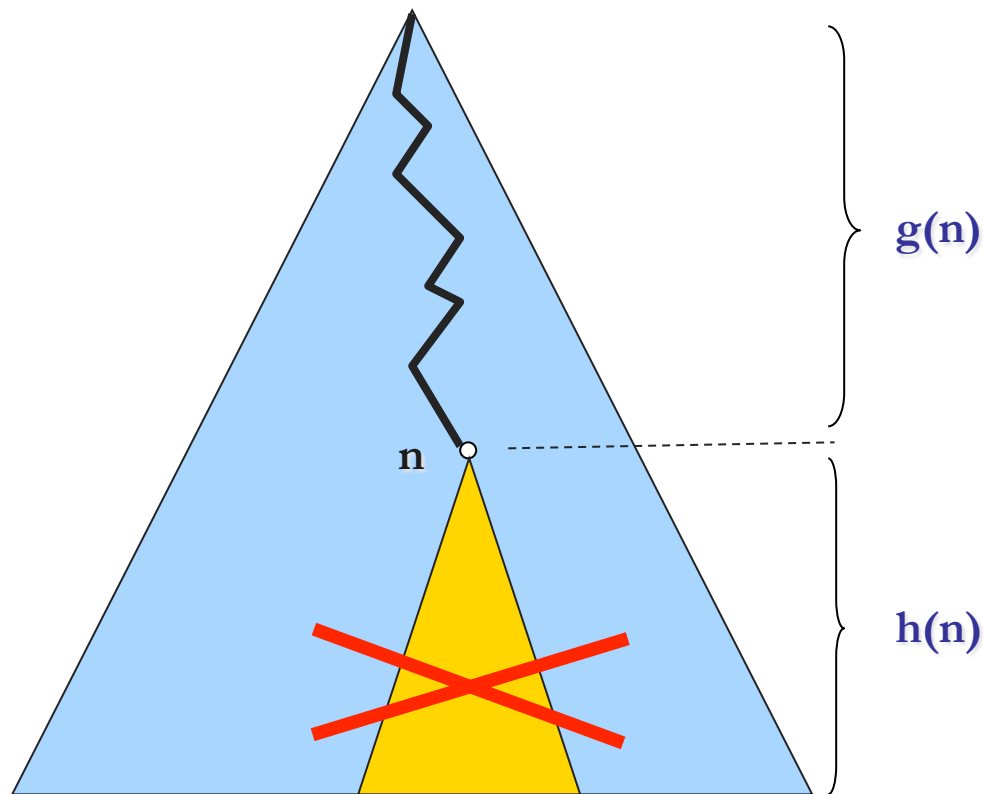
## 2.Best-First Search

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

Needs lots of memory



# Classic Branch-and-Bound



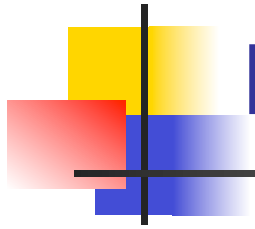
OR Search Tree

Upper Bound **UB**

Lower Bound **LB**

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

**Prune if  $LB(n) \geq UB$**



# How to Generate Heuristics

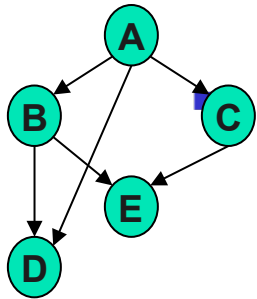
---

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



# Static MBE Heuristics

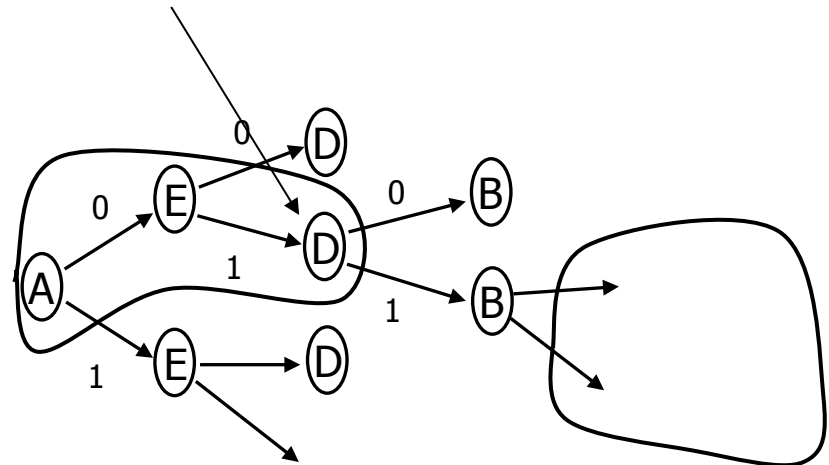
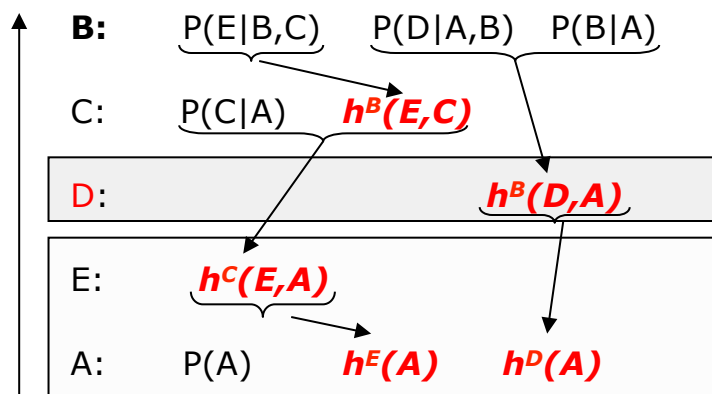
- Given a partial assignment  $x^p$ , estimate the cost of the best extension to a full solution



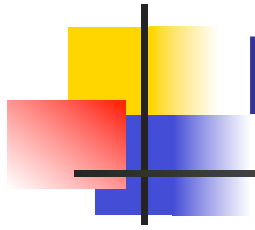
The evaluation function  $f(x^p)$  can be computed using function recorded by the Mini-Bucket scheme

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

Belief Network



$$f(a,e,D) = \underbrace{P(a)}_g + \underbrace{h^B(D,a) + h^C(e,a)}_{h - \text{is admissible}}$$



# 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:

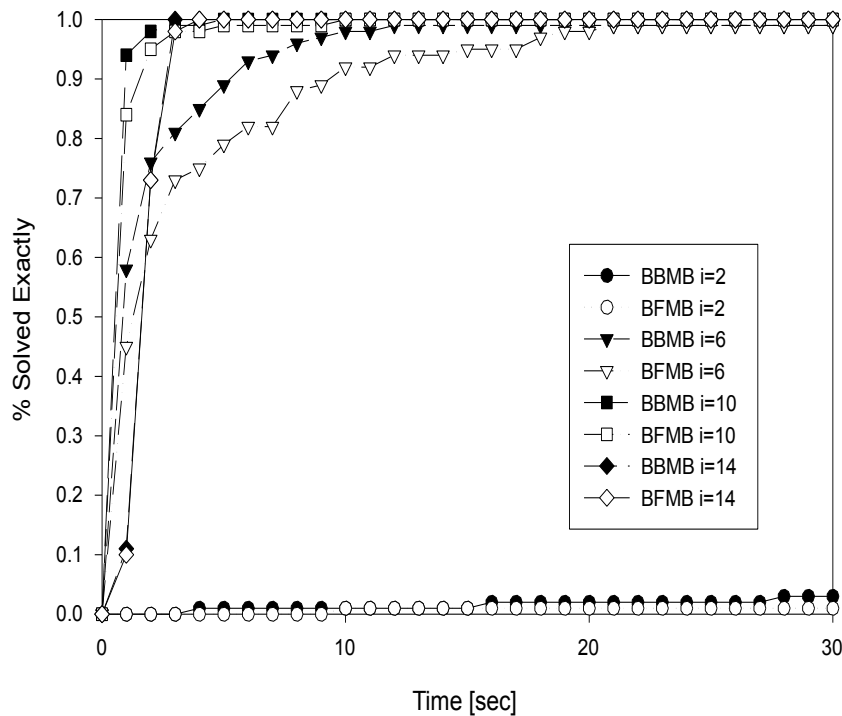
- Random Coding (Bayesian)
- CPCS (Bayesian)
- Random (CSP)

## Measures of performance

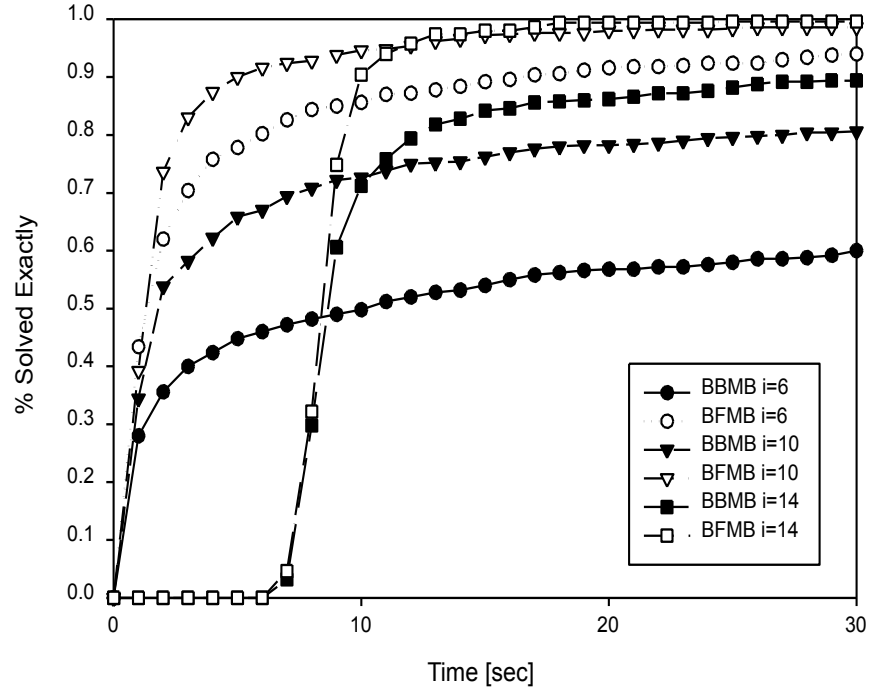
- Compare accuracy given a fixed amount of time - how close is the cost found to the optimal solution
- Compare trade-off performance as a function of time

# Empirical Evaluation of mini-bucket heuristics, Bayesian networks, coding

Random Coding,  $K=100$ , noise=0.28



Random Coding,  $K=100$ , noise=0.32



# Max-CSP experiments

(Kask and Dechter, 2000)

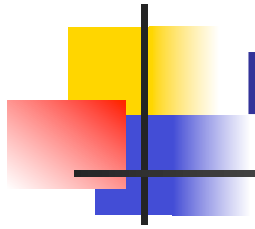
| T  | MBE<br>BBMB<br>BFMB<br>i=2<br>#/time | MBE<br>BBMB<br>BFMB<br>i=4<br>#/time          | MBE<br>BBMB<br>BFMB<br>i=6<br>#/time | MBE<br>BBMB<br>BFMB<br>i=8<br>#/time | MBE<br>BBMB<br>BFMB<br>i=10<br>#/time | MBE<br>BBMB<br>BFMB<br>i=12<br>#/time | PFC-MRDAC<br>#/time |
|--|--------------------------------------|---|--------------------------------------|----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



# Dynamic MB Heuristics

---

- Rather than pre-compiling, the mini-bucket heuristics can be generated during search
- *Dynamic 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

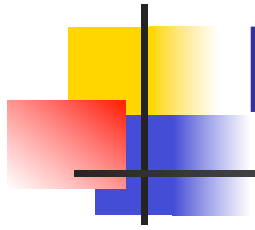




# Real World Benchmarks

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

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



# Empirical Results: Max-CSP

---

- **Random Binary Problems:**  $\langle N, K, C, T \rangle$ 
  - N: number of variables
  - K: domain size
  - C: number of constraints
  - T: Tightness
- **Task:** Max-CSP



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

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

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

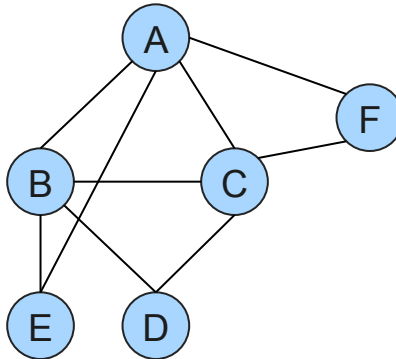


# Outline

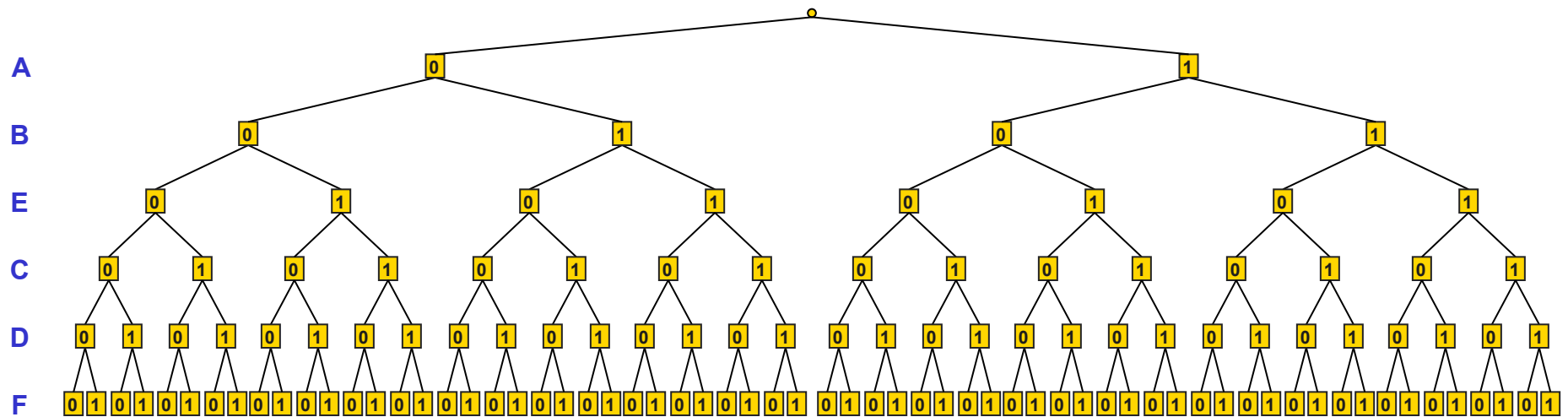
---

- **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

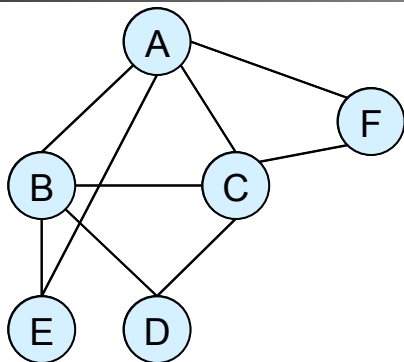
# Classic OR Search Space



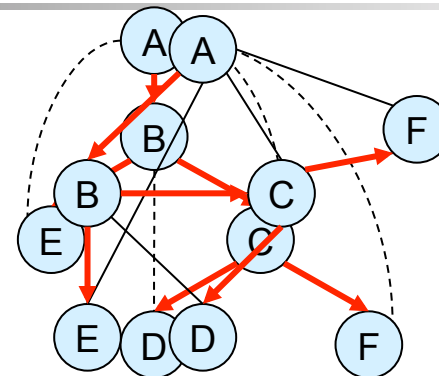
Ordering: A B E C D F



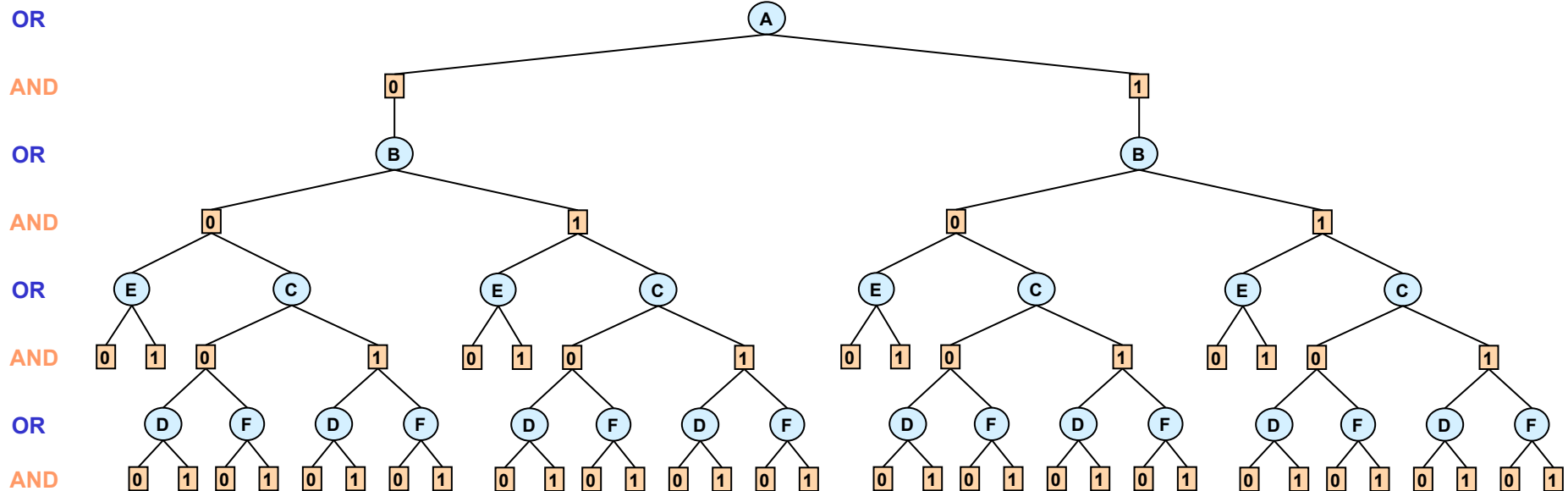
# AND/OR Search Space



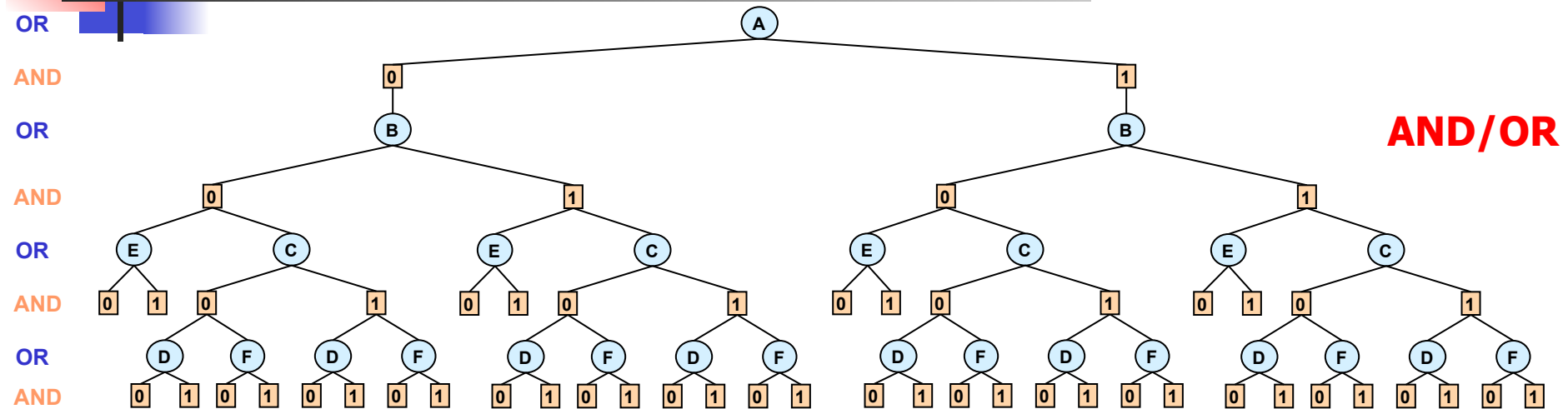
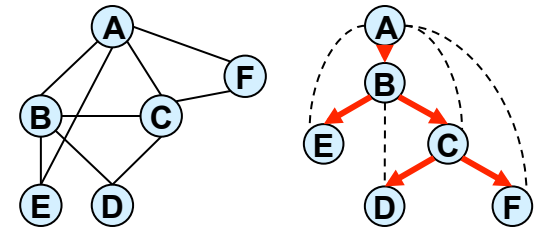
Primal graph



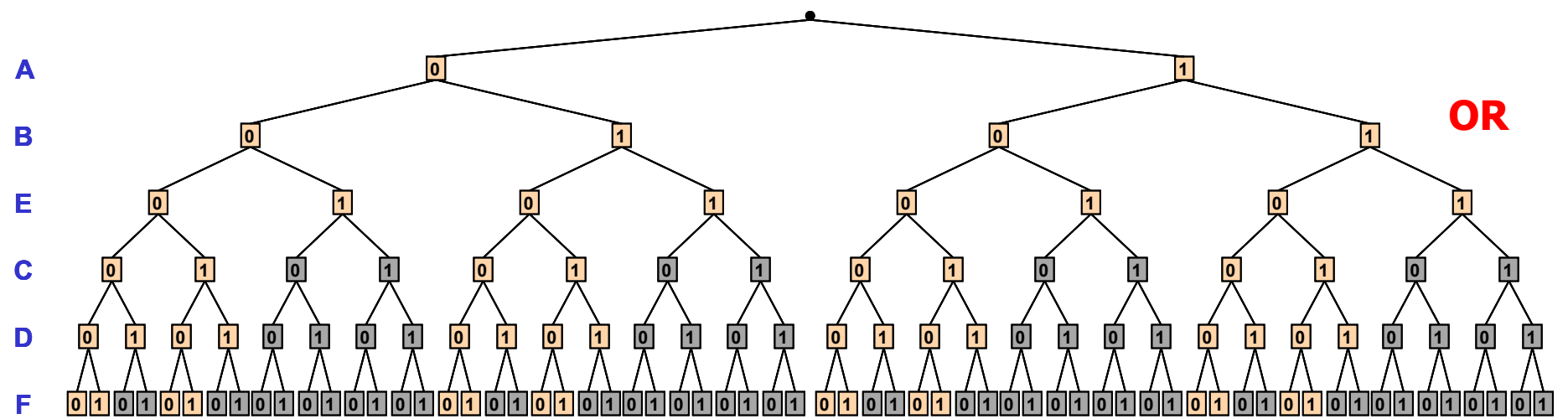
DFS tree



# AND/OR vs. OR



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





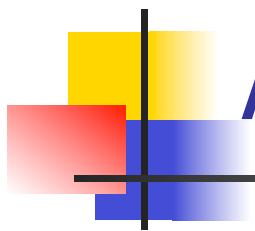
# OR space vs. AND/OR space

---

| width<br>h | height<br>t | 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   |

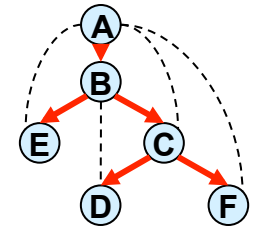
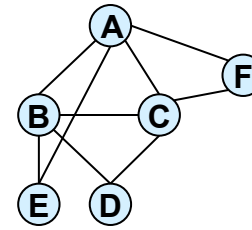
Random graphs with 20 nodes, 20 edges and 2 values per node.





# AND/OR vs. OR

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



OR

AND

OR

AND

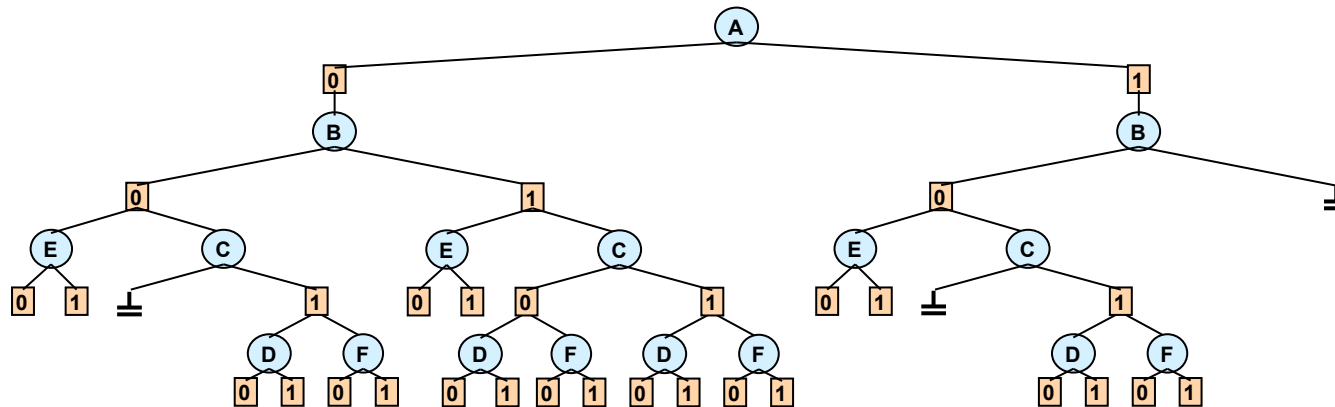
OR

AND

OR

AND

AND/OR



A

B

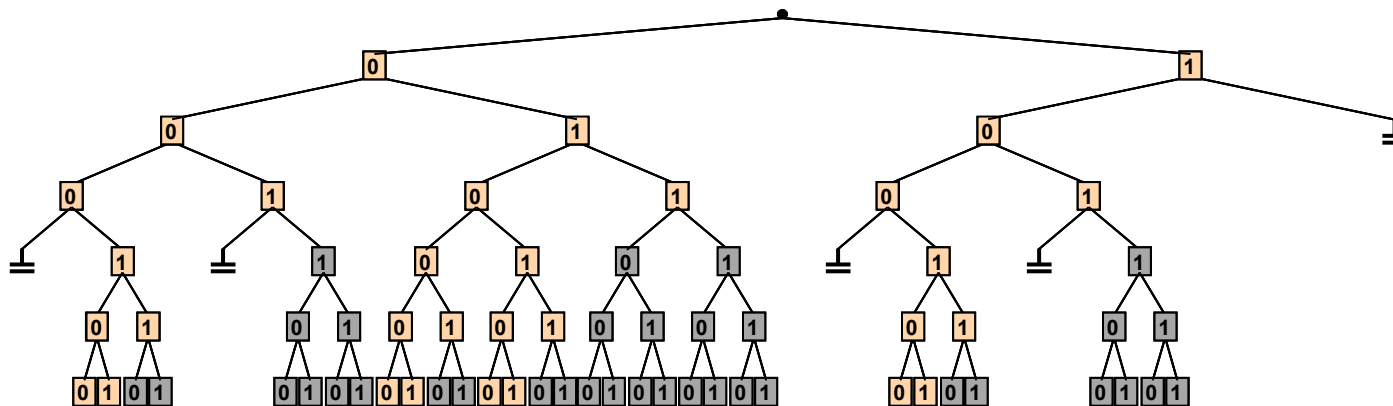
E

C

D

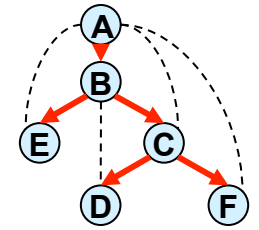
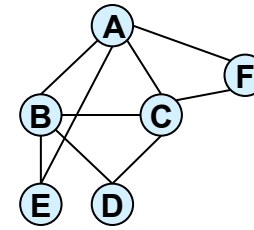
F

OR



# AND/OR vs. OR

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



OR

AND

OR

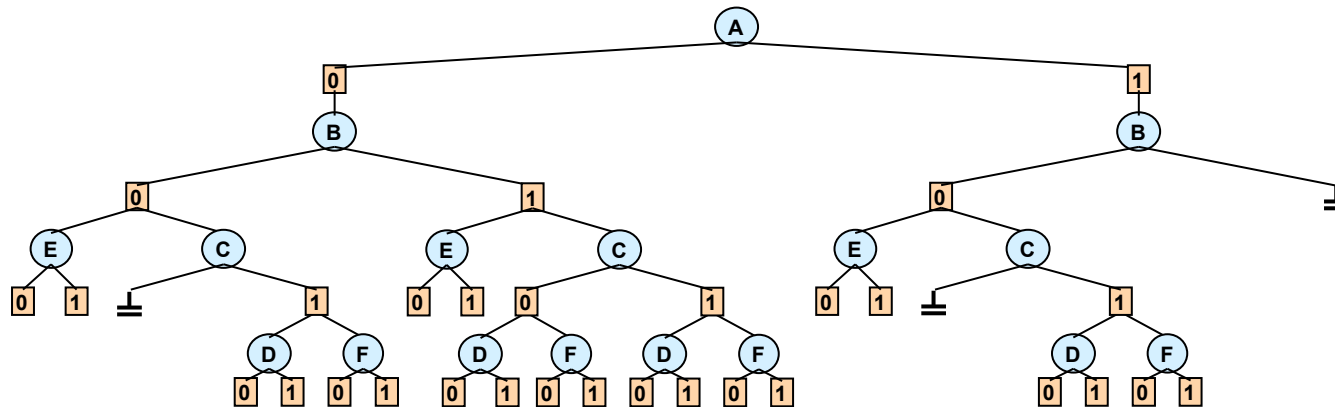
AND

OR

AND

OR

AND



**AND/OR**

Space: linear

Time:

$O(\exp(m))$

$O(w * \log n)$

A

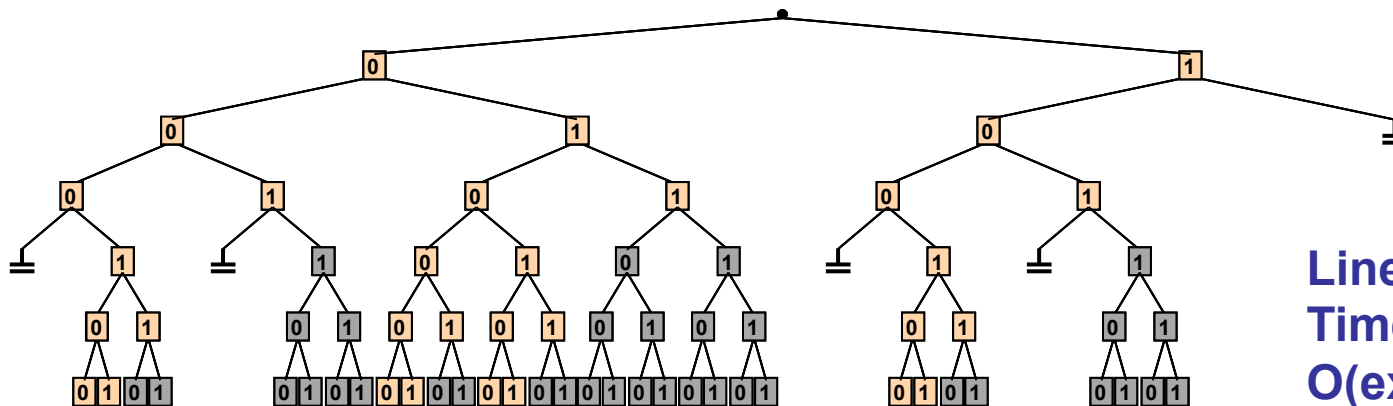
B

E

C

D

F

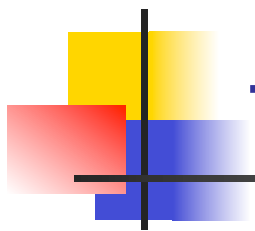


**OR**

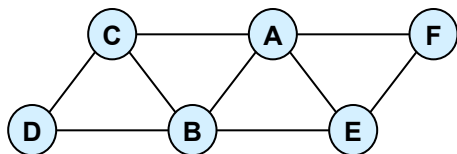
Linear space,

Time:

$O(\exp(n))$



# #CSP – AND/OR Search Tree

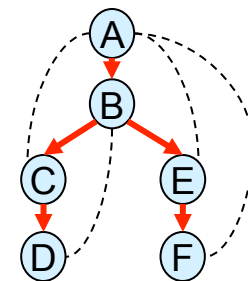


| A | B | C | $R_{ABC}$ |
|---|---|---|-----------|
| 0 | 0 | 0 | 1         |
| 0 | 0 | 1 | 1         |
| 0 | 1 | 0 | 0         |
| 0 | 1 | 1 | 1         |
| 1 | 0 | 0 | 1         |
| 1 | 0 | 1 | 1         |
| 1 | 1 | 0 | 1         |
| 1 | 1 | 1 | 0         |

| 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         |



OR

AND

OR

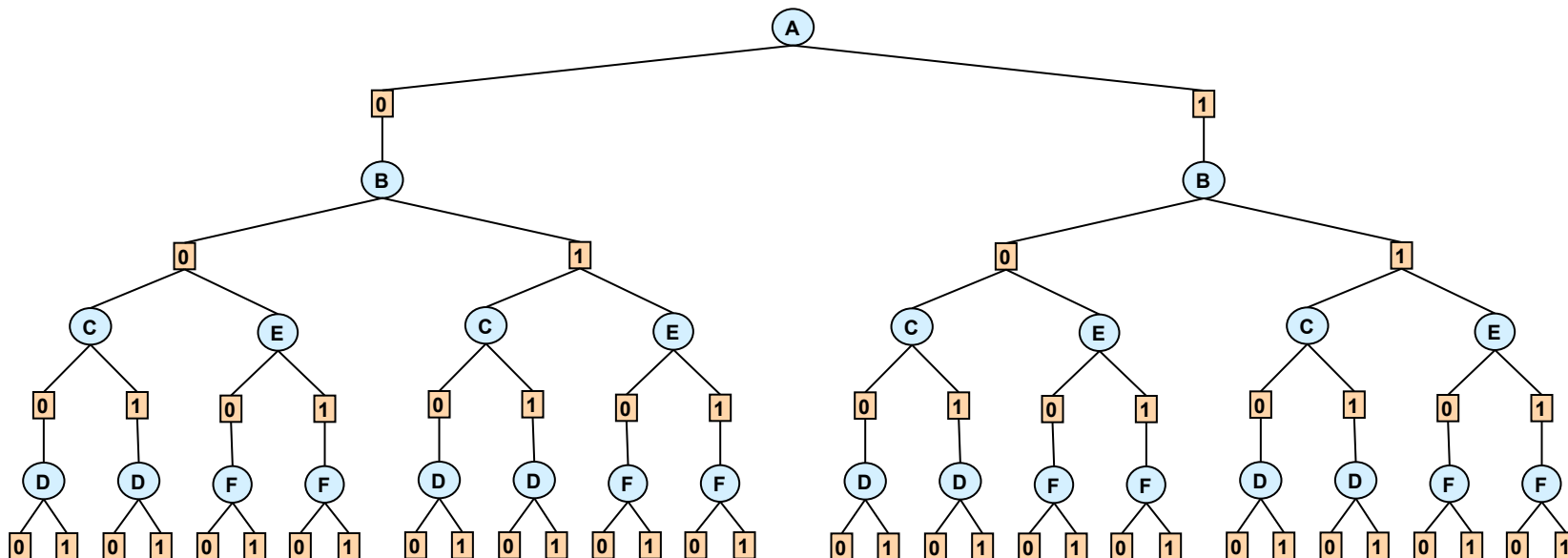
AND

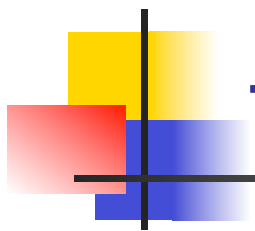
OR

AND

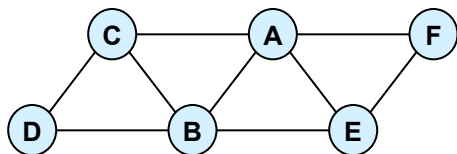
OR

AND





# #CSP – AND/OR Search Tree

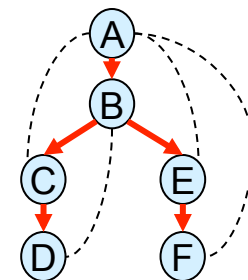


| A | B | C | $R_{ABC}$ |
|---|---|---|-----------|
| 0 | 0 | 0 | 1         |
| 0 | 0 | 1 | 1         |
| 0 | 1 | 0 | 0         |
| 0 | 1 | 1 | 1         |
| 1 | 0 | 0 | 1         |
| 1 | 0 | 1 | 1         |
| 1 | 1 | 0 | 1         |
| 1 | 1 | 1 | 0         |

| 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         |



OR

AND

OR

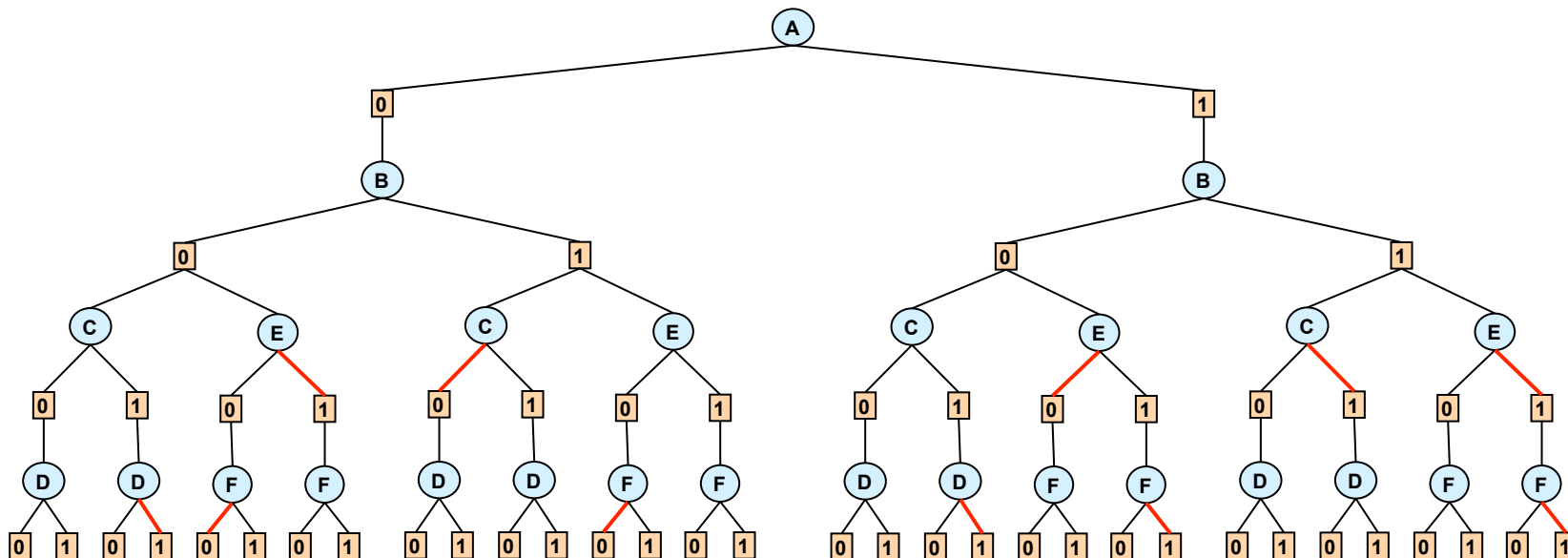
AND

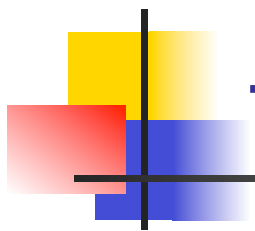
OR

AND

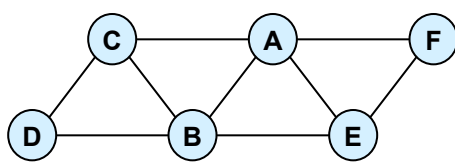
OR

AND





# #CSP – AND/OR Tree DFS

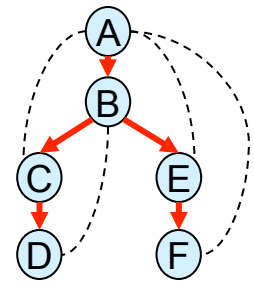


| A | B | C | $R_{ABC}$ |
|---|---|---|-----------|
| 0 | 0 | 0 | 1         |
| 0 | 0 | 1 | 1         |
| 0 | 1 | 0 | 0         |
| 0 | 1 | 1 | 1         |
| 1 | 0 | 0 | 1         |
| 1 | 0 | 1 | 1         |
| 1 | 1 | 0 | 1         |
| 1 | 1 | 1 | 0         |

| 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         |



OR

AND

OR

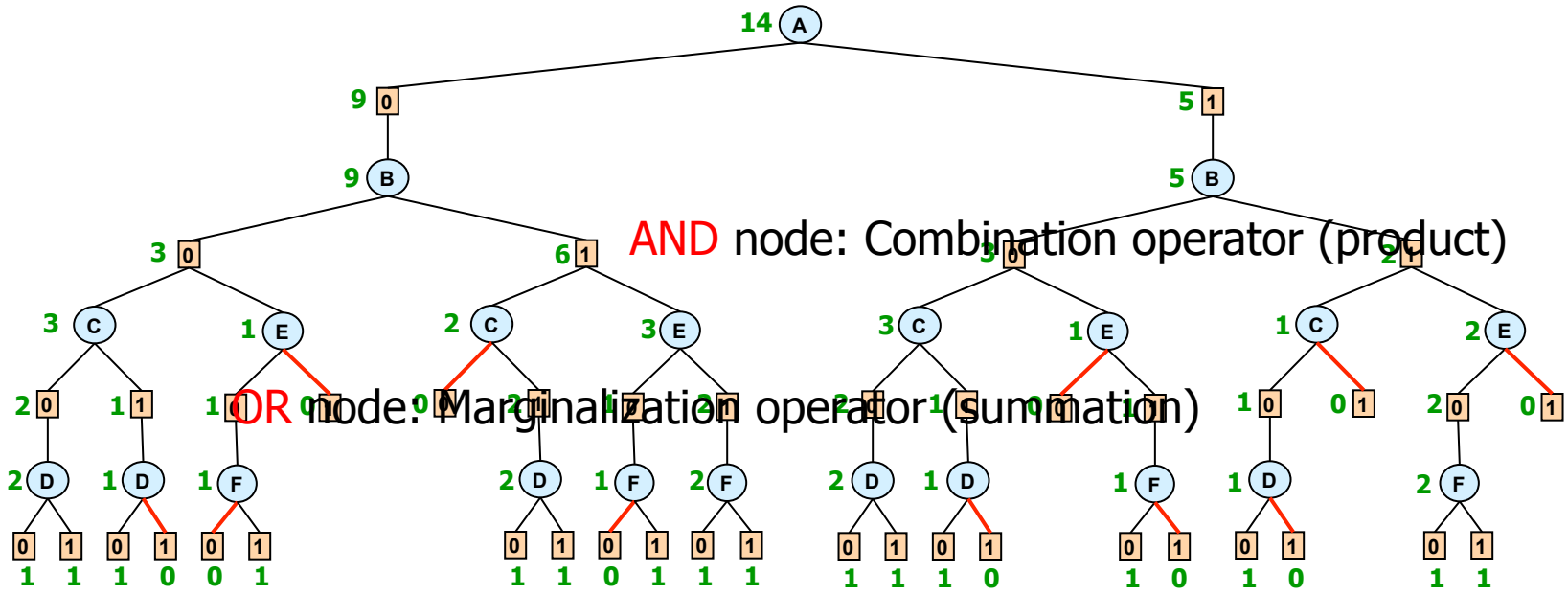
AND

OR

AND

OR

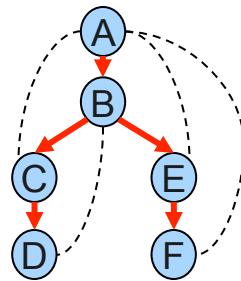
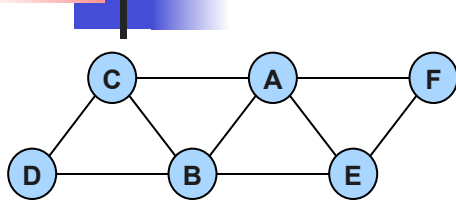
AND



AND node: Combination operator (product)

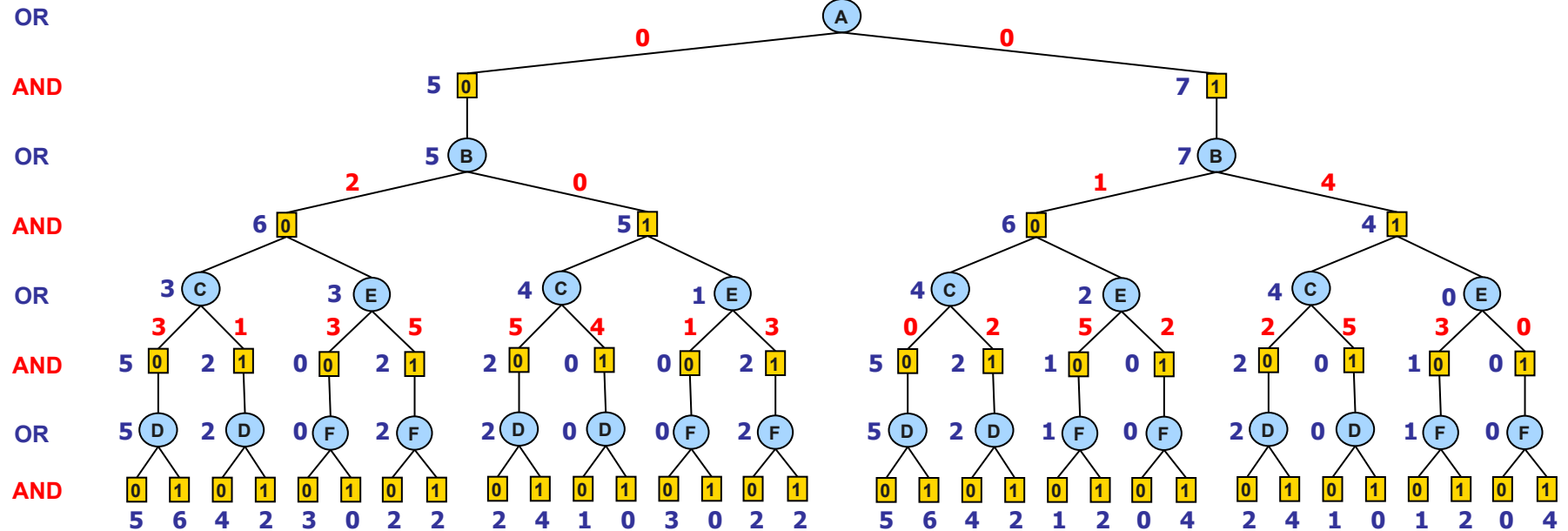
OR node: Marginalization operator (summation)

# AND/OR Tree Search for COP

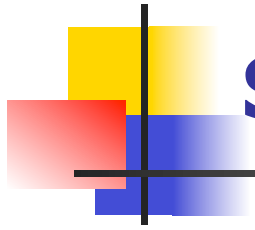


| A | B | $f_1$ | A | C | $f_2$ | A | E | $f_3$ | A | F | $f_4$ | B | C | $f_5$ | B | D | $f_6$ | B | E | $f_7$ | C | D | $f_8$ | E | F | $f_9$ |
|---|---|-------|---|---|-------|---|---|-------|---|---|-------|---|---|-------|---|---|-------|---|---|-------|---|---|-------|---|---|-------|
| 0 | 0 | 2     | 0 | 0 | 3     | 0 | 0 | 0     | 0 | 0 | 2     | 0 | 0 | 0     | 0 | 0 | 4     | 0 | 0 | 3     | 0 | 0 | 1     | 0 | 0 | 1     |
| 0 | 1 | 0     | 0 | 1 | 0     | 0 | 1 | 3     | 0 | 1 | 0     | 0 | 1 | 1     | 0 | 1 | 2     | 0 | 1 | 2     | 0 | 1 | 4     | 0 | 1 | 0     |
| 1 | 0 | 1     | 1 | 0 | 0     | 1 | 0 | 2     | 1 | 0 | 0     | 1 | 0 | 2     | 1 | 0 | 1     | 1 | 0 | 1     | 1 | 0 | 0     | 1 | 0 | 0     |
| 1 | 1 | 4     | 1 | 1 | 1     | 1 | 1 | 0     | 1 | 1 | 2     | 1 | 1 | 4     | 1 | 1 | 0     | 1 | 1 | 0     | 1 | 1 | 0     | 1 | 1 | 2     |

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



~~AND~~ node = Minimization operator (summinization)

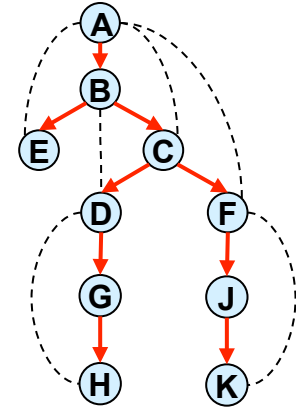
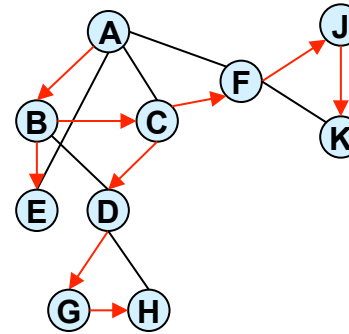


# Summary of AND/OR Search Trees

---

- 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}

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

AND context(D) = {D}

context(F) = {F}

OR

AND

OR

AND

OR

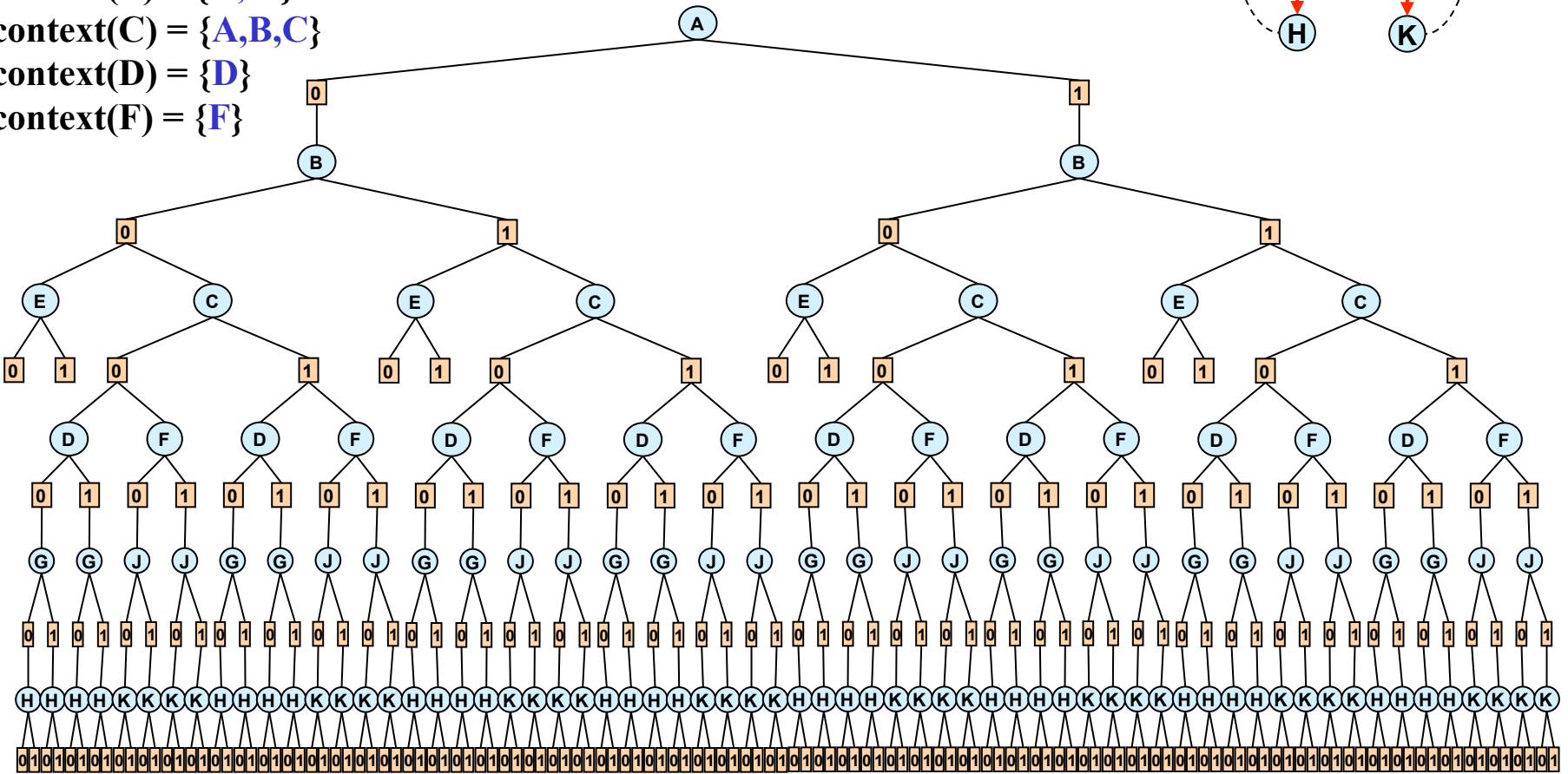
AND

OR

AND

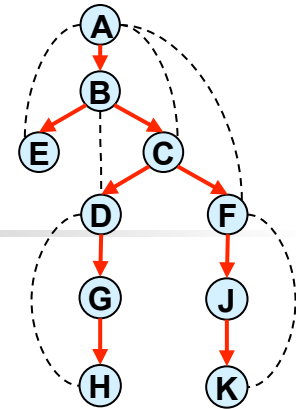
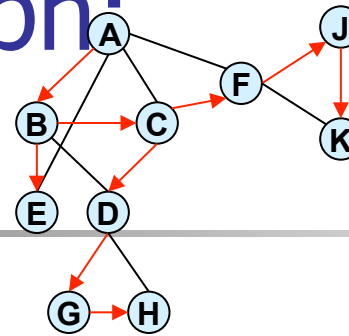
OR

AND





# An AND/OR Graph: Caching Goods



OR

AND

OR

AND

OR

AND

OR

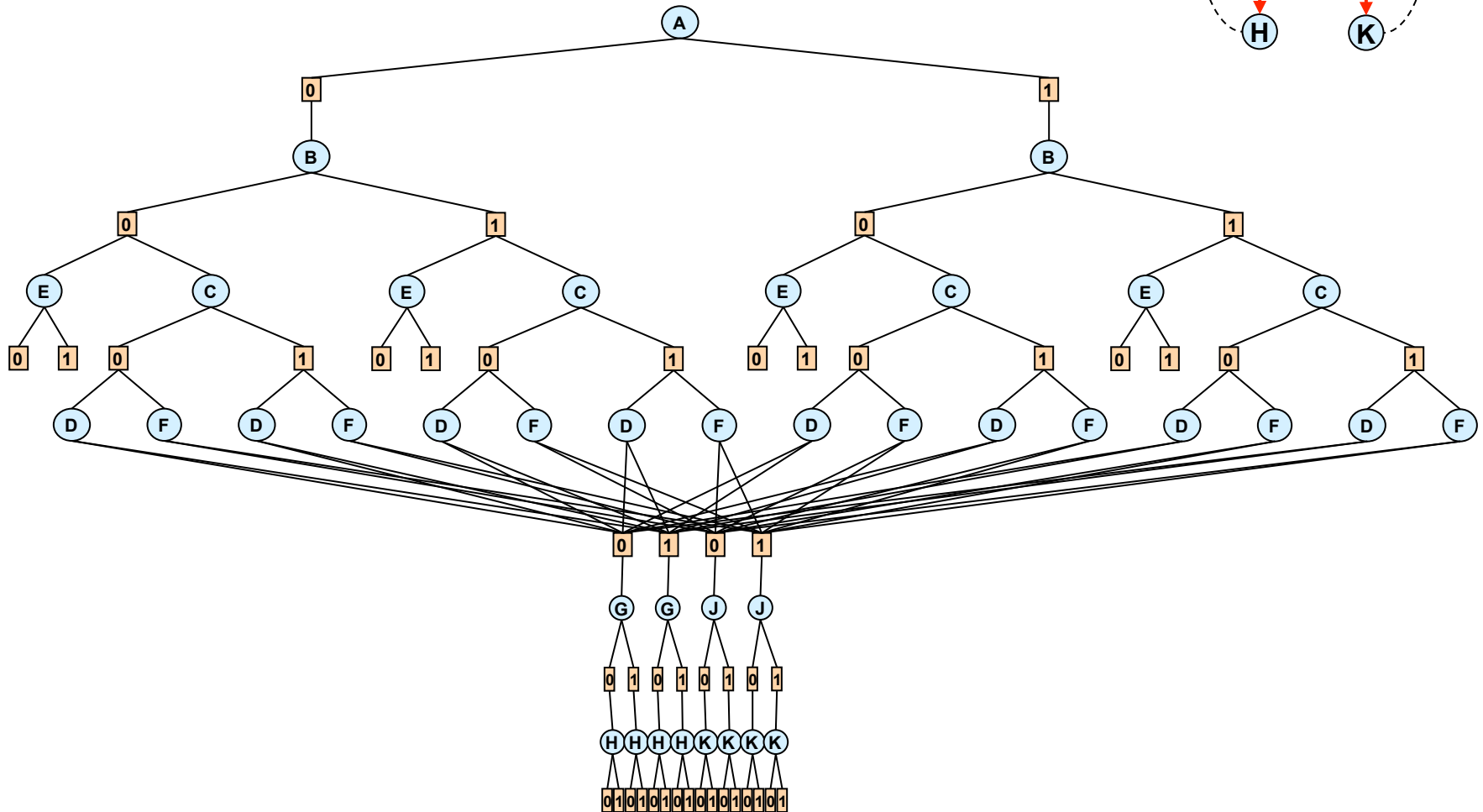
AND

OR

AND

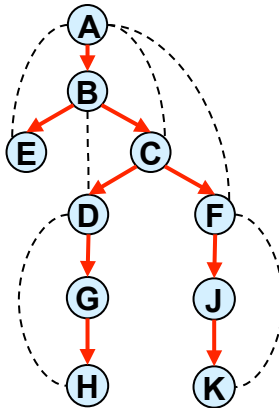
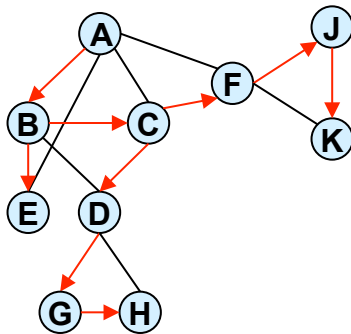
OR

AND

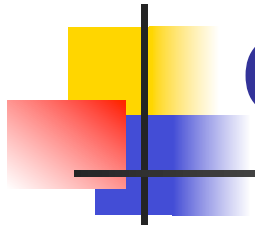


# Context-based Caching

- Caching is possible when **context** is the same
- **context** = parent-separator set in induced pseudo-graph  
= current variable +  
parents connected to subtree below



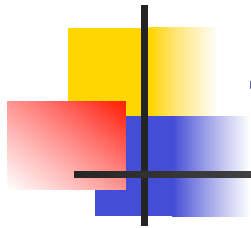
$\text{context}(B) = \{A, B\}$   
 $\text{context}(C) = \{A, B, C\}$   
 $\text{context}(D) = \{D\}$   
 $\text{context}(F) = \{F\}$



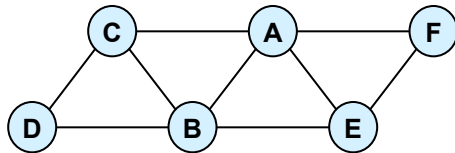
# Complexity of AND/OR Graph

---

- **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

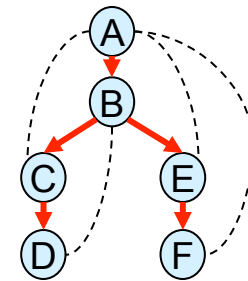


| A | B | C | $R_{ABC}$ |
|---|---|---|-----------|
| 0 | 0 | 0 | 1         |
| 0 | 0 | 1 | 1         |
| 0 | 1 | 0 | 0         |
| 0 | 1 | 1 | 1         |
| 1 | 0 | 0 | 1         |
| 1 | 0 | 1 | 1         |
| 1 | 1 | 0 | 1         |
| 1 | 1 | 1 | 0         |

| 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         |



OR

AND

OR

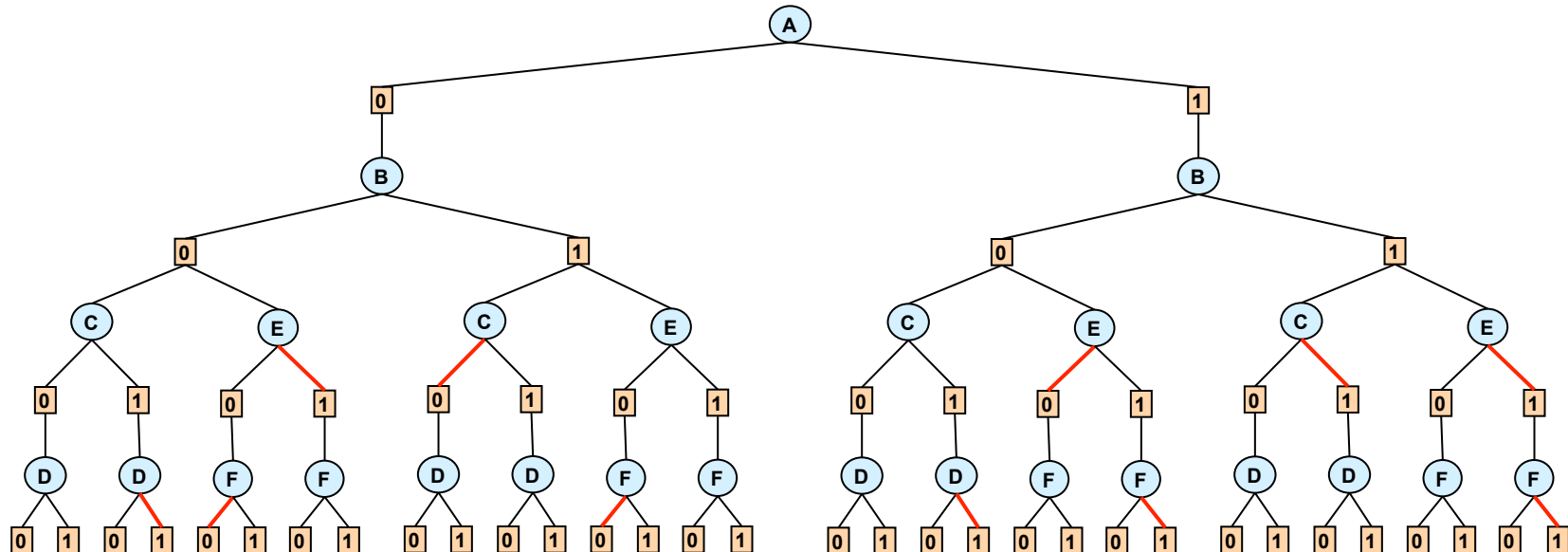
AND

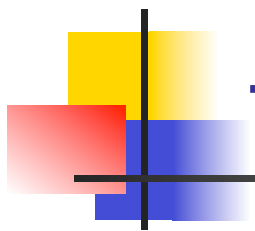
OR

AND

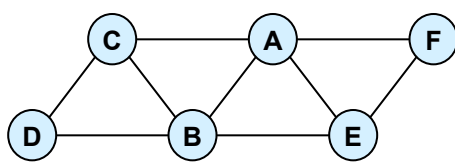
OR

AND





# #CSP – AND/OR Tree DFS

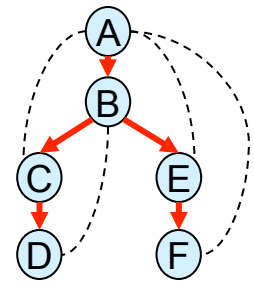


| A | B | C | R <sub>ABC</sub> |
|---|---|---|------------------|
| 0 | 0 | 0 | 1                |
| 0 | 0 | 1 | 1                |
| 0 | 1 | 0 | 0                |
| 0 | 1 | 1 | 1                |
| 1 | 0 | 0 | 1                |
| 1 | 0 | 1 | 1                |
| 1 | 1 | 0 | 1                |
| 1 | 1 | 1 | 0                |

| B | C | D | R <sub>BCD</sub> |
|---|---|---|------------------|
| 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 <sub>ABE</sub> |
|---|---|---|------------------|
| 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 <sub>AEF</sub> |
|---|---|---|------------------|
| 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                |



OR

AND

OR

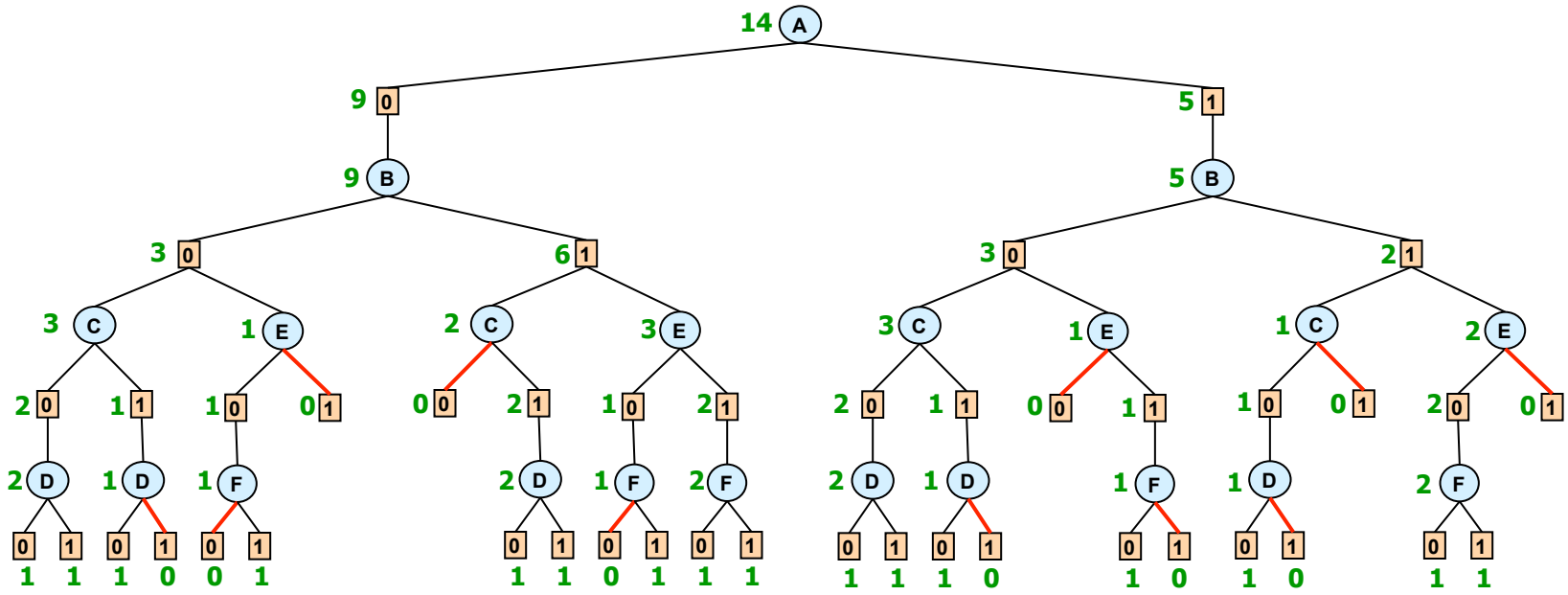
AND

OR

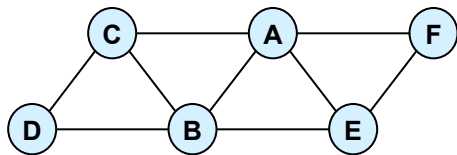
AND

OR

AND



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

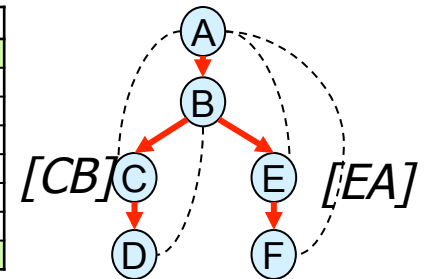


| A | B | C | $R_{ABC}$ |
|---|---|---|-----------|
| 0 | 0 | 0 | 1         |
| 0 | 0 | 1 | 1         |
| 0 | 1 | 0 | 0         |
| 0 | 1 | 1 | 1         |
| 1 | 0 | 0 | 1         |
| 1 | 0 | 1 | 1         |
| 1 | 1 | 0 | 1         |
| 1 | 1 | 1 | 0         |

| 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         |



OR

AND

OR

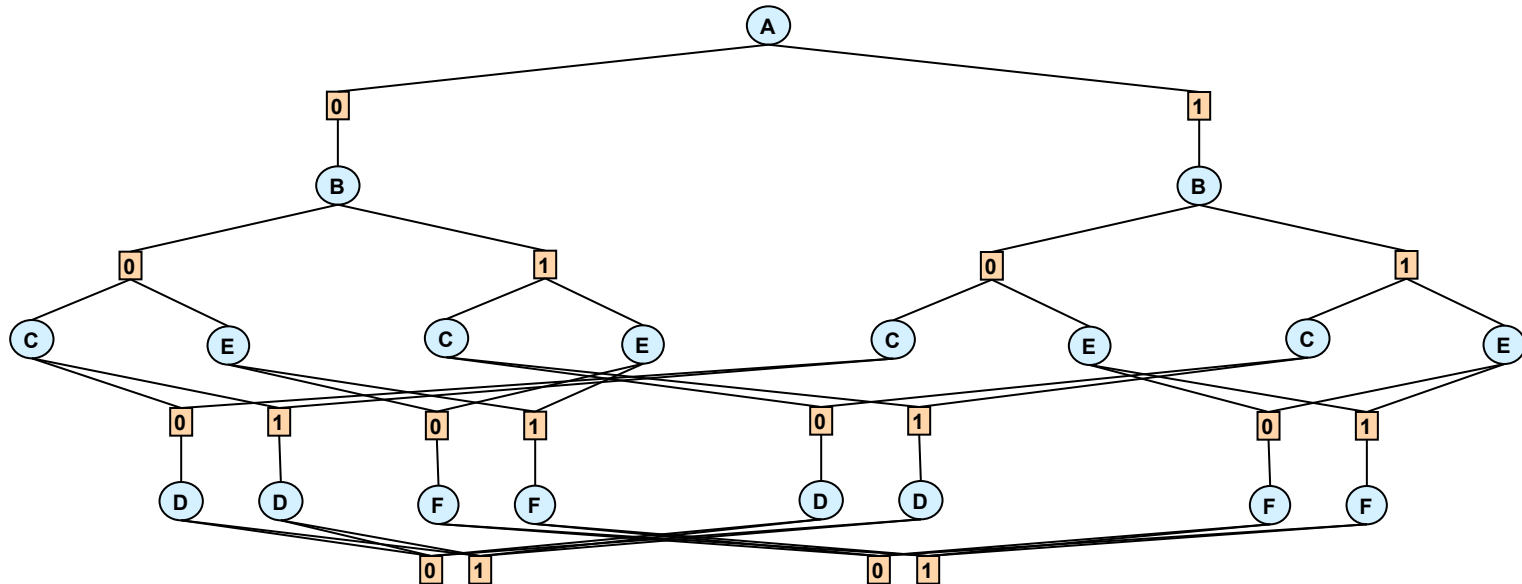
AND

OR

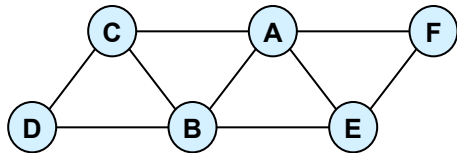
AND

OR

AND



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

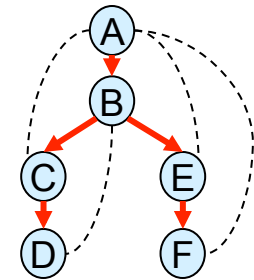


| A | B | C | $R_{ABC}$ |
|---|---|---|-----------|
| 0 | 0 | 0 | 1         |
| 0 | 0 | 1 | 1         |
| 0 | 1 | 0 | 0         |
| 0 | 1 | 1 | 1         |
| 1 | 0 | 0 | 1         |
| 1 | 0 | 1 | 1         |
| 1 | 1 | 0 | 1         |
| 1 | 1 | 1 | 0         |

| 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         |



OR

**Time and Space**  
 **$O(\exp(w^*))$**

AND

OR

AND

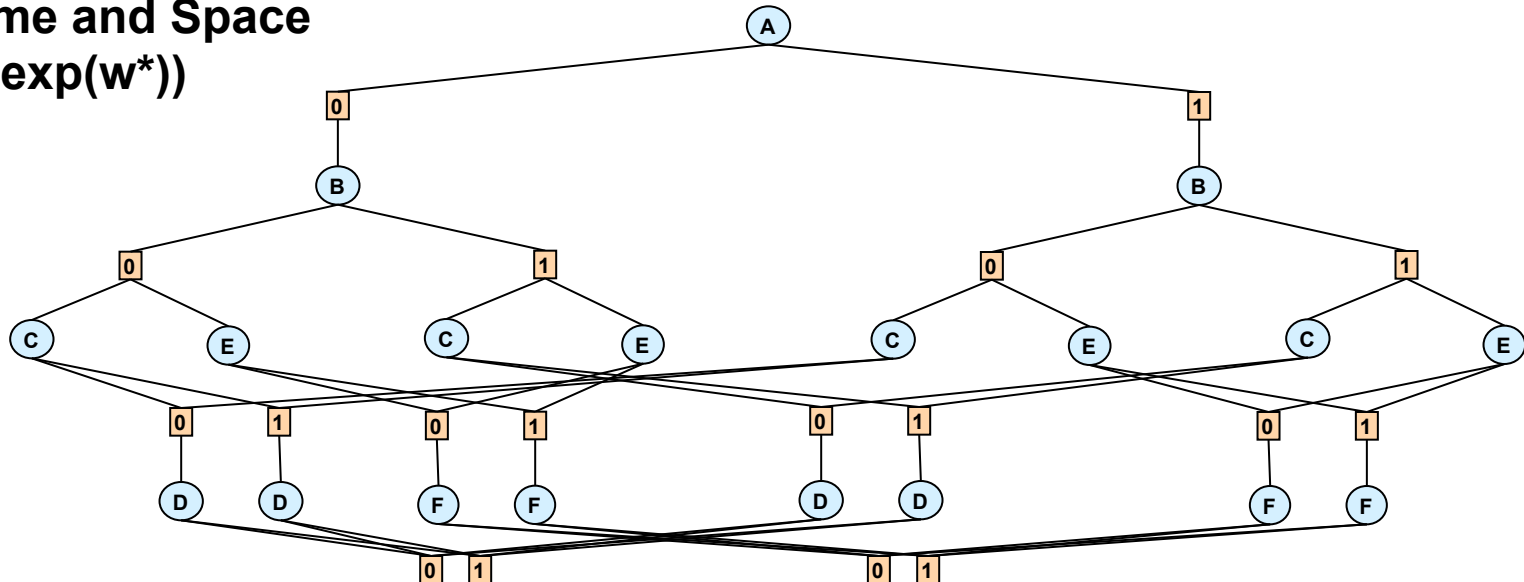
OR

AND

OR

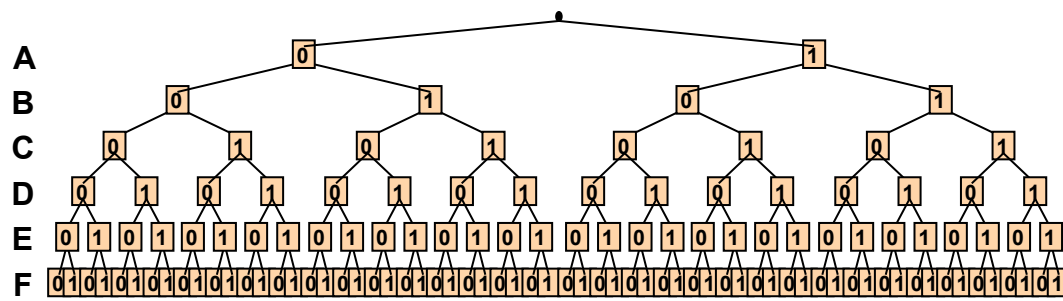
AND

**Space**  
 **$O(\exp(\text{sep}-w^*))$**



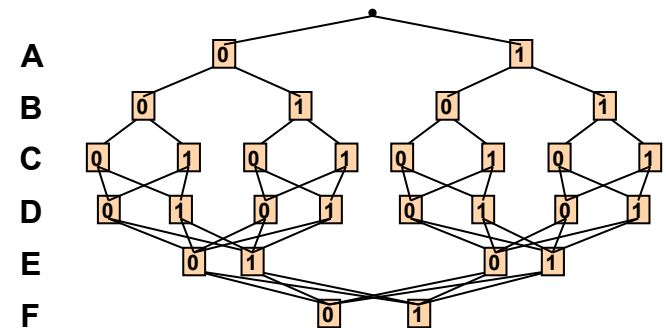


# All Four Search Spaces



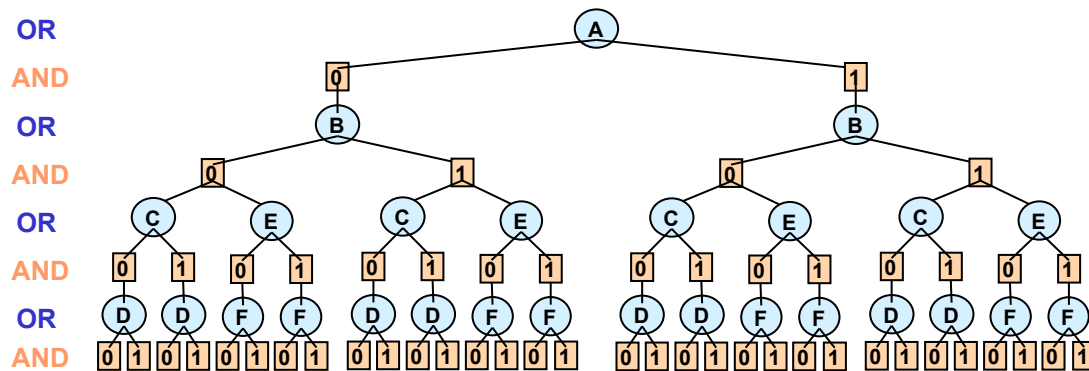
Full OR search tree

126 nodes



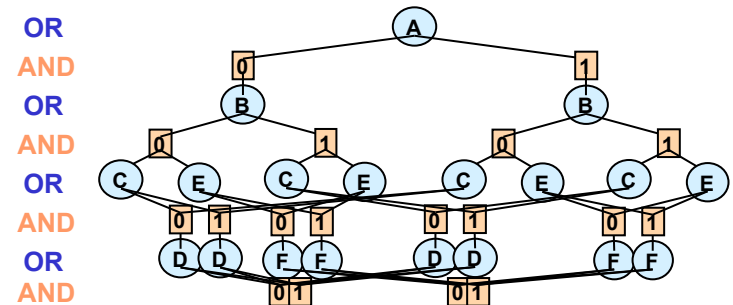
Context minimal OR search graph

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

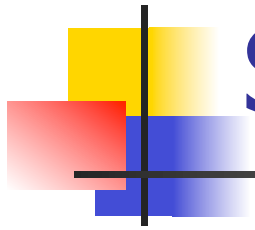
- Space:  $O(n)$
- Time:  $O(k^n)$

## ■ AND/OR graph

- Space:  $O(n k^{w^*})$
- Time:  $O(n k^{w^*})$

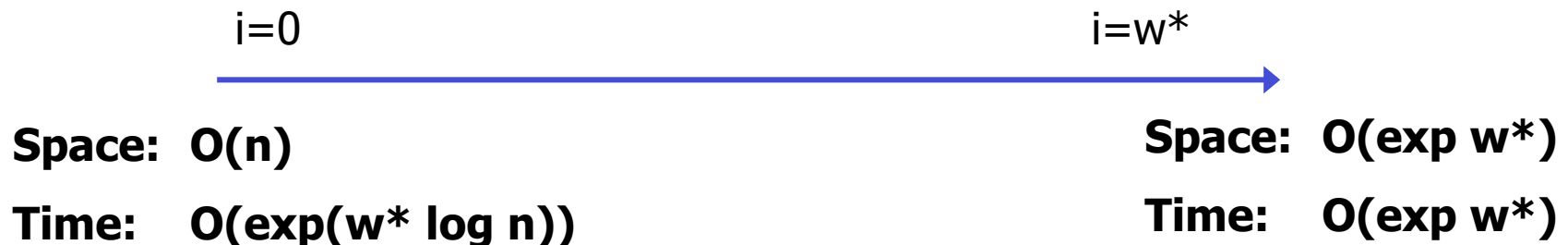
## ■ OR graph

- 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)



**$AO(i)$  time complexity?**

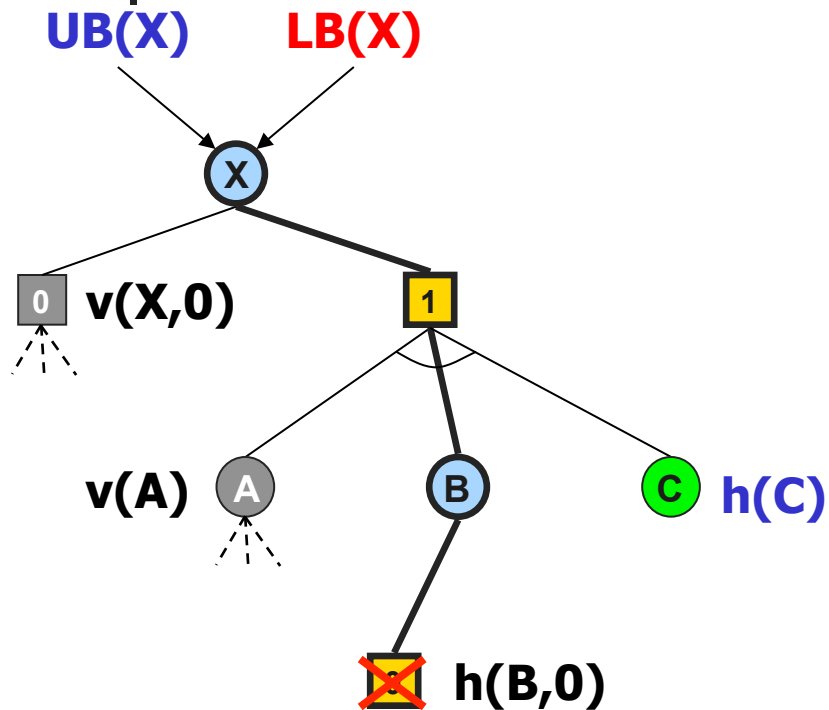


## AND/OR Branch-and-Bound (AOBB)

---

- 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.  $v(X,0)$ )

$LB(X) = LB(X,1)$

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

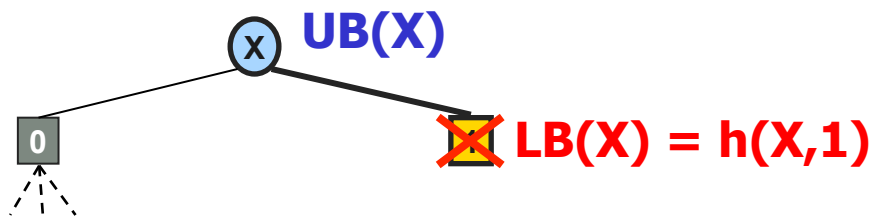
$LB(B) = LB(B,0)$

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

Prune below AND node (B,0) if  $LB(X) \geq UB(X)$

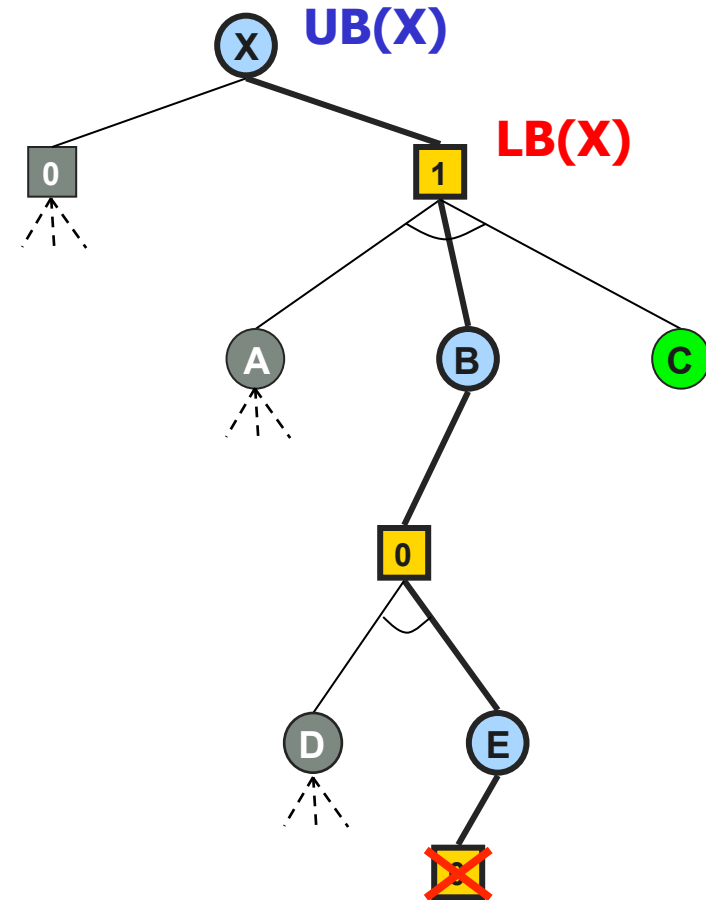
# Shallow/Deep Cutoffs

Prune if  $LB(X) \geq UB(X)$

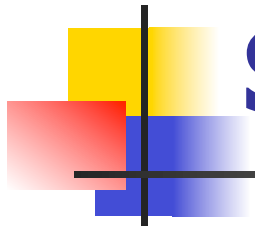


Shallow cutoff

Reminiscent of **Minimax** shallow/deep cutoffs



Deep cutoff



# Summary of AOBB

---

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



# Heuristics for AND/OR

---

- 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

---

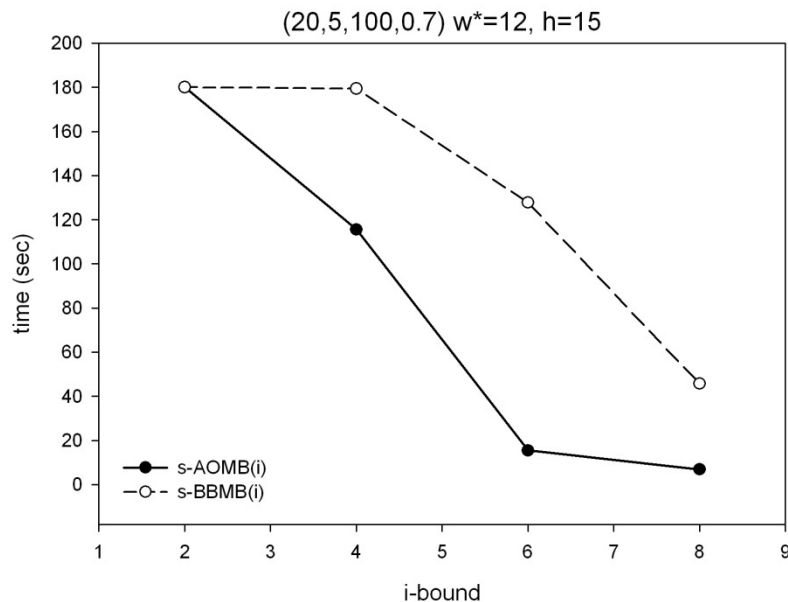
- Tasks
  - Solving WCSPs
  - Finding the MPE in belief networks
- Benchmarks (WCSP)
  - Random binary WCSPs
  - RLFAP networks (CELAR6)
  - Bayesian Networks Repository
- Algorithms
  - s-AOMB(i), d-AOMB(i), AOMFDAC
  - s-BBMB(i), d-BBMB(i), BBMFDAC
  - Static variable ordering (dfs traversal of the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



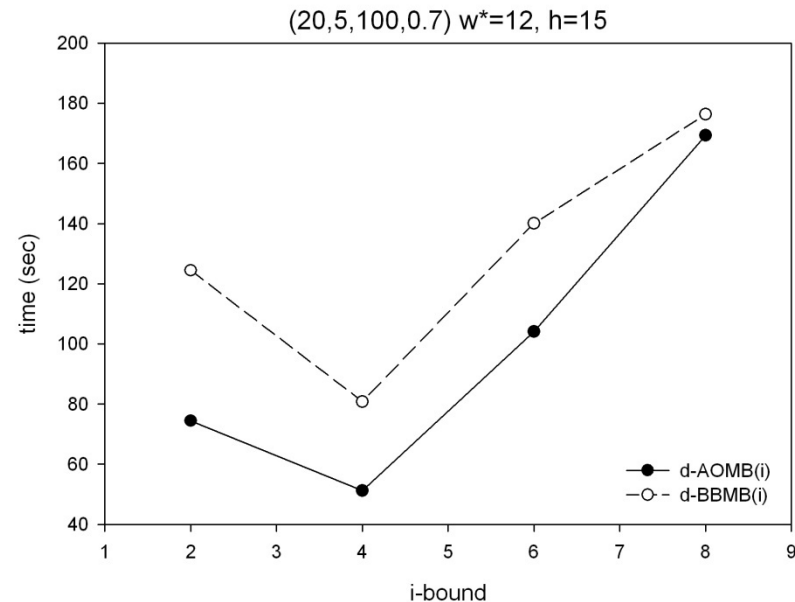
# Random Binary WCSPs

(Marinescu and Dechter, 2005)

## S-AOMB vs S-BBMB



## D-AOMB vs D-BBMB

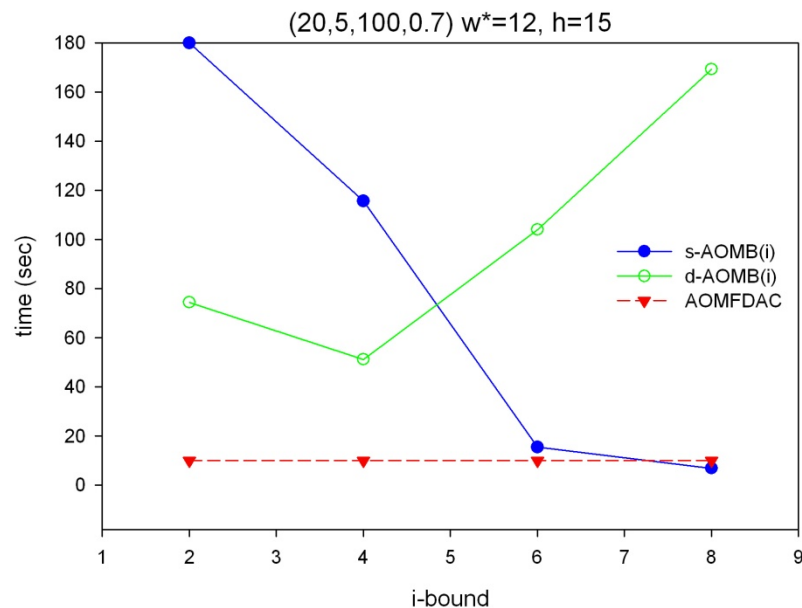


Random networks with  $n=20$  (number of variables),  $d=5$  (domain size),  $c=100$  (number of constraints),  $t=70\%$  (tightness). Time limit 180 seconds.

**AO search is superior to OR search**

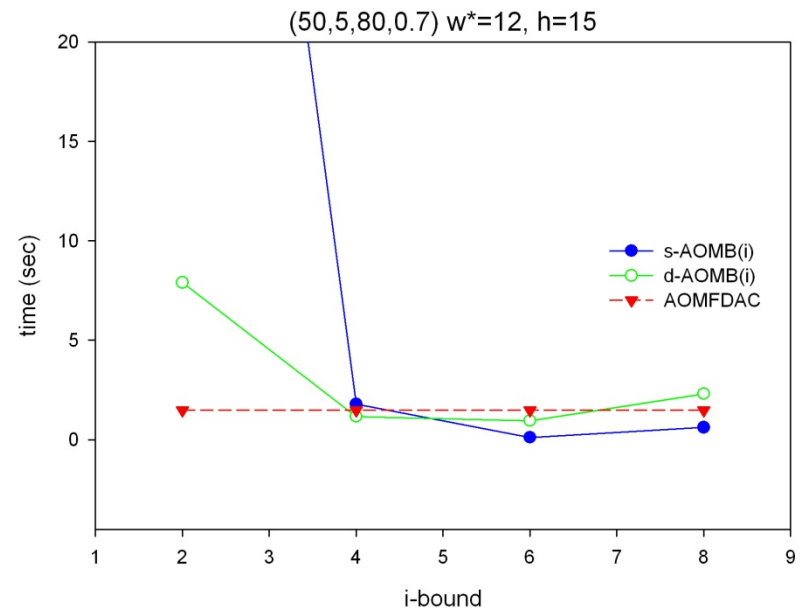
# Random Binary WCSPs (contd.)

**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

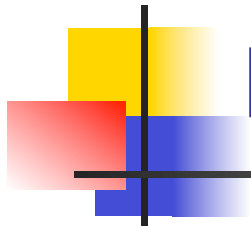
---

## Radio Link Frequency Assignment Problem (RLFAP)

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

CELAR6 sub-instances

**AOMFDAC** is superior to **ORMFDAC**



# Bayesian Networks Repository

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

Time limit 600 seconds

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

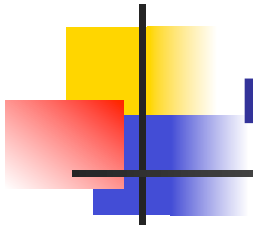
**Static** AO is better with accurate heuristic (**large i**)



# 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

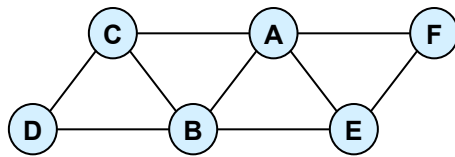


## From Searching Trees to Searching Graphs

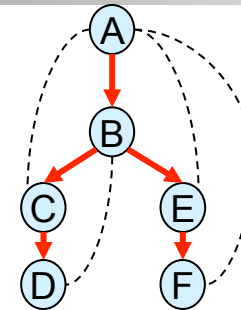
---

- 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



$\text{context}(A) = \{A\}$   
 $\text{context}(B) = \{B, A\}$   
 $\text{context}(C) = \{C, B\}$   
 $\text{context}(D) = \{D\}$   
 $\text{context}(E) = \{E, A\}$   
 $\text{context}(F) = \{F\}$

Pseudo-tree

OR

AND

OR

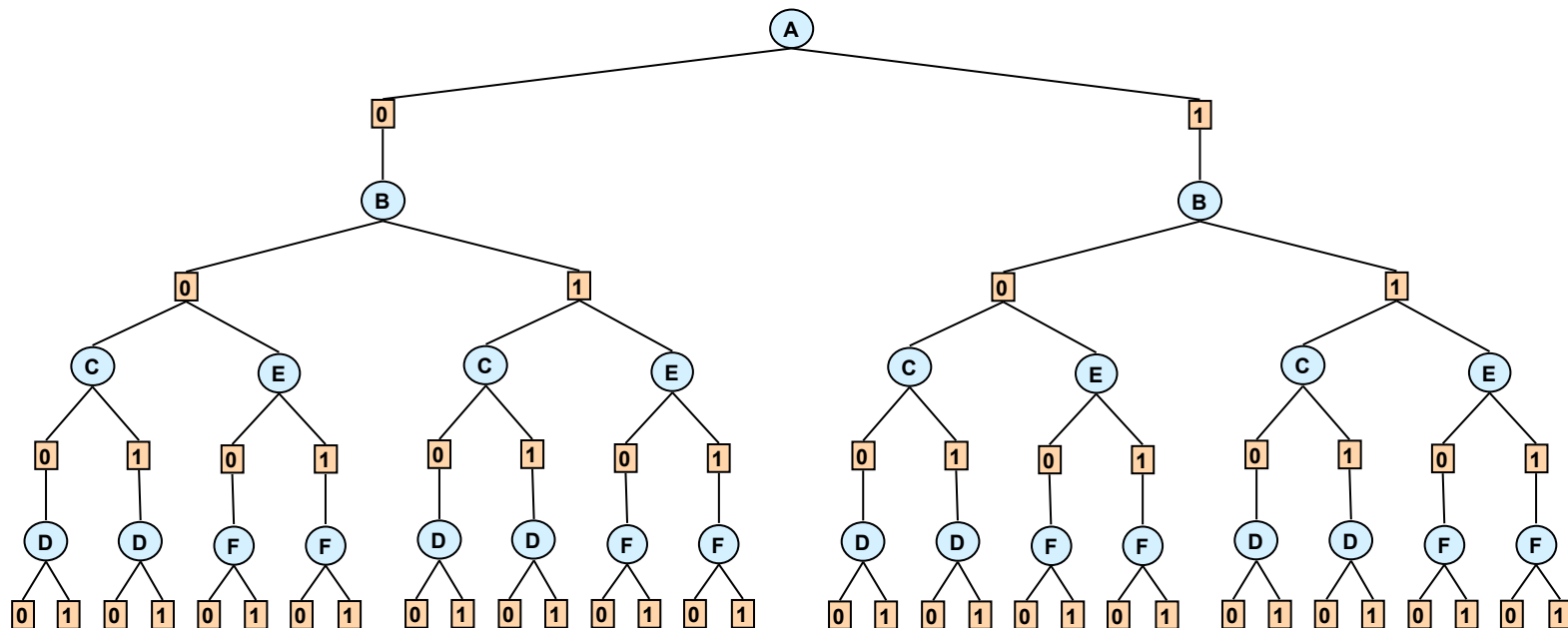
AND

OR

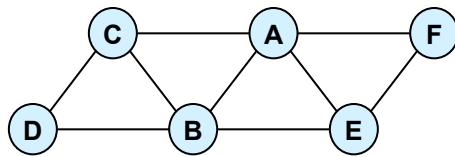
AND

OR

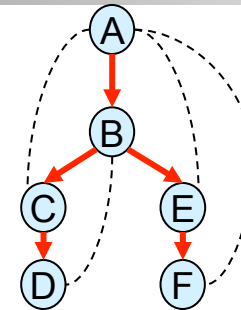
AND



# AND/OR Search Graph

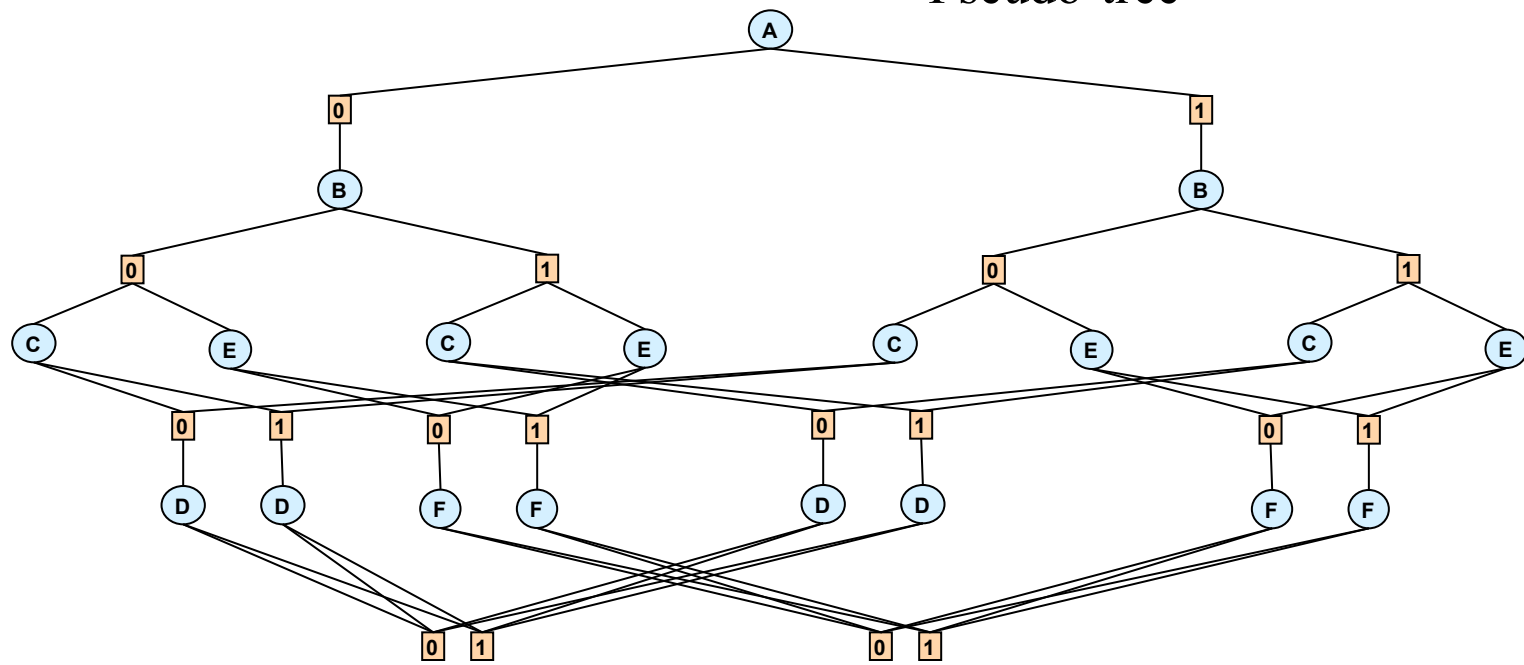


Primal graph



$\text{context}(A) = \{A\}$   
 $\text{context}(B) = \{B, A\}$   
 $\text{context}(C) = \{C, B\}$   
 $\text{context}(D) = \{D\}$   
 $\text{context}(E) = \{E, A\}$   
 $\text{context}(F) = \{F\}$

Pseudo-tree



OR

AND

OR

AND

OR

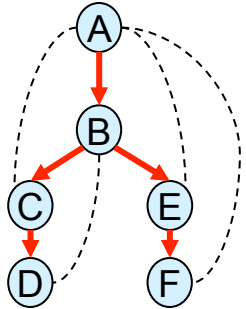
AND

OR

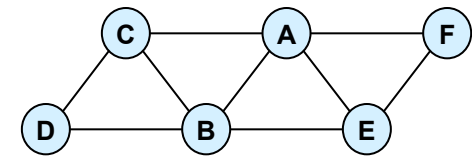
AND



# Context-based caching



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

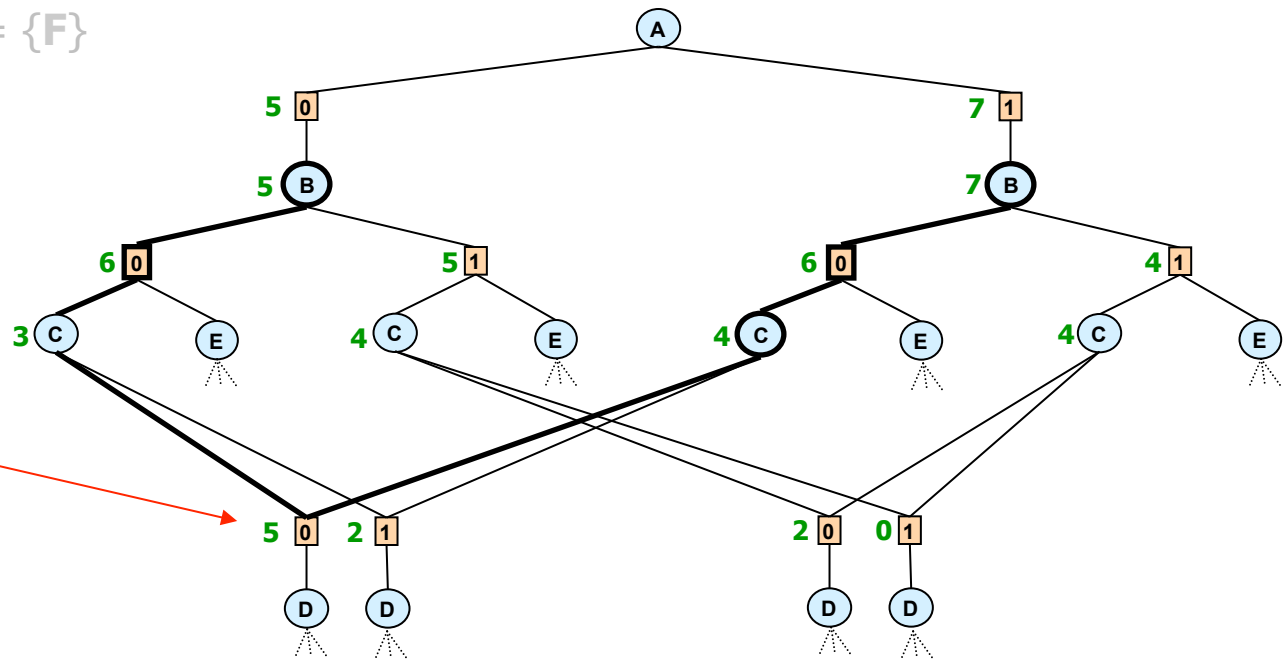


Primal graph

Cache Table (C)

| B | C | Value |
|---|---|-------|
| 0 | 0 | 5     |
| 0 | 1 | 2     |
| 1 | 0 | 2     |
| 1 | 1 | 0     |

Space:  $O(\exp(2))$

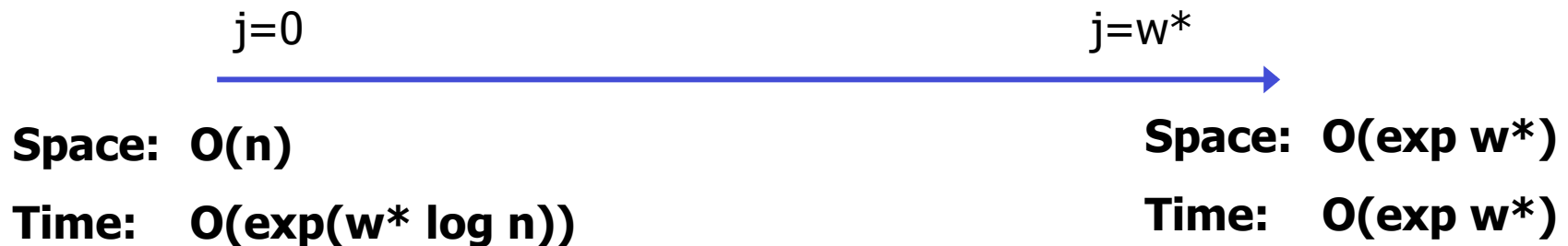




# Searching AND/OR Graphs

---

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



**AO(j) time complexity?**