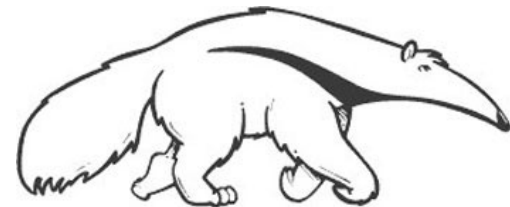


Algorithms for Causal Probabilistic Graphical Models

Class 3:
Search

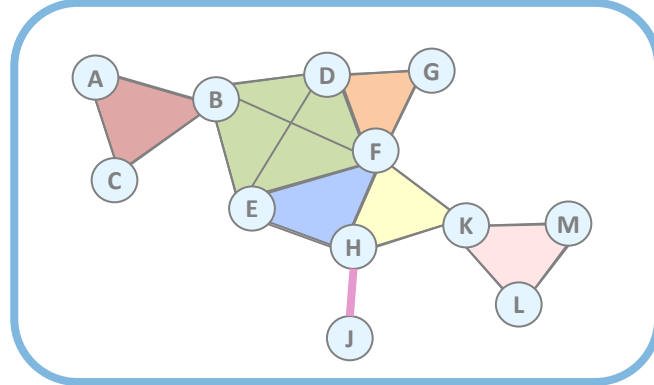
Athens Summer School on AI
July 2024

Prof. Rina Dechter
Prof. Alexander Ihler

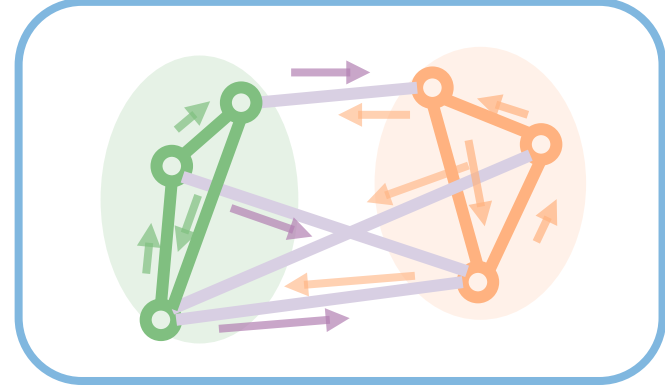


Outline of Lectures

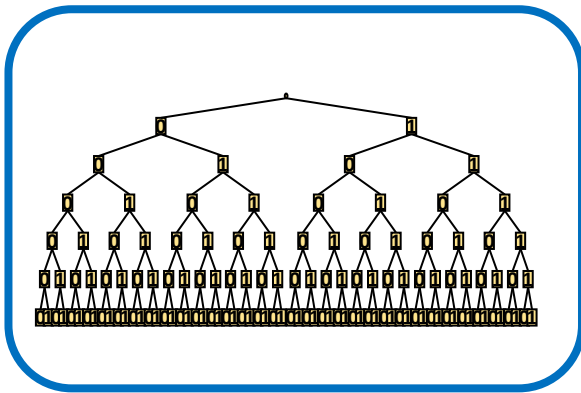
Class 1: Introduction & Inference



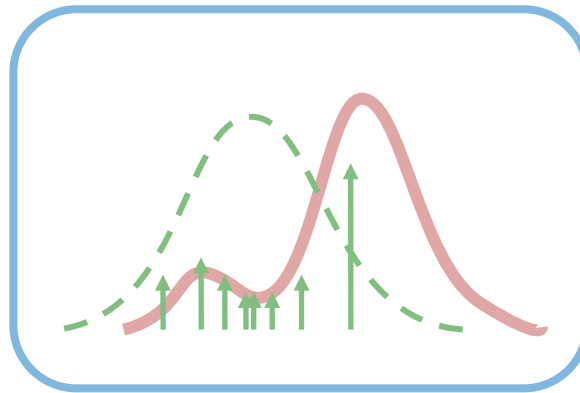
Class 2: Bounds & Variational Methods



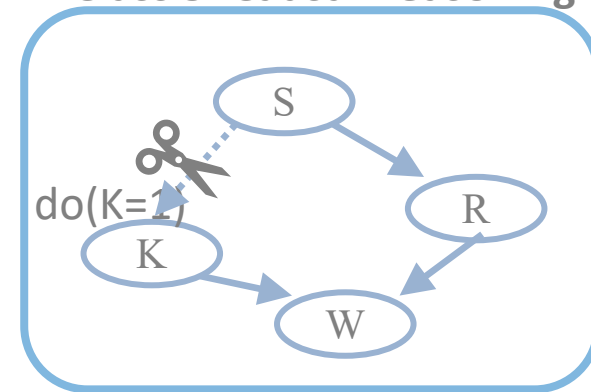
Class 3: Search Methods



Class 4: Monte Carlo Methods



Class 5: Causal Reasoning



Graphical Models

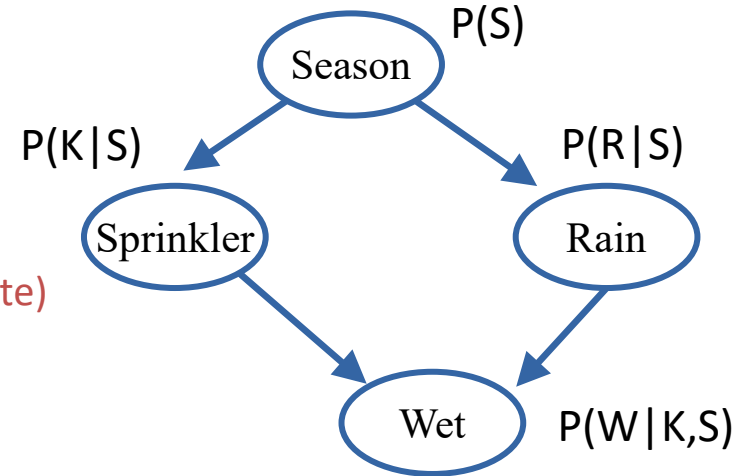
A *graphical model* consists of:

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

$D = \{D_1, \dots, D_n\}$ -- domains (we'll assume discrete)

$F = \{f_{\alpha_1}, \dots, f_{\alpha_m}\}$ -- functions or CPTs

and a *combination operator*



The *combination operator* defines an overall function from the individual factors,

e.g., “+” : $P(S, K, R, W) = P(S) \cdot P(K|S) \cdot P(R|S) \cdot P(W|K, S)$

Notation:

Discrete X_i values called “states”

“Tuple” or “configuration”: states taken by a set of variables

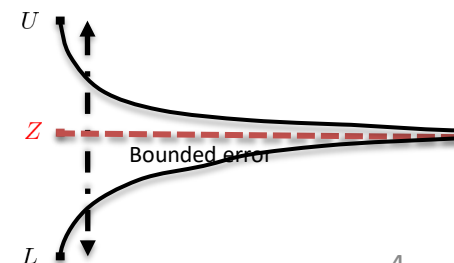
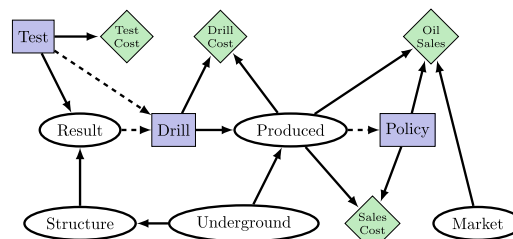
“Scope” of f : set of variables that are arguments to a factor f

often index factors by their scope, e.g., $f_{\alpha}(X_{\alpha})$, $X_{\alpha} \subseteq X$

Probabilistic Reasoning Problems

- Exact inference time, space exponential in induced width/tree-width
- Use **search** to trade memory for time and time for anytime

Max-Inference:	$f(x^*) = \max_x \prod_{\alpha} f_{\alpha}(x_{\alpha})$
Sum-Inference: (e.g., causal effects)	$Z = \sum_x \prod_{\alpha} f_{\alpha}(x_{\alpha})$
Mixed-Inference (MMAP):	$f_M(x_M^*) = \max_{x_M} \sum_{x_S} \prod_{\alpha} f_{\alpha}(x_{\alpha})$
Mixed-Inference (MEU): (e.g., decisions, planning)	$\text{MEU} = \max_{D_1, \dots, D_m} \sum_{X_1, \dots, X_n} \left(\prod_{P_i \in P} P_i \right) \times \left(\sum_{r_i \in R} r_i \right)$



Outline: Search



Outline: Search



The Probability Tree

$$P(a, e = 0) = P(a) \sum_b P(b | a) \sum_c P(c | a, b) \sum_b P(d | a, b) \sum_{e=0} P(e | b, c)$$

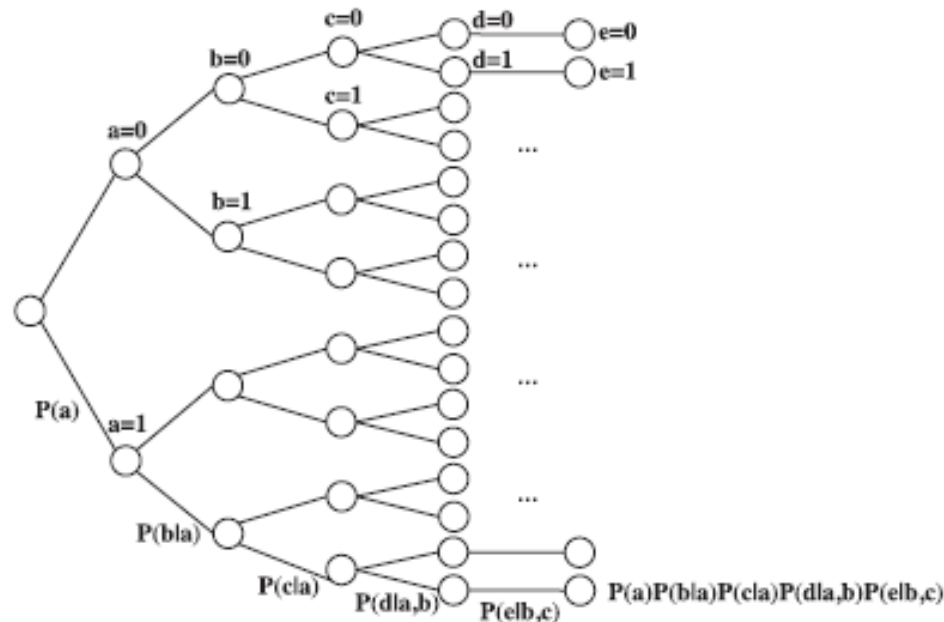
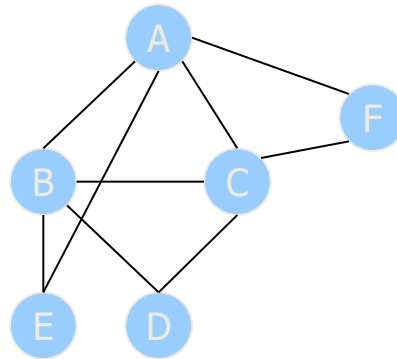


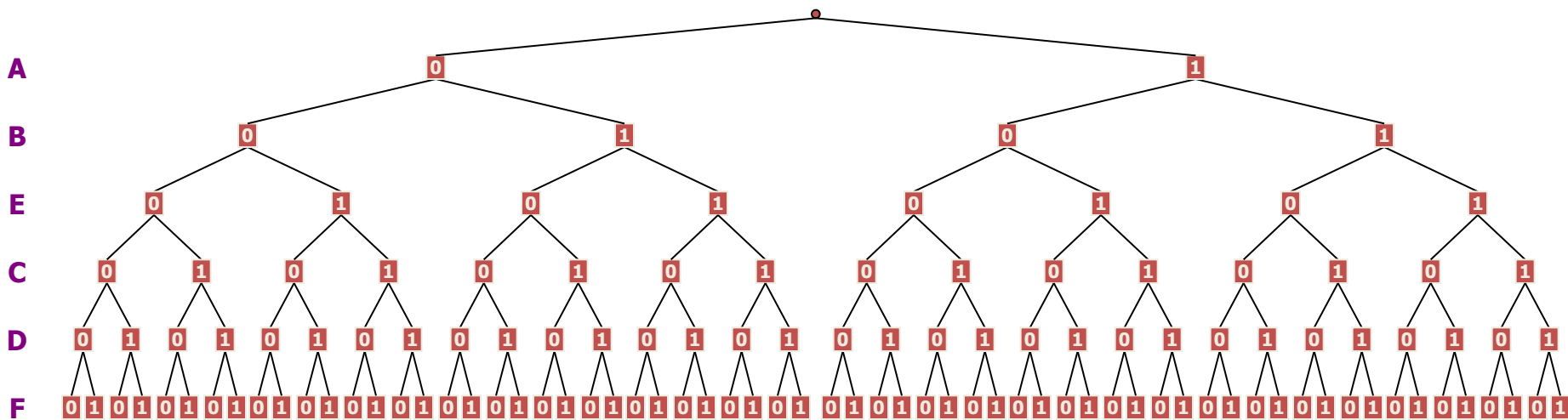
Figure 6.1: Probability tree for computing $P(d=1, g=0)$.

Complexity of conditioning: exponential time, linear space

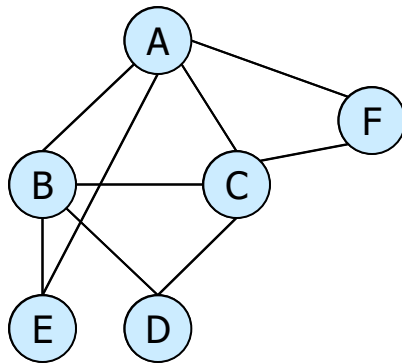
The Classic OR Search Space



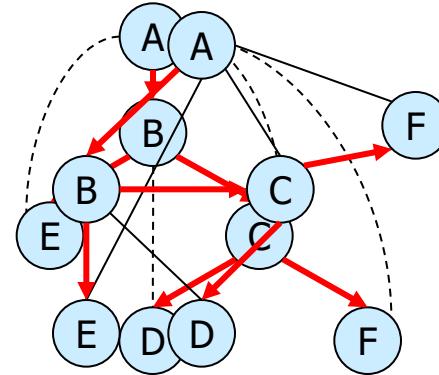
Ordering: A B E C D F



AND/OR Search Space



Primal graph



DFS tree

OR

AND

OR

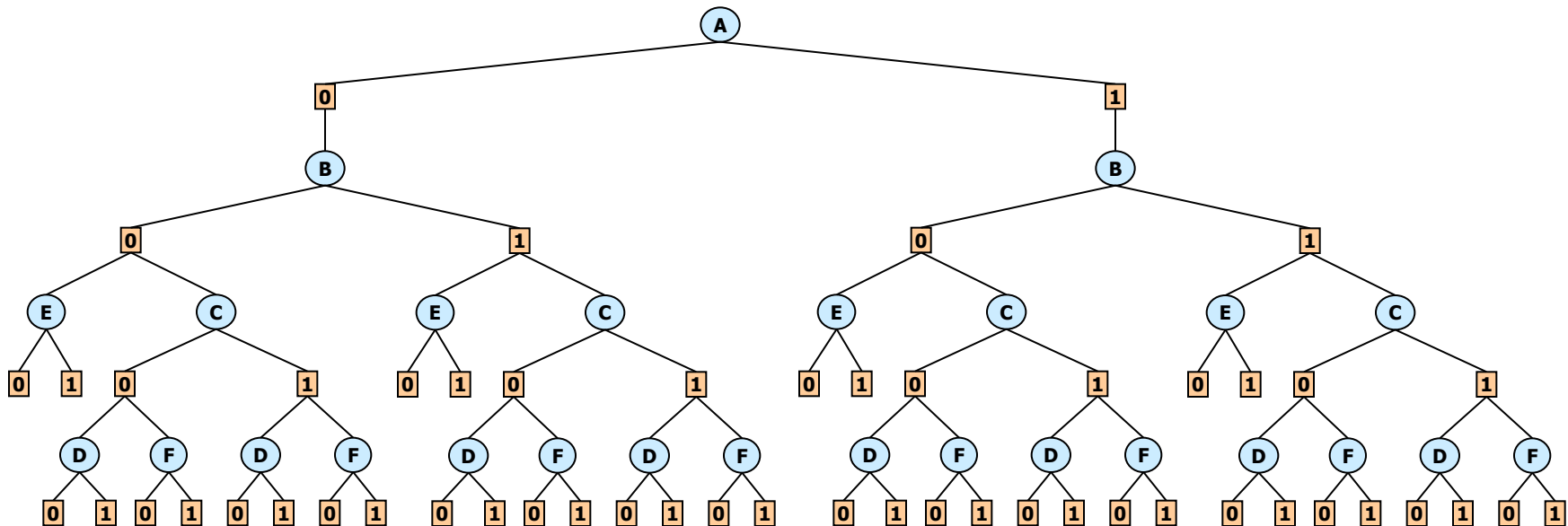
AND

OR

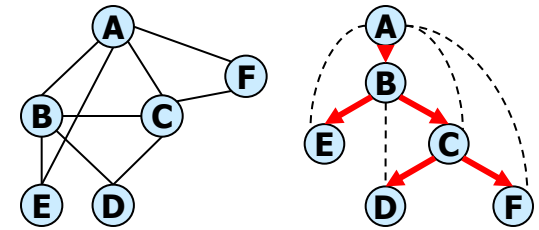
AND

OR

AND



AND/OR vs. OR



OR

AND

OR

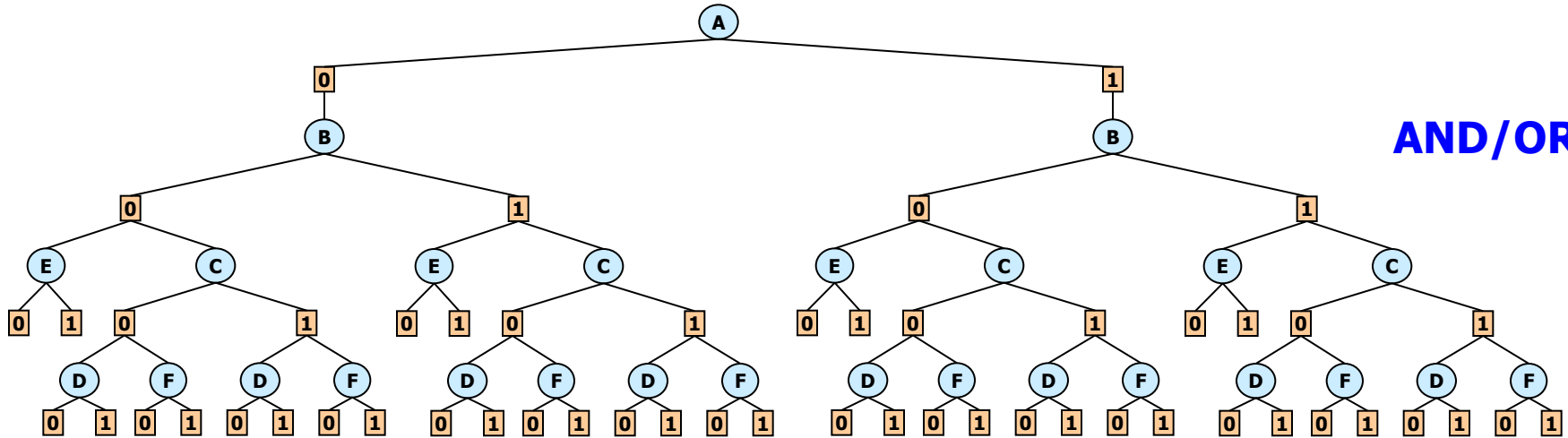
AND

OR

AND

OR

AND



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

A

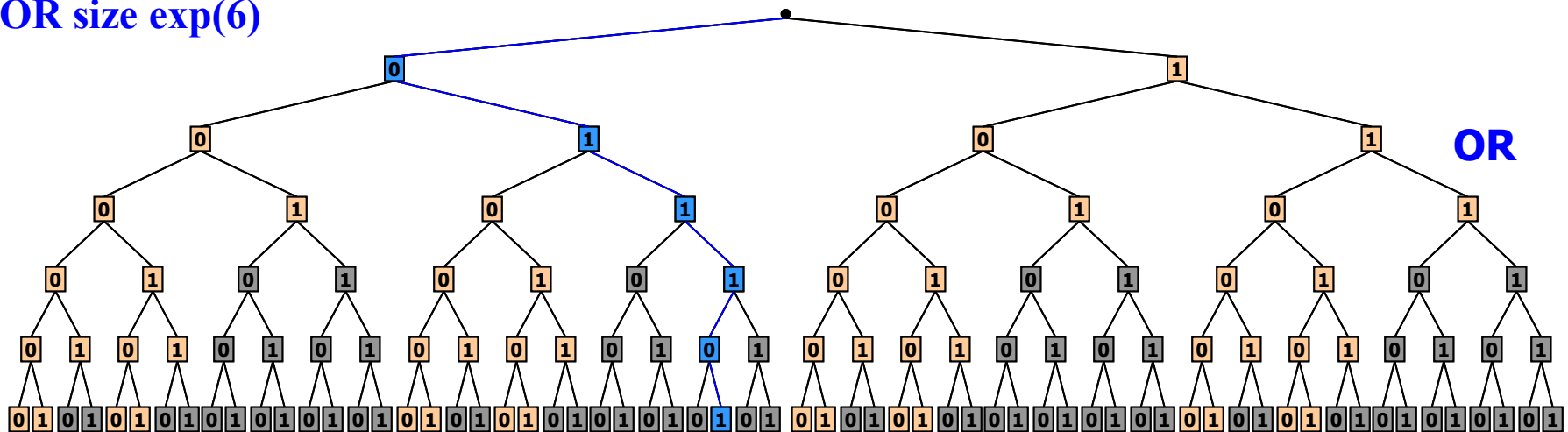
B

E

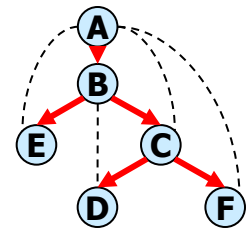
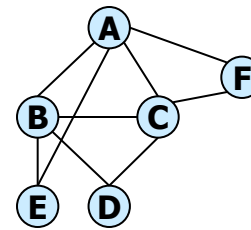
C

D

F



AND/OR vs. OR



OR

AND

OR

AND

OR

AND

OR

AND

A

B

E

C

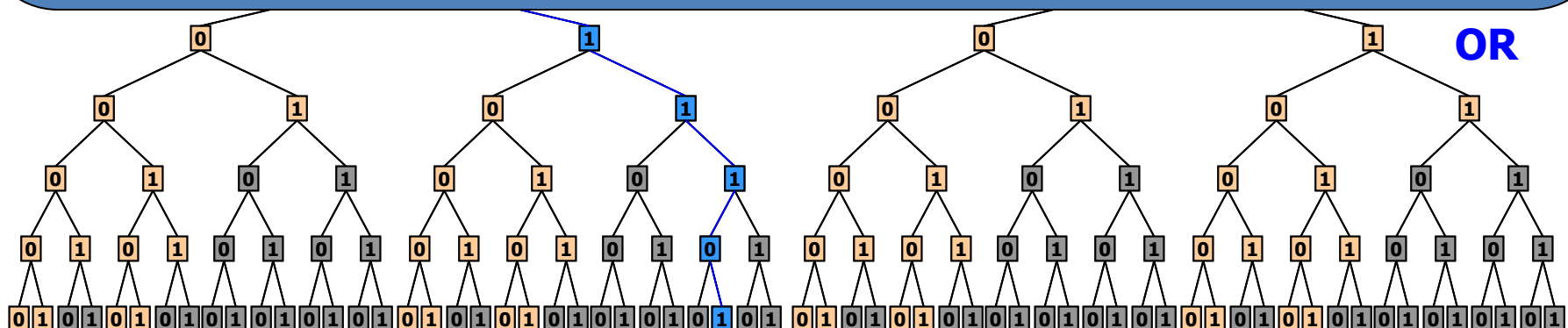
D

F

AND/OR

- *Size of tree $O(nk^h)$*
- *Can be traversed in*
- *Time $O(nk^h)$, Space $O(n)$*
- *All solution trees = all configurations*

OR





Arc weights
Cost of a solution tree
The value function

Arc Weights for AND/OR Trees

$$P(E | A, B)$$

A	B	E=0	E=1
0	0	.4	.6
0	1	.5	.5
1	0	.7	.3
1	1	.2	.8

Evidence: E=0

$$P(B | A)$$

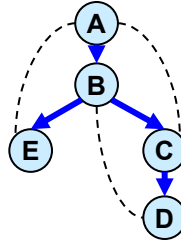
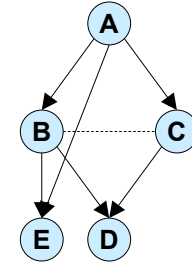
A	B=0	B=1
0	.4	.6
1	.1	.9

$$P(C | A)$$

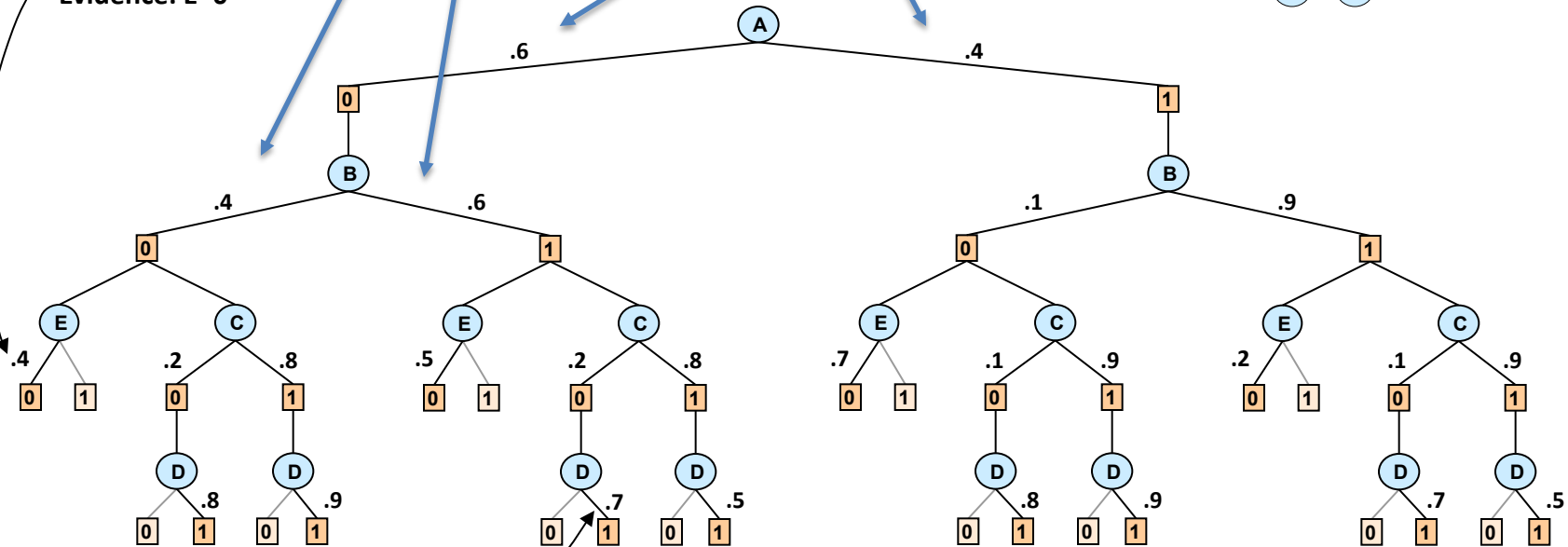
A	C=0	C=1
0	.2	.8
1	.7	.3

$$P(A)$$

A	P(A)
0	.6
1	.4



OR
AND
OR
AND
OR
AND
OR
AND



$$P(D | B, C)$$

B	C	D=0	D=1
0	0	.2	.8
0	1	.1	.9
1	0	.3	.7
1	1	.5	.5

Evidence: D=1

OR to AND arc weight $\langle X, x \rangle$ is the product of factors that all their arguments are just assigned at AND node $X=x$ but not before

Cost of a Solution Tree

$P(E | A, B)$

A	B	E=0	E=1
0	0	.4	.6
0	1	.5	.5
1	0	.7	.3
1	1	.2	.8

Evidence: E=0

$P(B | A)$

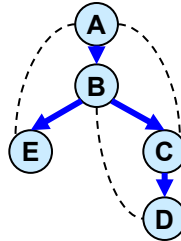
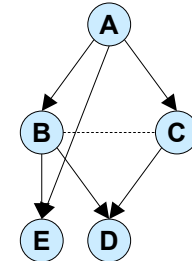
A	B=0	B=1
0	.4	.6
1	.1	.9

$P(C | A)$

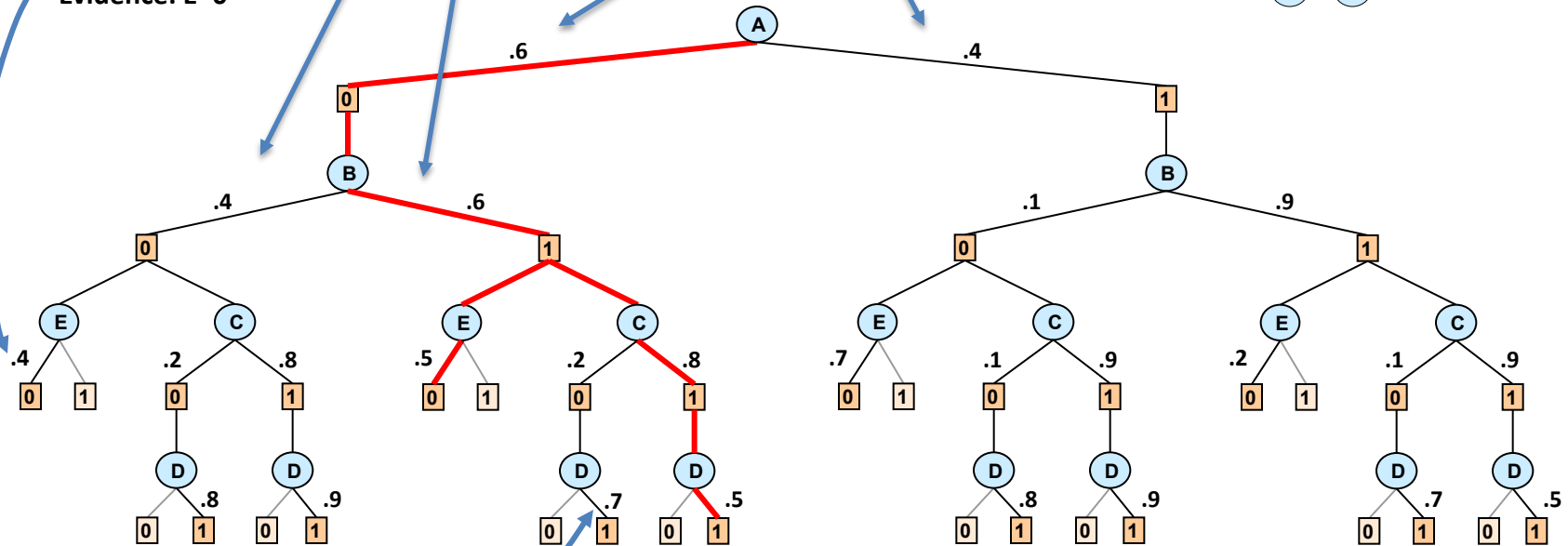
A	C=0	C=1
0	.2	.8
1	.7	.3

$P(A)$

A	P(A)
0	.6
1	.4



OR
AND
OR
AND
OR
AND
OR
AND



$P(D | B, C)$

B	C	D=0	D=1
0	0	.2	.8
0	1	.1	.9
1	0	.3	.7
1	1	.5	.5

Evidence: D=1

A solution tree includes the root and has a single child for any OR node, and all children of any of its AND nodes

Cost of the solution tree: the product of weights on its arcs

Cost of $(A=0, B=1, C=1, D=1, E=0) = 0.6 \cdot 0.6 \cdot 0.5 \cdot 0.8 \cdot 0.5 = 0.0720$

The Value Function for (Probability of Evidence)

$P(E | A, B)$

A	B	E=0	E=1
0	0	.4	.6
0	1	.5	.5
1	0	.7	.3
1	1	.2	.8

Evidence: E=0

$P(B | A)$

A	B=0	B=1
0	.4	.6
1	.1	.9

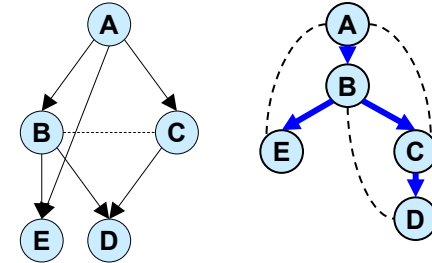
$P(C | A)$

A	C=0	C=1
0	.2	.8
1	.7	.3

$P(A)$

A	P(A)
0	.6
1	.4

$P(D=1, E=0)=?$



OR

AND

OR

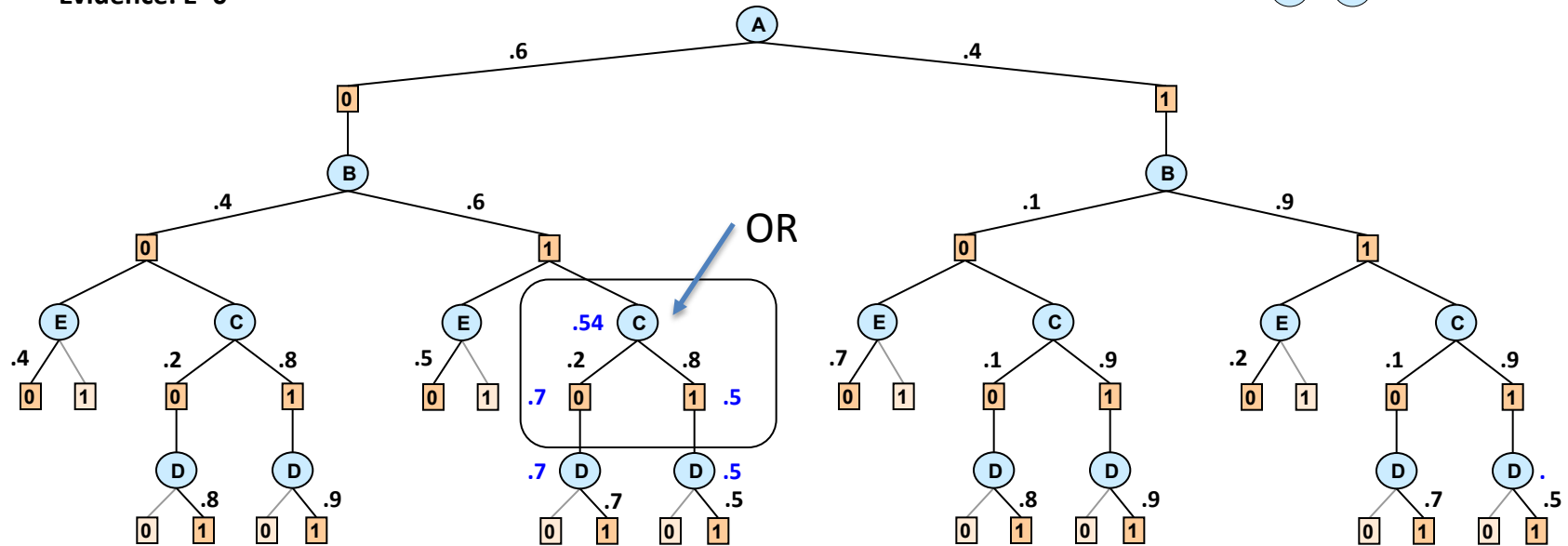
AND

OR

AND

OR

AND



$P(D | B, C)$

B	C	D=0	D=1
0	0	.2	.8
0	1	.1	.9
1	0	.3	.7
1	1	.5	.5

Evidence: D=1

Value of node = updated belief for sub-problem below

AND node: product

OR node: Marginalization by summation

$$\prod_{n' \in \text{children}(n)} v(n')$$

$$\sum_{n' \in \text{children}(n)} w(n, n') v(n')$$

The Value Function (Probability of Evidence)

$P(E | A, B)$

A	B	E=0	E=1
0	0	.4	.6
0	1	.5	.5
1	0	.7	.3
1	1	.2	.8

Evidence: E=0

$P(B | A)$

A	B=0	B=1
0	.4	.6
1	.1	.9

$P(C | A)$

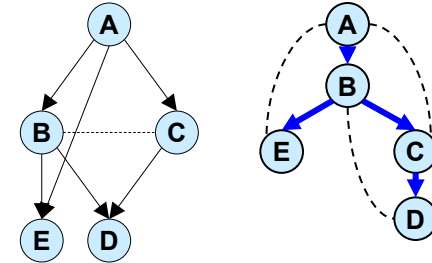
A	C=0	C=1
0	.2	.8
1	.7	.3

$P(A)$

A	P(A)
0	.6
1	.4

$P(D=1, E=0)=?$

.24408



AND .3028

.3028

OR

.8 .8 .9 .9

.7 .7 .5 .5

.8 .8 .9 .9

.7 .7 .5 .5

$P(D | B, C)$

B	C	D=0	D=1
0	0	.2	.8
0	1	.1	.9
1	0	.3	.7
1	1	.5	.5

Evidence: D=1

Value of node = updated belief for sub-problem below

AND node: product

OR node: Marginalization by summation

$$\prod_{n' \in \text{children}(n)} v(n')$$

$$\sum_{n' \in \text{children}(n)} w(n, n') v(n')$$

The Value Function

$$P(E | A, B)$$

A	B	E=0	E=1
0	0	.4	.6
0	1	.5	.5
1	0	.7	.3
1	1	.2	.8

Evidence: E=0

$$P(B | A)$$

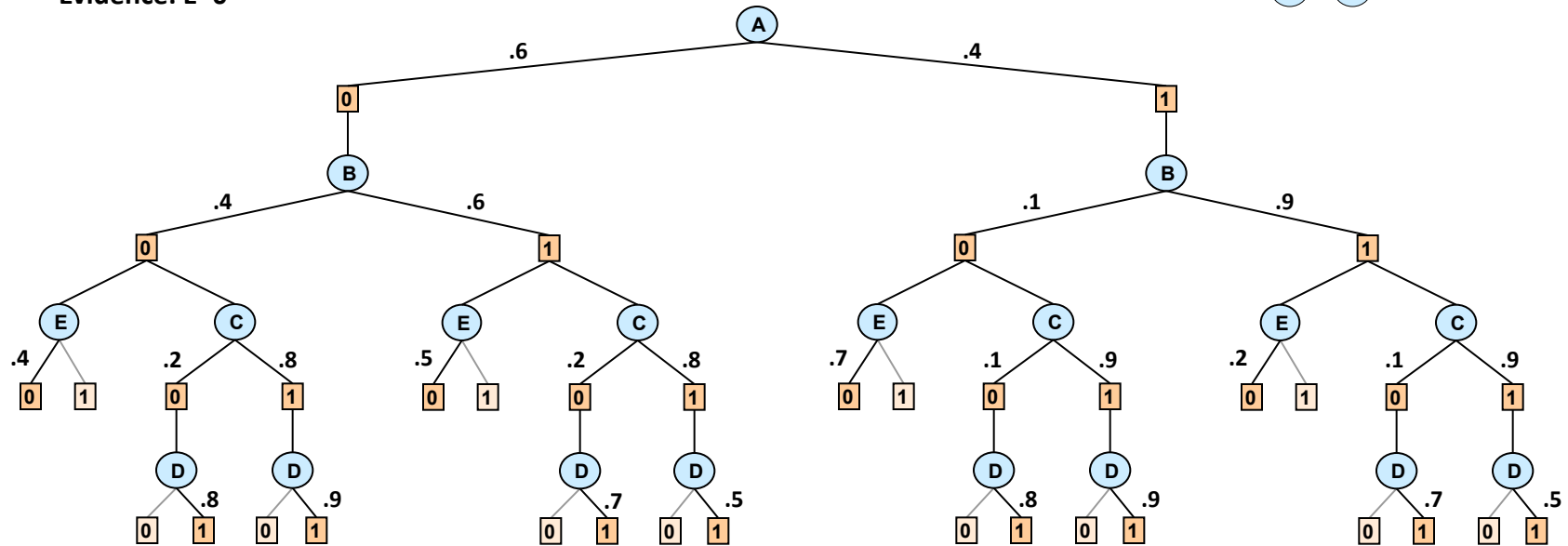
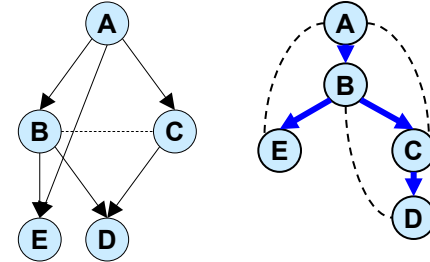
A	B=0	B=1
0	.4	.6
1	.1	.9

$$P(C | A)$$

A	C=0	C=1
0	.2	.8
1	.7	.3

$$P(A)$$

A	P(A)
0	.6
1	.4



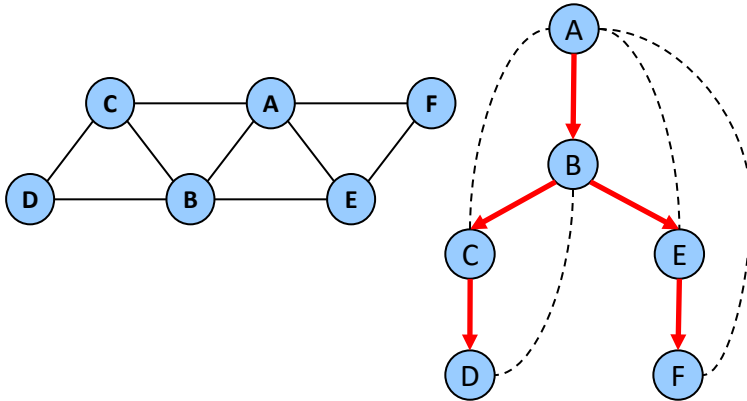
$$P(D | B, C)$$

B	C	D=0	D=1
0	0	.2	.8
0	1	.1	.9
1	0	.3	.7
1	1	.5	.5

Evidence: D=1

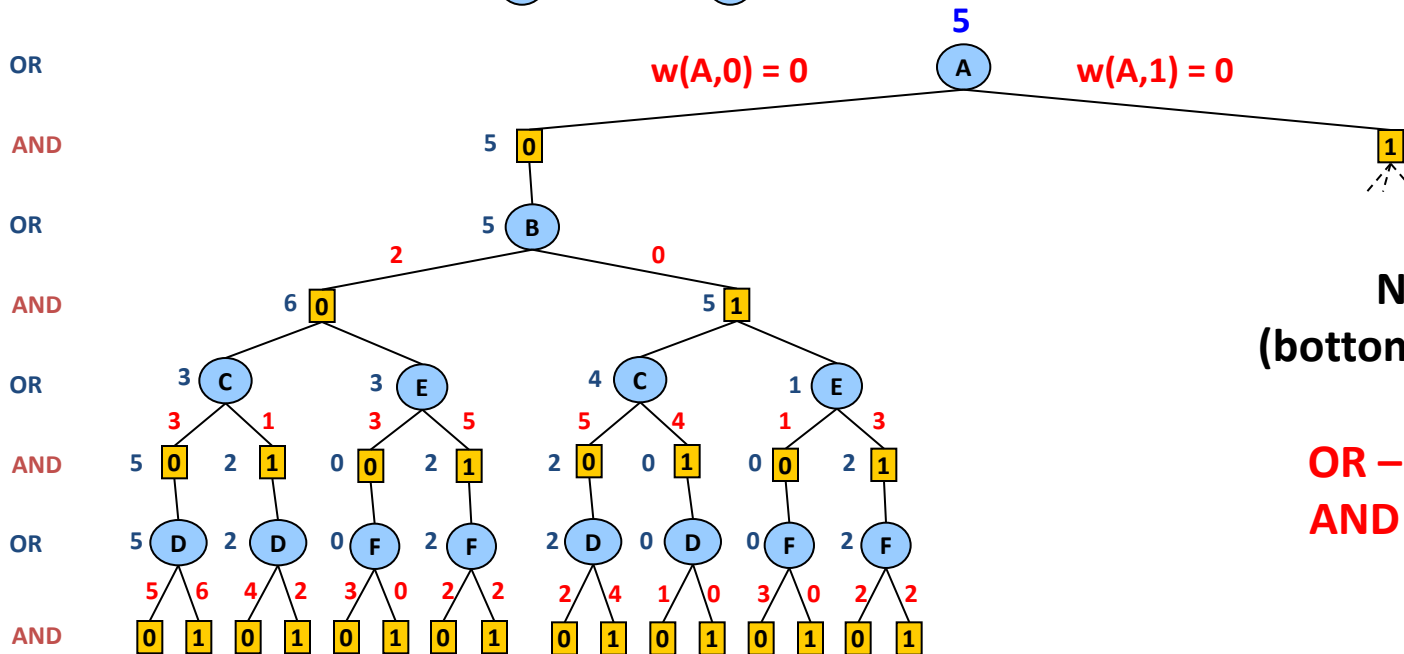
- $V(n)$ is dictated by the query of interest
- $V(n)$ the value of the sub-problem represented by $T(n)$
- For sum-inference it is the probability mass below n
- Can be computed recursively based on child values.

The Value Function for Optimization



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

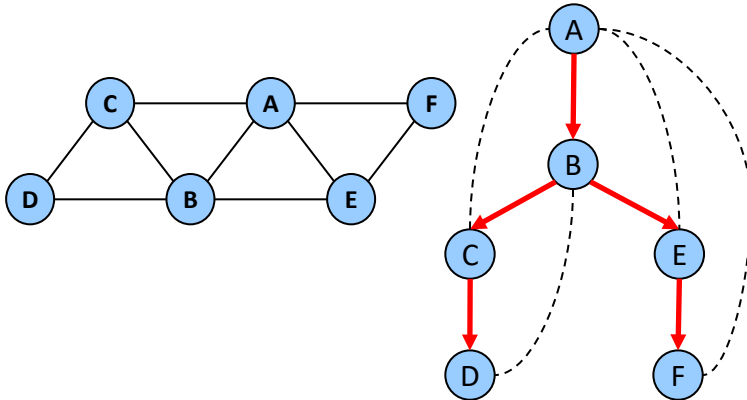
Objective function: $F^* = \min_x \sum_{\alpha} f_{\alpha}(x_{\alpha})$



Node Value
(bottom-up evaluation)

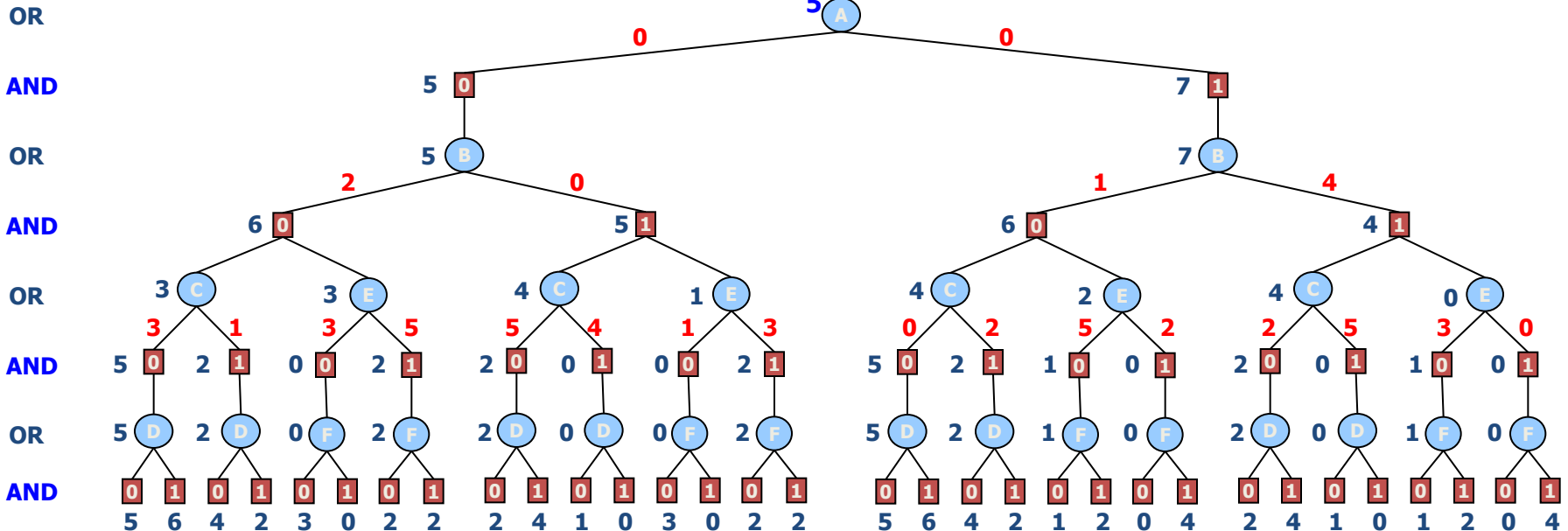
OR – minimization
AND – summation

The Value Function for Optimization



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: $F^* = \min_x \sum_{\alpha} f_{\alpha}(x_{\alpha})$

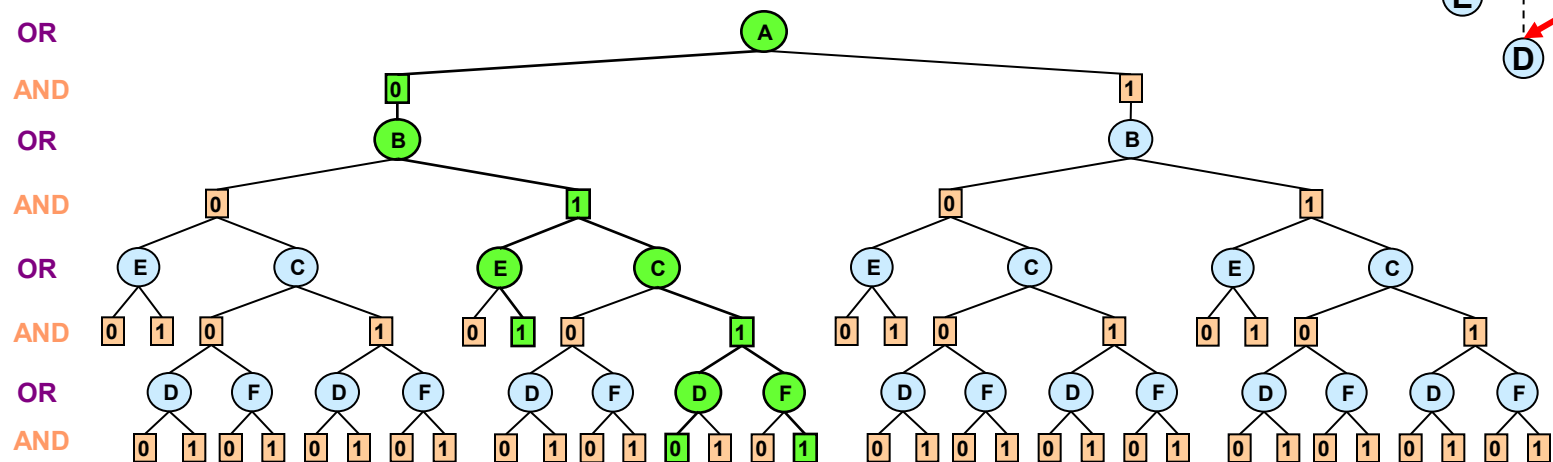
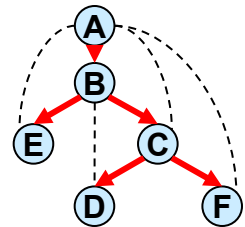
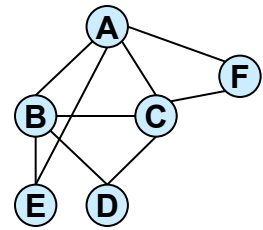


AND node = Combination operator (summation)

OR node = Marginalization operator (minimization)

Summary: AND/OR Search Trees for GMs

- The AND/OR search tree of R relative to a pseudo-tree, T, has:
 - Alternating levels of: **OR** nodes (variables) and **AND** nodes (values)
- Successor function:
 - The successors of **OR nodes X** are all its consistent values along its path
 - The successors of **AND $\langle X, v \rangle$** are all X child variables in T
 - Arc-weight are assigned from the model factors
- A **solution** is a consistent subtree. Its cost, the product of the weights.
- **Query:** compute the value of the root node



Size and Traversal of AND/OR Search Tree

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

k = domain size

h = height of pseudo-tree

n = number of variables

w^* = treewidth

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

AND/OR vs. OR Spaces

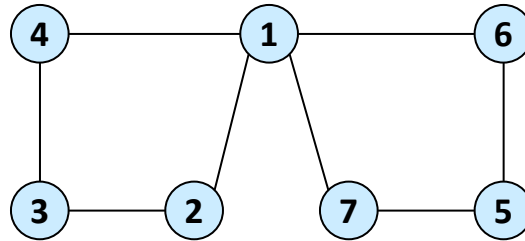
width	height	OR space		AND/OR space		
		Time (sec.)	Nodes	Time (sec.)	AND nodes	OR nodes
5	10	3.15	2,097,150	0.03	10,494	5,247
4	9	3.13	2,097,150	0.01	5,102	2,551
5	10	3.12	2,097,150	0.03	8,926	4,463
4	10	3.12	2,097,150	0.02	7,806	3,903
5	13	3.11	2,097,150	0.10	36,510	18,255

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

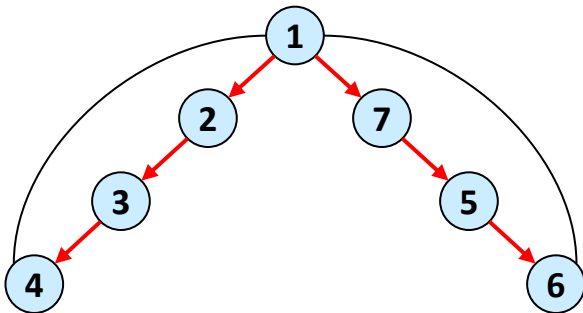
Pseudo Trees

A **pseudo-tree** of a graph is a tree spanning its nodes, where all arcs in the graph not in the tree are back-arcs

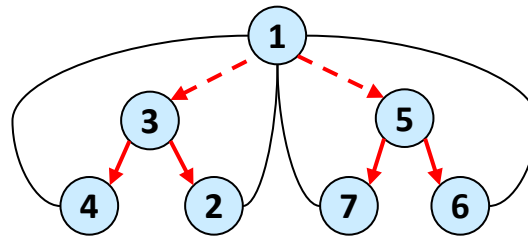
$$h \leq w^* \log n$$



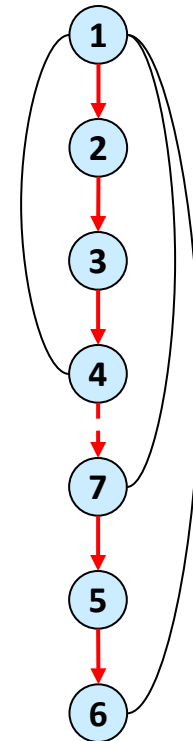
(a) Graph



(b) DFS tree
height=3

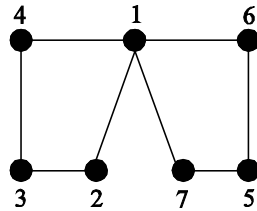


(c) Pseudo tree
height=2

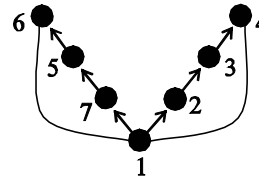


(d) Chain
height=6

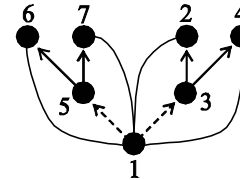
From DFS-Trees to Pseudo-Trees



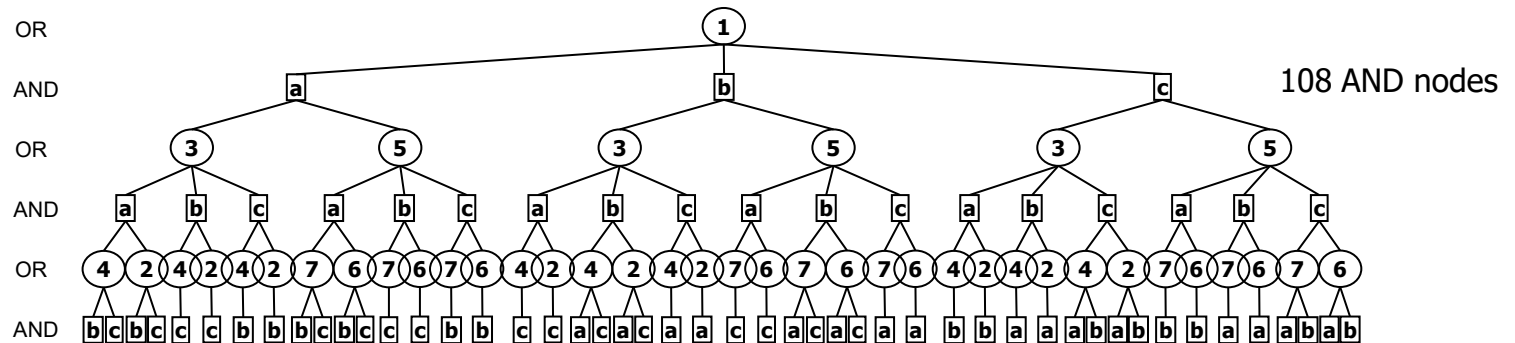
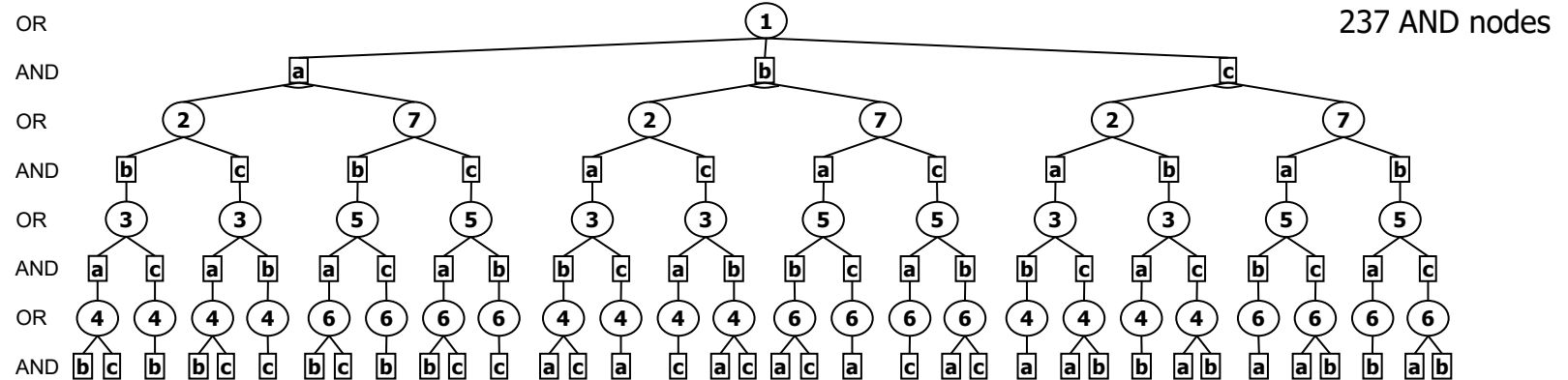
(a)



(b)



(c)



AND/OR Search-Tree Properties

(k = domain size, h = pseudo-tree height. n = number of variables)

- **Theorem:** Any AND/OR search tree based on a pseudo-tree is sound and complete (expresses all and only solutions)
- **Theorem:** Size of AND/OR search tree is $O(n k^h)$
Size of OR search tree is $O(k^n)$
- **Theorem:** Size of AND/OR search tree can be bounded by $O(\exp(w^* \log n))$
- When the pseudo-tree is a chain we get an OR space

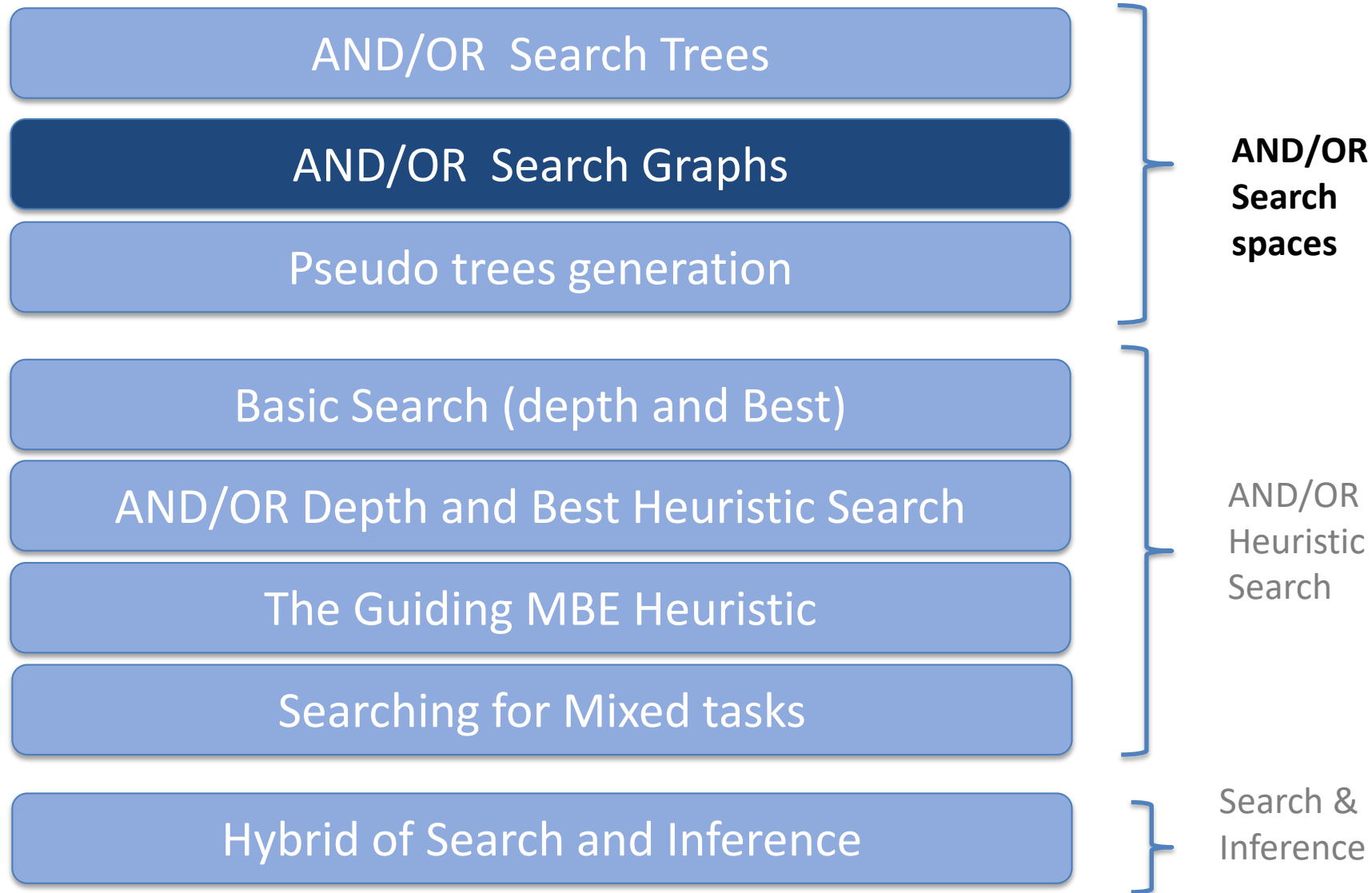
Summary: Queries and Value of Nodes

- $V(n)$ is the value of the tree $T(n)$ for the task:
 - **Max-Inference**: $v(n)$ is the optimal solution in $T(n)$
 - **Sum-Inference**: $v(n)$ is probability of evidence in $T(n)$.
 - **Mixed-Inference**: $v(n)$ is the marginal map in $T(n)$.
 - **Mixed-Inference**: $v(n)$ is the max-expect utility in $T(n)$ of ID.
- **Goal**: compute the value of the root node recursively traversing the AND/OR tree.

Complexity of searching depth-first is

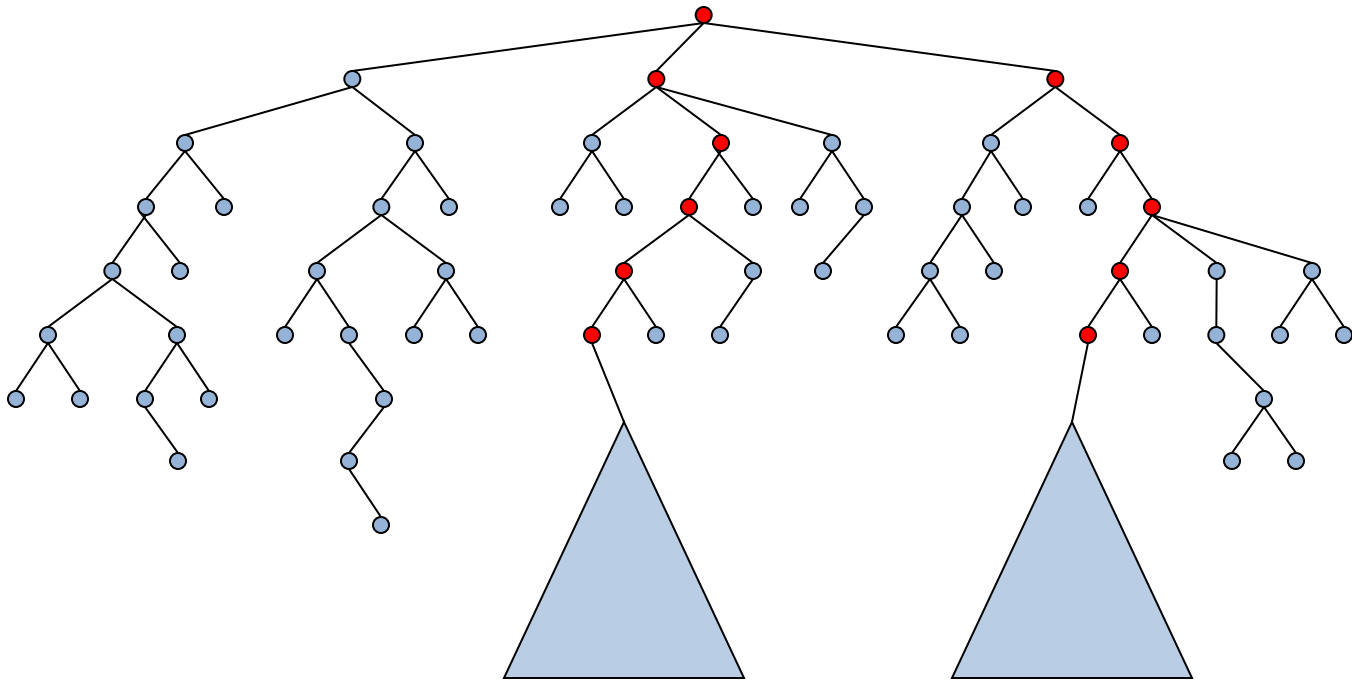
- Space: $O(n)$
- Time: $O(nk^h)$
- Time: $O(k^{w \cdot \log n})$

Outline: Search



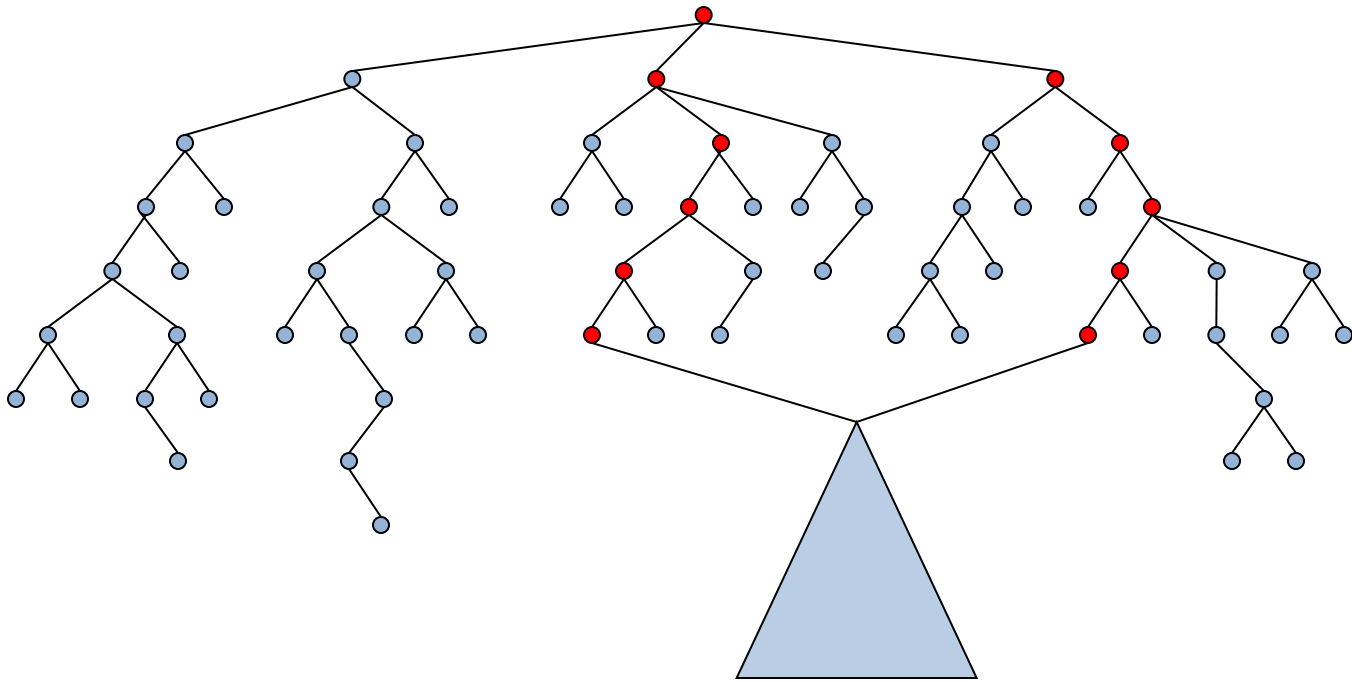
From Search Trees to Search Graphs

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

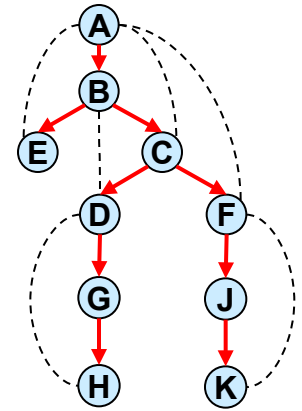
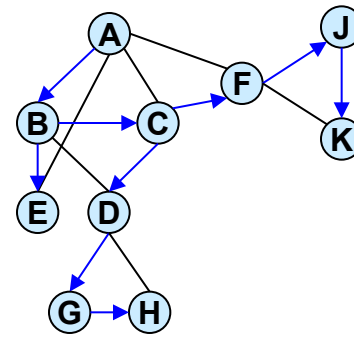


From Search Trees to Search Graphs

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



AND/OR Tree



OR

AND

OR

AND

OR

AND

OR

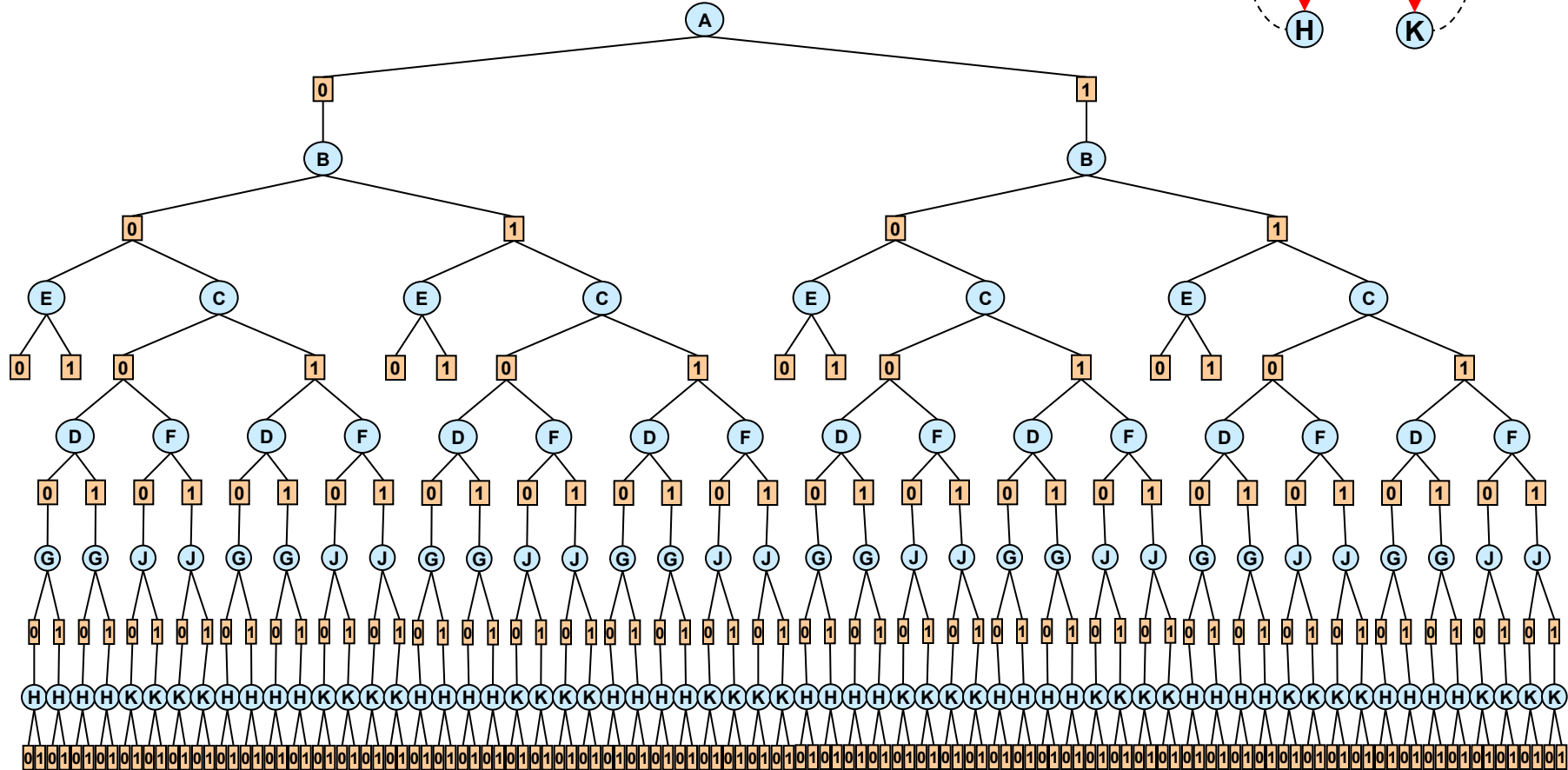
AND

OR

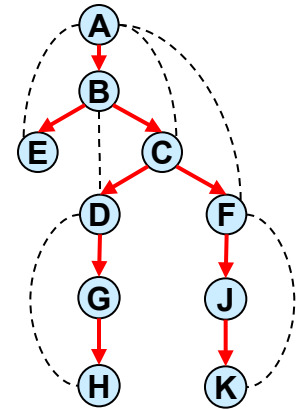
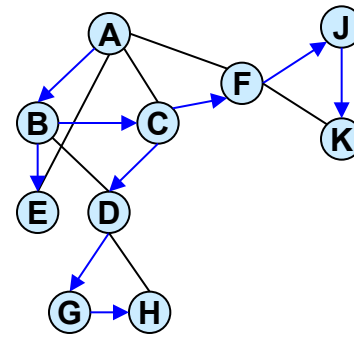
AND

OR

AND



AND/OR Graph



OR

AND

OR

AND

OR

AND

OR

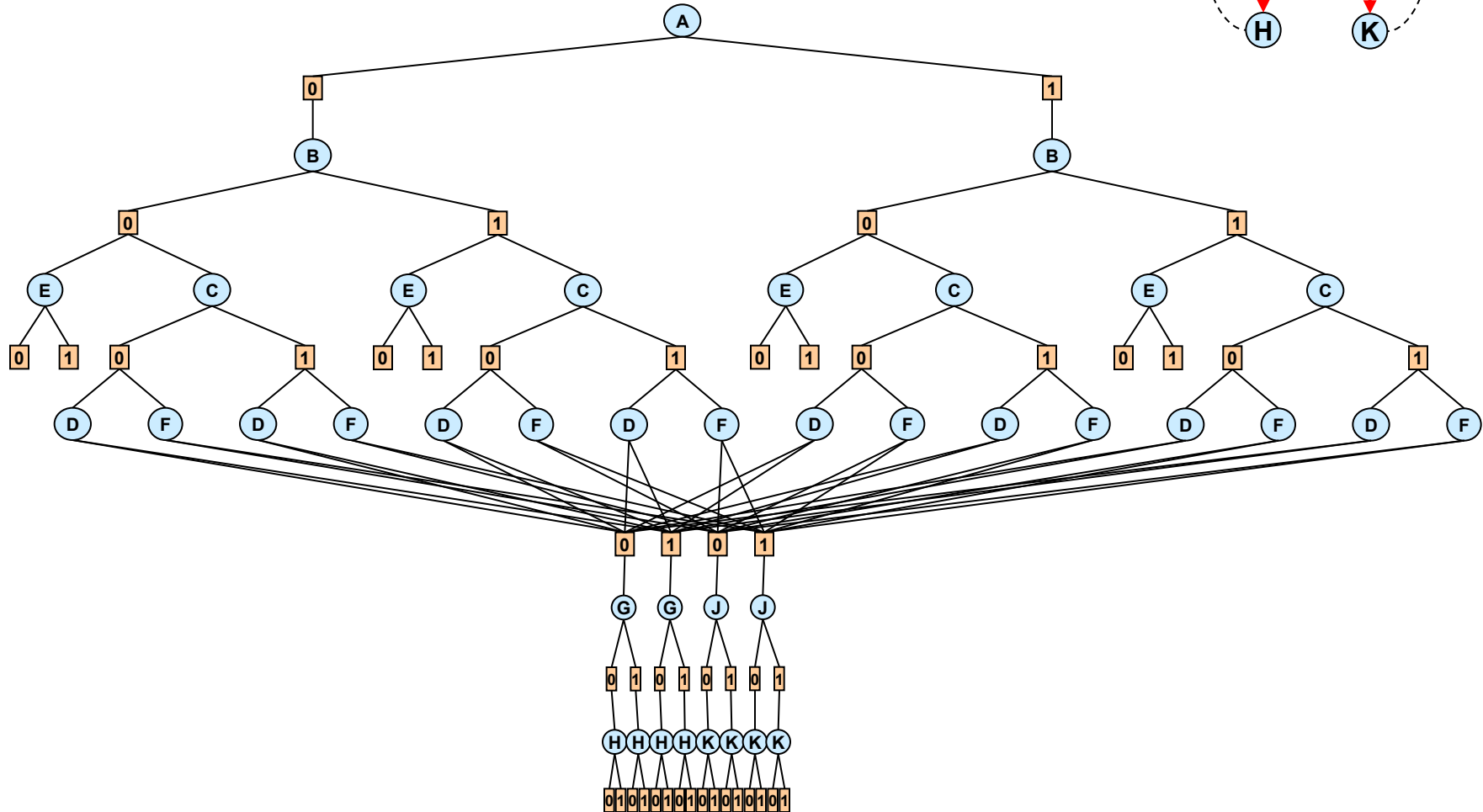
AND

OR

AND

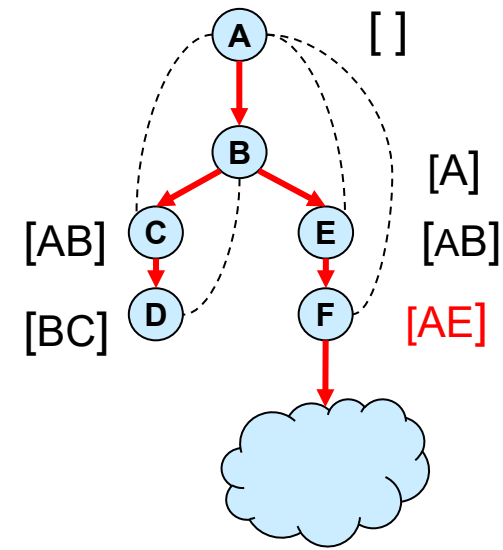
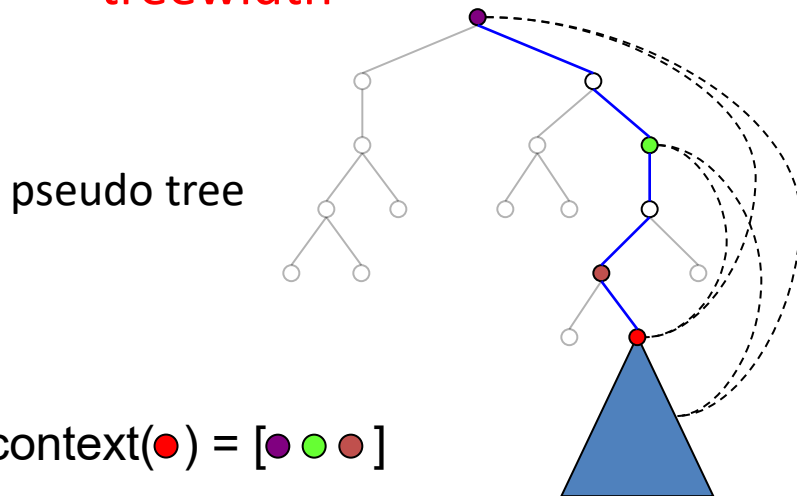
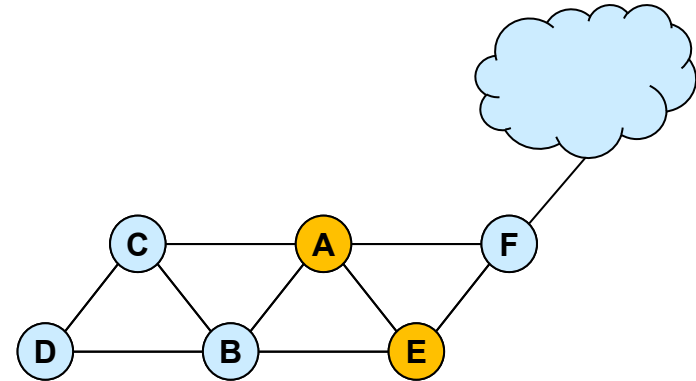
OR

AND



Merging Based on Context

- context (X) = ancestors of X in pseudo tree, connected to X, or to descendants of X
- context (X) = parents in the induced graph
- max |context| = induced width = treewidth



Context-Based Minimal AND/OR Search Graph

Definition 7.2.13 (context minimal AND/OR search graph) *The AND/OR search graph of M guided by a pseudo-tree T that is closed under context-based merge operator, is called the context minimal AND/OR search graph and is denoted by $C_T(R)$.*

AND/OR Tree DFS Algorithm (Value=Sum-Product)

$P(E | A, B)$

A	B	E=0	E=1
0	0	.4	.6
0	1	.5	.5
1	0	.7	.3
1	1	.2	.8

Evidence: E=0

$P(B | A)$

A	B=0	B=1
0	.4	.6
1	.1	.9

$P(C | A)$

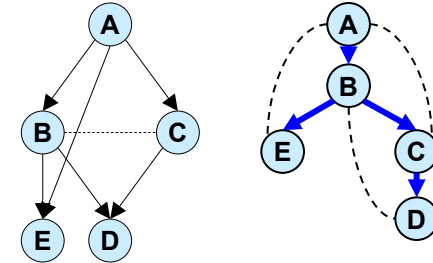
A	C=0	C=1
0	.2	.8
1	.7	.3

$P(A)$

A	P(A)
0	.6
1	.4

Result: $P(D=1, E=0)$

.24408



OR

AND

OR

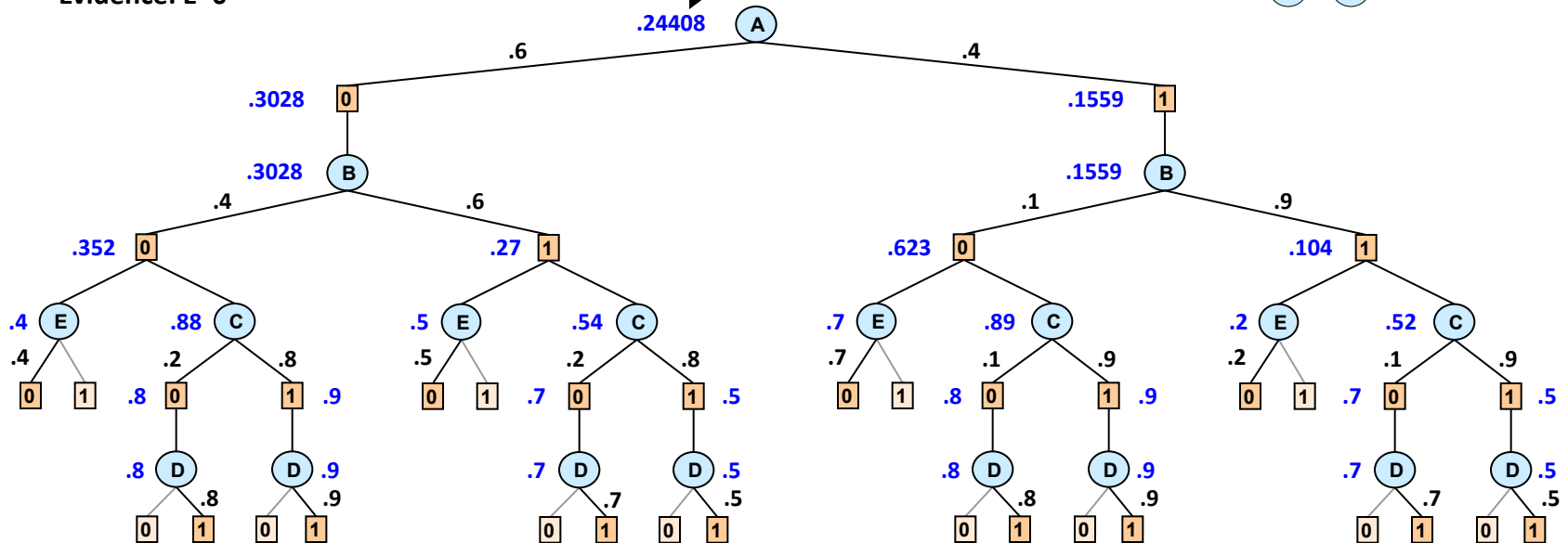
AND

OR

AND

OR

AND



$P(D | B, C)$

B	C	D=0	D=1
0	0	.2	.8
0	1	.1	.9
1	0	.3	.7
1	1	.5	.5

Evidence: D=1

AND/OR Search Graph (Value=Sum-Product)

$P(E | A, B)$

A	B	E=0	E=1
0	0	.4	.6
0	1	.5	.5
1	0	.7	.3
1	1	.2	.8

Evidence: E=0

$P(B | A)$

A	B=0	B=1
0	.4	.6
1	.1	.9

$P(C | A)$

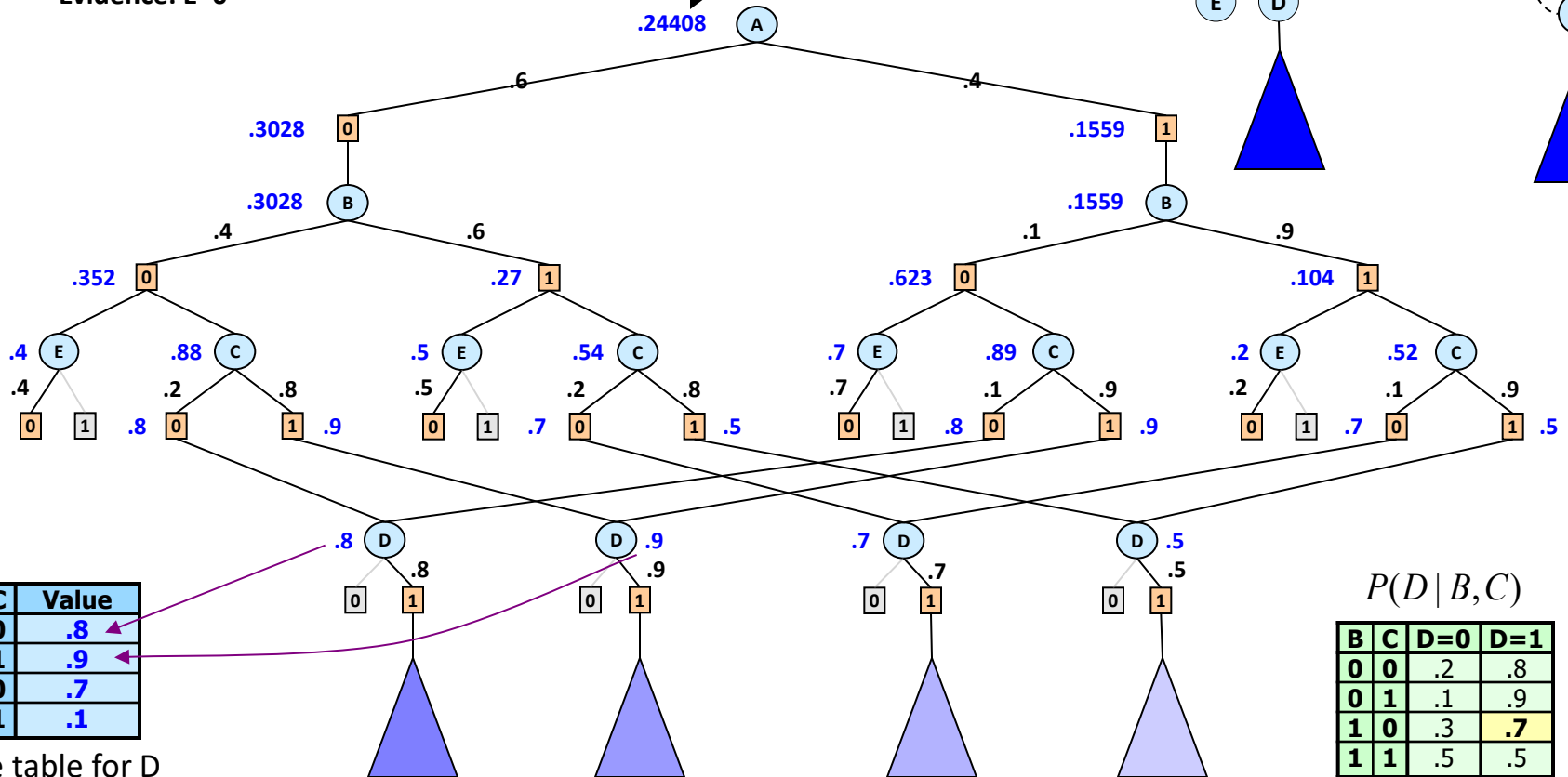
A	C=0	C=1
0	.2	.8
1	.7	.3

$P(A)$

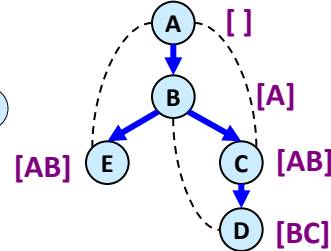
A	P(A)
0	.6
1	.4

Result: $P(D=1, E=0)$

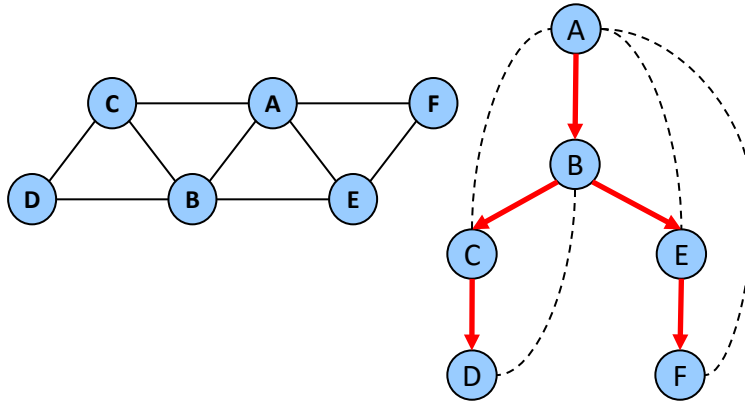
.24408



Context



AND/OR Search Graph (Optimization)



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: $F^* = \min_x \sum_{\alpha} f_{\alpha}(x_{\alpha})$

OR

AND

OR

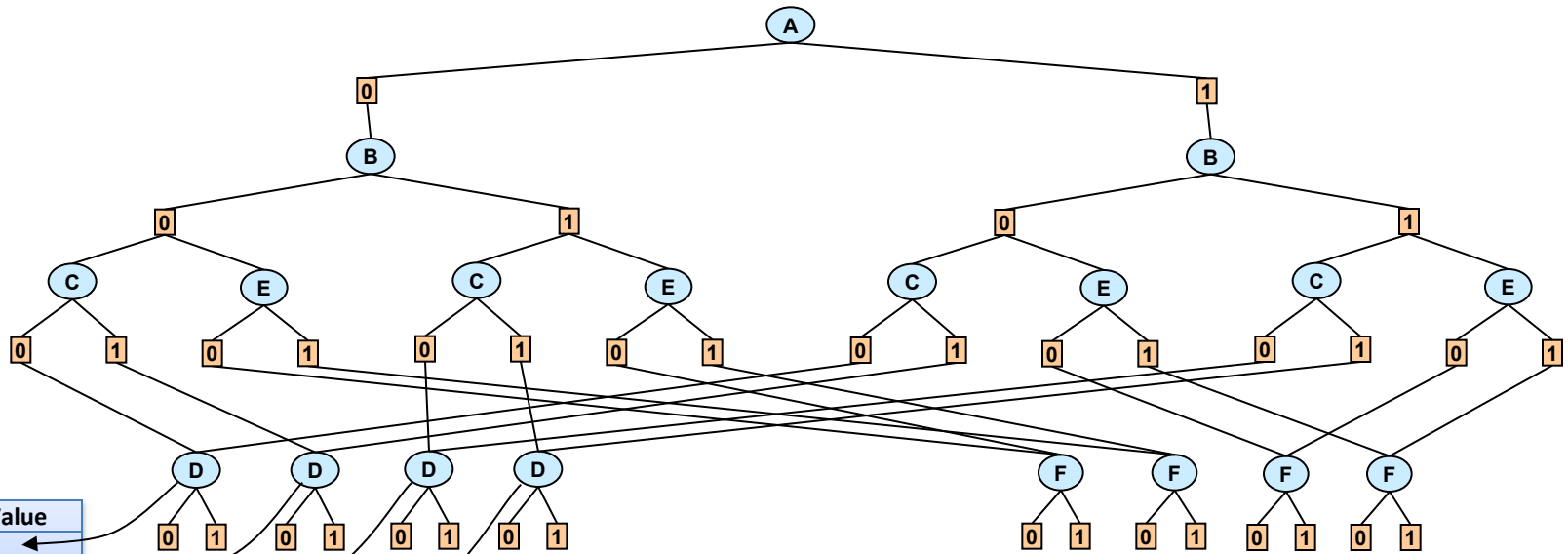
AND

OR

AND

OR

AND

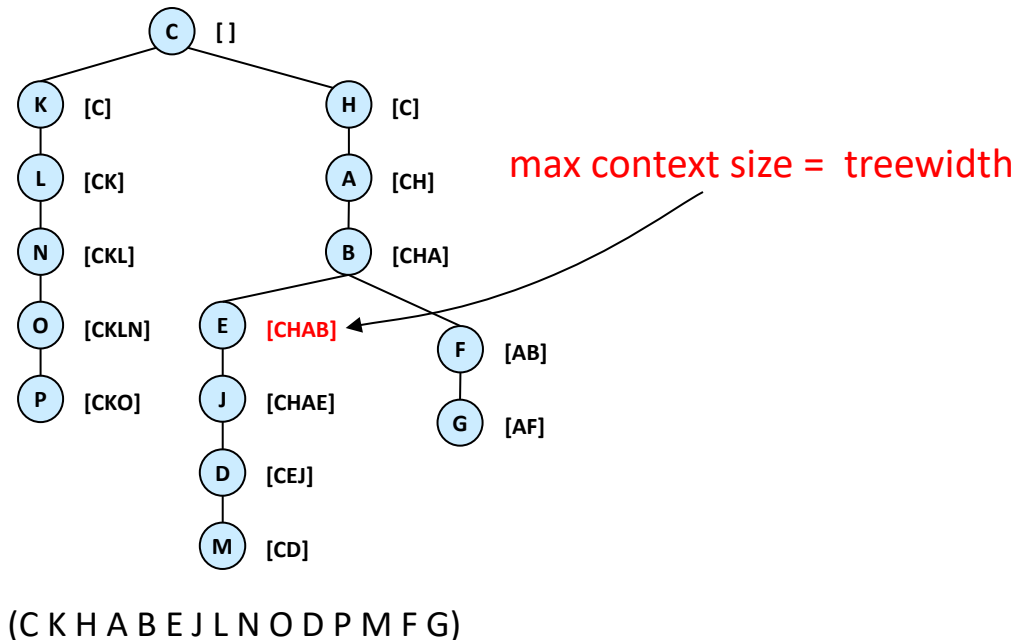
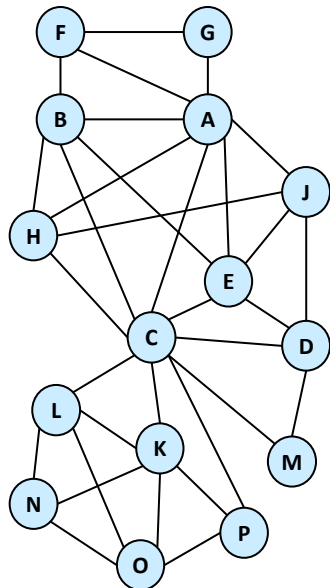


B	C	Value
0	0	←
0	1	←
1	0	←
1	1	←

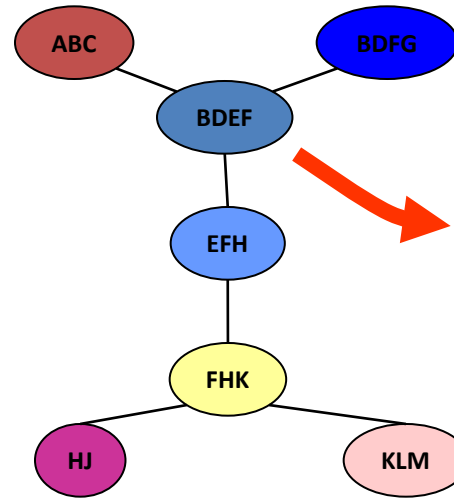
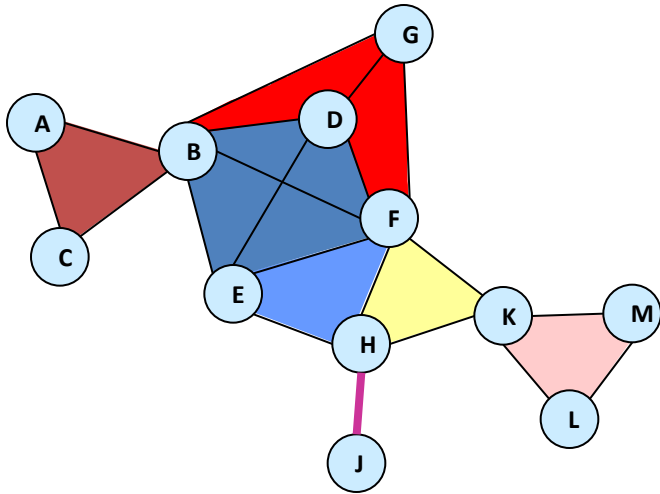
Context minimal AND/OR search graph

How Big Is The Context?

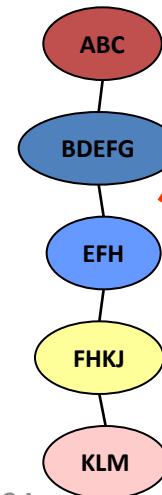
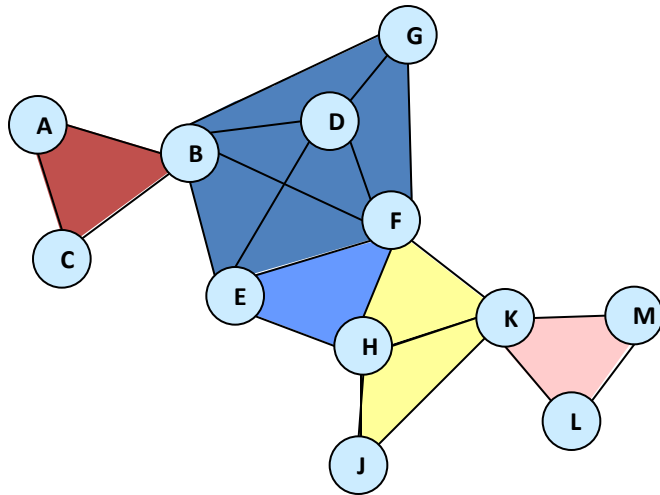
- Theorem:** The maximum context-size of a pseudo-tree equals the **treewidth** along the pseudo tree.



Treewidth vs. Pathwidth

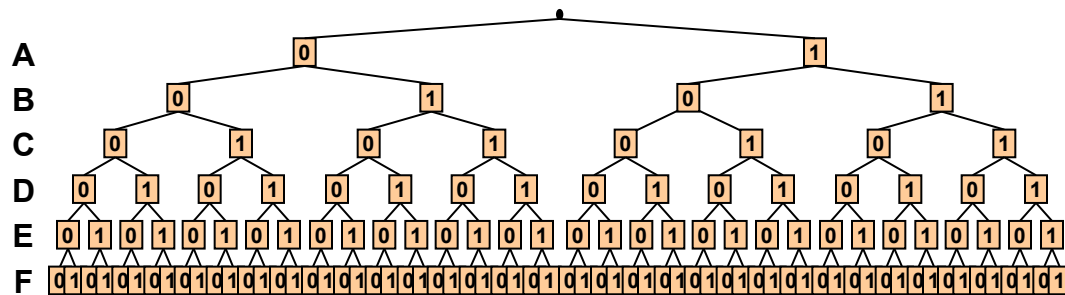
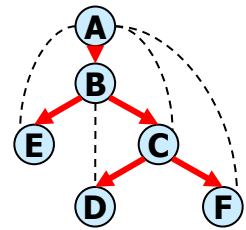


treewidth = 3
= (max cluster size) - 1



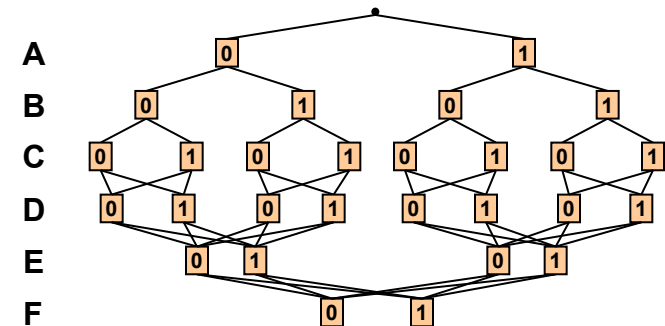
pathwidth = 4
= (max cluster size) - 1

All Four Search Spaces



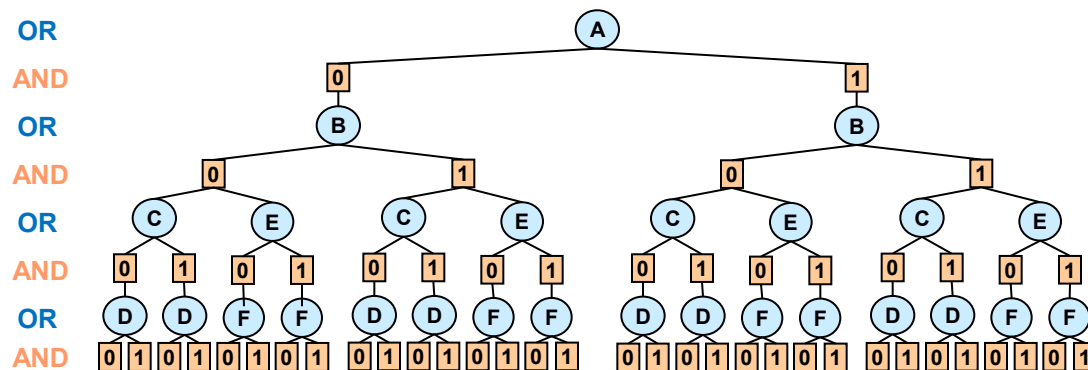
Full OR search tree

126 nodes



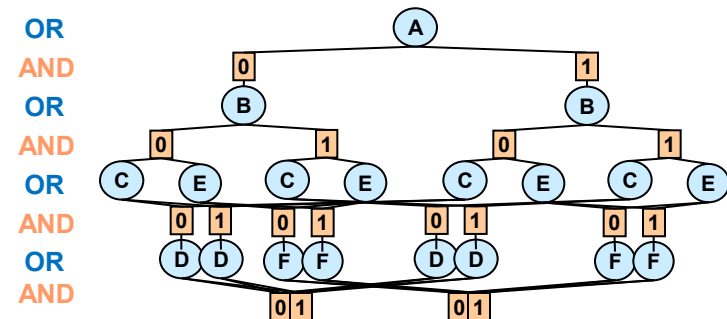
Context minimal OR search graph

28 nodes



Full AND/OR search tree

54 AND nodes



Context minimal AND/OR search graph

18 AND nodes

k = domain size

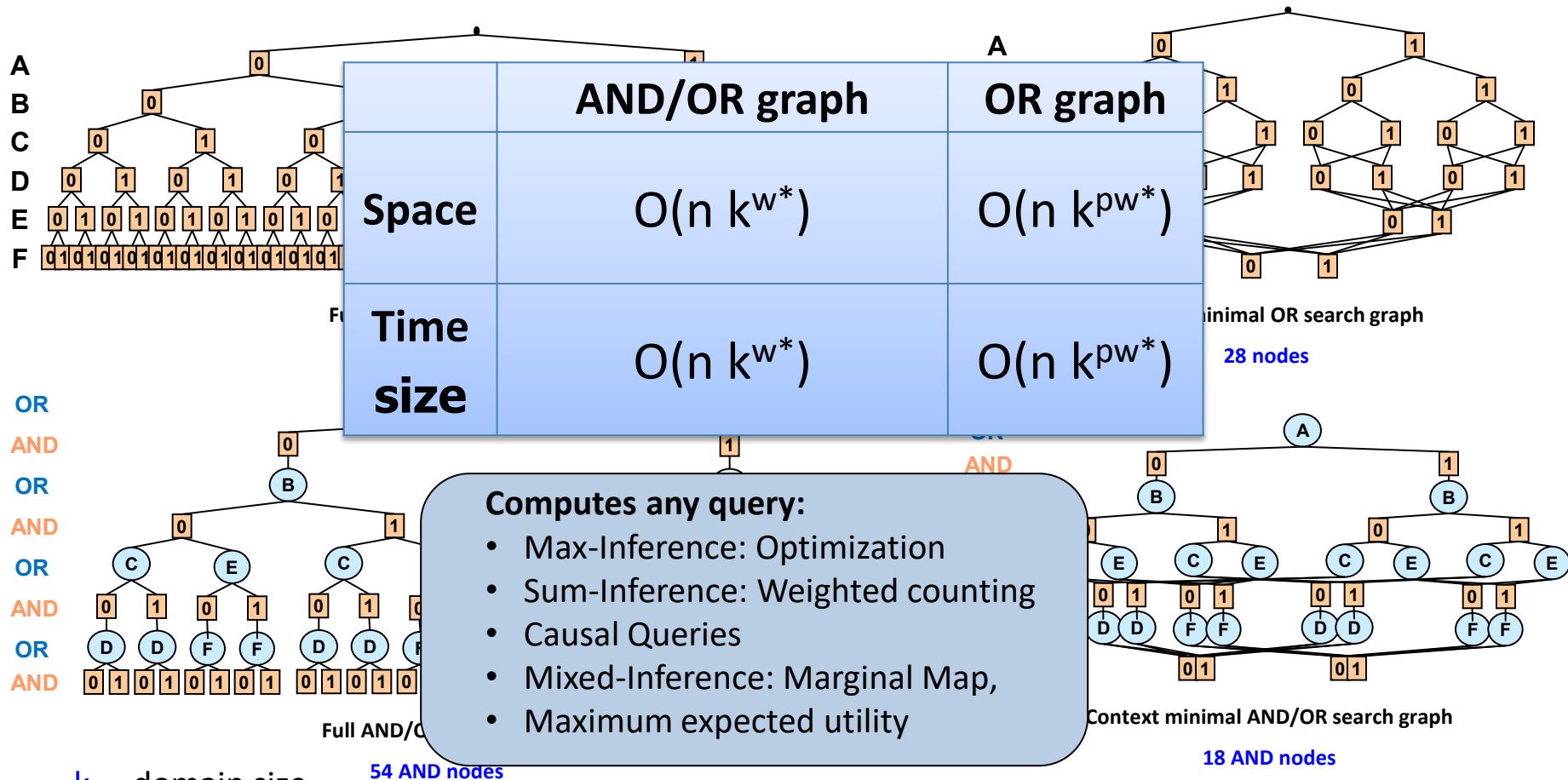
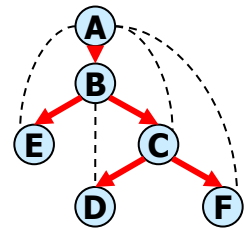
n = number of variables

w^* = treewidth

pw^* = pathwidth

Any query is best computed
over the context-minimal AND/OR space

All Four Search Spaces



k = domain size

n = number of variables

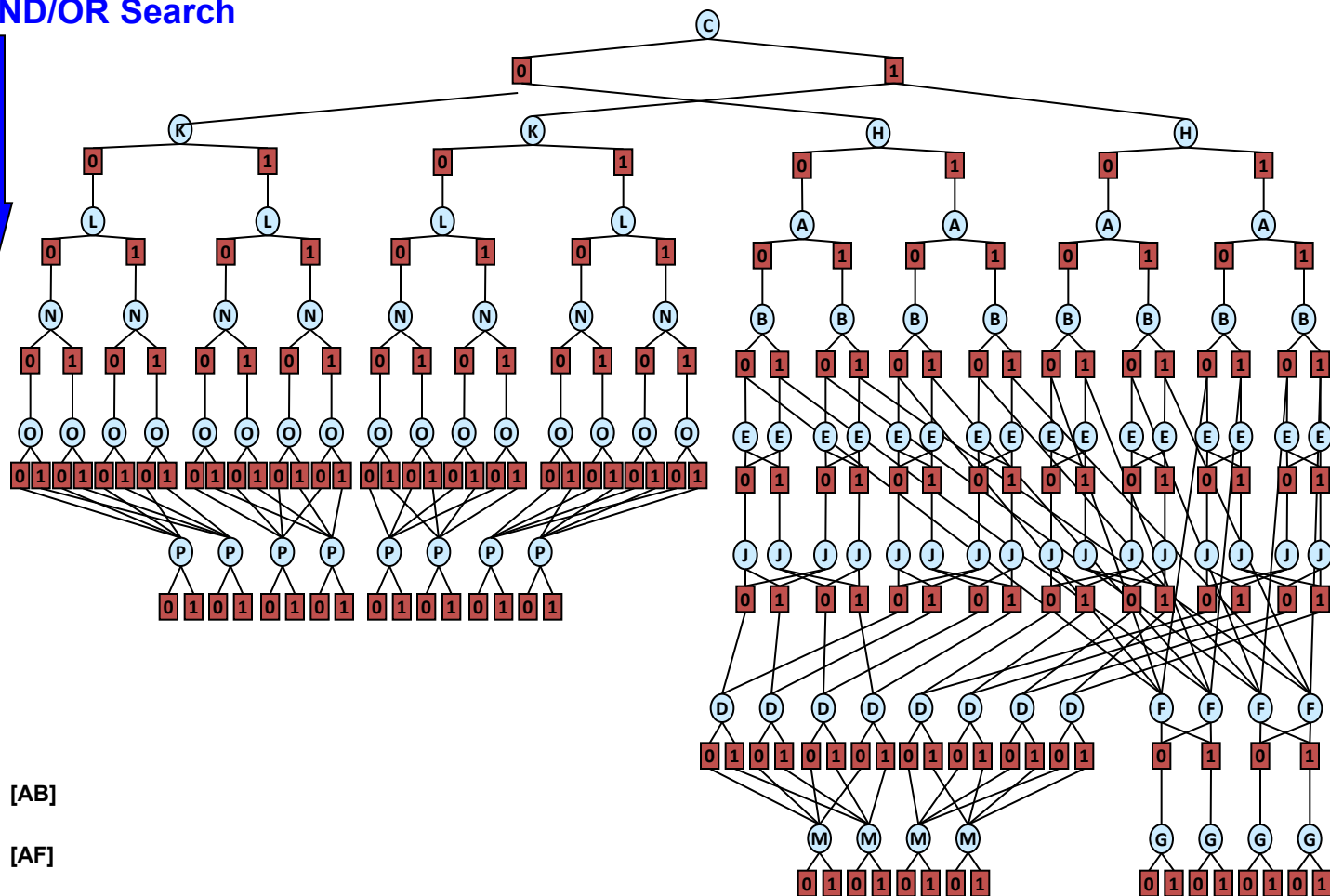
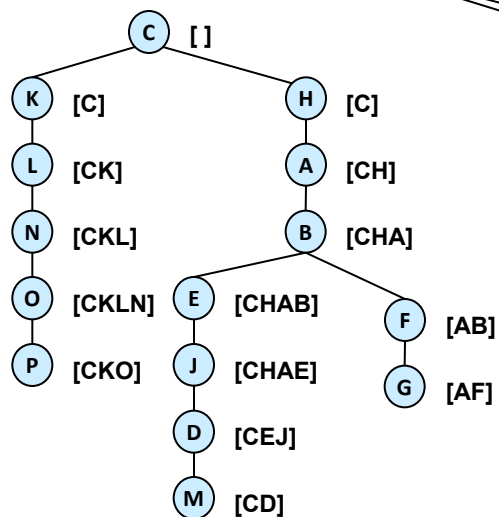
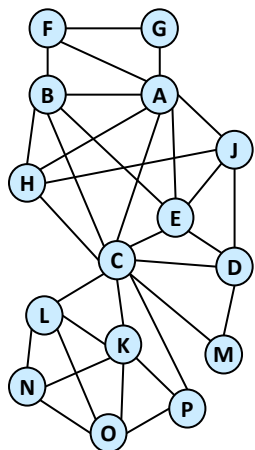
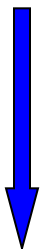
w^* = treewidth

pw^* = pathwidth

**Any query is best computed
over the context-minimal AND/OR space**

AND/OR Search and Variable Elimination

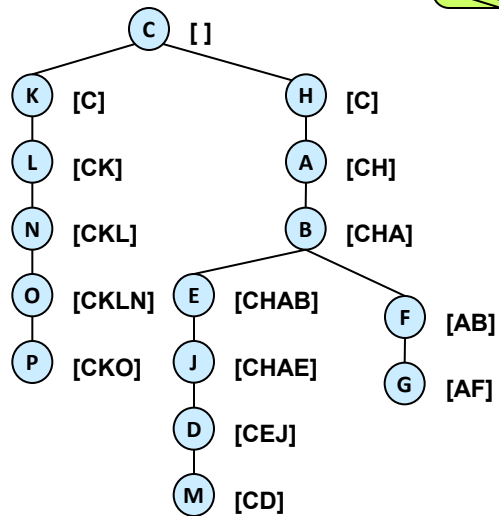
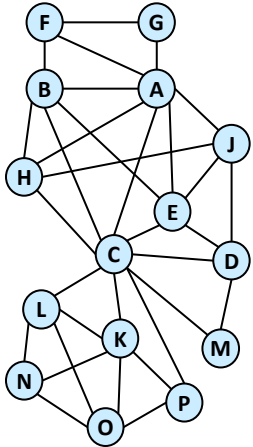
AND/OR Search



(CKHABEJLNODPMFG)

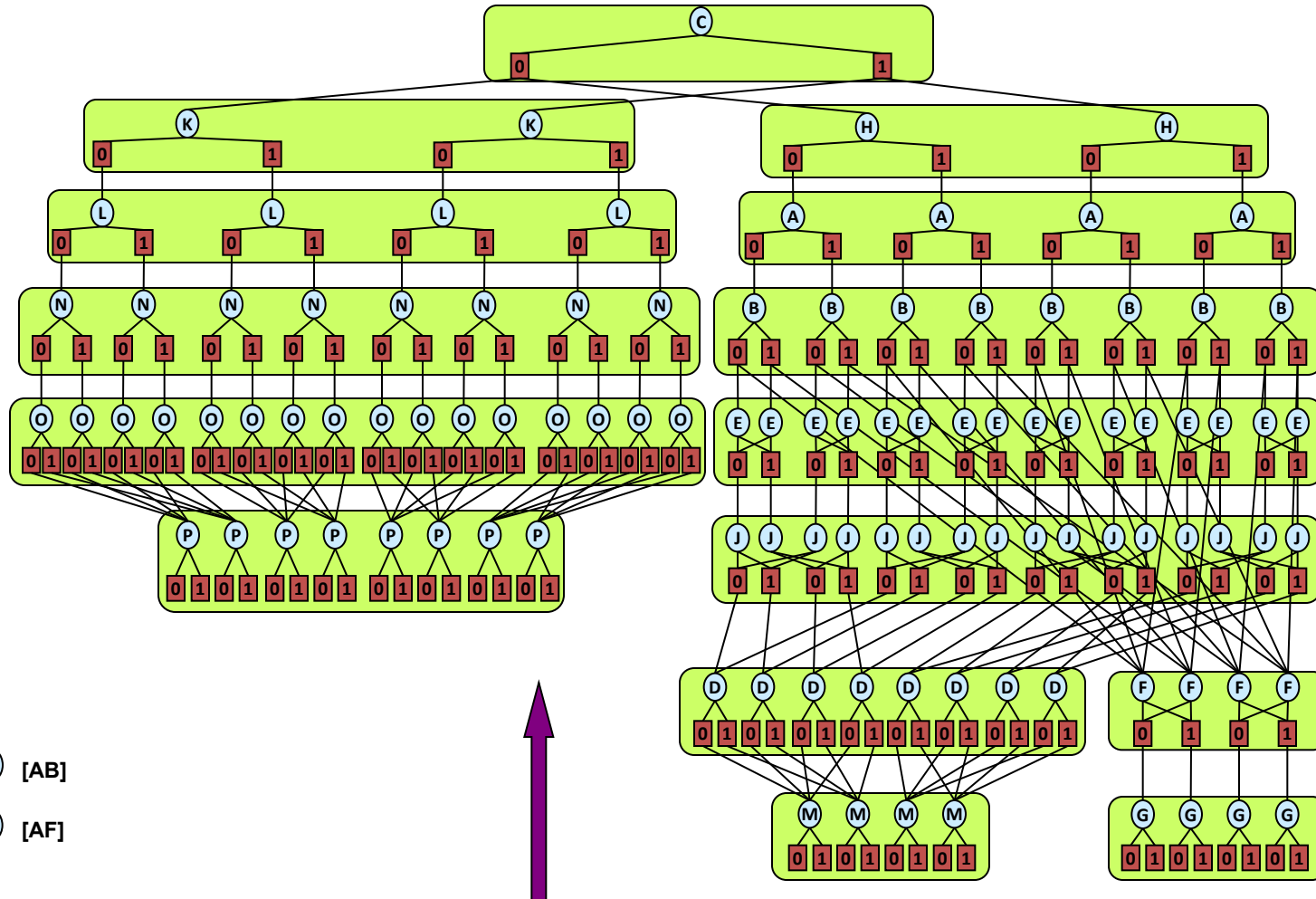
Dechter & Ihler

AND/OR Search and Variable Elimination



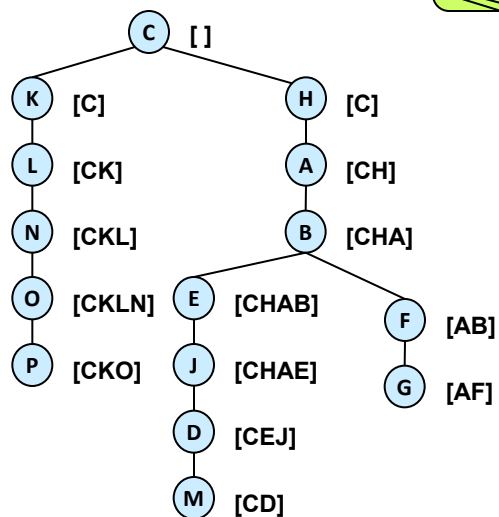
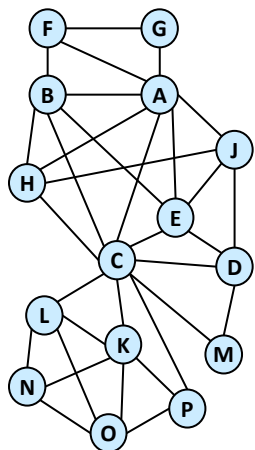
(CKHABEJLNODPMFG)

Dechter & Ihler



Variable Elimination

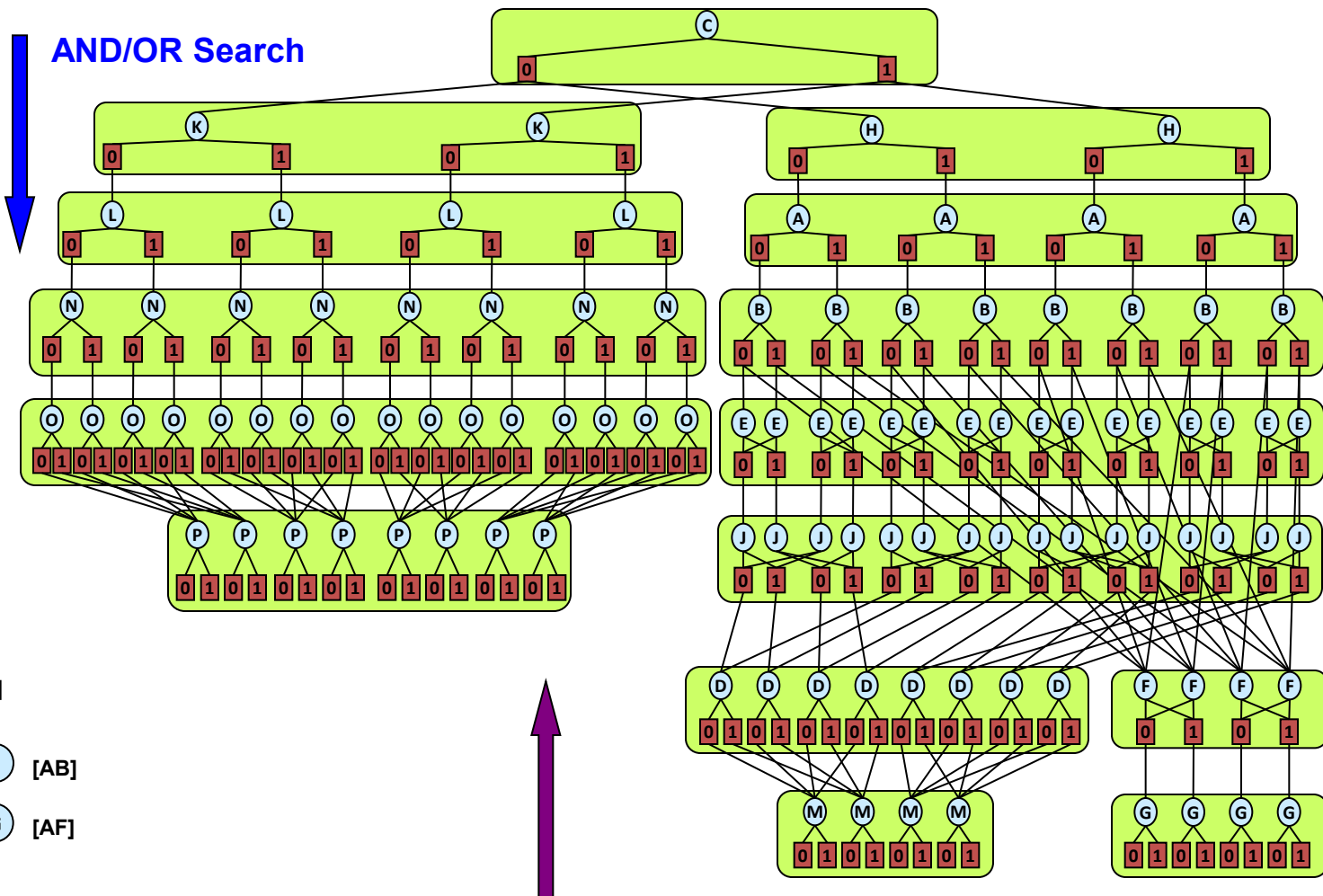
AND/OR Search and Variable Elimination



(CKHABEJLNODPMFG)

Dechter & Ihler

AND/OR Search



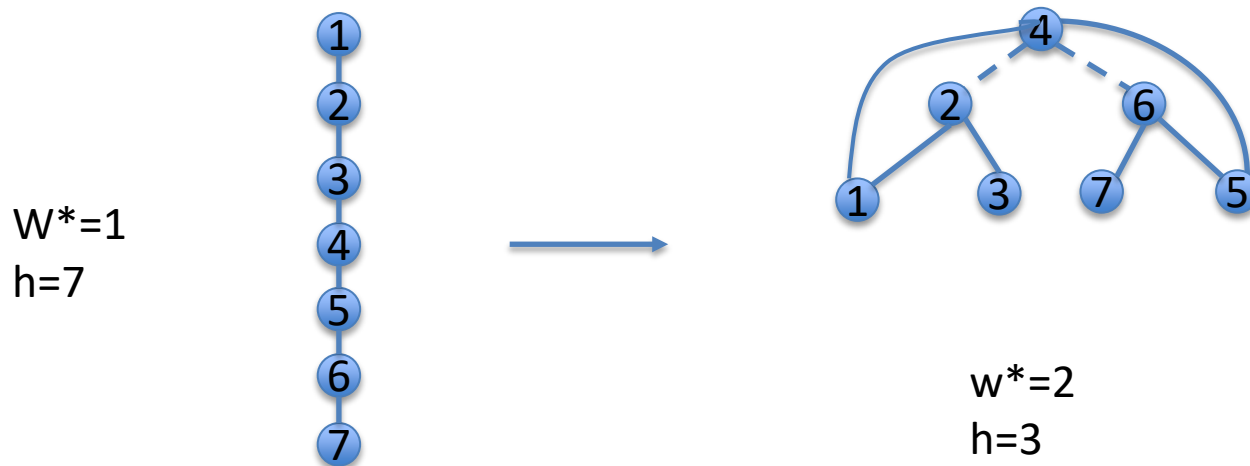
Variable Elimination

Outline: Search



Finding Min-height Pseudo-Trees

- Finding a min height pseudo-tree is NP-complete, but:
- Given a tree-decomposition with treewidth w^* , there exists a pseudo-tree whose height satisfies
 - $h \leq w^* \log n$
- Optimality of h and w^* cannot be achieved at once.



Constructing Pseudo-Trees

- **Min-Fill** [Kjaerulff, 1990]
 - Depth-first traversal of the induced graph obtained along the **min-fill** elimination order
 - Variables ordered according to the smallest “fill-set”
- **Hypergraph Partitioning** [Karypis and Kumar, 2000]
 - Functions are vertices in the hypergraph and variables are hyperedges
 - Recursive decomposition of the hypergraph while minimizing the separator size at each step
 - Using state-of-the-art software package **hMeTiS**

Quality of Pseudo-Trees

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

Bayesian Networks Repository

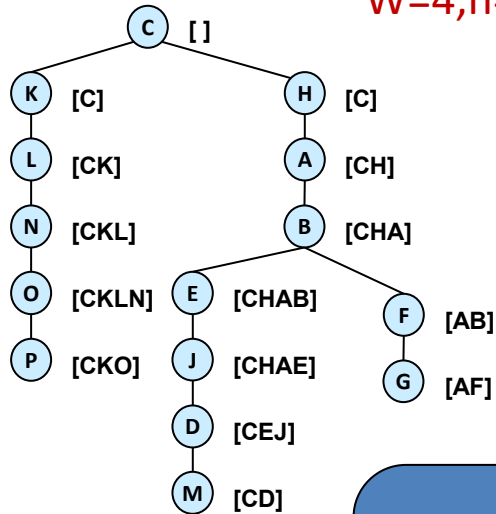
Network	hypergraph		min-fill	
	width	depth	width	depth
spot5	47	152	39	204
spot28	108	138	79	199
spot29	16	23	14	42
spot42	36	48	33	87
spot54	12	16	11	33
spot404	19	26	19	42
spot408	47	52	35	97
spot503	11	20	9	39
spot505	29	42	23	74
spot507	70	122	59	160

SPOT5 Benchmarks

For more see [Dechter 2003]

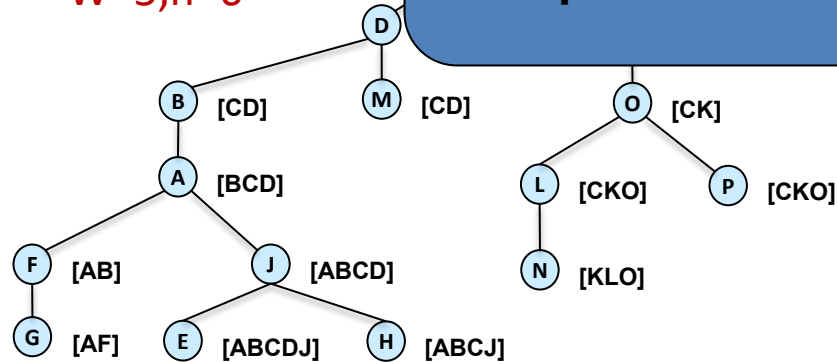
The Impact of the Pseudo-Tree

$W=4, h=8$



(C K H A B E J L N O)

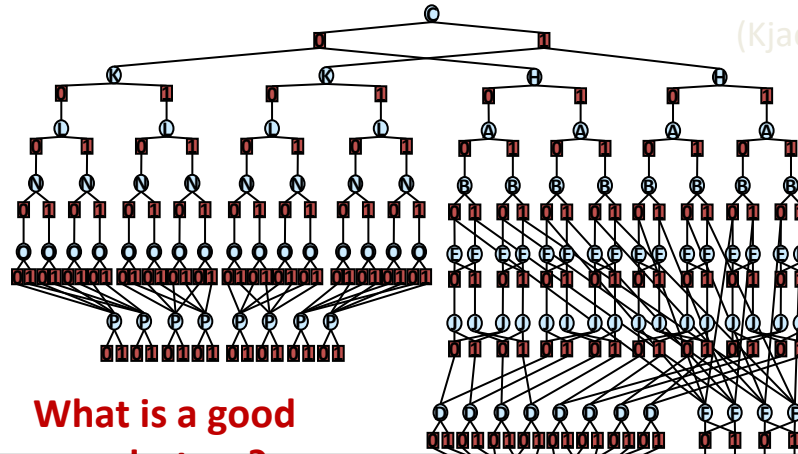
$W=5, h=6$



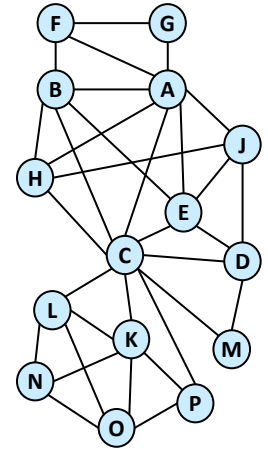
DS-602.41b1er

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

Min-Fill
(Kjaerulff90)

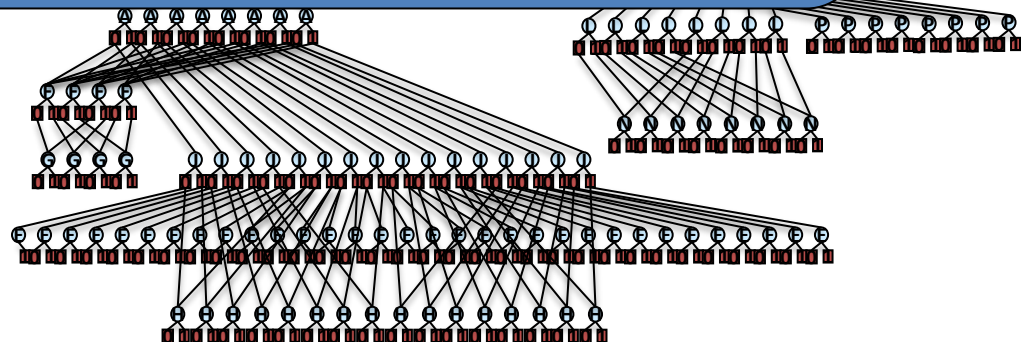


What is a good

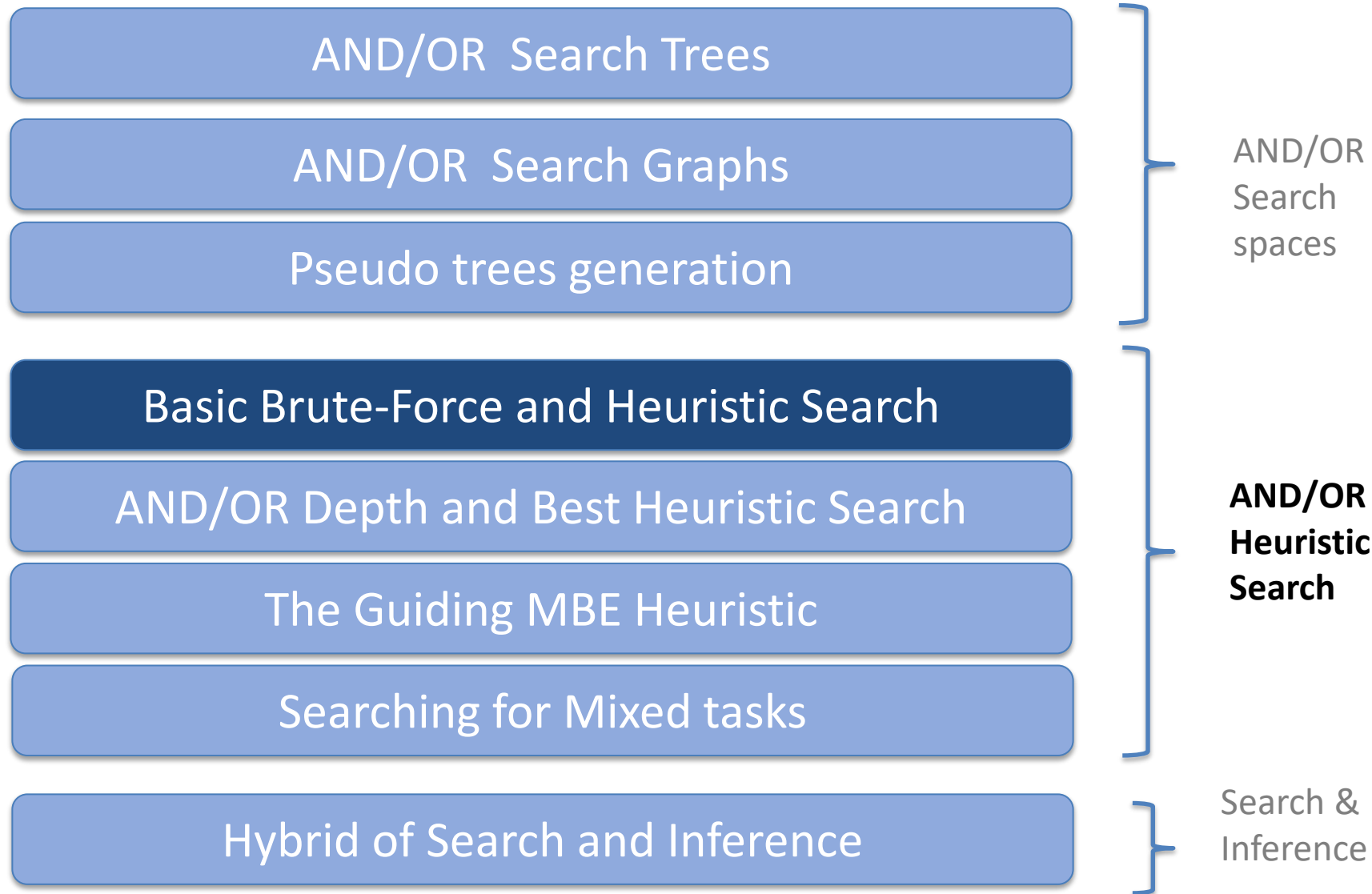


graph
onning
(s)

- Choose pseudo-tree with a minimal search graph
- But determinism and pruning for optimization is unpredictable



Outline: Search



Probabilistic Reasoning Problems

- Exact Inference by elimination or search
- Complexity:

Causal effects

Max-Inference:	$f(x^*) = \max_x \prod_{\alpha} f_{\alpha}(x_{\alpha})$
Sum-Inference:	$Z = \sum_x \prod_{\alpha} f_{\alpha}(x_{\alpha})$
Mixed-Inference (MMAP):	$f_M(x_M^*) = \max_{x_M} \sum_{x_S} \prod_{\alpha} f_{\alpha}(x_{\alpha})$
Mixed-Inference (MEU):	$\text{MEU} = \max_{\tilde{x}} \sum (\prod P_i) \times (\sum r_i)$

$e^{\text{tree-width}}$

Harder

- All solved by AND/OR Depth-first search,
 - Linear memory, $\exp(h)$ time or
 - $\exp(w^*)$ memory and time
- But, we can do better by:
 - Pruning while searching
 - Generating upper and lower bounds anytime

AND/OR Tree DFS Algorithm (Belief Updating)

$P(E | A, B)$

A	B	E=0	E=1
0	0	.4	.6
0	1	.5	.5
1	0	.7	.3
1	1	.2	.8

Evidence: E=0

$P(B | A)$

A	B=0	B=1
0	.4	.6
1	.1	.9

$P(C | A)$

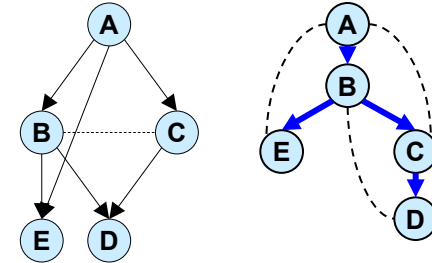
A	C=0	C=1
0	.2	.8
1	.7	.3

$P(A)$

A	P(A)
0	.6
1	.4

Result: $P(D=1, E=0)$

.24408



OR

AND

OR

AND

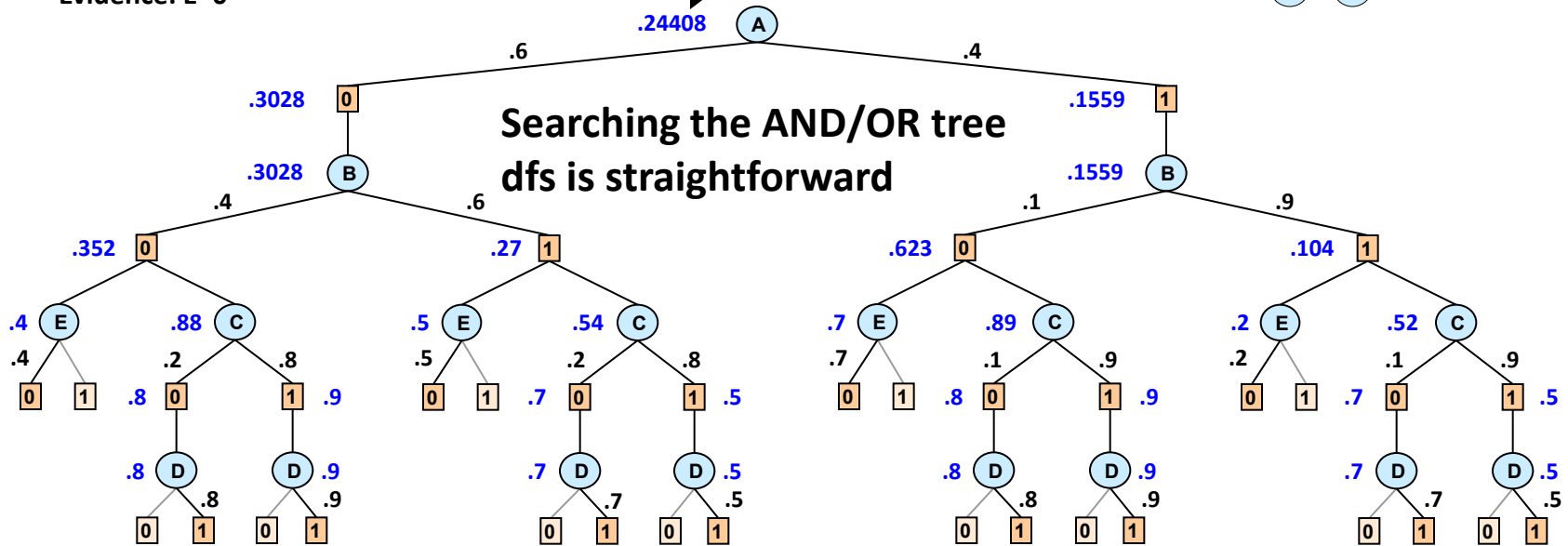
OR

AND

OR

AND

Searching the AND/OR tree
dfs is straightforward



$P(D | B, C)$

B	C	D=0	D=1
0	0	.2	.8
0	1	.1	.9
1	0	.3	.7
1	1	.5	.5

Evidence: D=1

OR node: Marginalization operator (summation)

AND node: Combination operator (product)

Value of node = updated belief for sub-problem below

AND/OR Graph DFS Algorithm (Belief Updating)

$P(E | A, B)$

A	B	E=0	E=1
0	0	.4	.6
0	1	.5	.5
1	0	.7	.3
1	1	.2	.8

Evidence: E=0

$P(B | A)$

A	B=0	B=1
0	.4	.6
1	.1	.9

$P(C | A)$

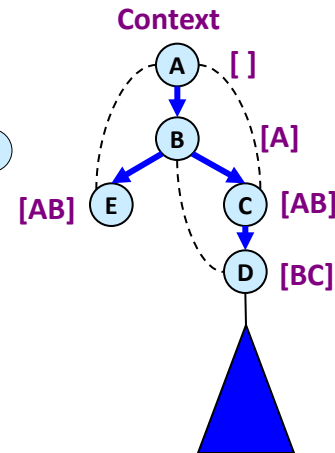
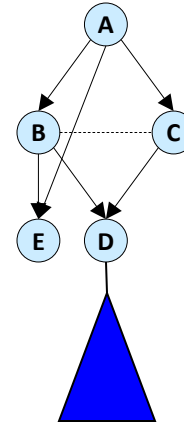
A	C=0	C=1
0	.2	.8
1	.7	.3

$P(A)$

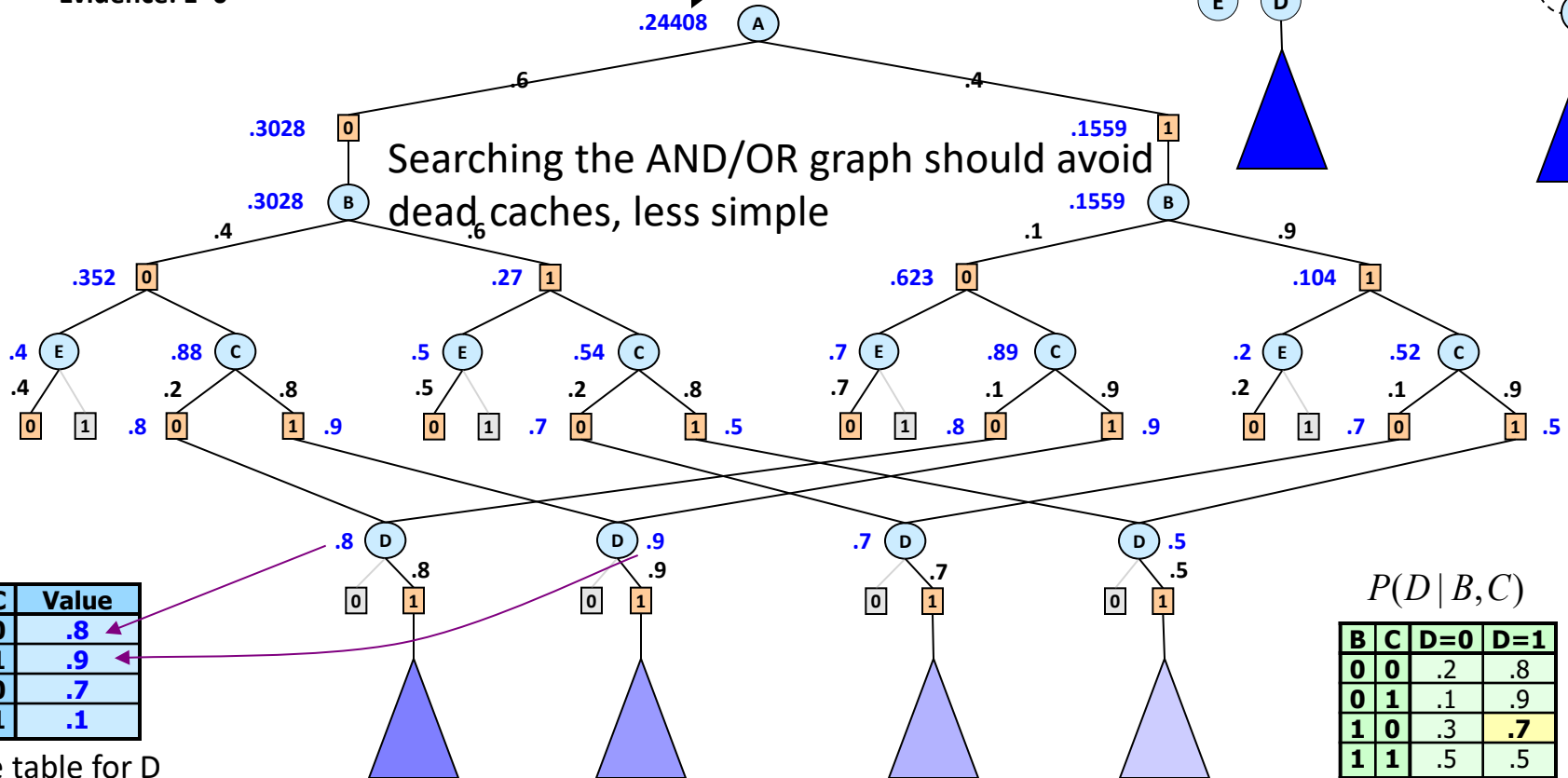
A	P(A)
0	.6
1	.4

Result: $P(D=1, E=0)$

.24408



Searching the AND/OR graph should avoid dead caches, less simple



B	C	Value
0	0	.8
0	1	.9
1	0	.7
1	1	.1

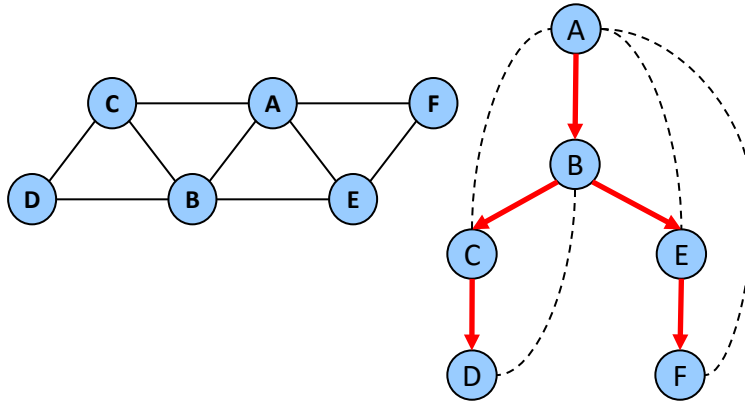
Cache table for D

$P(D | B, C)$

B	C	D=0	D=1
0	0	.2	.8
0	1	.1	.9
1	0	.3	.7
1	1	.5	.5

Evidence: D=1

AND/OR Search Graph (Optimization)



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: $F^* = \min_x \sum_{\alpha} f_{\alpha}(x_{\alpha})$

OR

AND

OR

AND

OR

AND

OR

AND

B	C	Value
0	0	←
0	1	←
1	0	←
1	1	←

Context minimal AND/OR search graph

Basic Heuristic Search

We assume min-sum problems in the following

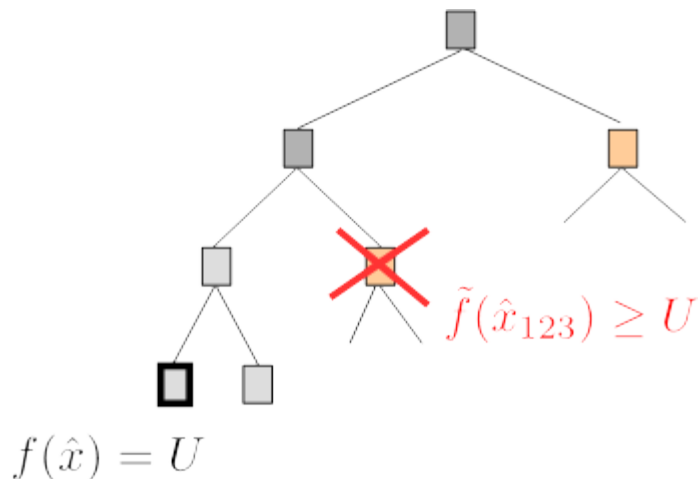
Heuristic function $\tilde{f}(\hat{x}_p)$ computes a lower bound on the best extension of partial configuration $\hat{x}_p$ and can be used to guide heuristic search.

We focus on:

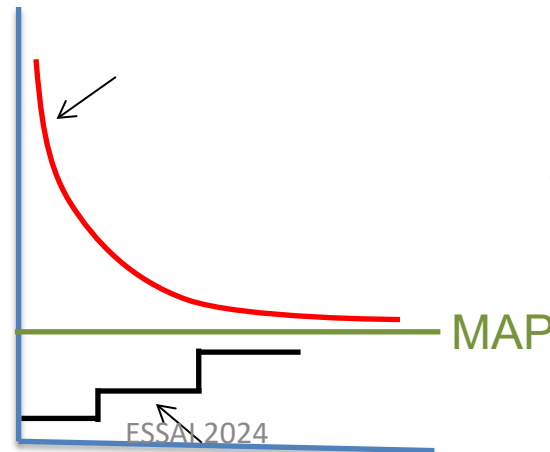
1. Branch-and-Bound

Use heuristic function $\tilde{f}(\hat{x}_p)$ to prune the depth-first search tree

Linear space



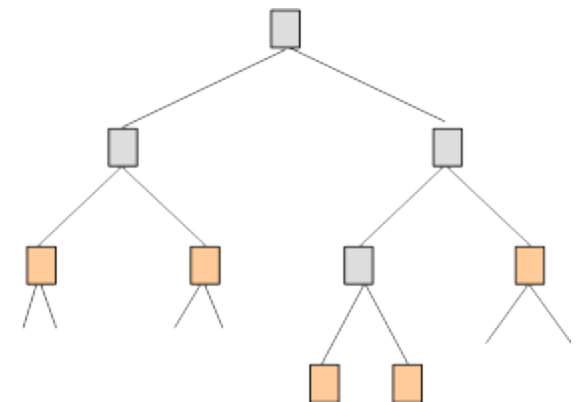
BnB is upper-bound anytime



2. Best-First Search

Always expand the node with the lowest heuristic value $\tilde{f}(\hat{x}_p)$

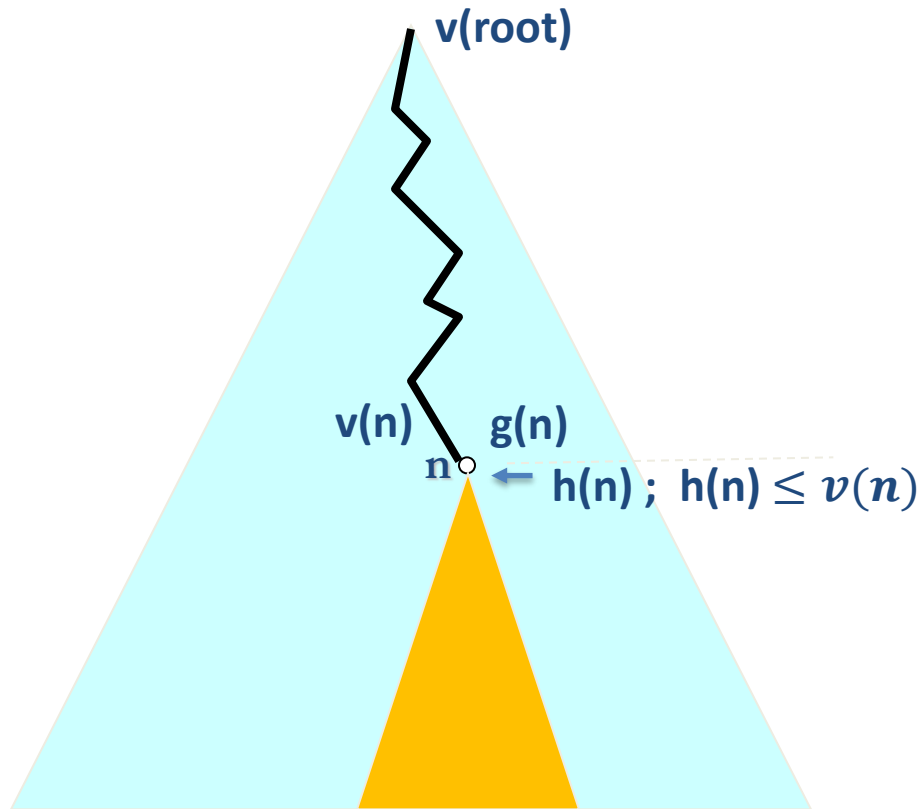
Needs lots of memory



Basic Heuristic Search; Best-First

Task: compute $v(\text{root})$: MAP, Marginal, MMAP

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

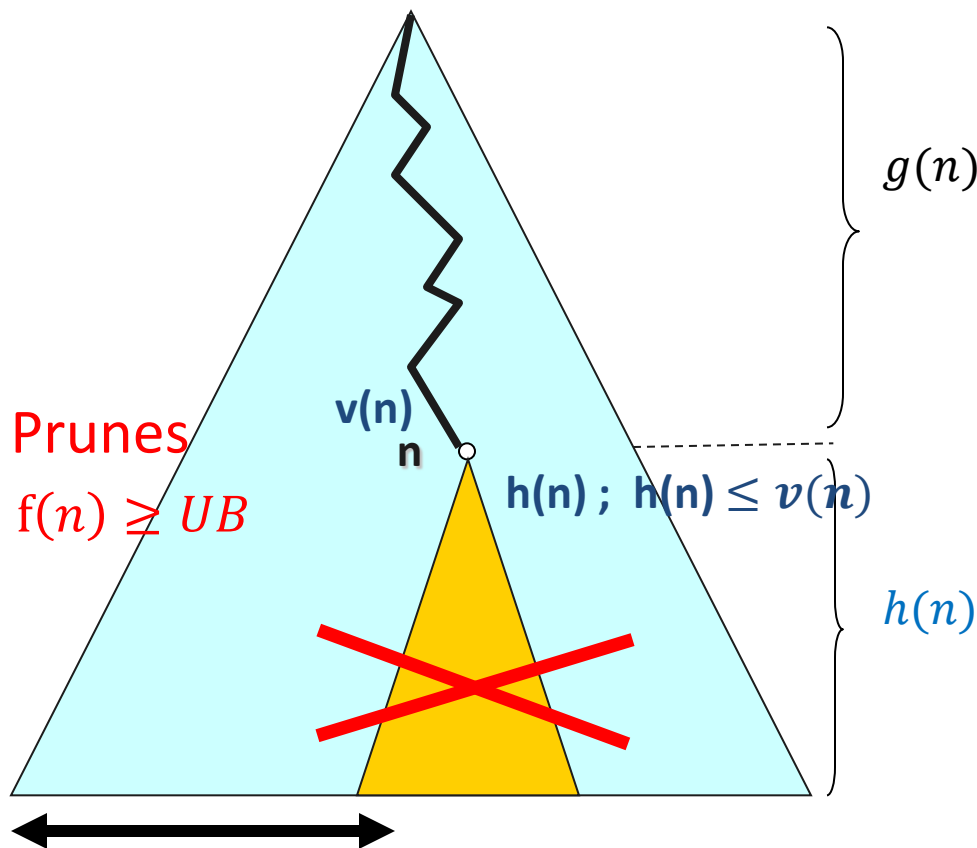


$$f(n) = g(n) + h(n) \leq g(n) + v(n) = f^*(n)$$

$f(n)$ is a lower bound on best cost through n

- **Best-First Algorithms, (A^*)**
 - Expand nodes in OPEN list in order of min $f(n)$
 - Terminates with first full solution (for MAP)
- **Properties**
 - Optimal, if $h(n) \leq v(n)$
 - Expands least set of nodes
 - exponential memory
 - **Not anytime solution for MAP**
 - **Yields lower bounds on value, anytime**

Basic Heuristic Search; Depth-First



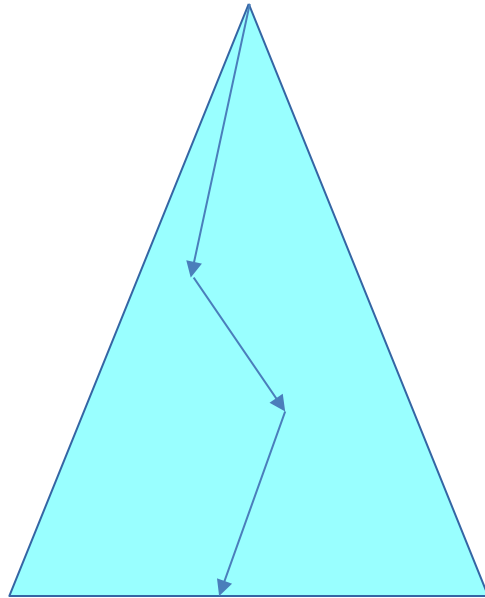
(UB) Upper Bound = best solution so far

- **Depth-First (B&B for MAP)**
 - Expand in dfs order
 - Update UB with each solution
 - Prunes if $f(n) \geq UB$
- **Properties**
 - Can use only linear memory
 - Yields upper bounds anytime

Best+Depth-First Search

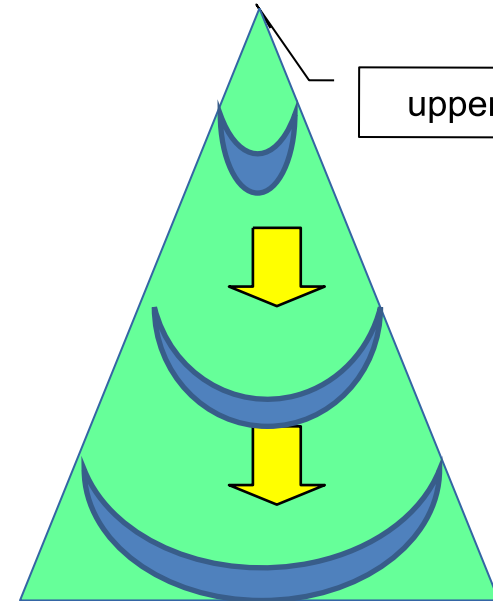
- Yields upper and lower bounds anytime

Depth-First search

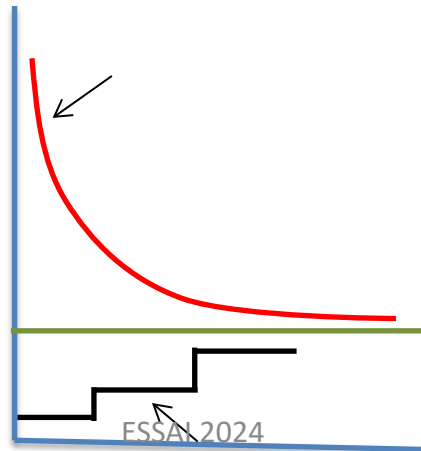
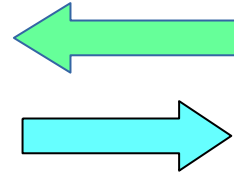


Lower bound

Best-First search



upper bound



MAP, Marginal, MMAP

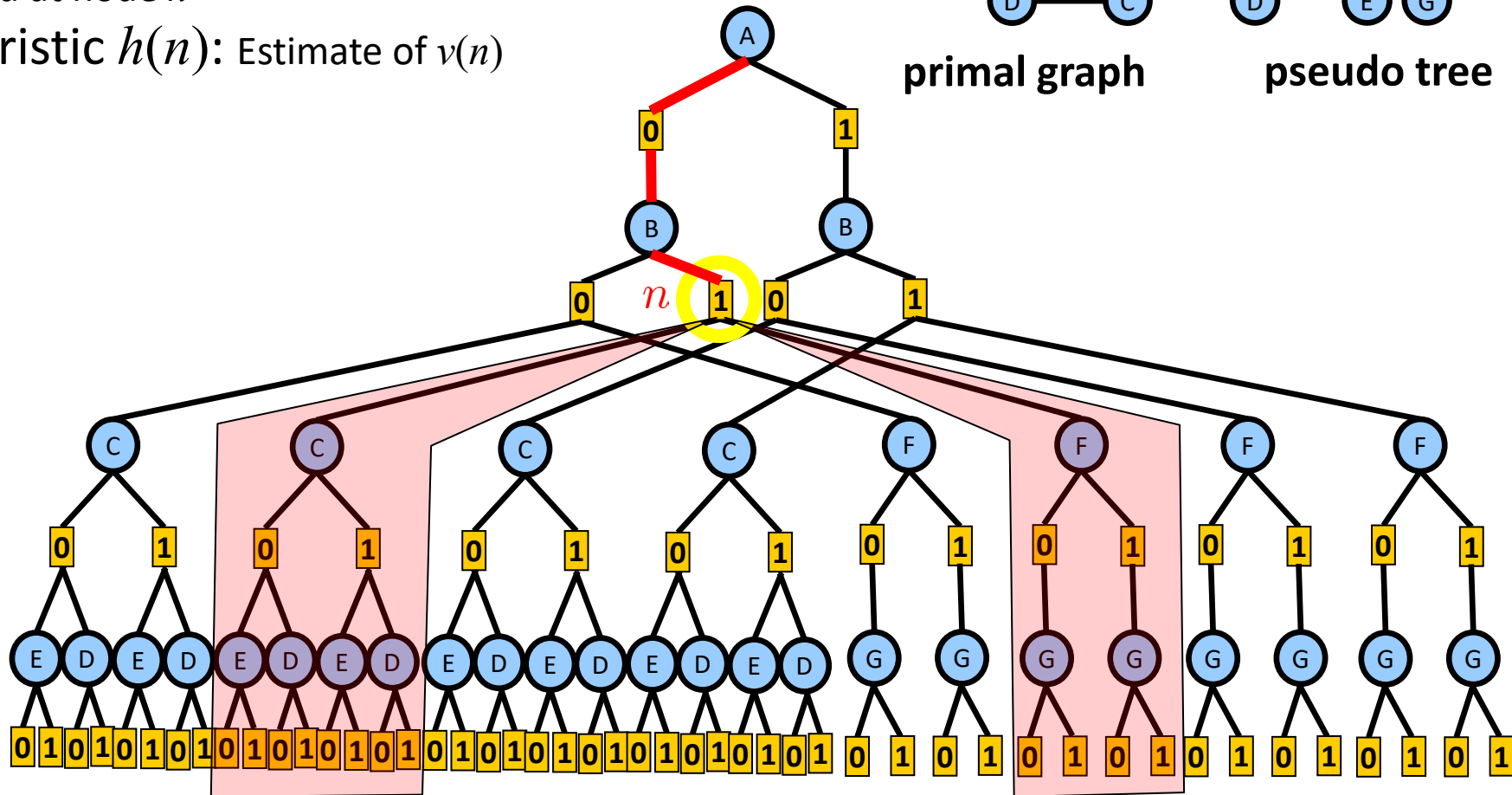
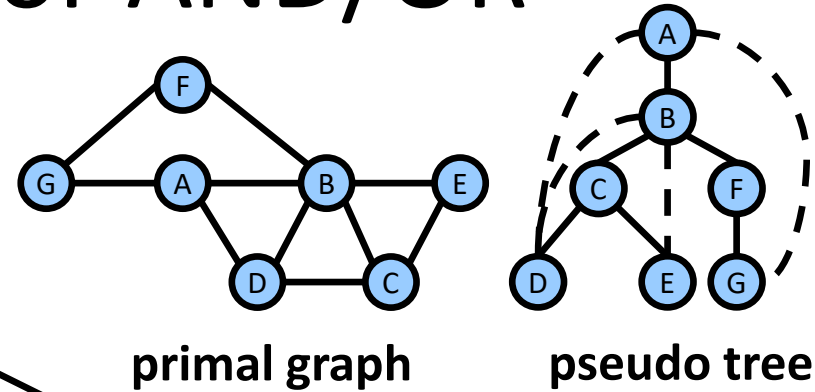
Outline: Search



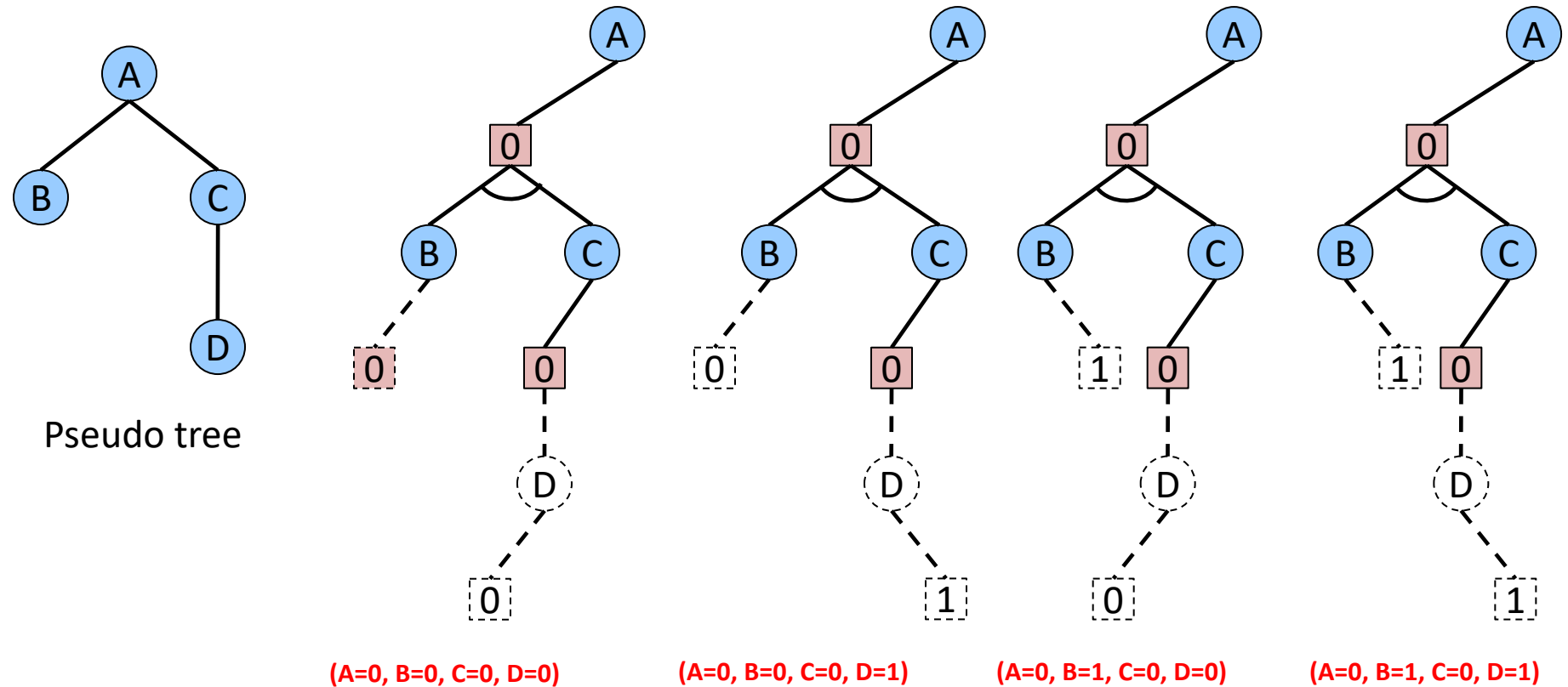
Value and Heuristic for AND/OR

Value $v(n)$: answer of the subtree rooted at node n

Heuristic $h(n)$: Estimate of $v(n)$



Partial Solution Tree



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

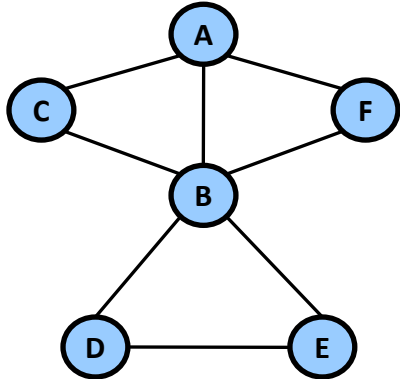
$g(T')$ = conditioned value of a node

$V(T')$ = the combined value below T'

$f^*(T')$ = conditioned value through T'

Exact Evaluation Function

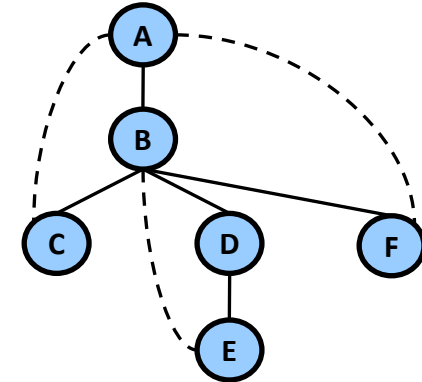
Conditioned value of a node



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

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

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



OR

AND

OR

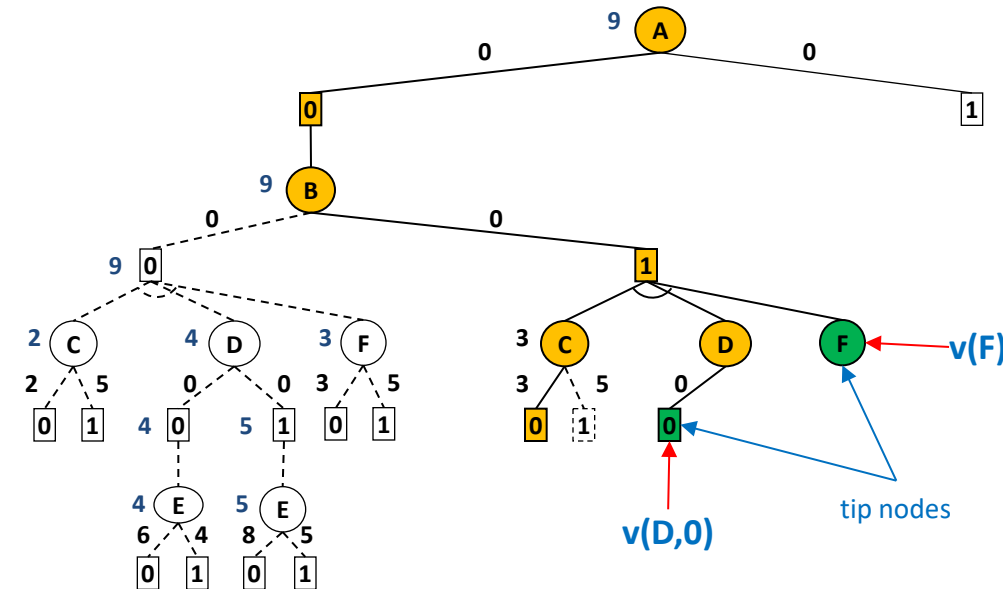
AND

OR

AND

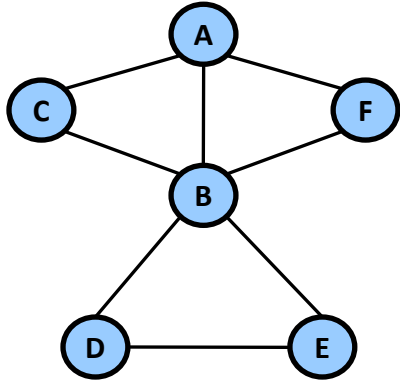
OR

AND



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

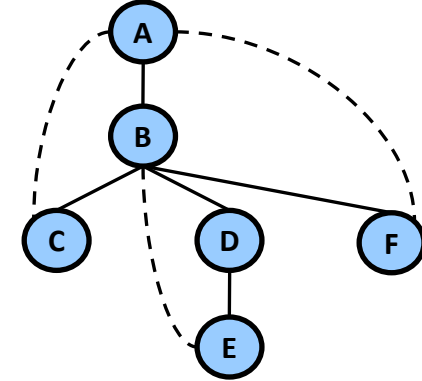
Heuristic Evaluation Function



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

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

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



OR

AND

OR

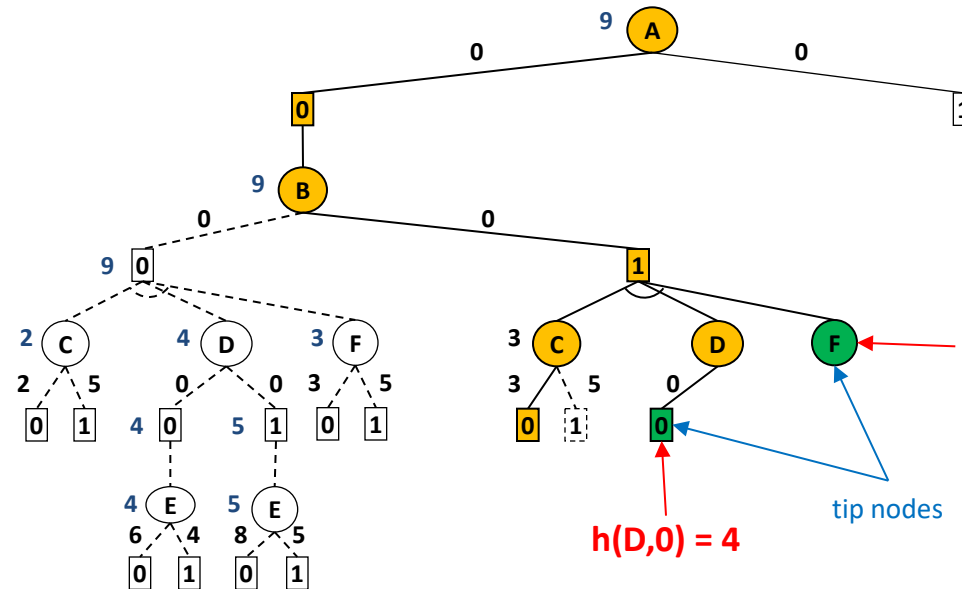
AND

OR

AND

OR

AND



$$h(n) \leq v(n)$$

$$h(F) = 5$$

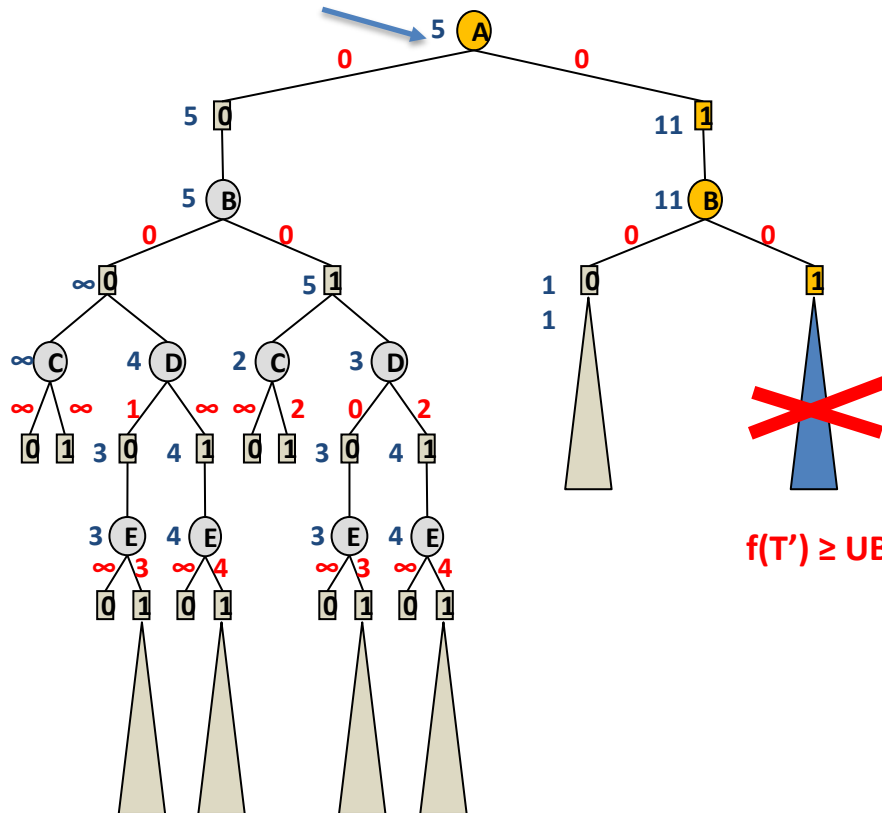
$$h(D,0) = 4$$

tip nodes

$$f(T') = w(A,0) + w(B,1) + w(C,0) + w(D,0) + h(D,0) + h(F) = 12 \leq f^*(T')$$

Depth-First AND/OR Branch-and-Bound

UB (best solution so far)



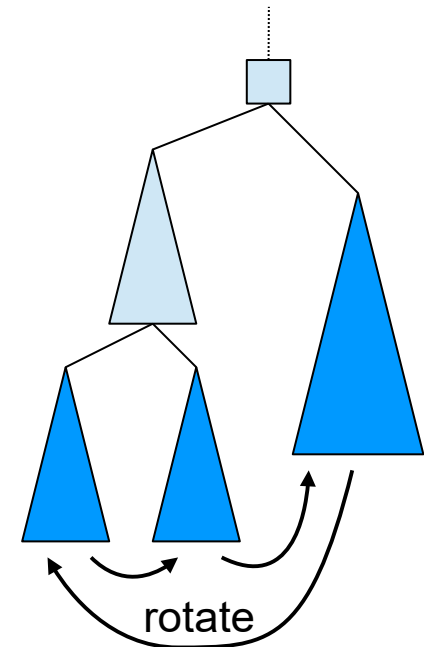
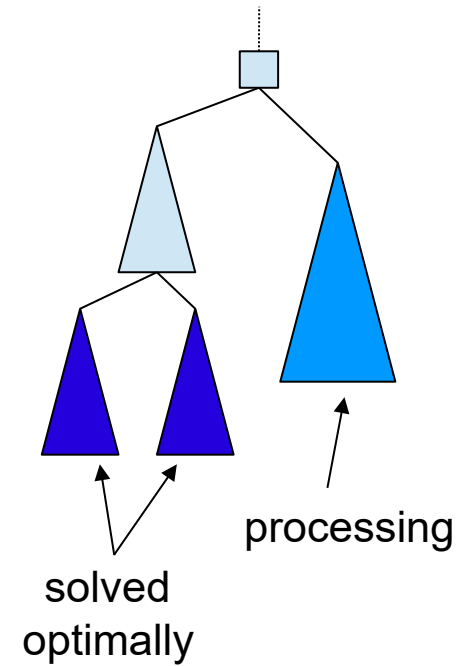
- Associate each node n with a heuristic lower bound $h(n)$ on $v(n)$

Algorithm AOBB:

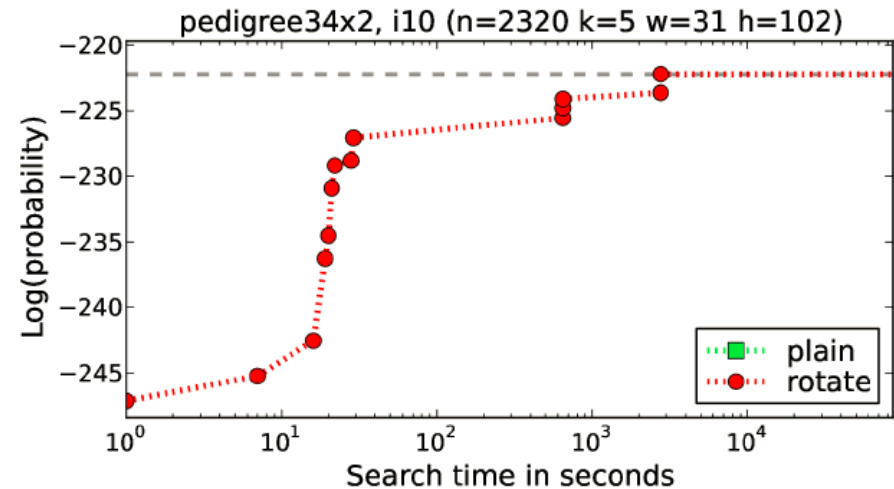
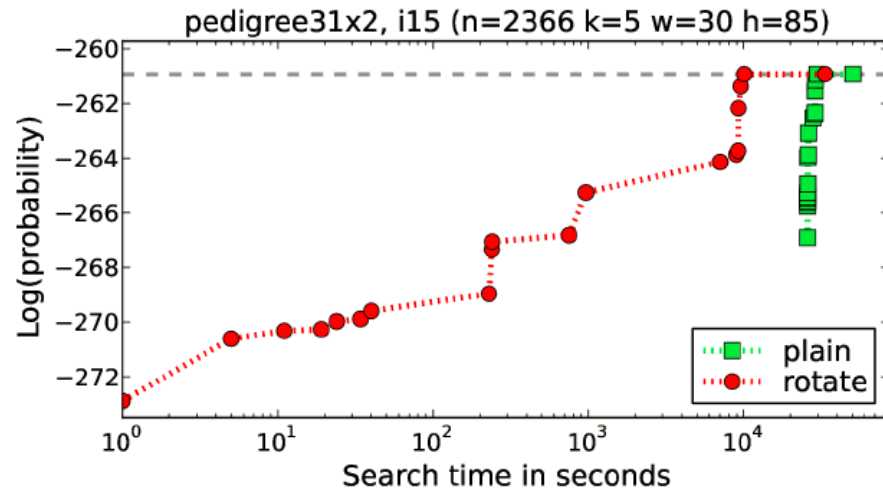
- EXPAND** (top-down)
 - Evaluate $f(T')$ and prune search if $f(T') \geq UB$
 - If not in cache, generate successors of the tip node n
- PROPAGATE** (bottom-up)
 - Update value of the parent p of n
 - OR nodes: minimization
 - AND nodes: summation
 - Cache value of n based on context

Anytime Performance

- OR Branch-and-Bound is anytime
- But AND/OR breaks anytime behavior of depth-first scheme:
 - First anytime solution delayed until last sub-problem starts processing
- **Breadth-Rotating AOBB:**
 - Take turns processing sub-problems
 - Limit number of expansions per visit
 - Solve each sub-problem depth-first
 - Maintain favorable complexity bounds

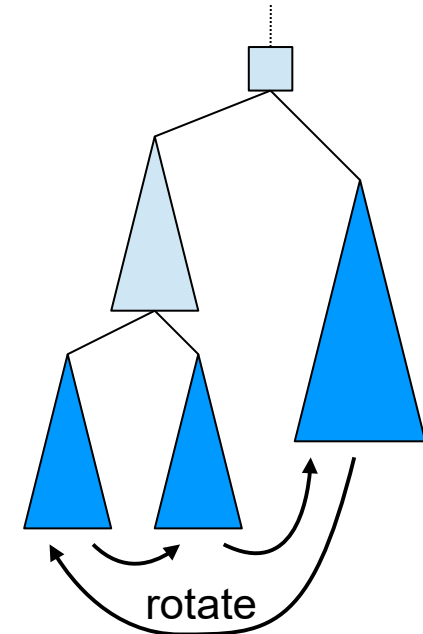


Anytime Performance



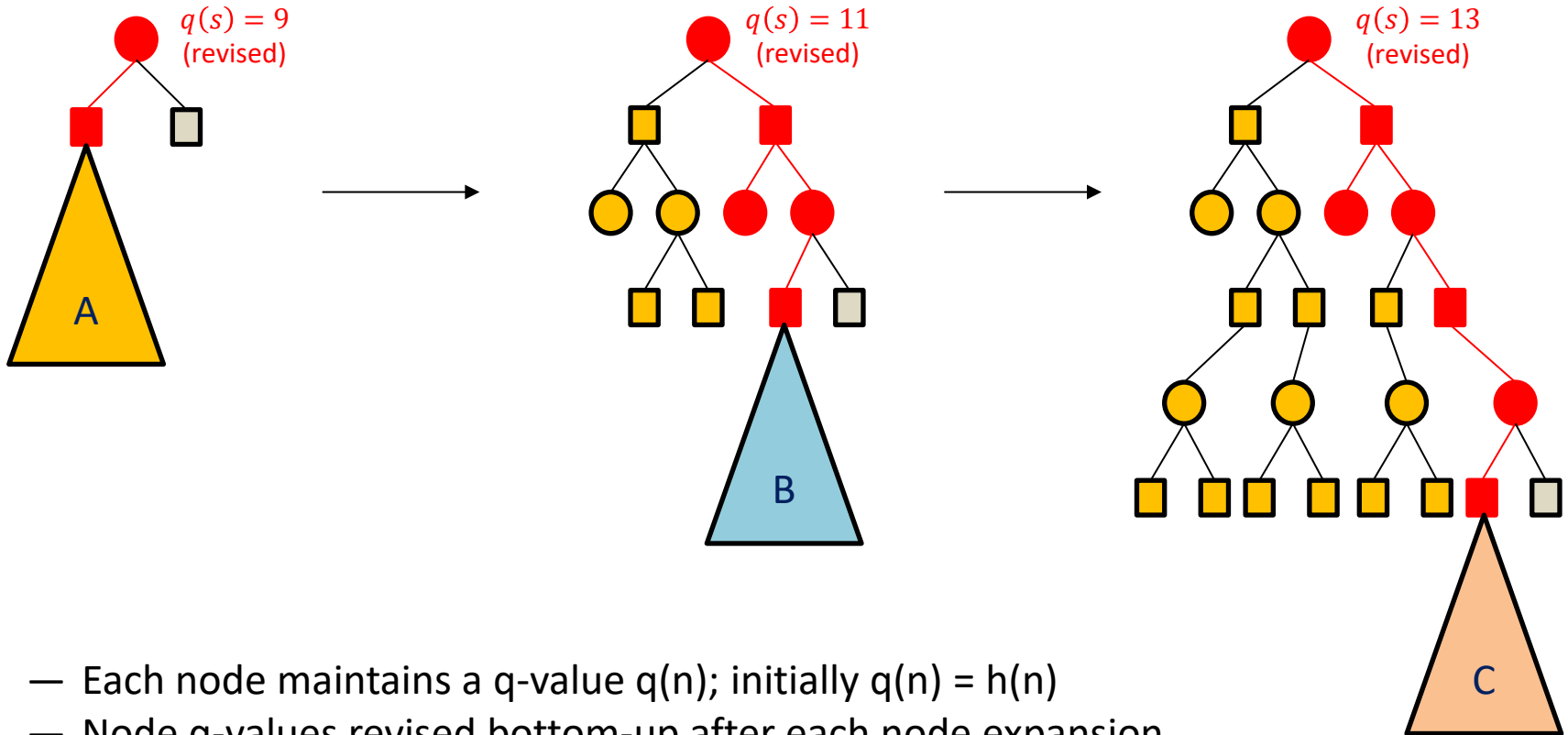
- **Breadth-Rotating AOBB:**

- Take turns processing sub-problems
 - Limit number of expansions per visit
- Solve each sub-problem depth-first
 - Maintain favorable complexity bounds



[Otten and Dechter, 2012]

AOBF: Best-First AND/OR Search



- Each node maintains a q-value $q(n)$; initially $q(n) = h(n)$
- Node q-values revised bottom-up after each node expansion
- Update current best partial solution subtree (a tip node expanded next)
- All expanded nodes are stored in memory
- Search terminates with optimal solution (cost)

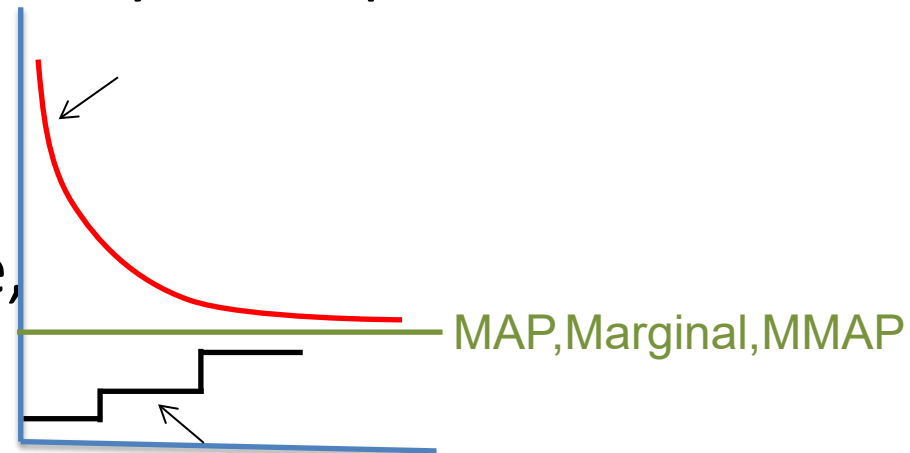
AOBF: Best-First AND/OR Search

- AO*-traverses the context-minimal AND/OR graph
 - All nodes expanded are stored in memory
 - Each node maintains a q-value: $q(n)$, (Best lower bound below n)
- Node q-values are revised bottom-up after each expansion
 - OR: minimization: $q(n) = \min_{n' \in \text{succ}(n)} (w(n, n') + q(n'))$
 - AND: summation: $q(n) = \sum_{n' \in \text{succ}(n)} q(n')$, (initially, $q(n) = h(n)$)

AOBF versus AOBB

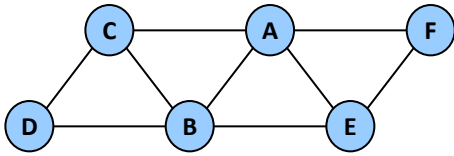
- **AOBF** expands a smallest subset of the AO search space
 - This translates into significant time savings
- **AOBB** can use far less memory by avoiding dead-caches, whereas **AOBF** keeps in memory the explicated search graph

- **AOBB (BRAOBB)** is anytime,
- **AOBF** generates lower bounds anytime, but not anytime solutions (configuration)



Outline: Search





Given a cost function:

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

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

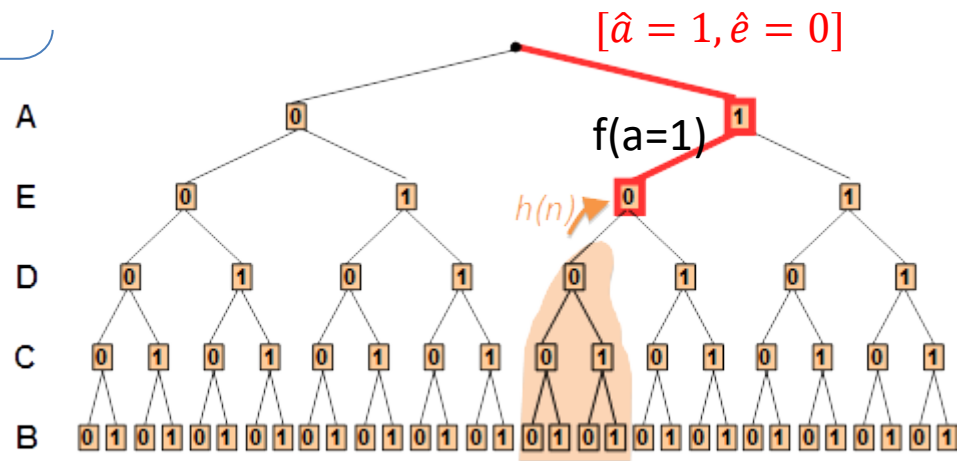
$$f^*(\hat{a}, \hat{e}, D) = \min_{b, c} F(\hat{a}, b, c, D, \hat{e})$$

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

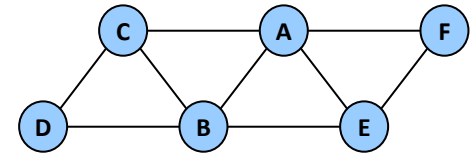
$$(h^* = v)$$

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

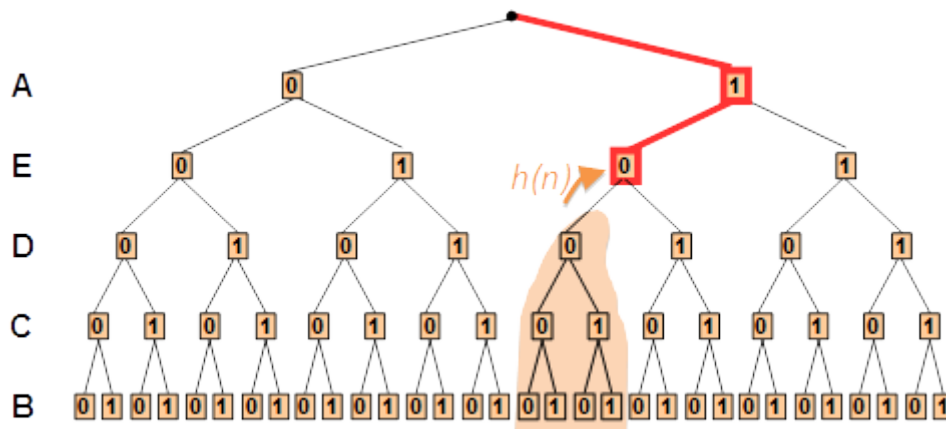
[Kask and Dechter, 2001]



Static Mini-Bucket Heuristics



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



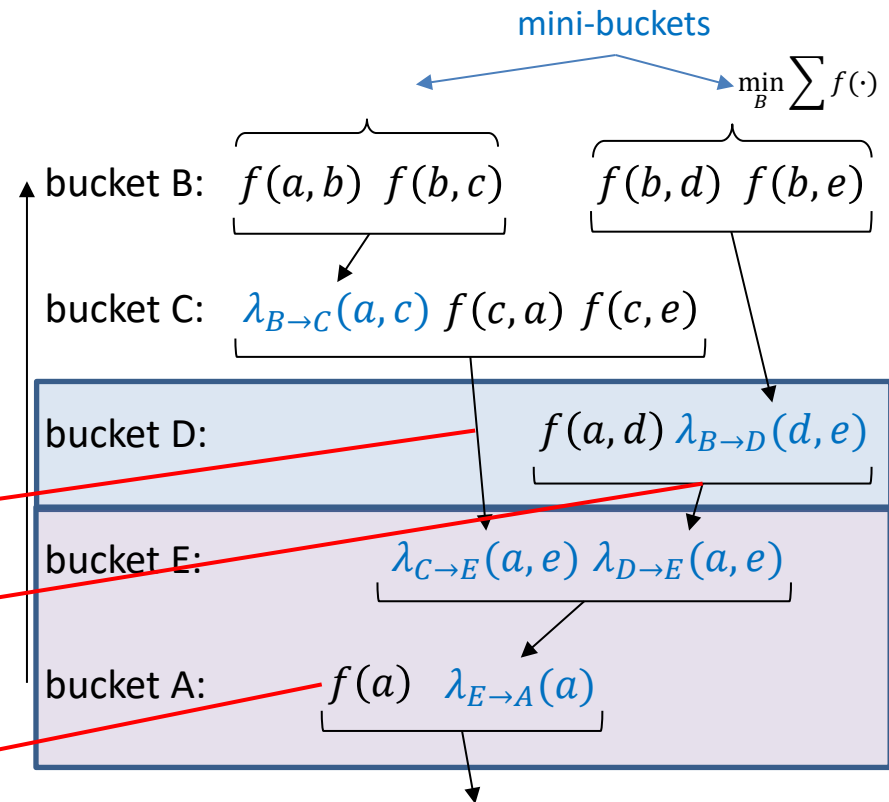
cost to go:

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

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

cost so far:

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



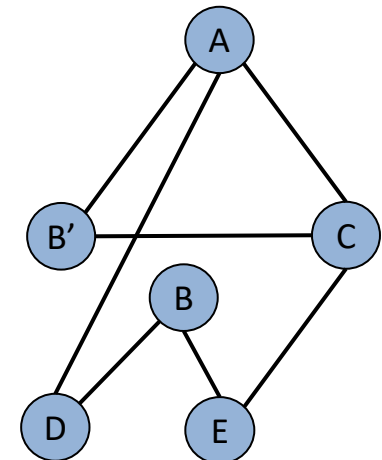
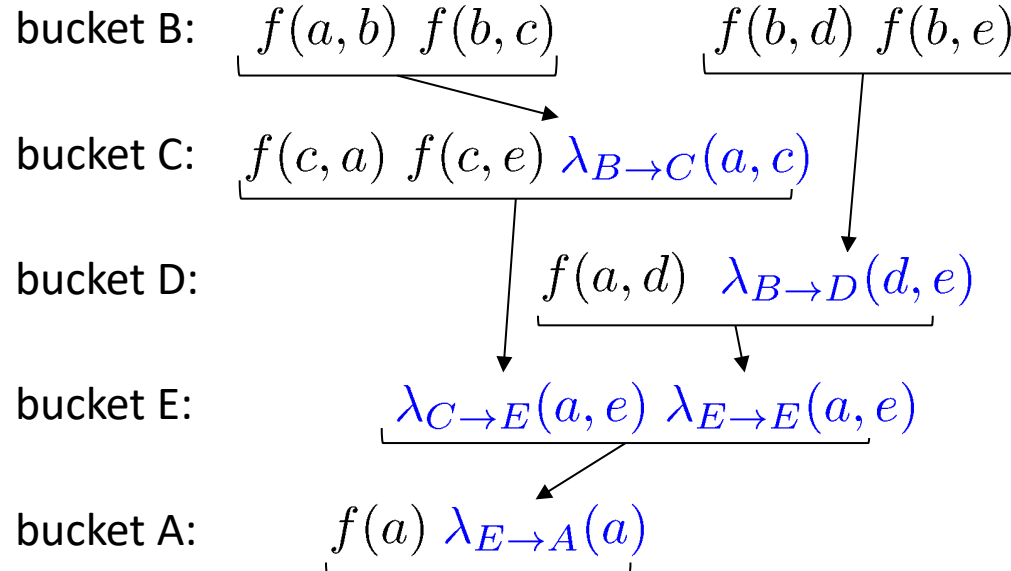
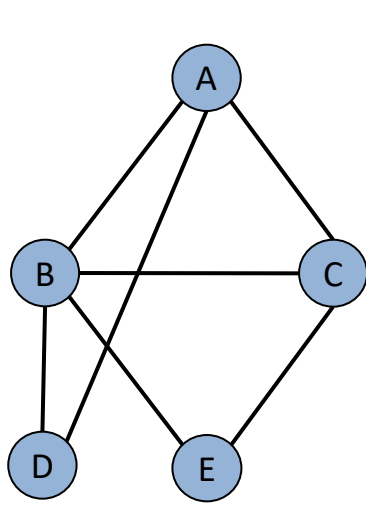
L = lower bound

Properties of the MBE Heuristics

- MBE heuristic is monotone, admissible
- Computed in linear time (during search)
- Important:
 - Heuristic strength can vary by $MBE(i)$
 - Higher i -bound $\rightarrow$ more pre-processing $\rightarrow$ more accurate heuristic $\rightarrow$ less search
- Allows controlled trade-off between pre-processing and search
- Can be computed **statically** or **dynamically** during search

Review: Mini-bucket Elimination

For Max-Inference: max-product or min-sum mini-buckets



$$\lambda_{B \rightarrow C}(a, c) = \max_b f(a, b) f(b, c)$$

$$\lambda_{B \rightarrow D}(d, e) = \max_b f(b, d) f(b, e)$$

$$\lambda_{C \rightarrow E}(a, e) = \max_c \dots$$

Review: Weighted Mini-bucket

[Liu & Ihler 2011]

For Sum-Inference

$$\lambda_{B \rightarrow C} = \sum_b^{w_{B1}} f(a, b) \cdot f(b, c) \quad w_{B1} + w_{B2} = 1$$

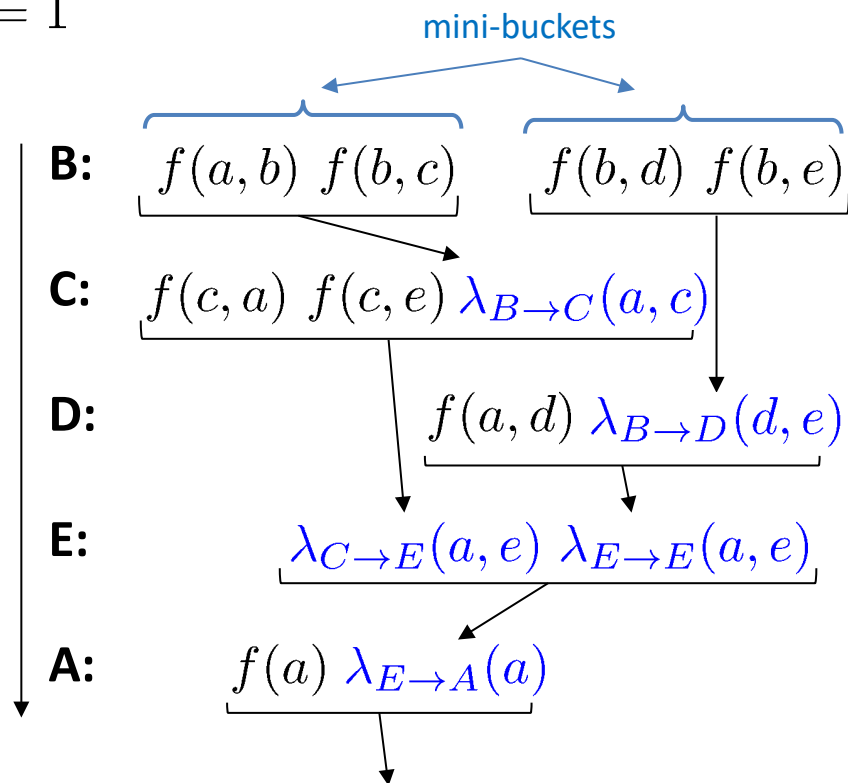
$$\lambda_{B \rightarrow D} = \sum_b^{w_{B2}} f(b, d) \cdot f(b, e)$$

$$\lambda_{C \rightarrow E} = \sum_c f(c, a) \cdot f(c, e) \cdot \lambda_{B \rightarrow C}$$

$$\vdots$$

Compute downward messages
using weighted sum

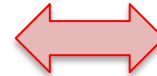
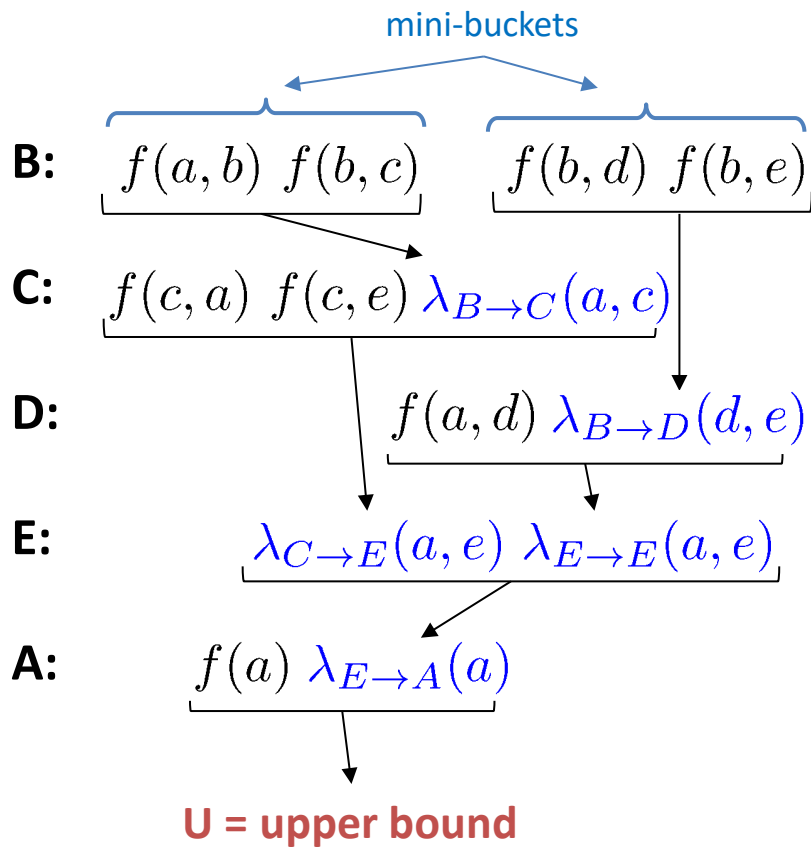
Upper bound if all weights positive
(corresponding lower bound if only one positive, rest negative)



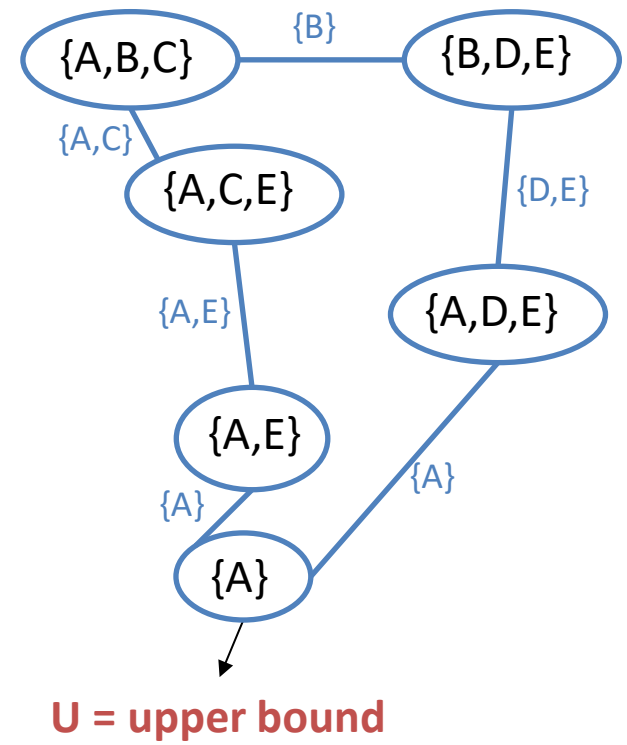
Review: MBE+Moment Matching

For all queries

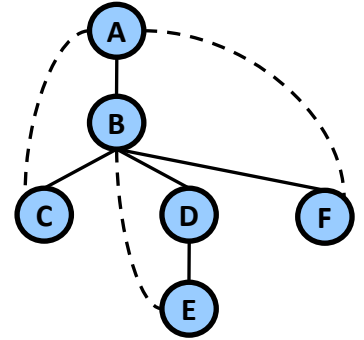
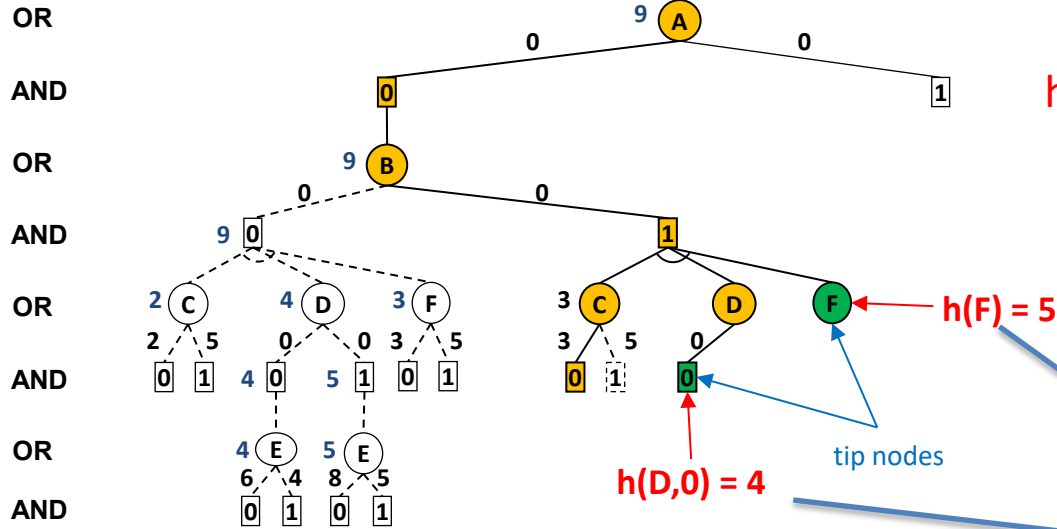
- Mini-bucket elimination defines regions with bounded complexity



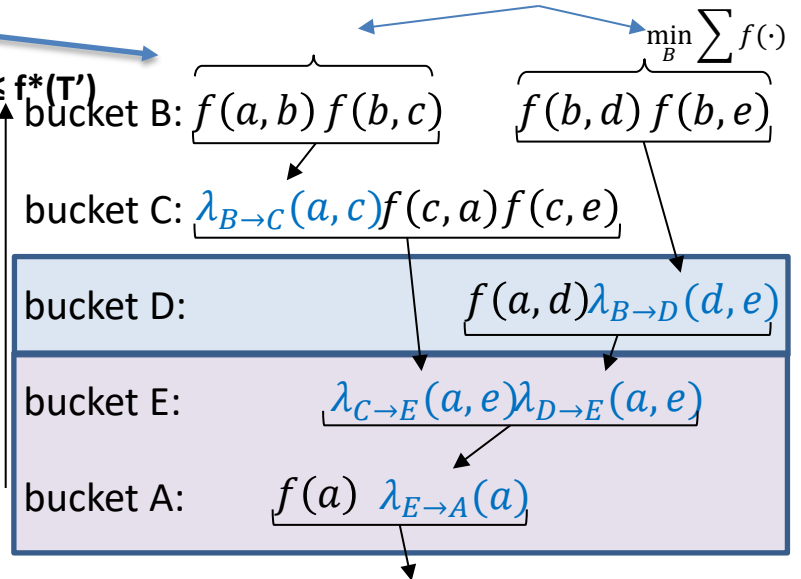
Join graph:



MBE Heuristic Guides AO Search

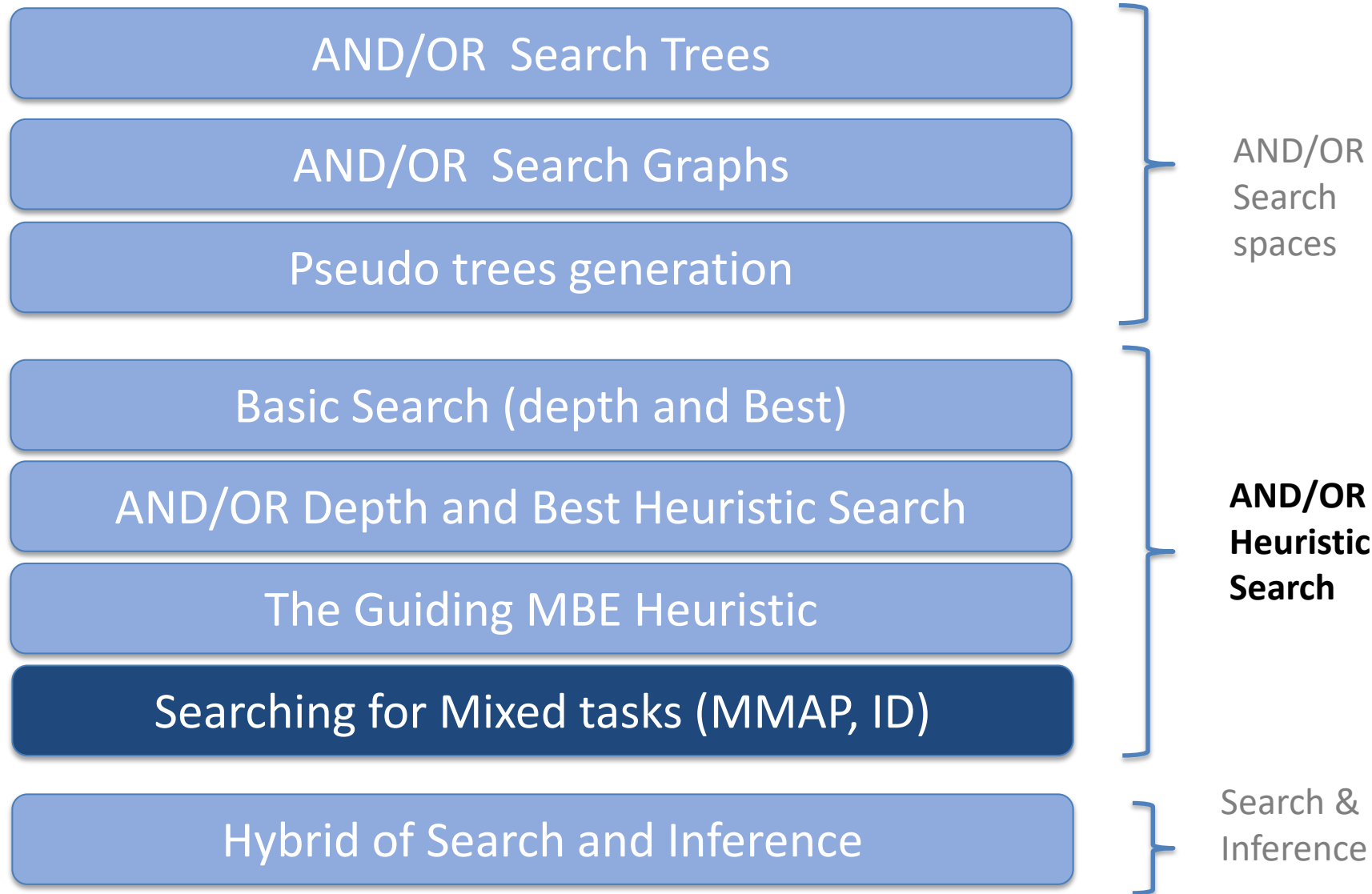


$$f(T') = w(A,0) + w(B,1) + w(C,0) + w(D,0) + h(D,0) + h(F) = 12 \leq f^*(T')$$



$L = \text{lower bound}$

Outline: Search



Probabilistic Reasoning Problems

- Exact Inference by elimination or search
- Complexity:

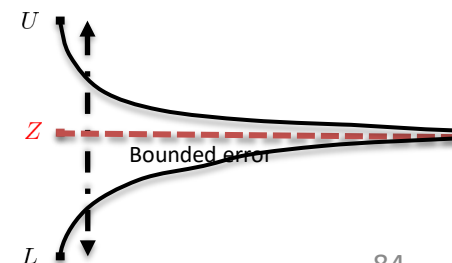
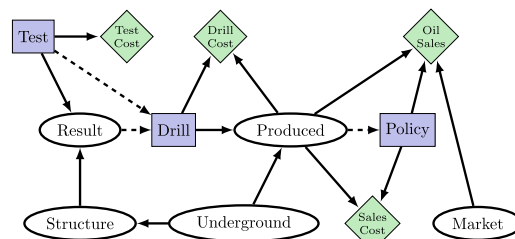
Causal effects

Max-Inference:	$f(x^*) = \max_x \prod_{\alpha} f_{\alpha}(x_{\alpha})$
Sum-Inference:	$Z = \sum_x \prod_{\alpha} f_{\alpha}(x_{\alpha})$
Mixed-Inference (MMAP):	$f_M(x_M^*) = \max_{x_M} \sum_{x_S} \prod_{\alpha} f_{\alpha}(x_{\alpha})$
Mixed-Inference (MEU):	$MEU = \max_{D_1, \dots, D_m} \sum_{X_1, \dots, X_n} \left(\prod_{P_i \in P} P_i \right) \times \left(\sum_{r_i \in R} r_i \right)$

$e^{\text{tree-width}}$

Harder

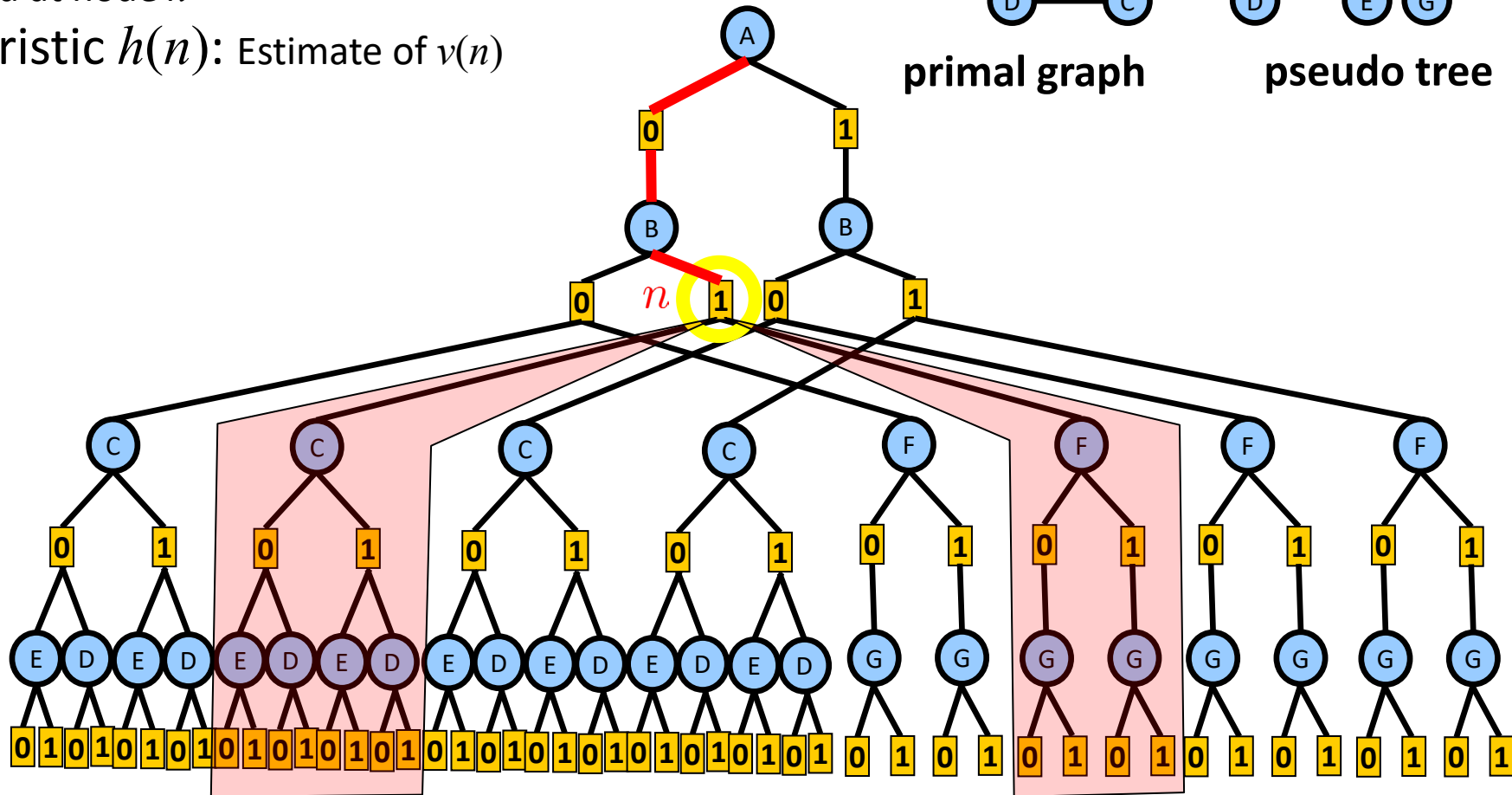
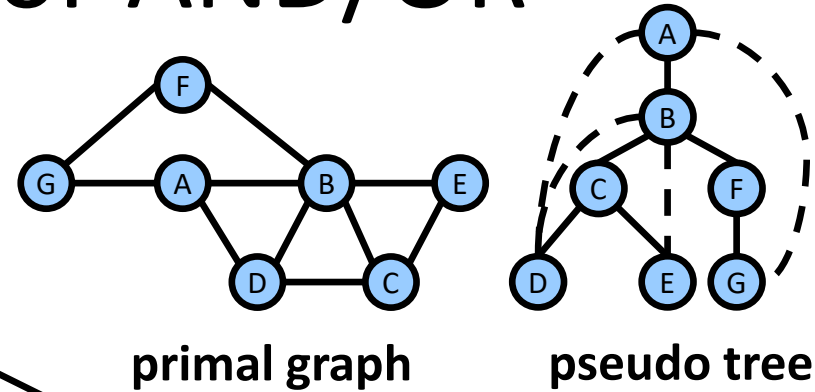
Influence
diagrams &
planning



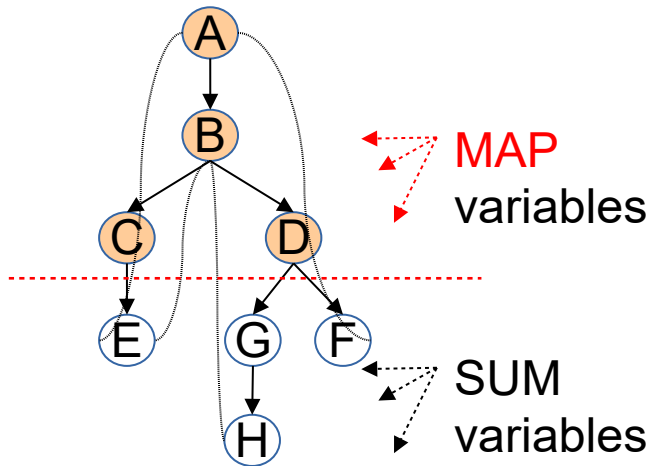
Value and Heuristic for AND/OR

Value $v(n)$: answer of the subtree rooted at node n

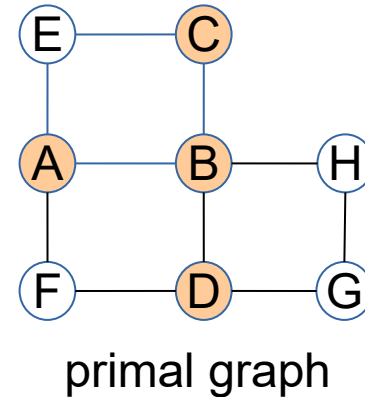
Heuristic $h(n)$: Estimate of $v(n)$



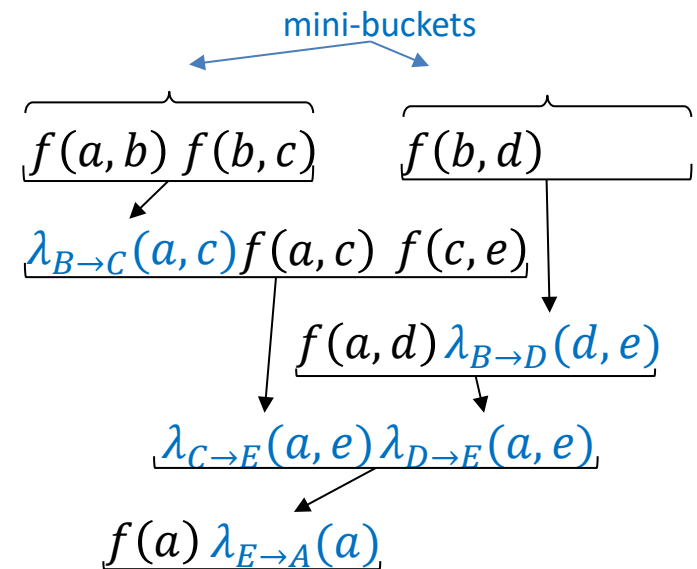
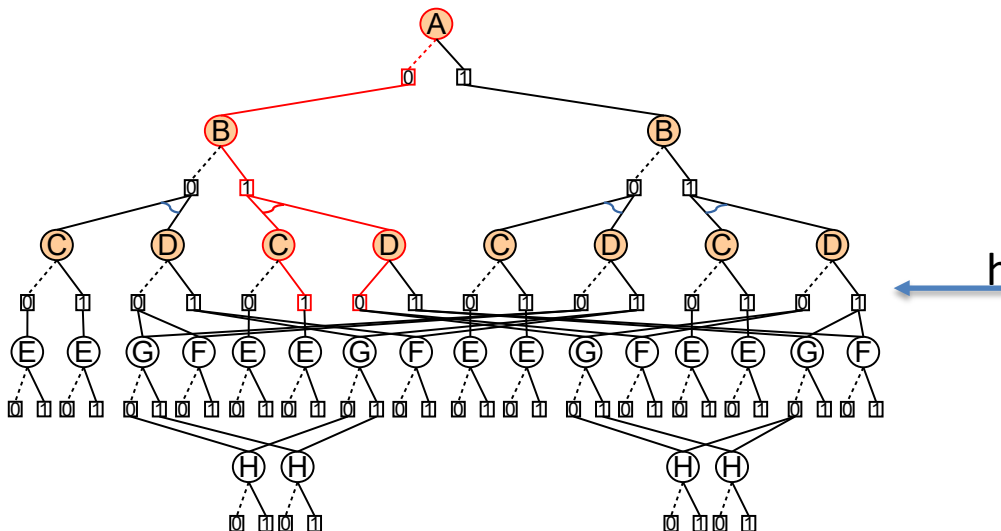
AND/OR Search for Marginal MAP



constrained
pseudo tree

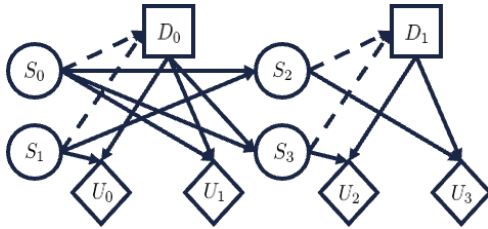


Anytime Depth+Best to yield upper and lower bounds

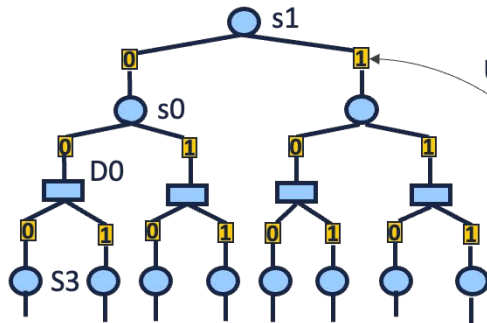


AND/OR Search for Influence Diagrams

Heuristic AND/OR search with decomposition bounds

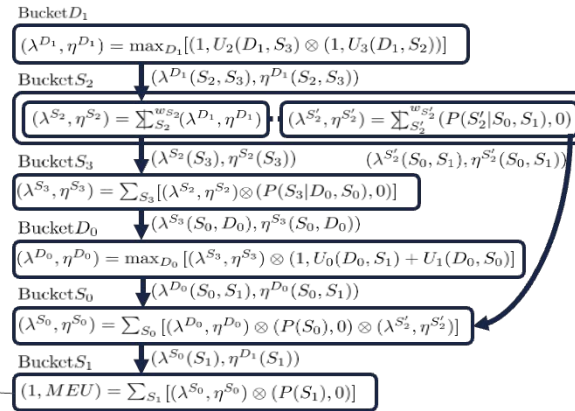


$$\sum_{S_0, S_1} \max_{D_0} \sum_{S_2, S_3} \max_{D_1} [\prod_{P_i \in \mathbf{P}} P_i] [\sum_{U_i \in \mathbf{U}} U_i] [\prod_{\Delta_i \in \Delta} \Delta_i]$$



[Marinescu 2010]

AND/OR Search Graph for Influence Diagrams

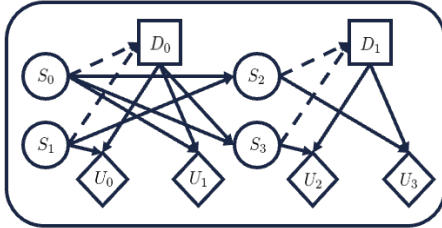


[Lee et.al 2019]

Weighted Mini-bucket elimination bound for Influence Diagrams

AND/OR Search for Influence Diagrams

Influence Diagram [Howard and Matheson 1981]



[Jensen et al 1994; Maua et. al 2012]

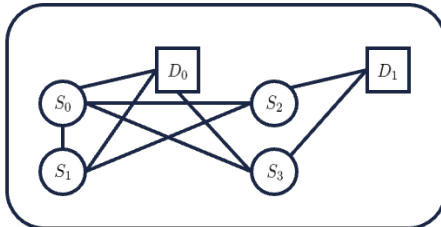
Valuation Algebra for Influence Diagrams

$$\Psi := \{(P_i, 0) | P_i \in \mathbf{P}\} \cup \{(1, U_i) | U_i \in \mathbf{U}\}$$

$$\Psi_1 \otimes \Psi_2 := (P_1 P_2, P_1 V_2 + P_2 V_1)$$

$$\sum_Y^w \Psi := (\sum_Y^w P, \sum_Y^w V)$$

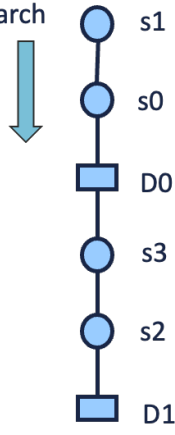
Primal Graph



[Freuder and Quinn 1985]

Pseudo-tree

Search

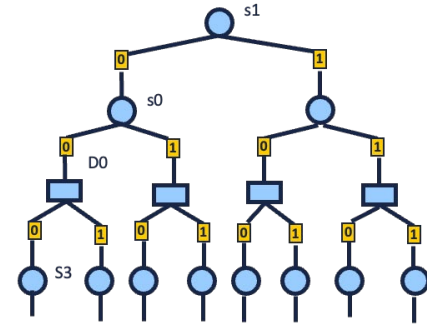


Inference [Dechter 1999]

By bucket elimination

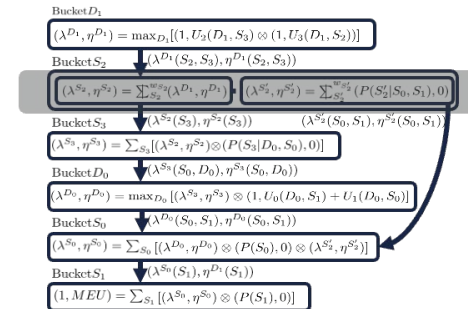
[Dechter et al 2007; Marinescu, et al 2010]

AND/OR Search Space



[Dechter et al 2003; Liu et al 2011; Lee et al 2019]

Decomposition bounds from bucket tree

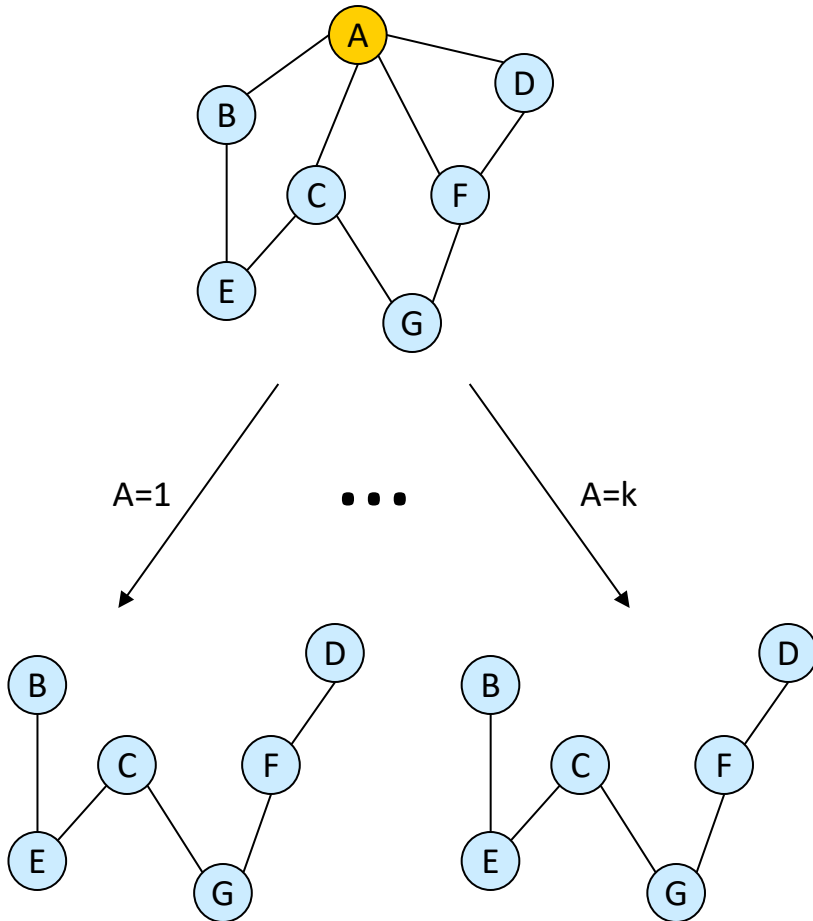


Outline: Search



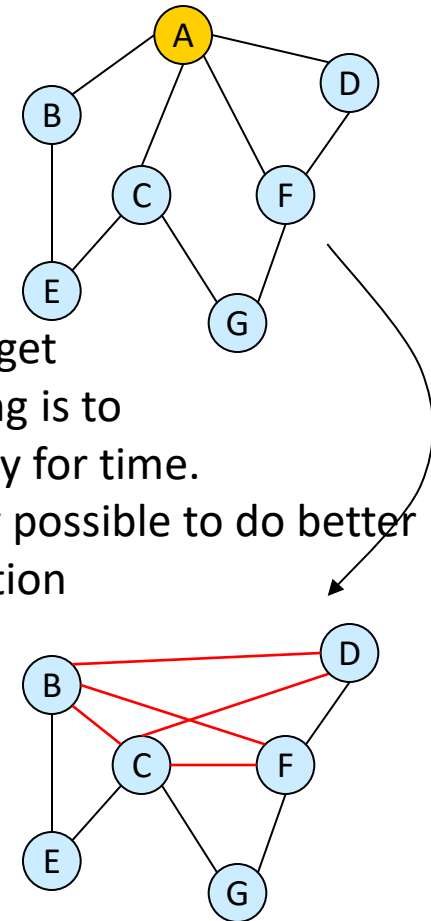
Conditioning versus Elimination

Conditioning (search)



k "sparser" problems

Elimination (inference)

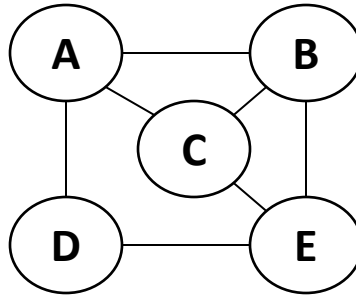


1 "denser" problem

The main target
In conditioning is to
Trade memory for time.
It is not really possible to do better
Then elimination

Hybrid: Cutset-Conditioning

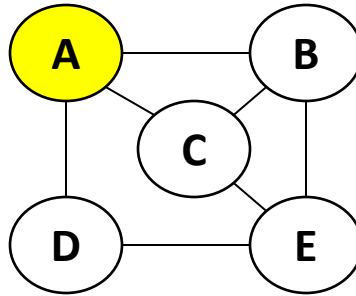
Variable Branching by Conditioning



Hybrid: Cutset-Conditioning

Variable Branching by Conditioning

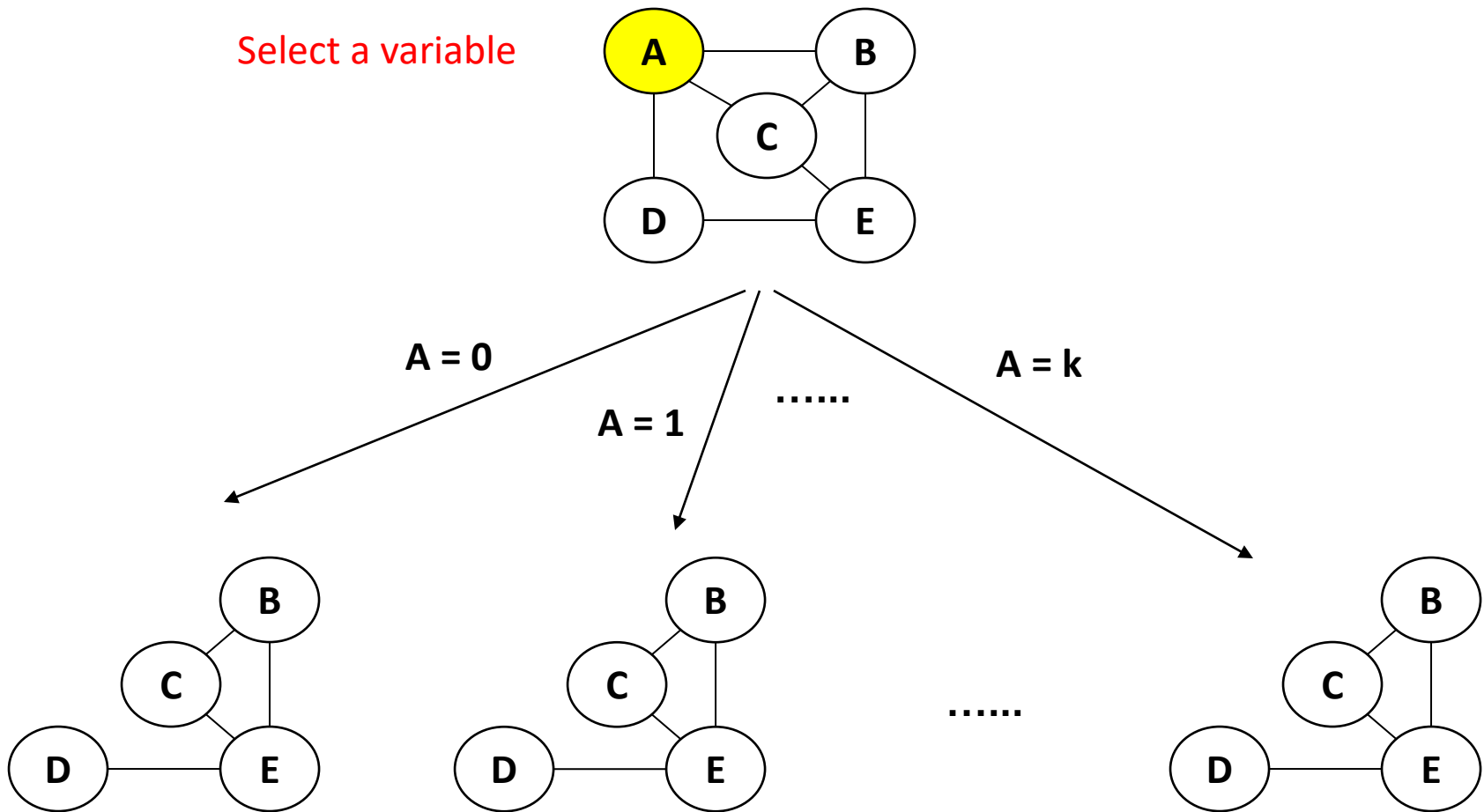
Select a variable



Hybrid: Cutset-Conditioning

Variable Branching by Conditioning

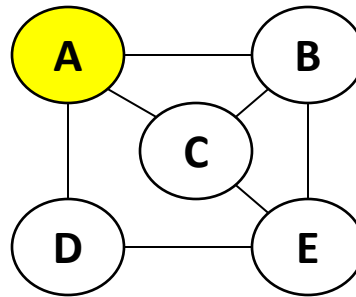
Select a variable



Hybrid: Cutset-Conditioning

Variable Branching by Conditioning

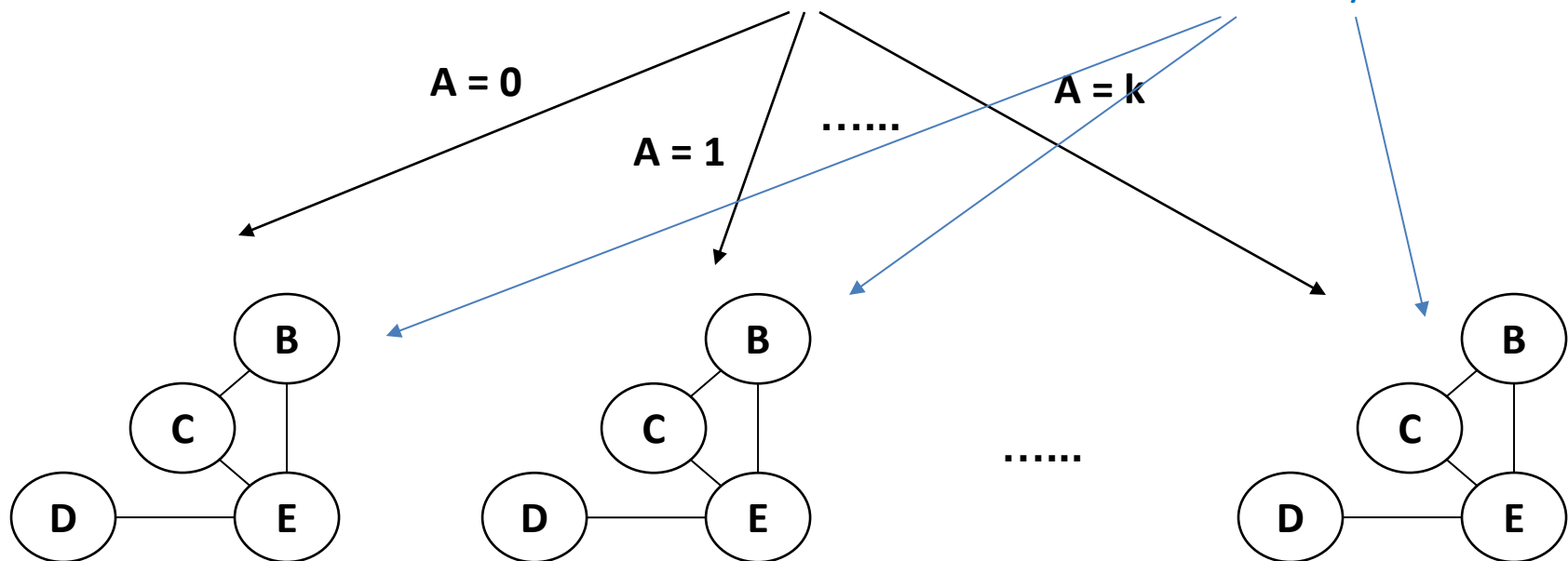
Select a variable



General principle:

Condition until tractable

Solve each sub-problem efficiently

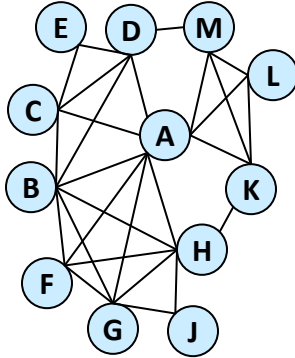


Hybrids Variants

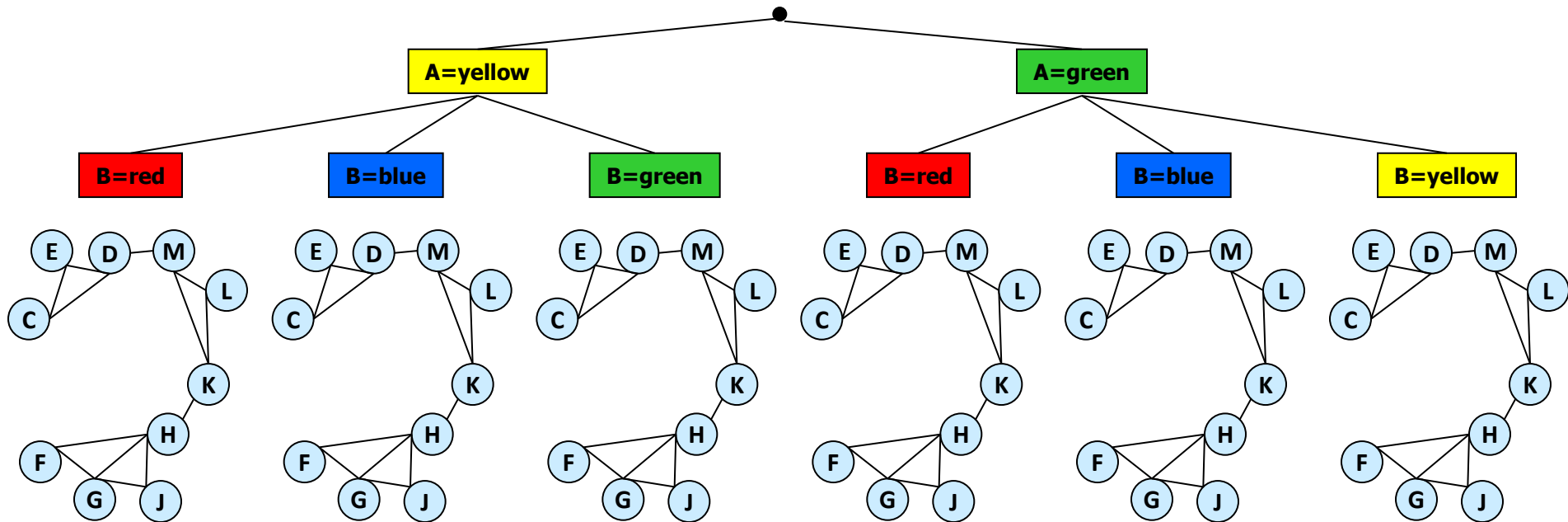
- **Condition, condition, condition**, ... and then only eliminate (w-cutset, cycle-cutset VEC(i))
- **Eliminate, eliminate, eliminate**, ... and then only search
- **Alternate** conditioning and elimination steps (elim-cond(i), ALT-VEC(i))

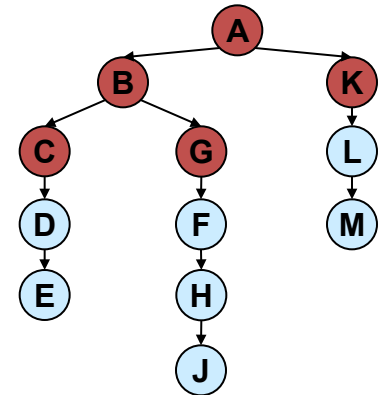
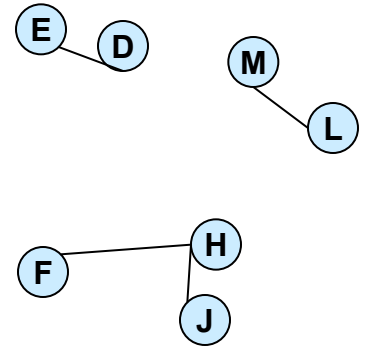
OR w-Cutset

Graph
Coloring
problem



- Inference may require too much memory
- **Condition** on some of the variables





1-cutset

Summary: Search methods

- **AND/OR search spaces** exploit the structure of the graphical model and create a far more compact search space.
 - **AND/OR Trees** are $\exp(\text{height})$ of the pseudo-tree and can be traversed in linear memory
 - **AND/OR Graphs** are $\exp(\text{induced-width})$ of the pseudo-tree and require $\exp(w)$ memory when traversed.
 - **The pseudo-tree structure** is instrumental in facilitating effective search
- **The MBE heuristic** can guide heuristic search (depth-first, Best-first or hybrid) pruning search further.
- **Tasks:** this schemes are applicable to a large class of tasks:
 - Belief updating, marginal map and Influence diagram search
- **Mixed schemes of Inference and Search** like cutset schemes facilitate tradeoff between memory and time.