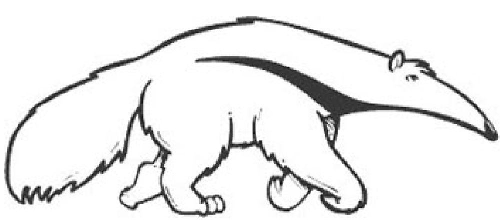
Algorithms for Reasoning with Probabilistic Graphical Models

Class 2: Search

International Summer School on Deep Learning
July 2017

Prof. Rina Dechter
Prof. Alexander Ihler

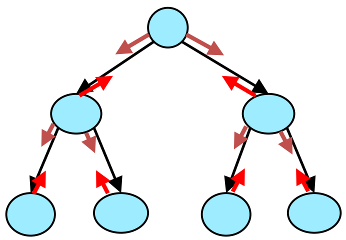
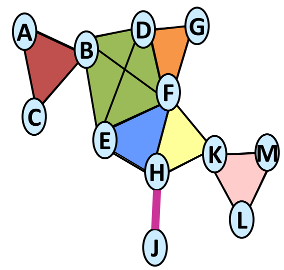


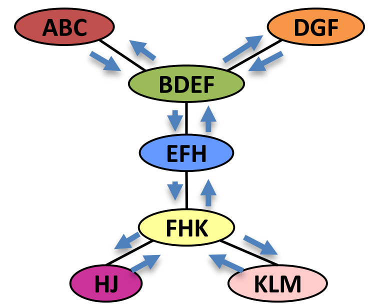
BREN:ICS INFORMATION AND COMPUTER SCIENCES

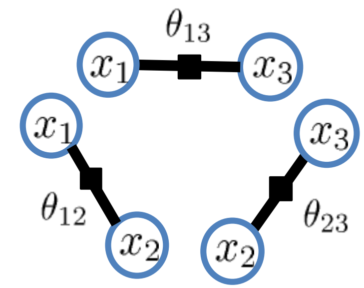
UNIVERSITY of CALIFORNIA IRVINE

Outline of Lectures

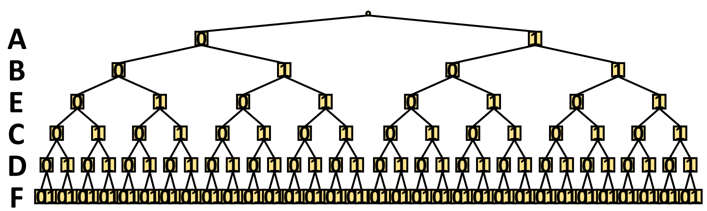
- Class 1: Introduction and Inference

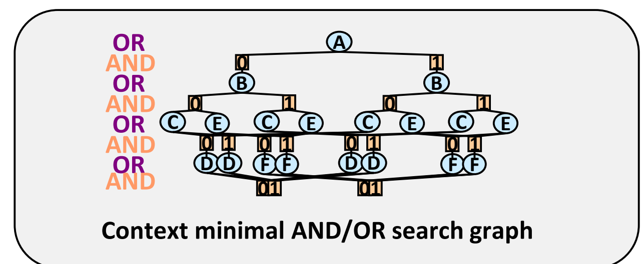







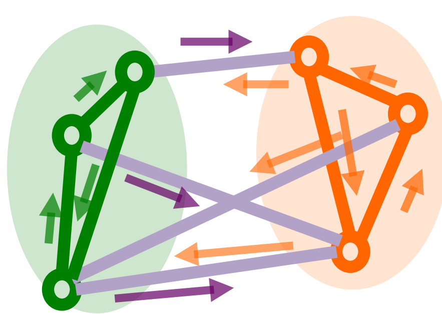
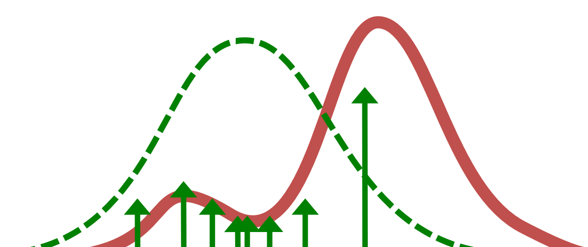
- Class 2: Search





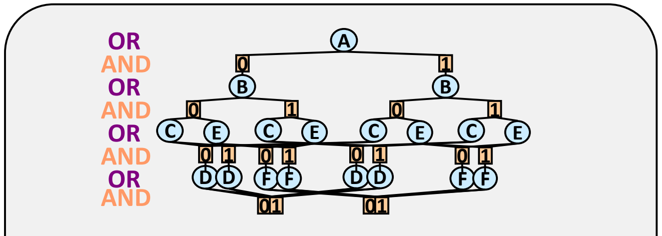
18 AND nodes

- Class 3: Variational Methods and Monte-Carlo Sampling

Road Map: Search

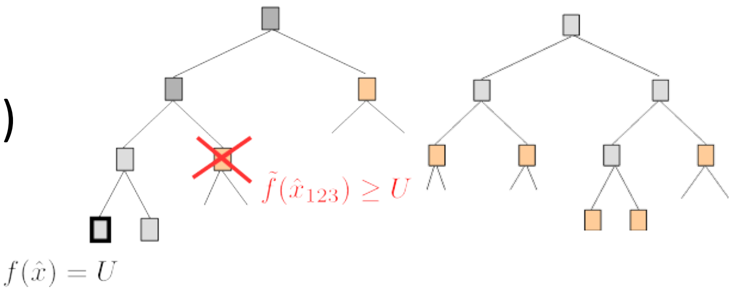
- Review Graphical Modes
- AND/OR search spaces, pseudo-trees
  - AND/OR search trees
  - AND/OR search graphs
  - Generating good pseudo-trees
  - Brute-force AND/OR



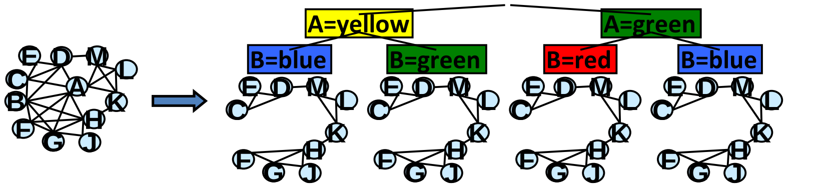
Context minimal AND/OR search graph

18 AND nodes

- Heuristic search (HS) for AND/OR spaces
  - Basic Heuristic search (Depth and Best)
  - AND/OR Depth-first HS (branch and bound)
  - AND/OR Best-first heuristic search
  - The Guiding MBE heuristic
  - Marginal Map (max-sum-product)



- Hybrids of search and Inference
- Summary and Class 2



Graphical models

A *graphical model* consists of:

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

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

$F = \{f_{\alpha_1}, \ldots, f_{\alpha_m}\}$ -- functions or "factors"

and a *combination operator*

Example:

$A \in \{0, 1\}$

$B \in \{0, 1\}$

$C \in \{0, 1\}$

$f_{AB}(A, B), \quad f_{BC}(B, C)$

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

e.g., "+" : $F(A, B, C) = f_{AB}(A, B) + f_{BC}(B, C)$

Notation:

Discrete Xi 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., $f_\alpha(X_\alpha), \quad X_\alpha \subseteq X$

Graphical models

A *graphical model* consists of:

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

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

$F = \{f_{\alpha_1}, \ldots, f_{\alpha_m}\}$ -- functions or "factors"

and a *combination operator*

Example:

$A \in \{0, 1\}$

$B \in \{0, 1\}$

$C \in \{0, 1\}$

$f_{AB}(A, B), \quad f_{BC}(B, C)$

$$F(A, B, C) = f_{AB}(A, B) + f_{BC}(B, C)$$

For discrete variables, think of functions as "tables"
(though we might represent them more efficiently)

A	B	f(A,B)
0	0	6
0	1	0
1	0	0
1	1	6

+

B	C	f(B,C)
0	0	6
0	1	0
1	0	0
1	1	6

=

A	B	C	f(A,B,C)
0	0	0	12
0	0	1	6
0	1	0	0
0	1	1	6
1	0	0	6
1	0	1	0
1	1	0	6
1	1	1	12

$F(A = 0, B = 1, C = 1)$ = 0 + 6

Canonical forms

A *graphical model* consists of:

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

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

$F = \{f_{\alpha_1}, \ldots, f_{\alpha_m}\}$ -- functions or "factors"

and a *combination operator*

Typically either multiplication or summation; mostly equivalent:

$$f_\alpha(X_\alpha) \geq 0$$

$$F(X) = \prod_\alpha f_\alpha(X_\alpha)$$

log / exp

$$\theta_\alpha(X_\alpha) = \log f_\alpha(X_\alpha) \in \mathbb{R}$$

$$\theta(X) = \log F(x) = \sum_\alpha \theta_\alpha(X_\alpha)$$

Product of nonnegative factors
(probabilities, 0/1, etc.)

Sum of factors
(costs, utilities, etc.)

Graphical visualization

A *graphical model* consists of:

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

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

$F = \{f_{\alpha_1}, \ldots, f_{\alpha_m}\}$ -- functions or “factors”

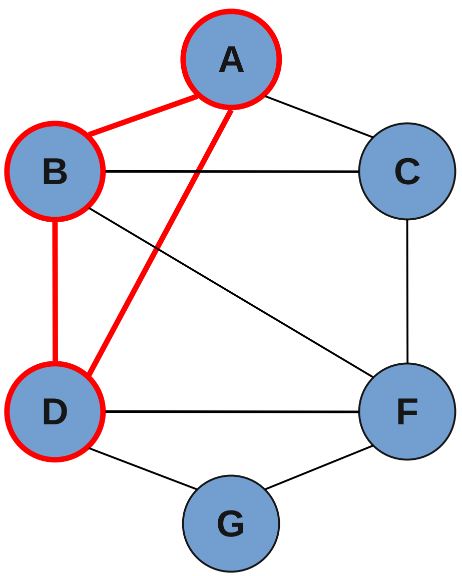
and a *combination operator*

Primal graph:

variables → nodes

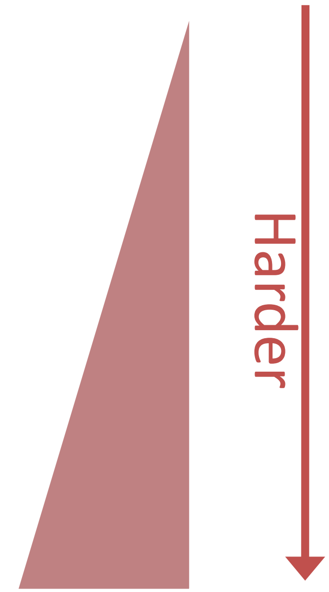
factors → cliques

$$F(A,B,C,D,F,G) = f_1(A,B,D) + f_2(D,F,G) + f_3(B,C,F) + f_4(A,C)$$



Types of queries

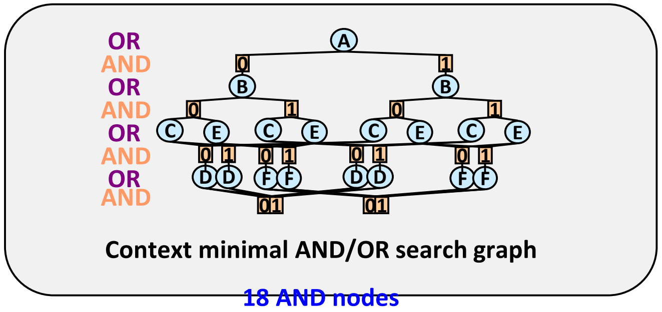
Max-Inference	$f(\mathbf{x}^*) = \max_{\mathbf{x}} \prod_{\alpha} f_\alpha(\mathbf{x}_\alpha)$
Sum-Inference	$Z = \sum_{\mathbf{x}} \prod_{\alpha} f_\alpha(\mathbf{x}_\alpha)$
Mixed-Inference	$f(\mathbf{x}_M^*) = \max_{\mathbf{x}_M} \sum_{\mathbf{x}_S} \prod_{\alpha} f_\alpha(\mathbf{x}_\alpha)$

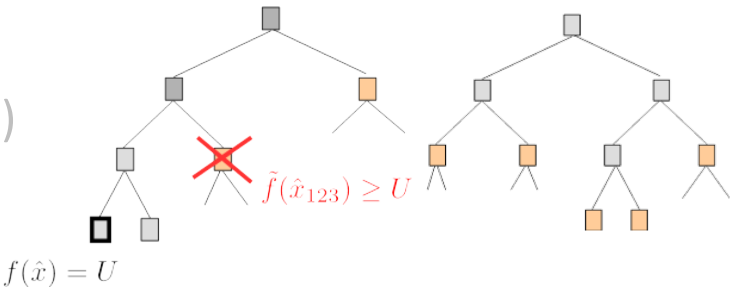


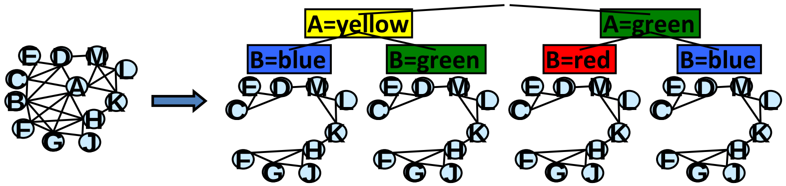
- **NP-hard**: exponentially many terms
- We will focus on **approximation** algorithms
  - **Anytime**: very fast & very approximate ! Slower & more accurate

Road Map: Search

- Review Graphical Modes
- AND/OR search spaces, pseudo-trees
  - AND/OR search trees
  - AND/OR search graphs
  - Generating good pseudo-trees
  - Brute-force AND/OR
- Heuristic search (HS) for AND/OR spaces
  - Basic Heuristic search (Depth and Best)
  - AND/OR Depth-first HS (branch and bound)
  - AND/OR Best-first heuristic search
  - The Guiding MBE heuristic
  - Marginal Map (max-sum-product)
- Hybrids of search and Inference
- Summary and Class 2







Conditioning - the Probability Tree

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

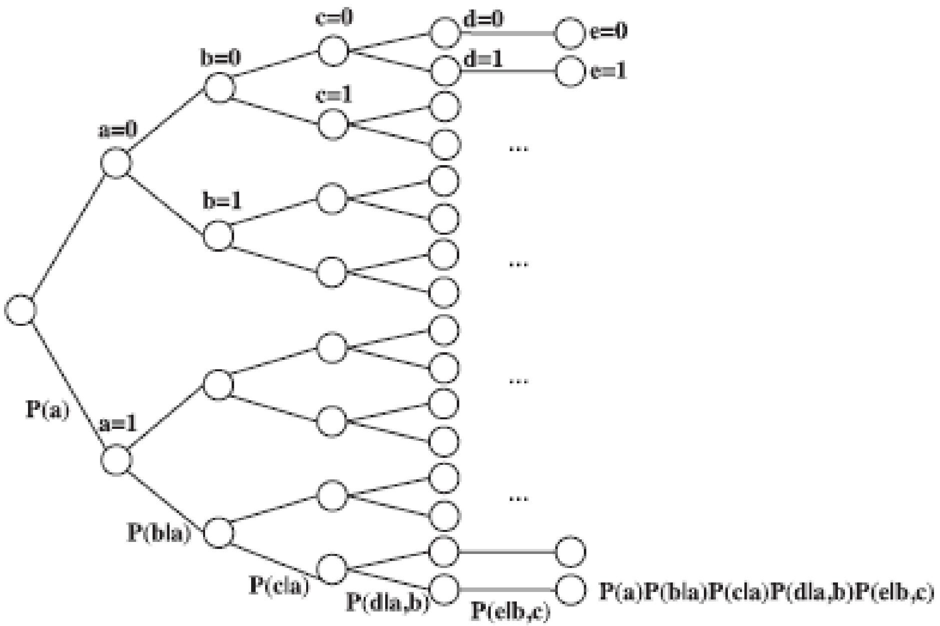
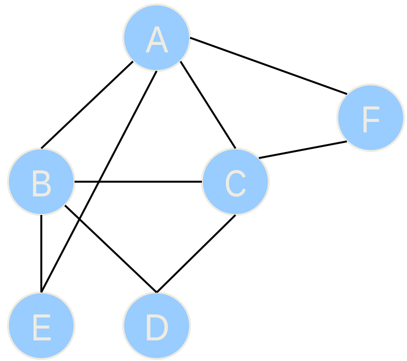


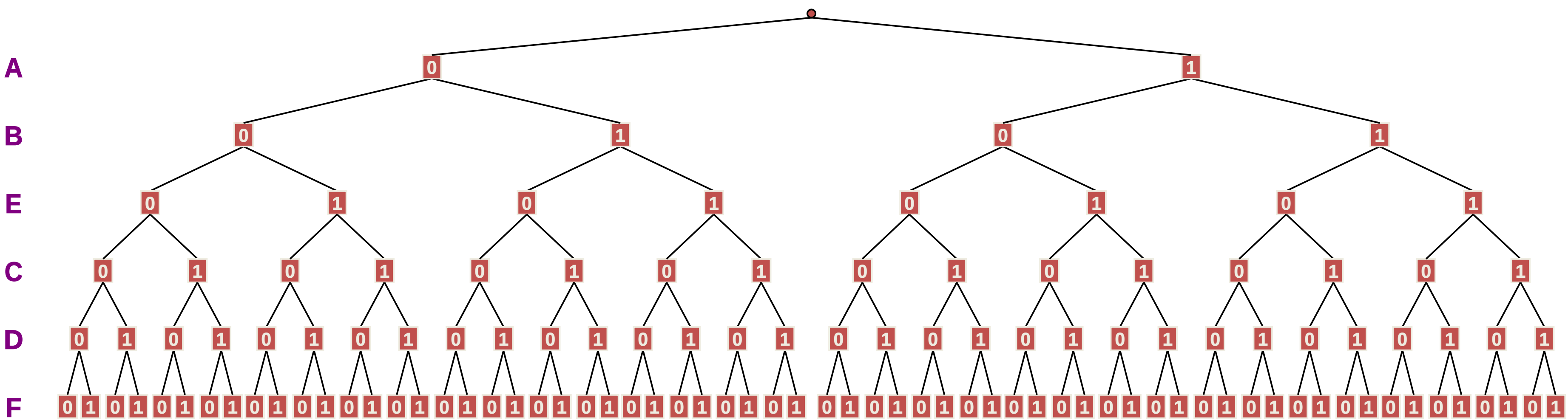
Figure 6.1: Probability tree for computing P(d=1,g=0).

Complexity of conditioning: exponential time, linear space

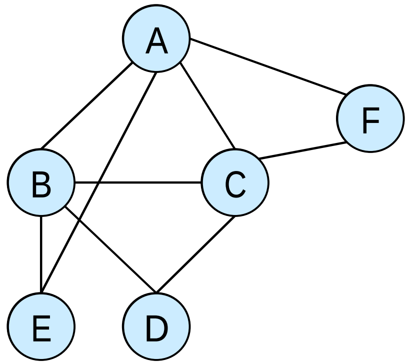
The Classic OR Search Space



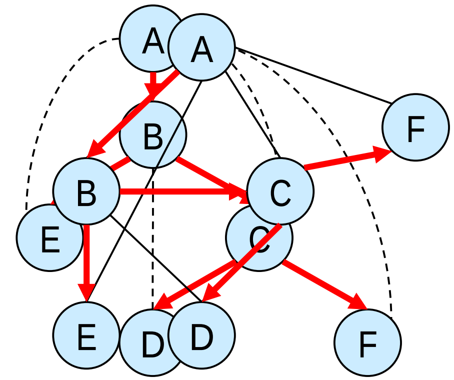
Ordering: A B E C D F



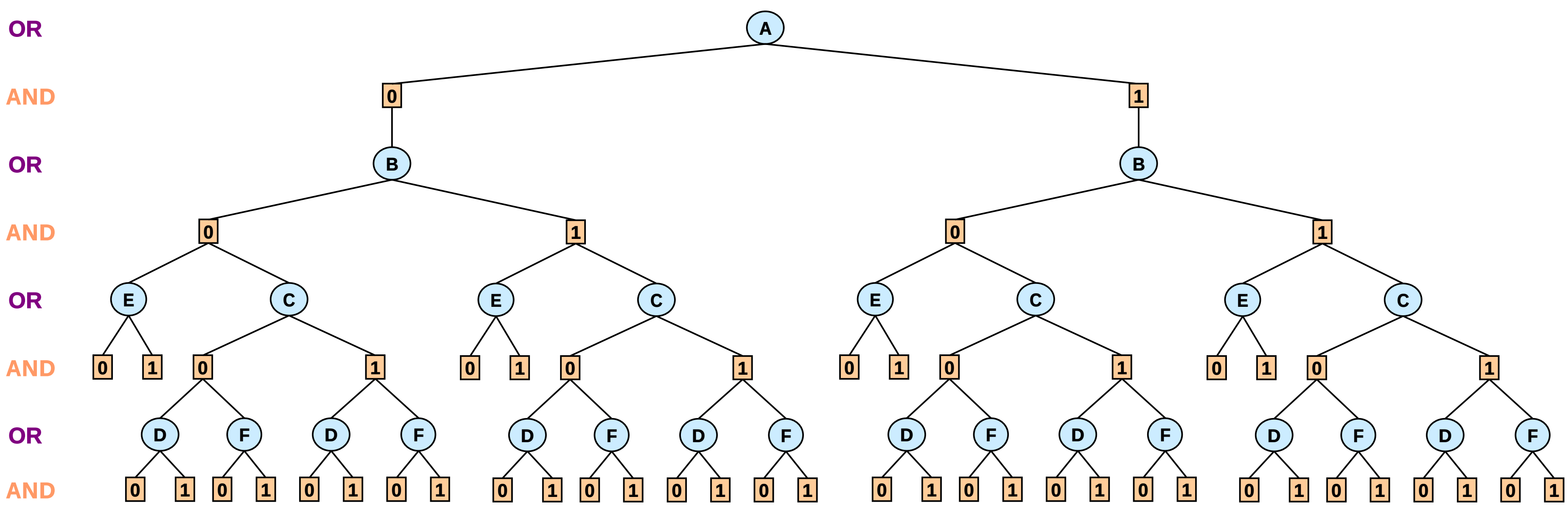
AND/OR Search Space



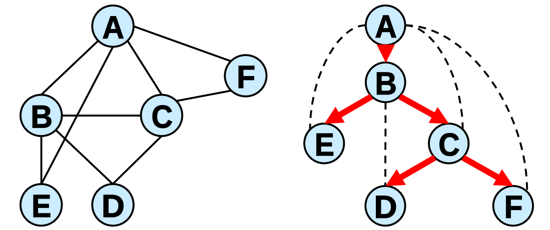
Primal graph

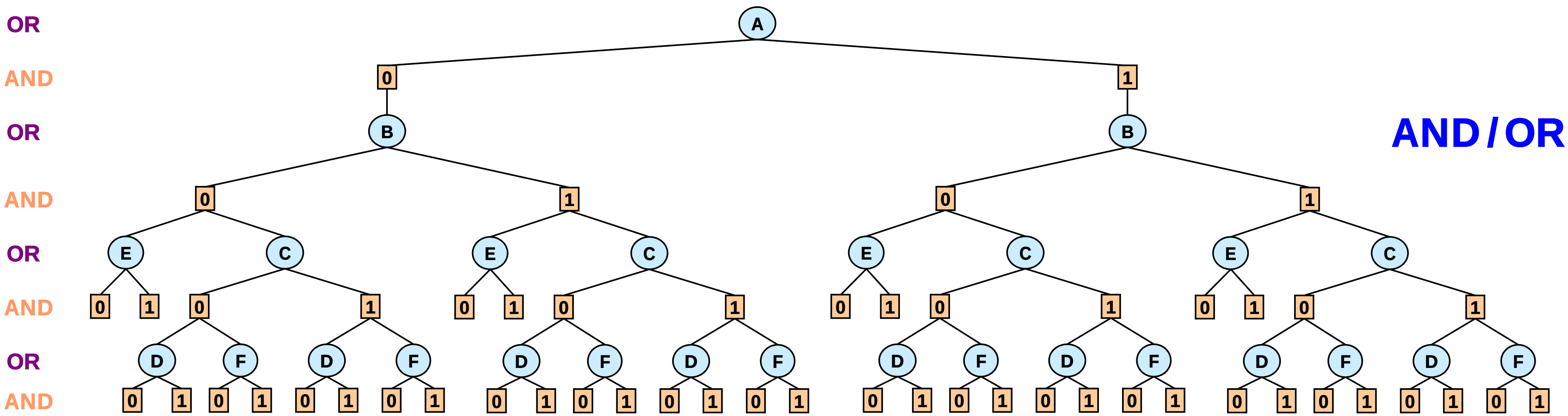


DFS tree

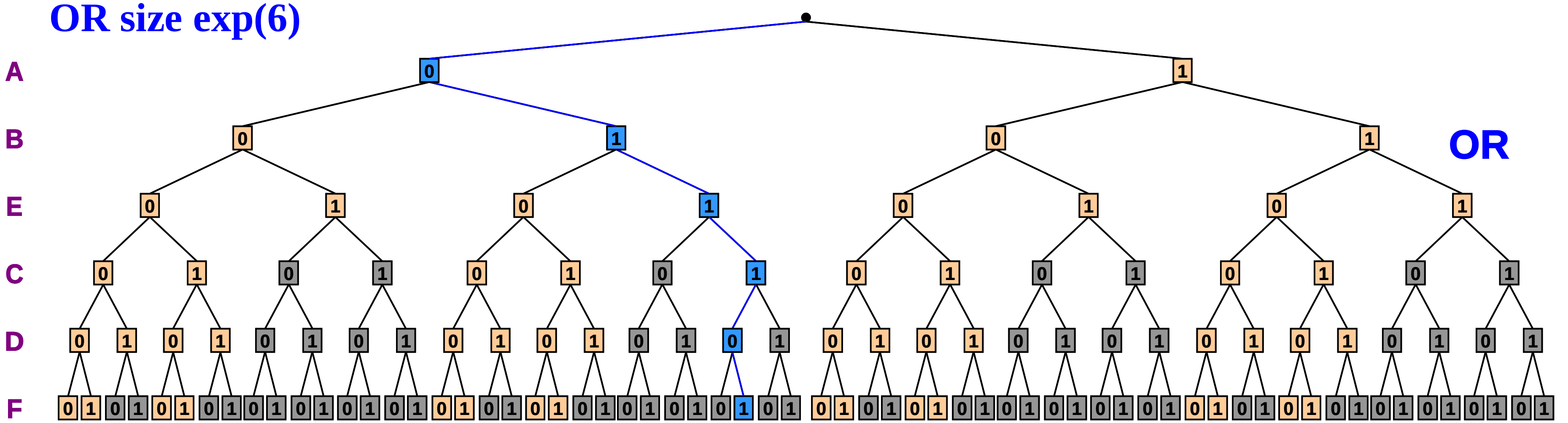


AND/OR vs. OR

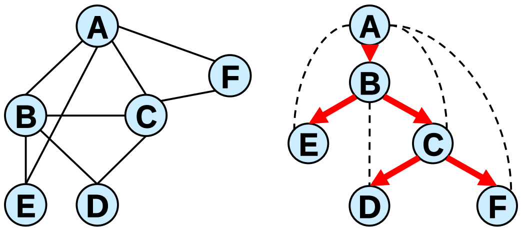


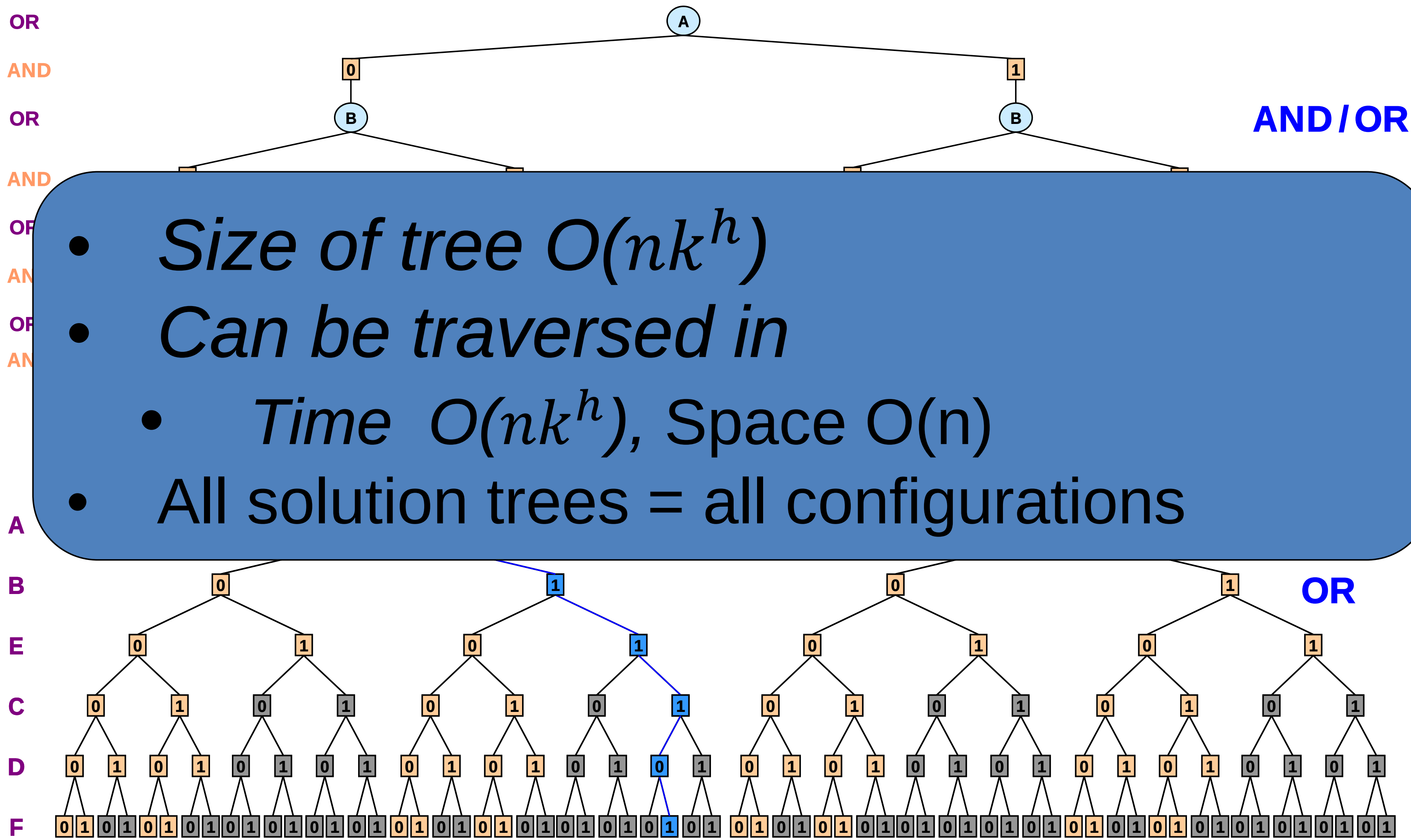


AND/OR size: exp(4),
OR size exp(6)



AND/OR vs. OR





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

Arc weights
Cost of a solution tree
The value function

Arc Weights for AND/OR Trees

$P(E \mid A,B)$

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

$P(B \mid A)$

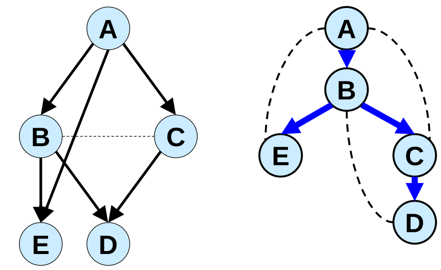
A	B=0	B=1
0	.4	.6
1	.1	.9

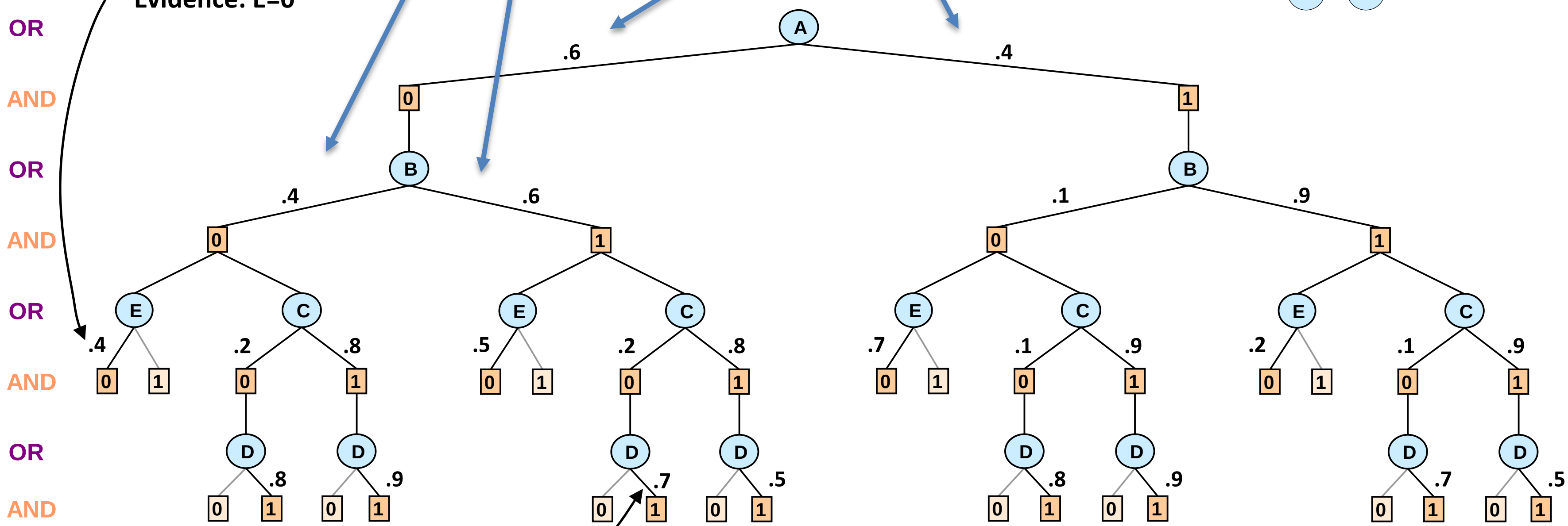
$P(C \mid A)$

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

$P(A)$

A	P(A)
0	.6
1	.4





$P(D \mid B,C)$

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 <X,x> 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

$P(E \mid A,B)$

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

$P(B \mid A)$

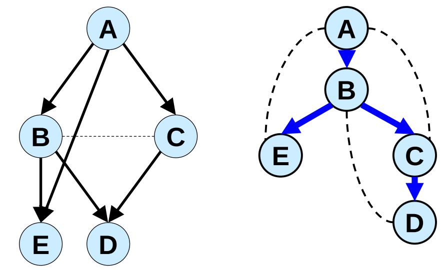
A	B=0	B=1
0	.4	.6
1	.1	.9

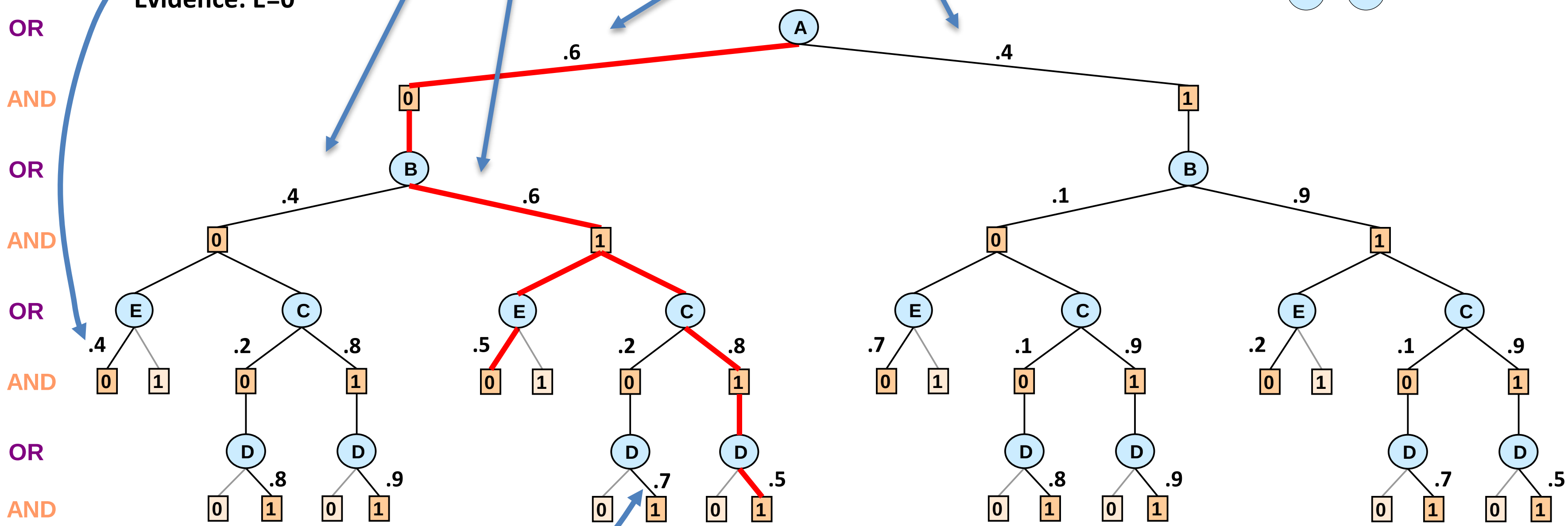
$P(C \mid A)$

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

$P(A)$

A	P(A)
0	.6
1	.4





$P(D \mid B,C)$

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 · 0.6 · 0.5 · 0.8 · 0.5 = 0.0720

The Value Function for (Probability of Evidence)

$P(E \mid A, B)$

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

$P(B \mid A)$

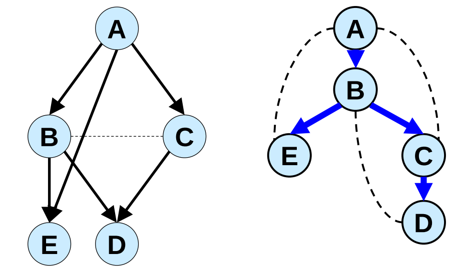
A	B=0	B=1
0	.4	.6
1	.1	.9

$P(C \mid A)$

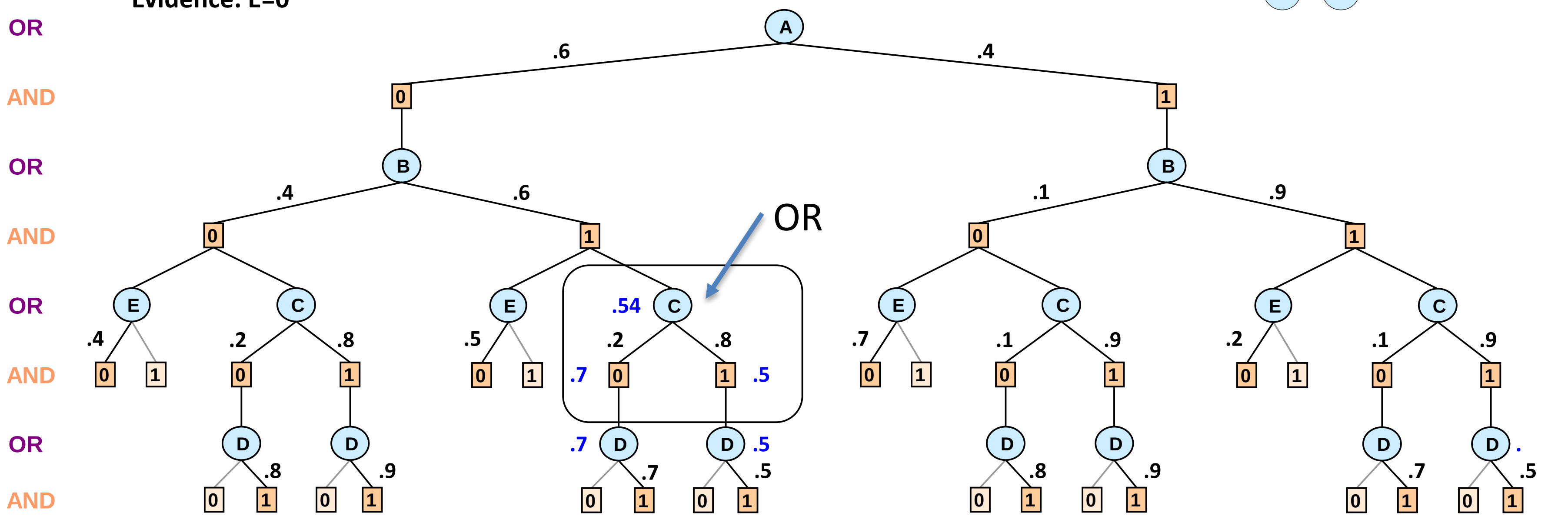
A	C=0	C=1
0	.2	.8
1	.7	.3

$P(A)$

A	P(A)
0	.6
1	.4



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



$P(D \mid B, C)$

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

OR node: Marginalization by summation $\sum_{n' \in children(n)} w(n,n')\, v(n')$

The Value Function (Probability of Evidence)

$P(E \mid A,B)$

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

$P(B \mid A)$

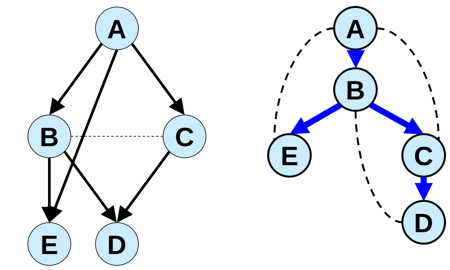
A	B=0	B=1
0	.4	.6
1	.1	.9

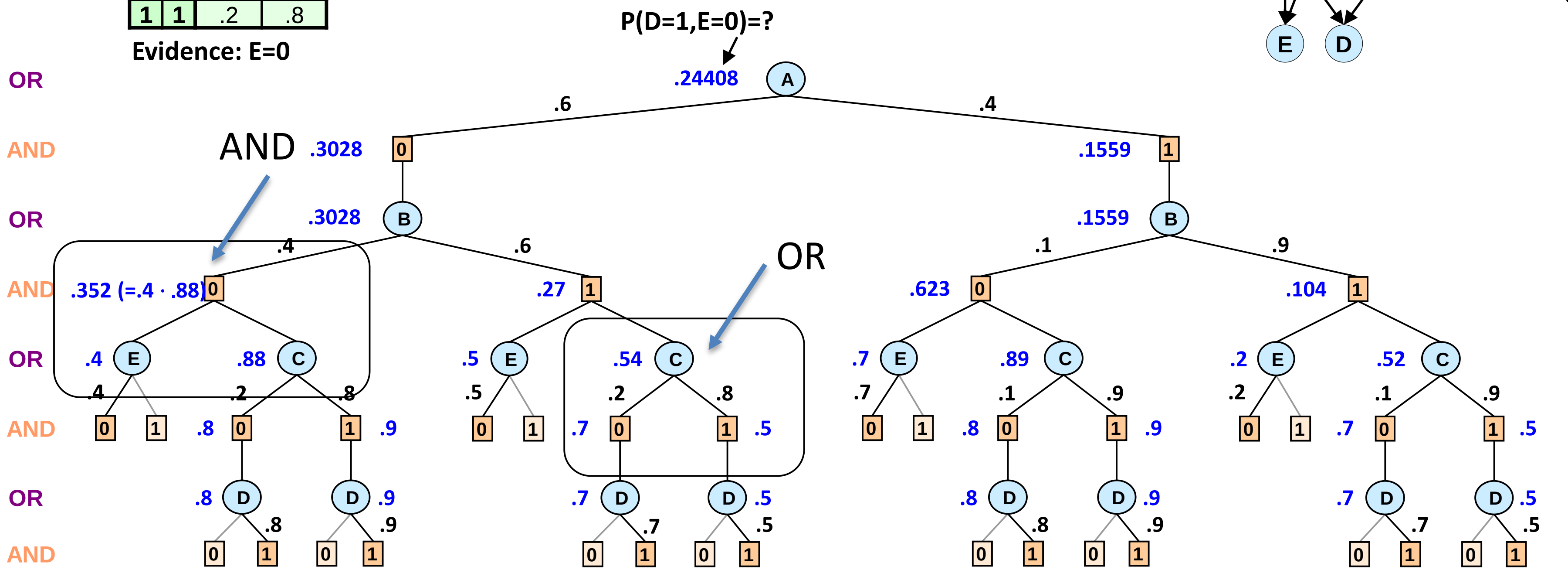
$P(C \mid A)$

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

$P(A)$

A	P(A)
0	.6
1	.4





$P(D \mid B,C)$

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

OR node: Marginalization by summation $\sum_{n' \in children(n)} w(n,n')\, v(n')$

The Value Function

$P(E \mid A,B)$

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

$P(B \mid A)$

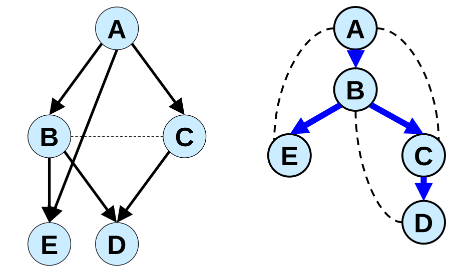
A	B=0	B=1
0	.4	.6
1	.1	.9

$P(C \mid A)$

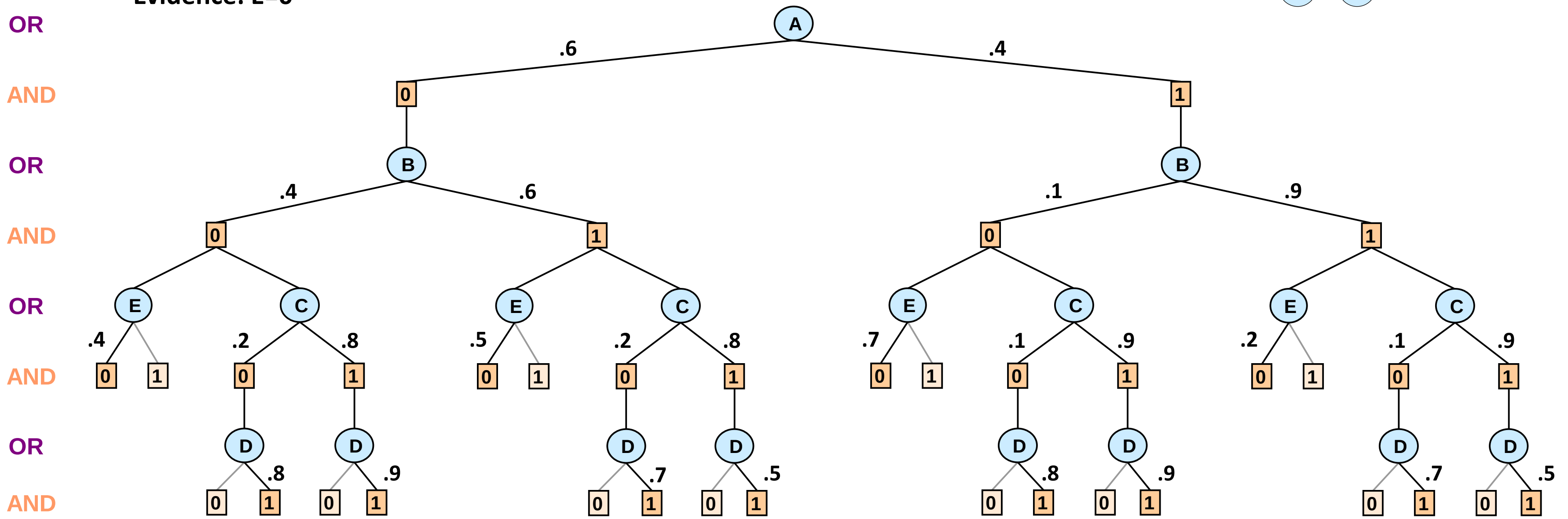
A	C=0	C=1
0	.2	.8
1	.7	.3

$P(A)$

A	P(A)
0	.6
1	.4



Evidence: E=0



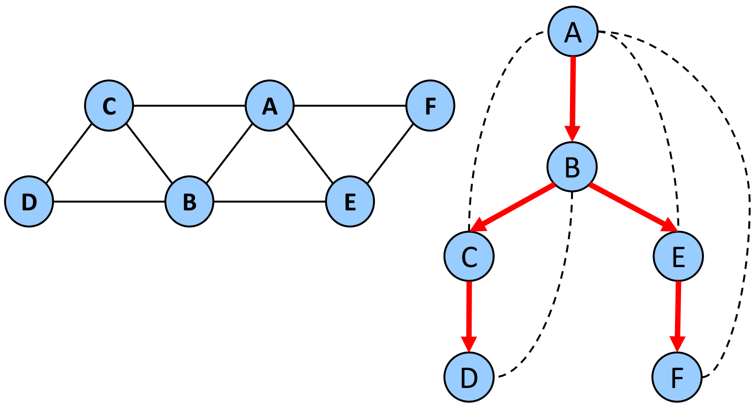
$P(D \mid B,C)$

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 mess below n
- Can be computed recursively based on child values.

The Value Function for Optimization



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

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

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

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

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

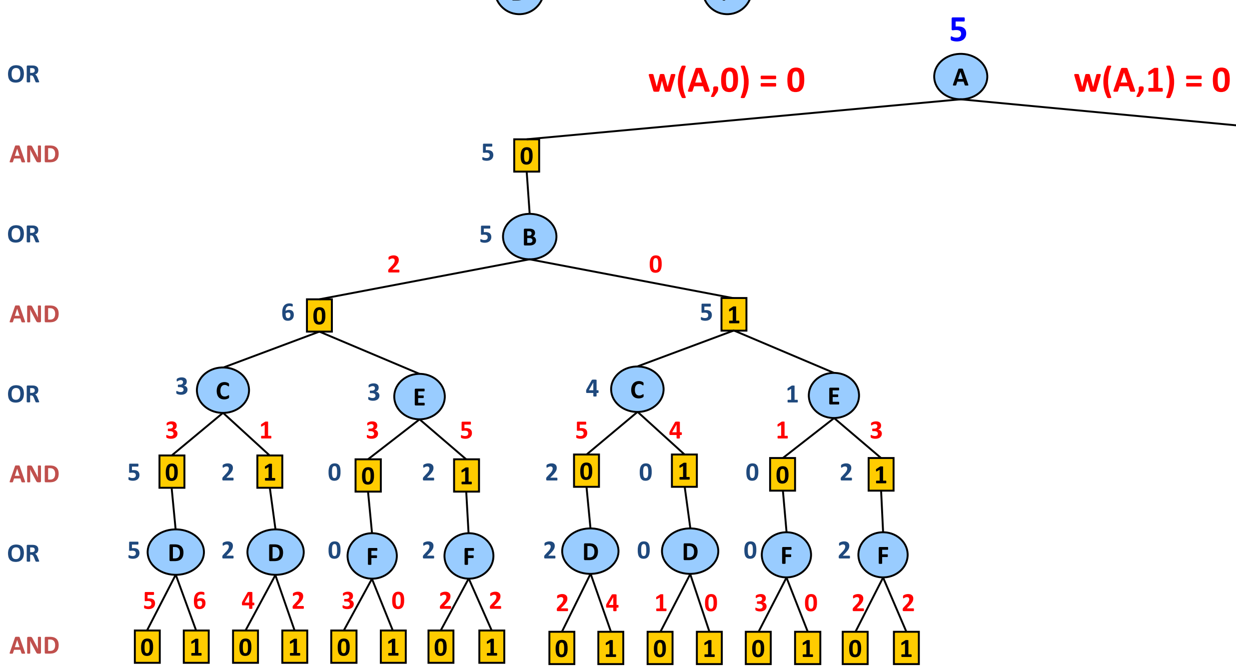
B	D	f_6
0	0	4
0	1	2
1	0	1
1	1	0

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

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

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

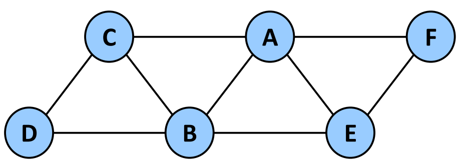
Objective function: $F^* = \min_x \sum_\alpha f_\alpha(x_\alpha)$

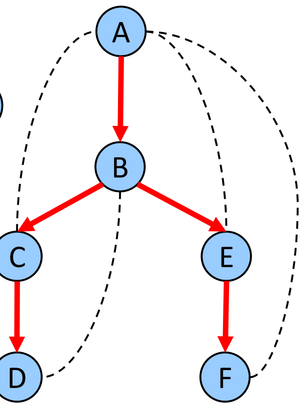


Node Value
(bottom-up evaluation)

OR – minimization
AND – summation

The Value Function for Optimization





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

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

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

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

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

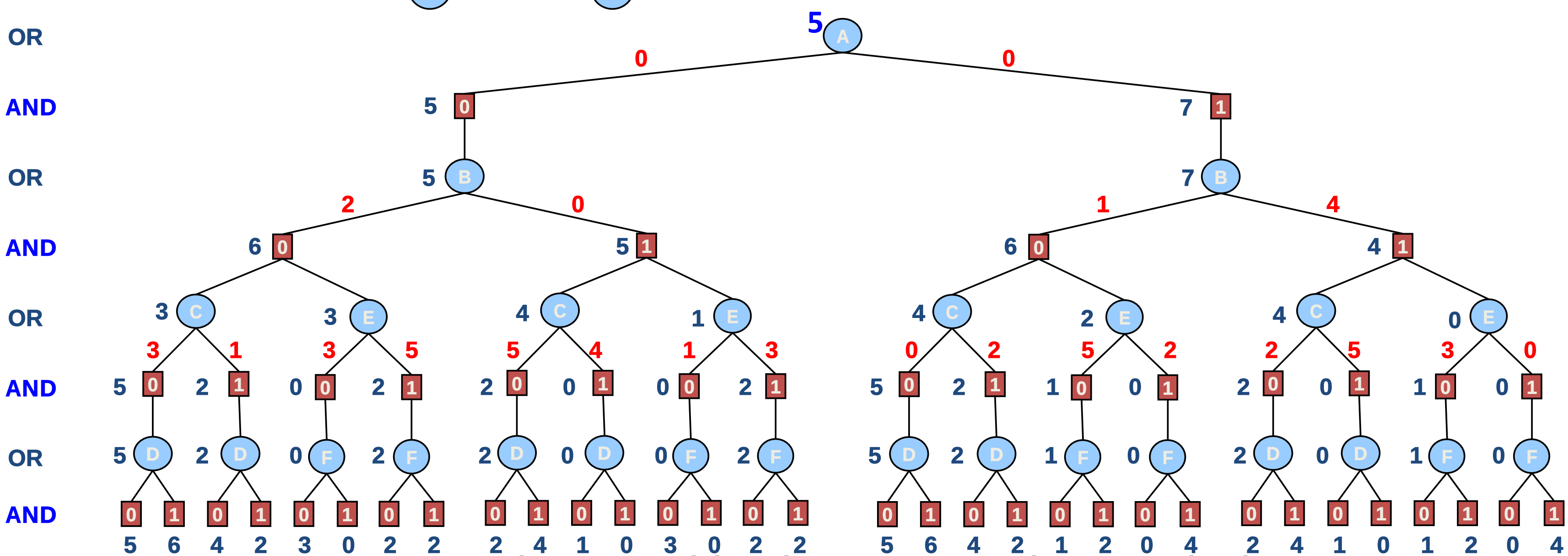
B	D	f_6
0	0	4
0	1	2
1	0	1
1	1	0

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

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

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

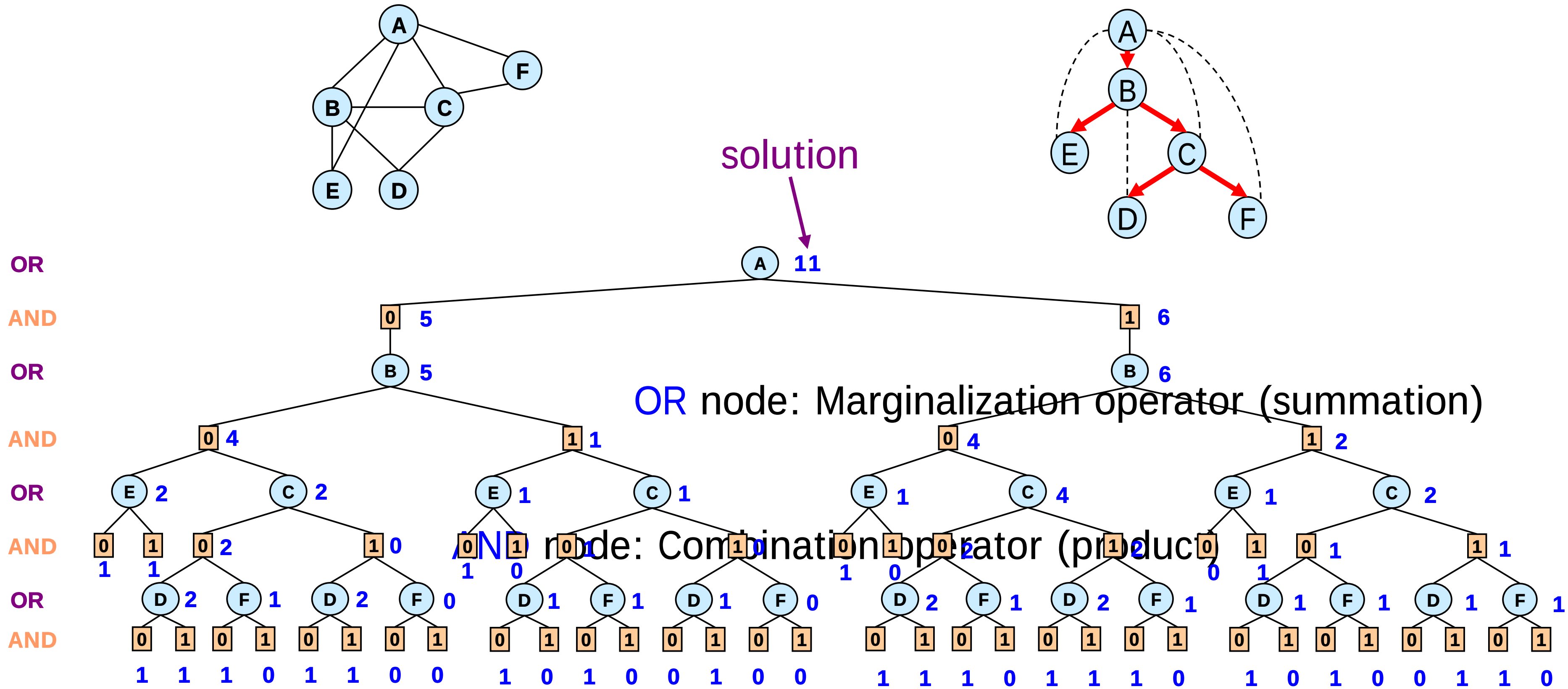
Objective function: $F^* = \min_x \sum_\alpha f_\alpha(x_\alpha)$



AND node = Combination operator (summation)

OR node = Marginalization operator (minimization)

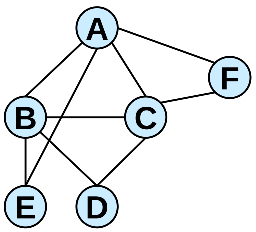
The AND/OR Counting Value(#CSP)



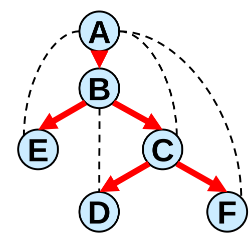
Value of node = number of solutions below it

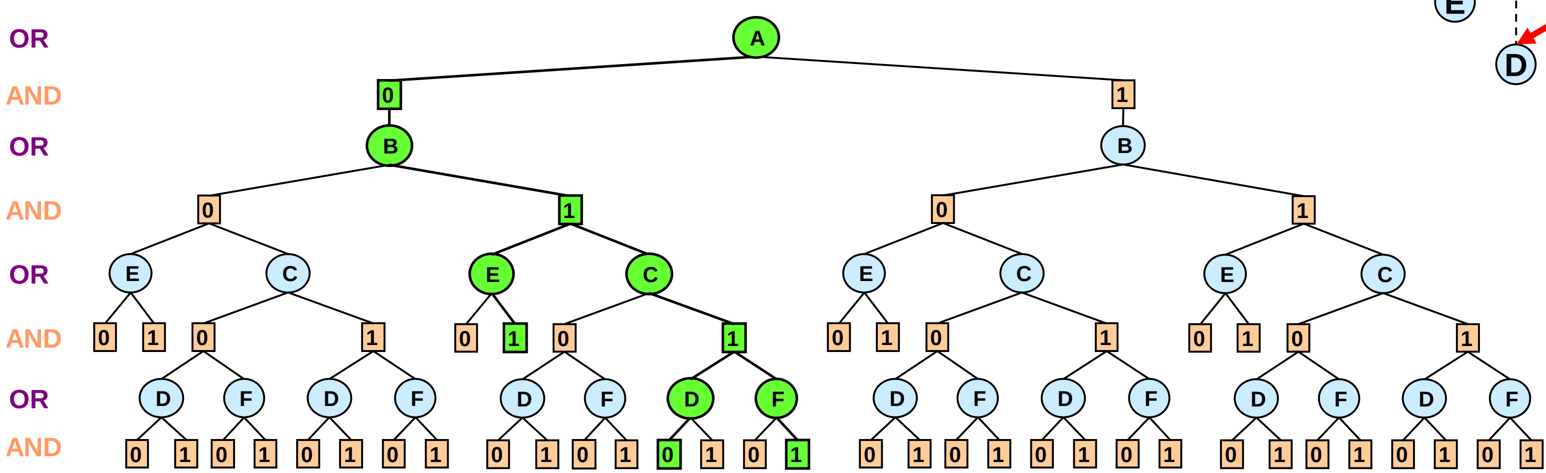
Summary: AND/OR Search Tree for GMs

- **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 <X,v> 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

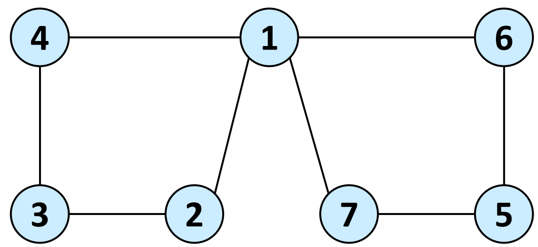
width	height	OR space		AND/OR space		
		Time (sec.)	Nodes	Time (sec.)	AND nodes	OR nodes
5	10	3.15	2,097,150	0.03	**10,494**	5,247
4	9	3.13	2,097,150	0.01	**5,102**	2,551
5	10	3.12	2,097,150	0.03	**8,926**	4,463
4	10	3.12	2,097,150	0.02	**7,806**	3,903
5	13	3.11	2,097,150	0.10	**36,510**	18,255

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

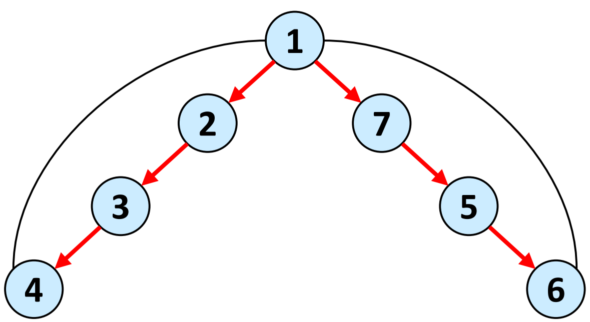
Pseudo Trees

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

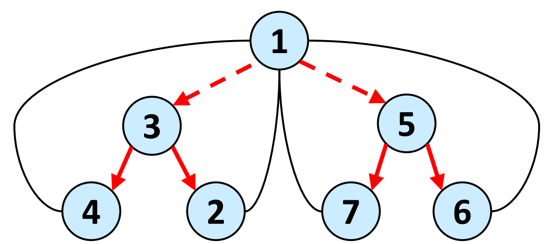
$$h \leq w^* \log n$$



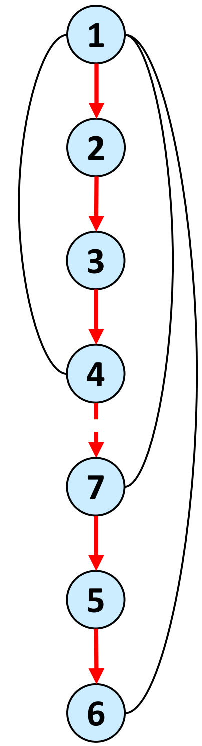
(a) Graph



(b) DFS tree
height=3

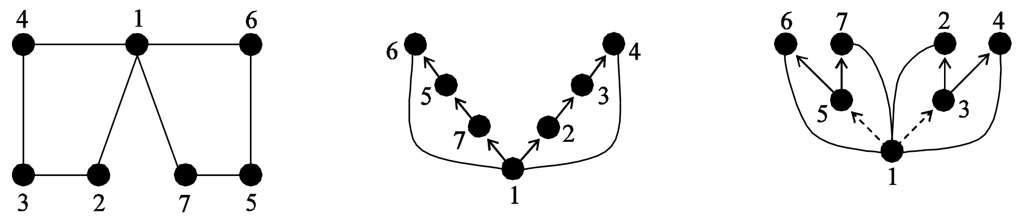


(c) Pseudo tree
height=2

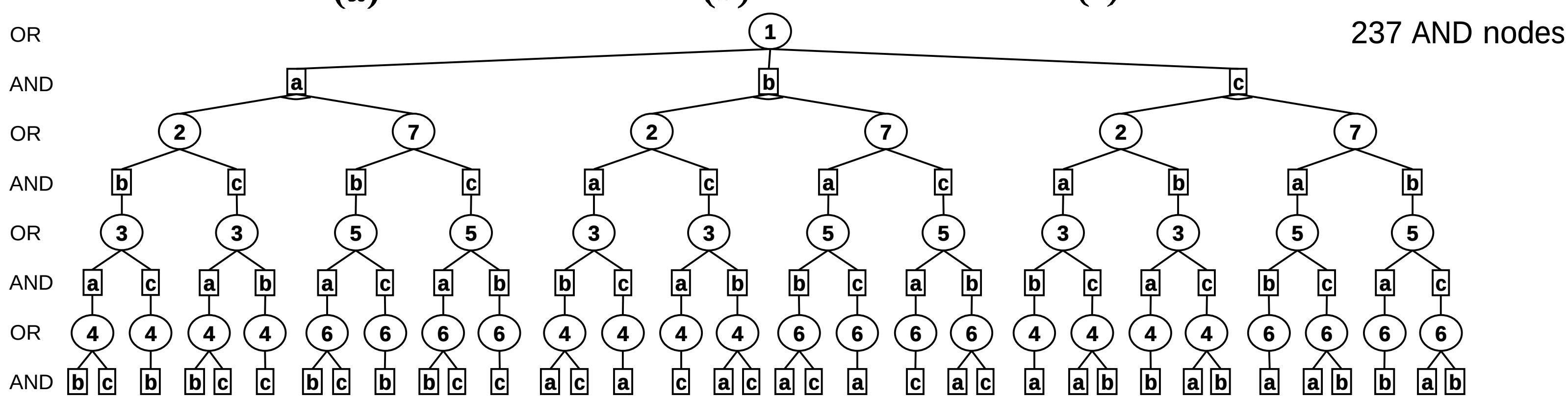


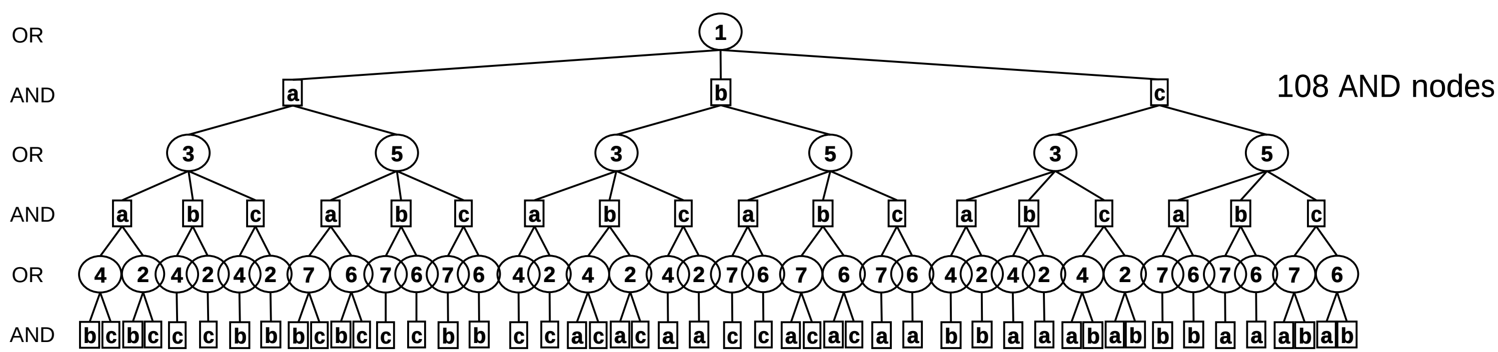
(d) Chain
height=6

From DFS-Trees to Pseudo-Trees



(a) (b) (c)





Summary: Queries and Value of Nodes

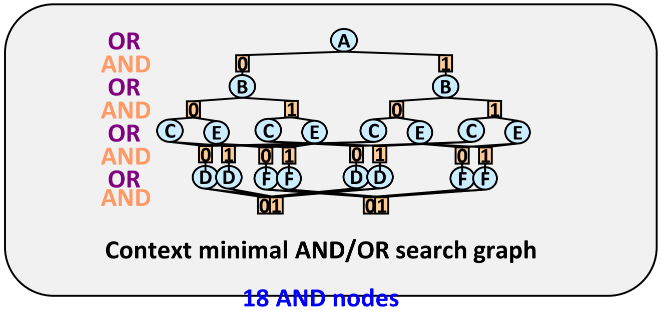
- V(n) is the value of the tree T(n) for the task:
  - Counting: v(n) is number of solutions in T(n)
  - ~~Consistency: v(n) is 0 if T(n) inconsistent, 1 othewise.~~
  - Max-Inference: v(n) is the optimal solution in T(n)
  - Sum-Inference: v(n) is probability of evidence in T(n).
  - Mixed-Inference: v(n) is the marginal map in T(n).
- Goal: compute the value of the root node recursively traversing the AND/OR tree.

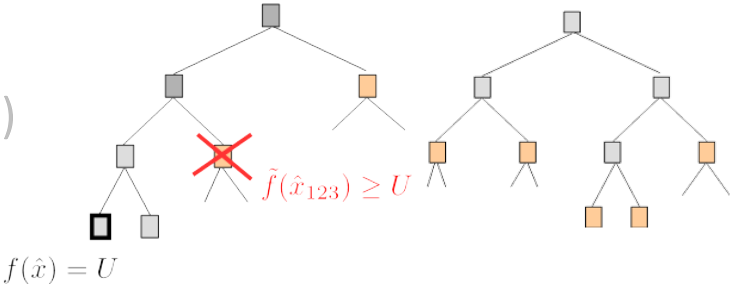
Complexity of searching depth-first is

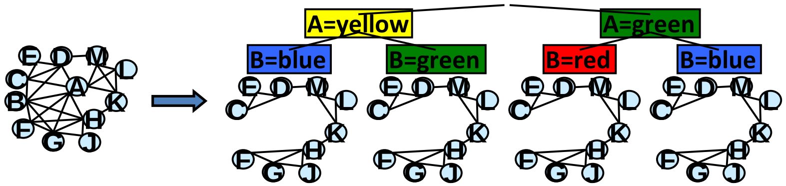
- Space: $O(n)$
- Time: $O(nk^h)$
- Time: $O(k^{w* \log n})$

Road Map: Search

- Review Graphical Modes
- AND/OR search spaces, pseudo-trees
  - AND/OR search trees
  - **AND/OR search graphs**
  - Generating good pseudo-trees
  - Brute-force AND/OR
- Heuristic search (HS) for AND/OR spaces
  - Basic Heuristic search (Depth and Best)
  - AND/OR Depth-first HS (branch and bound)
  - AND/OR Best-first heuristic search
  - The Guiding MBE heuristic
  - Marginal Map (max-sum-product)
- Hybrids of search and Inference
- Summary and Class 2

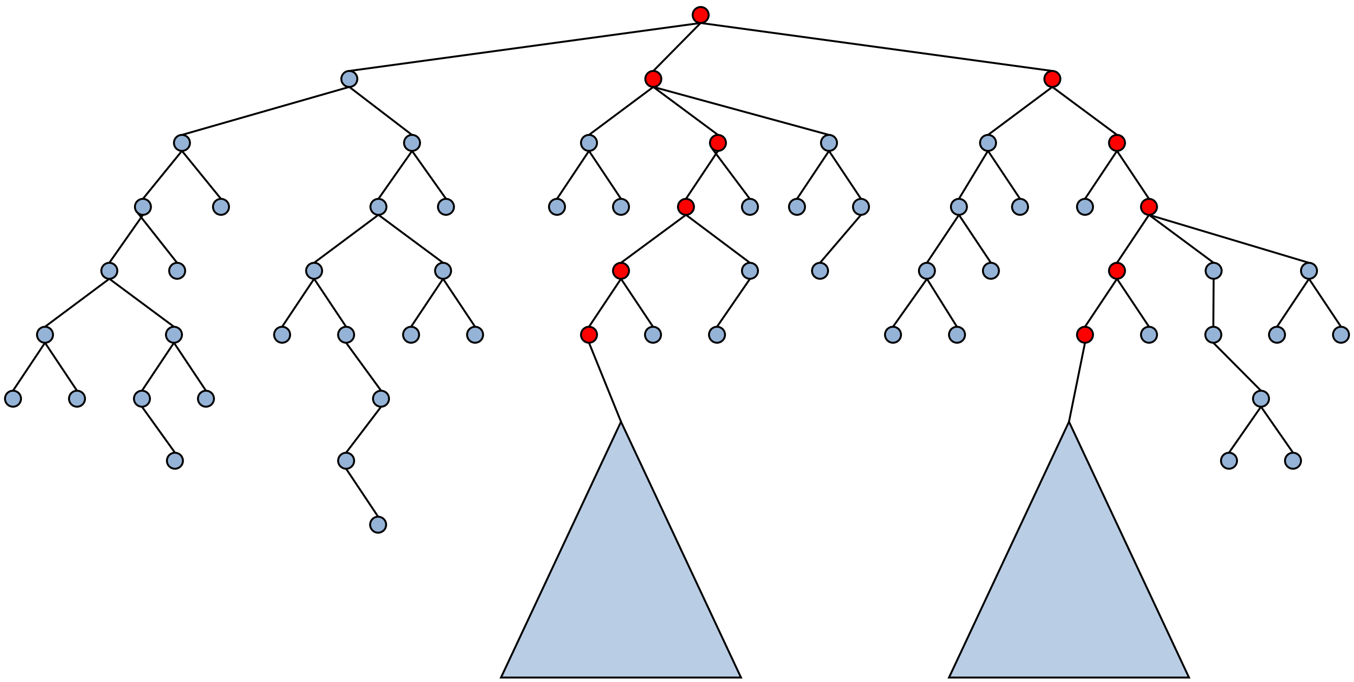






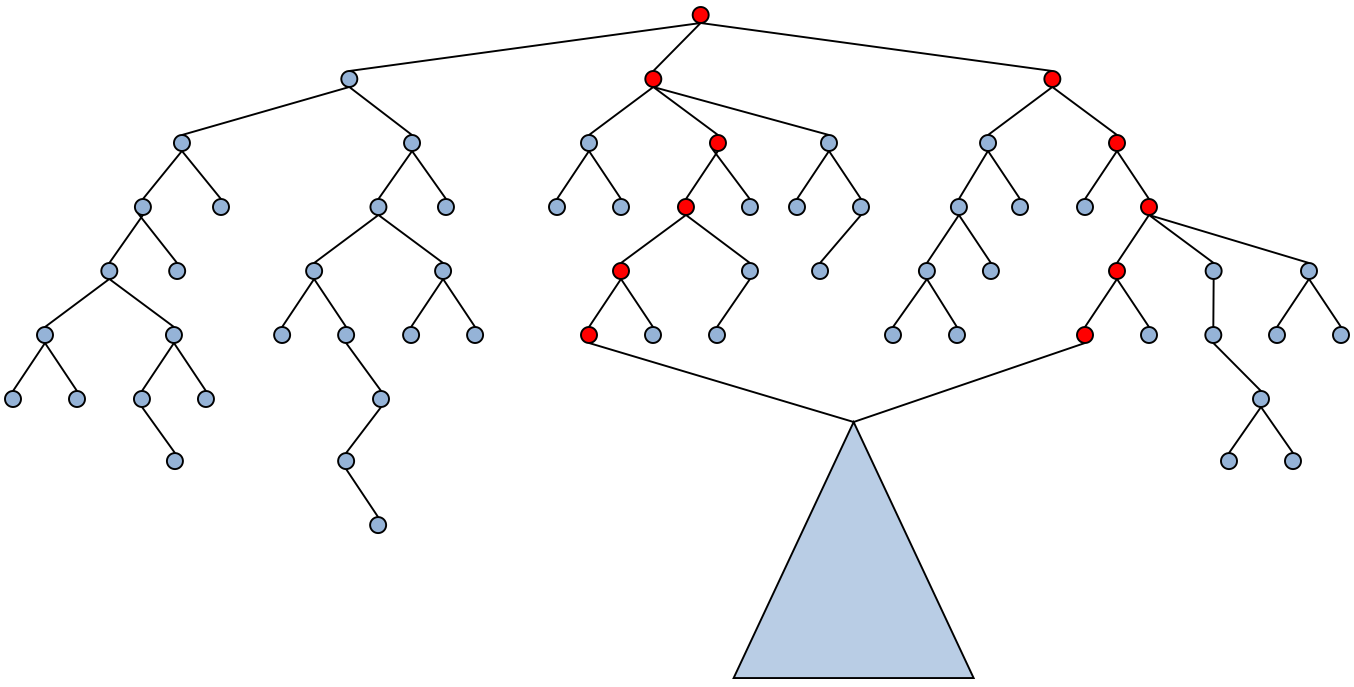
From Search Trees to Search Graphs

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

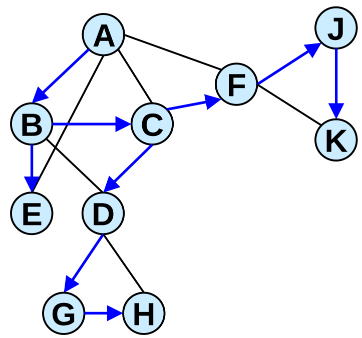


From Search Trees to Search Graphs

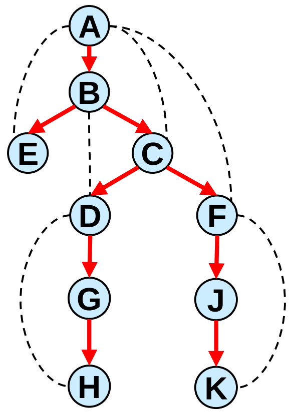
- Any two nodes that root identical subtrees or subgraphs can be merged



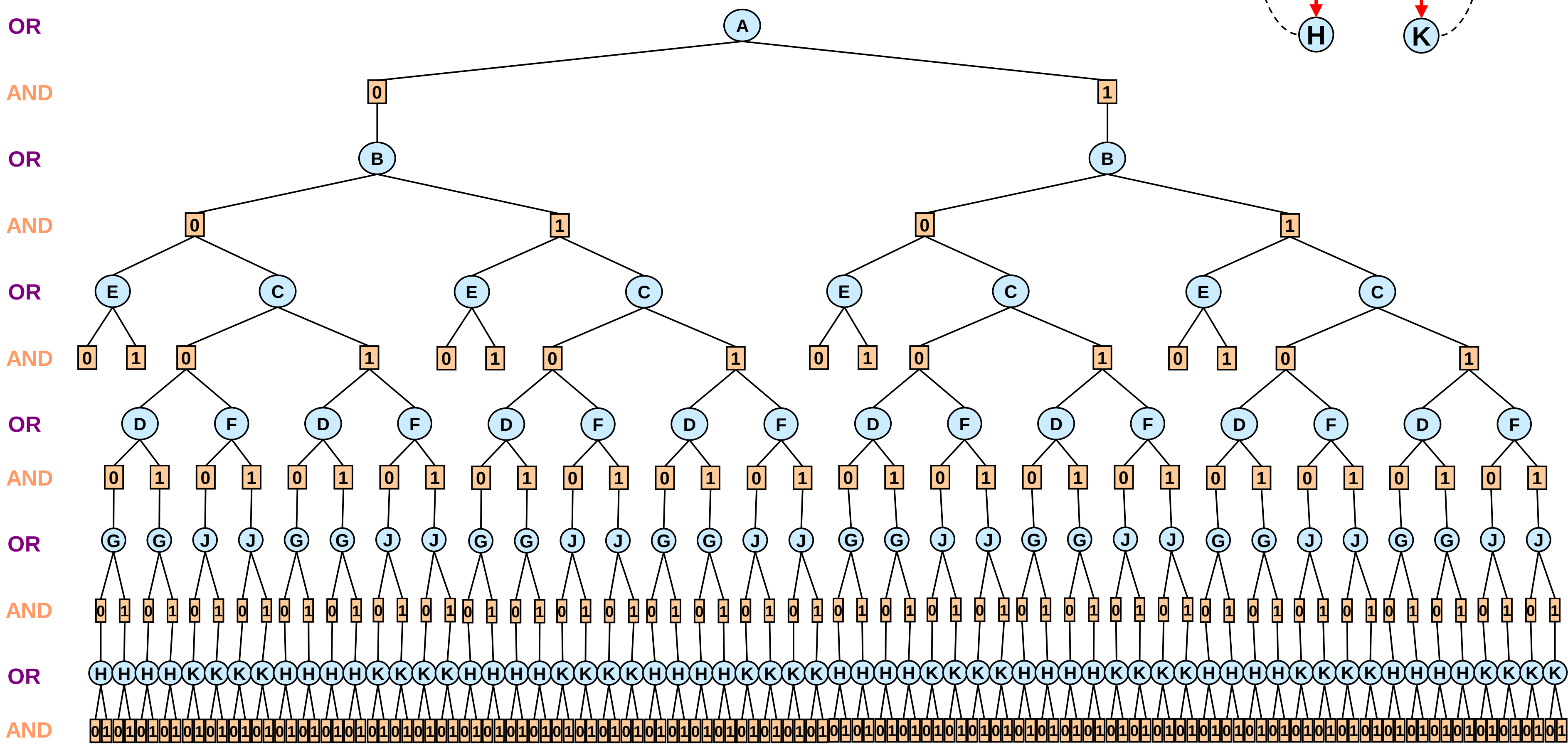
AND/OR Tree



A
B
C
D
E
F
G
H
J
K

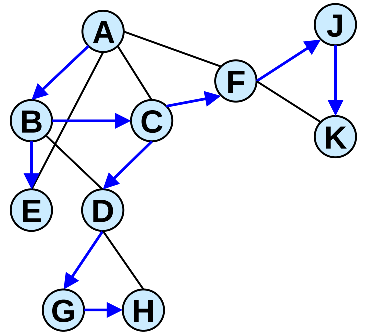


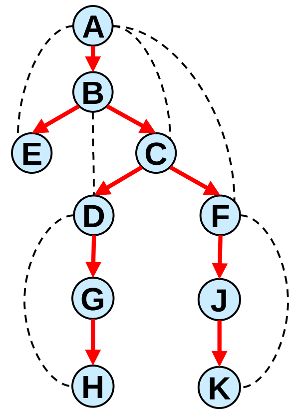
A
B
C
D
E
F
G
H
J
K

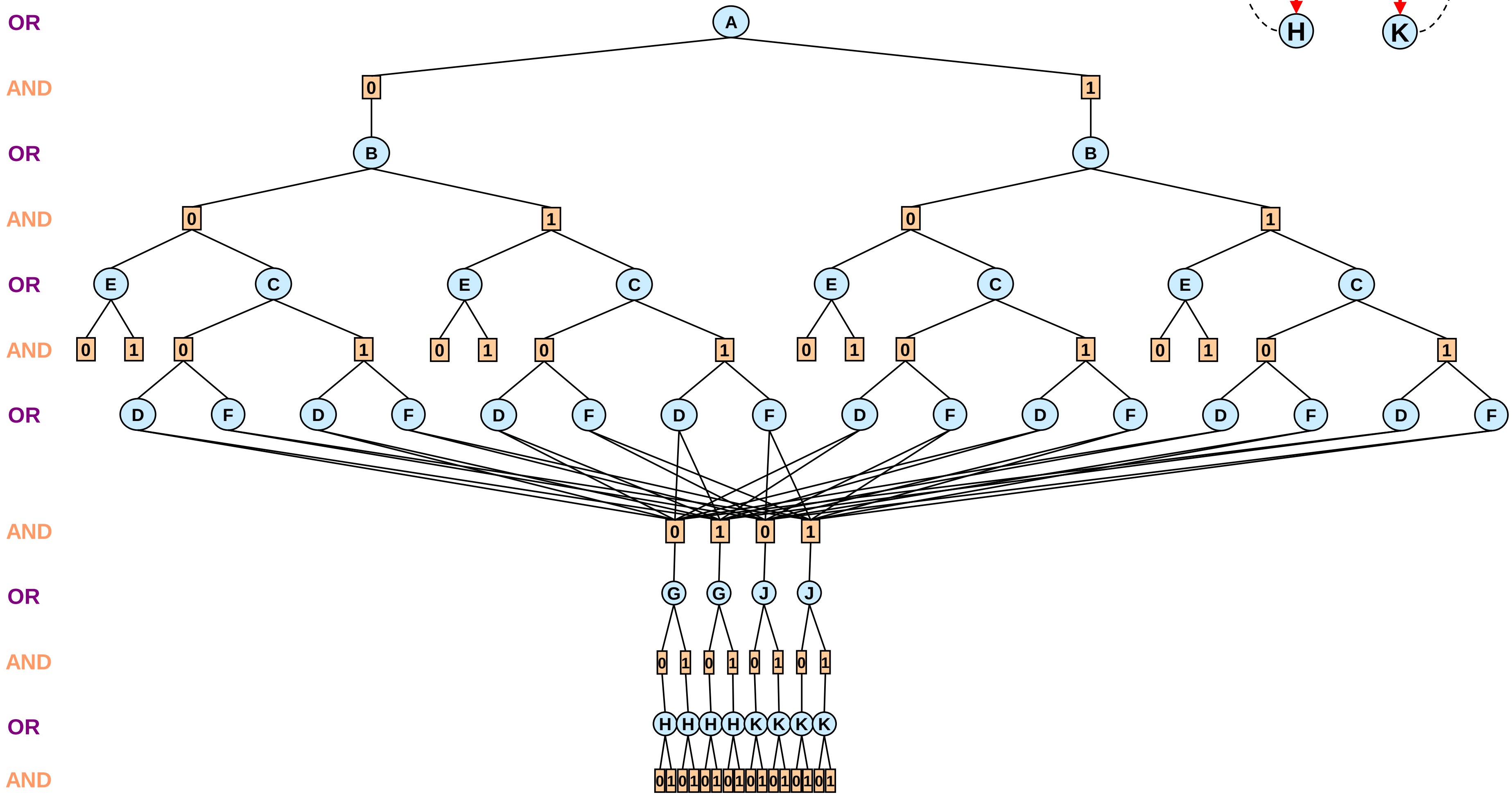


OR
AND
OR
AND
OR
AND
OR
AND
OR
AND
OR
AND

AND/OR Graph

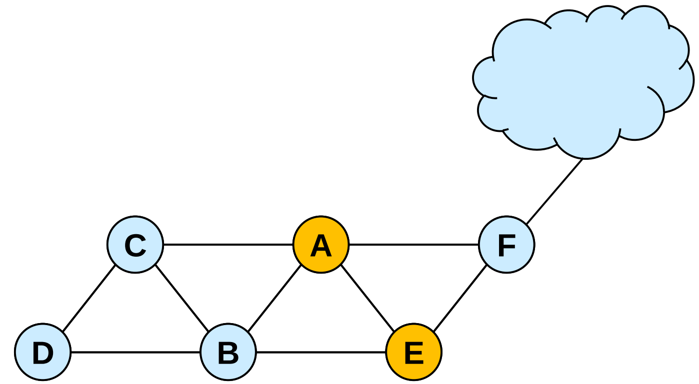

A
B
C
D
E
F
G
H
J
K

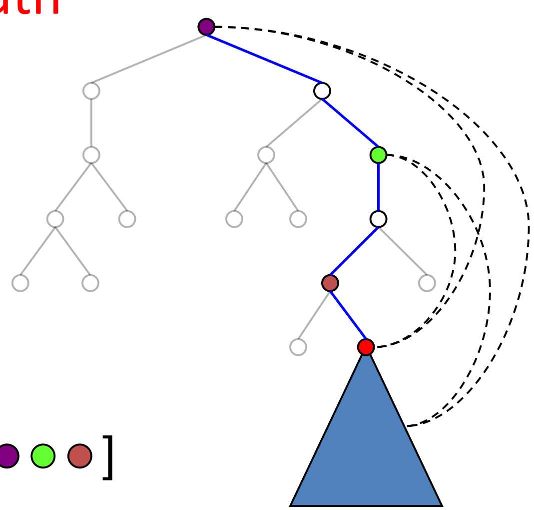

A
B
C
D
E
F
G
H
J
K


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

Merging Based on Context

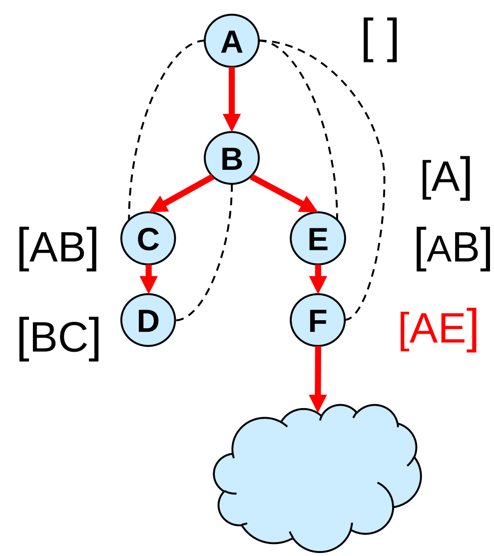
- context (X) = ancestors of X in pseudo tree, connected to X, or to descendants of X
- context (X) = parents in the induced graph
- max |context| = induced width = treewidth





pseudo tree

context(●) = [● ● ●]



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

$P(E \mid A, B)$

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

$P(B \mid A)$

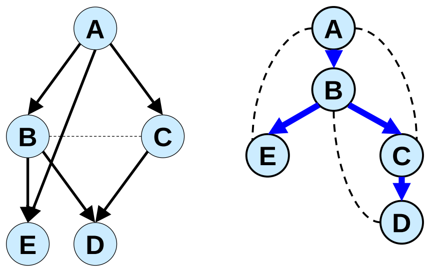
A	B=0	B=1
0	.4	.6
1	.1	.9

$P(C \mid A)$

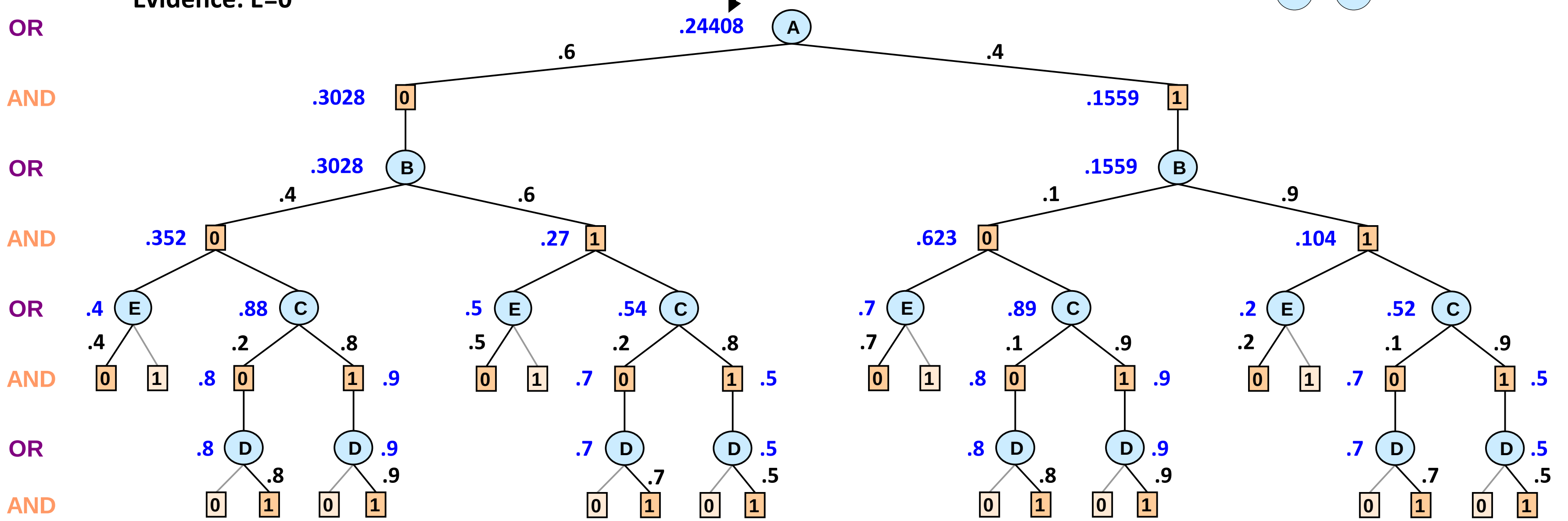
A	C=0	C=1
0	.2	.8
1	.7	.3

$P(A)$

A	P(A)
0	.6
1	.4



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



$P(D \mid B, C)$

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

Evidenc: D=1

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

$P(E \mid A,B)$

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

$P(B \mid A)$

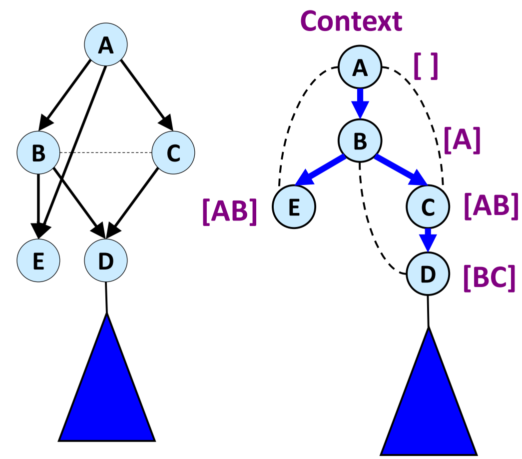
A	B=0	B=1
0	.4	.6
1	.1	.9

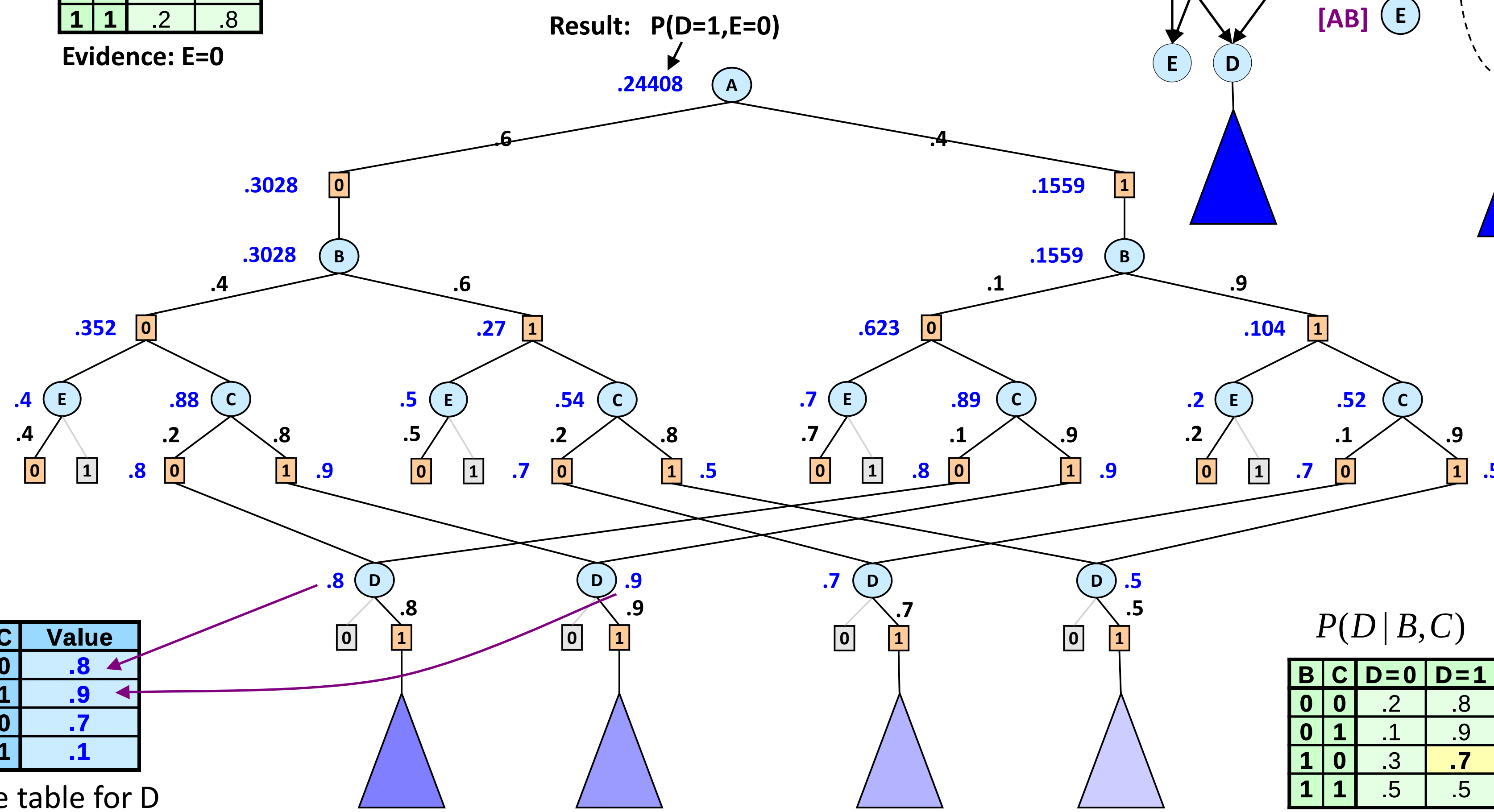
$P(C \mid A)$

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

$P(A)$

A	P(A)
0	.6
1	.4





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

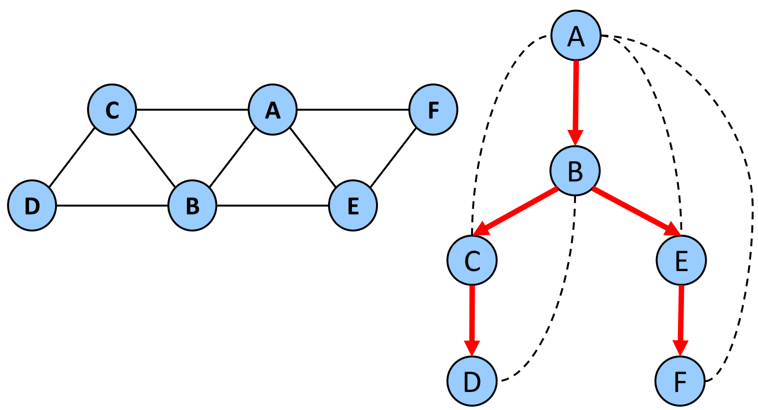
Cache table for D

$P(D \mid B,C)$

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
0	0	2
0	1	0
1	0	1
1	1	4

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

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

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

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

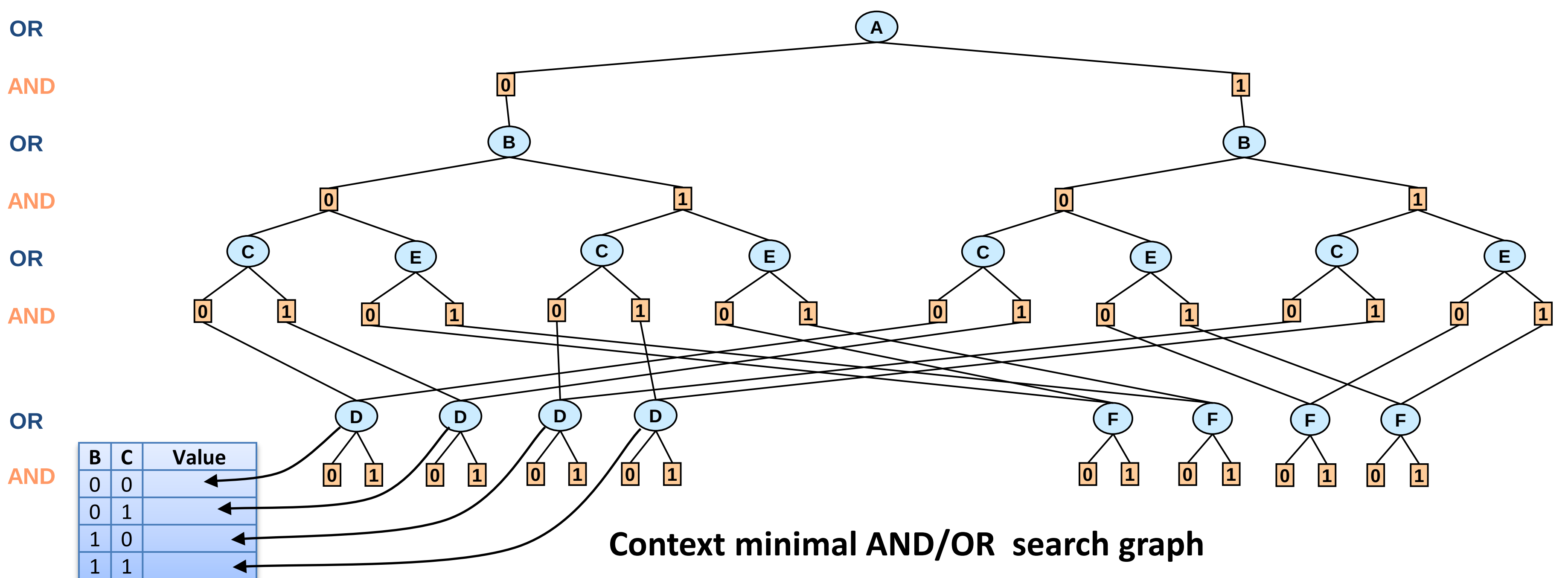
B	D	f_6
0	0	4
0	1	2
1	0	1
1	1	0

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

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

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

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



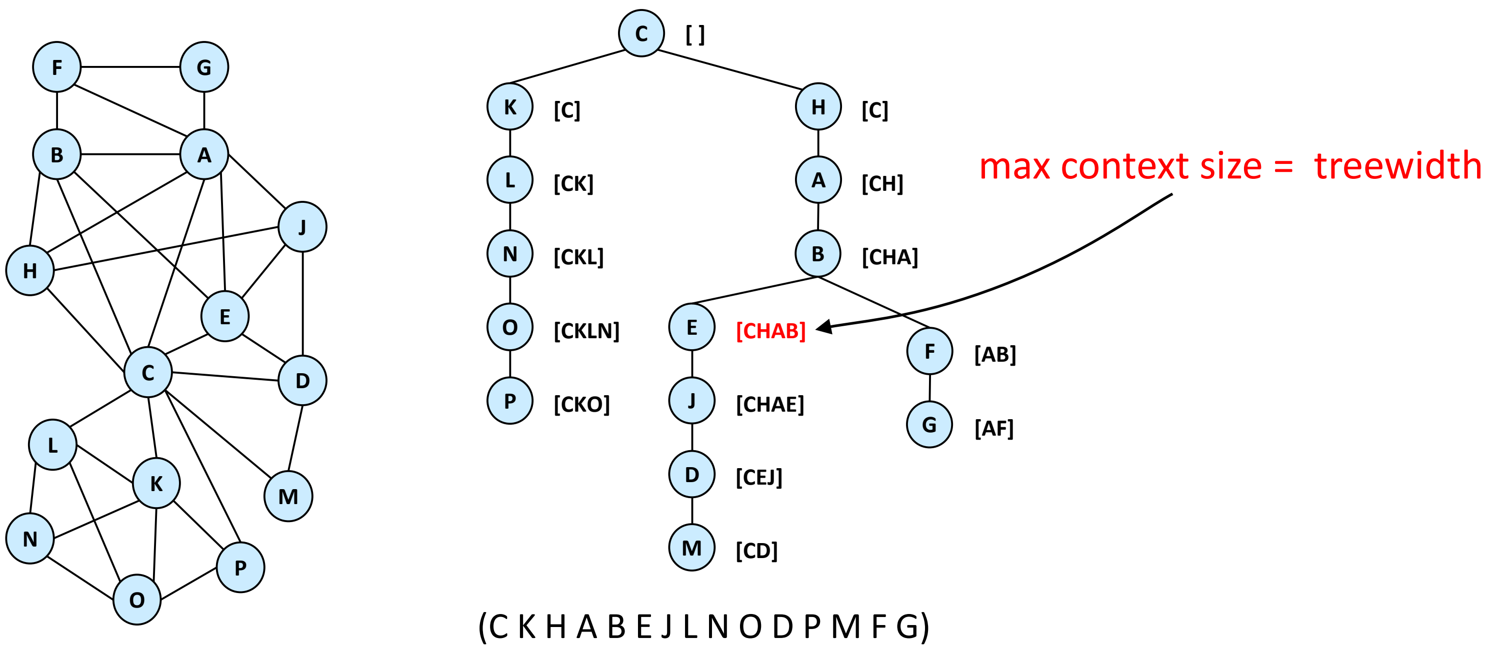
B	C	Value
0	0	
0	1	
1	0	
1	1	

Cache table for **D**

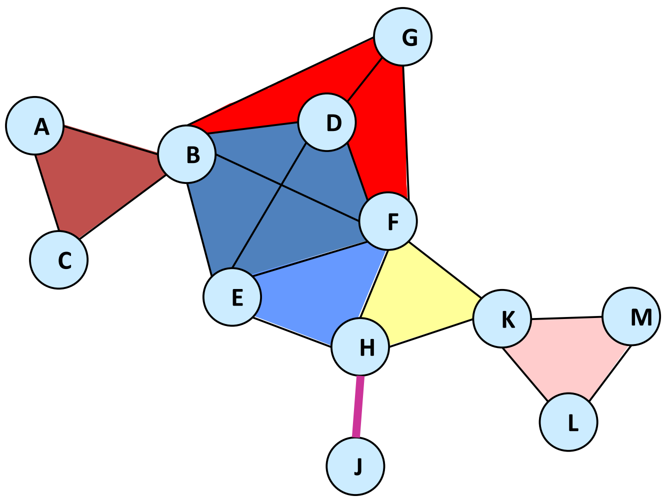
Context minimal AND/OR search graph

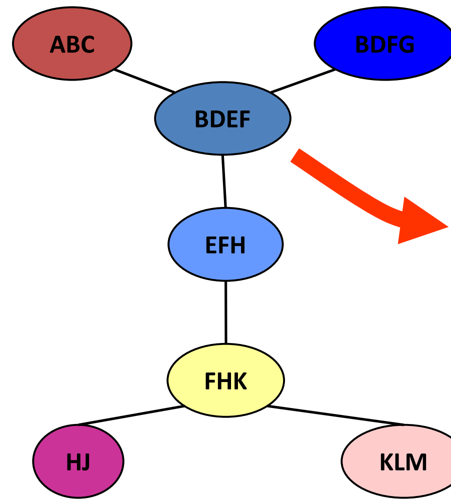
How Big Is The Context?

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

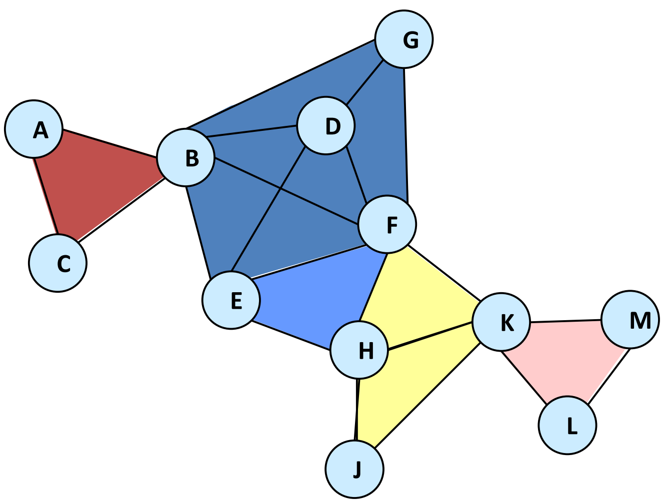


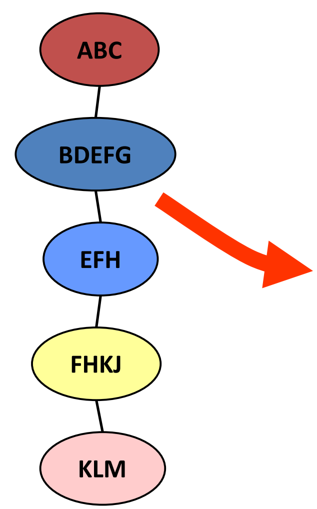
Treewidth vs. Pathwidth





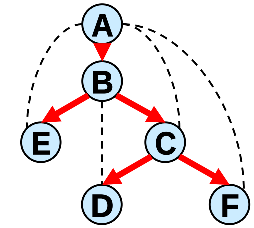
***treewidth* = 3**
= (max cluster size) - 1

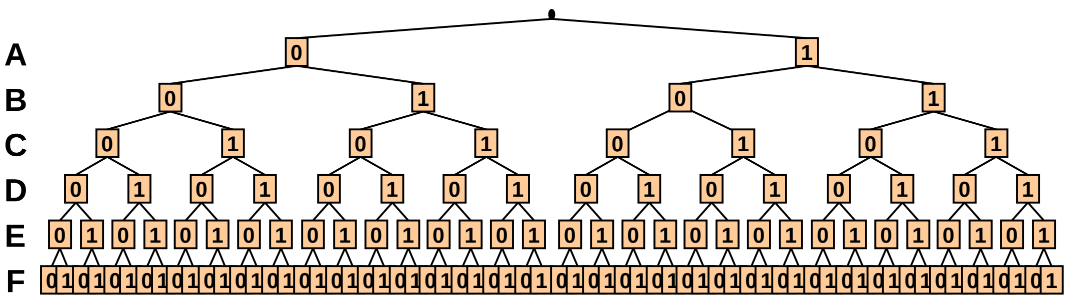




***pathwidth* = 4**
= (max cluster size) - 1

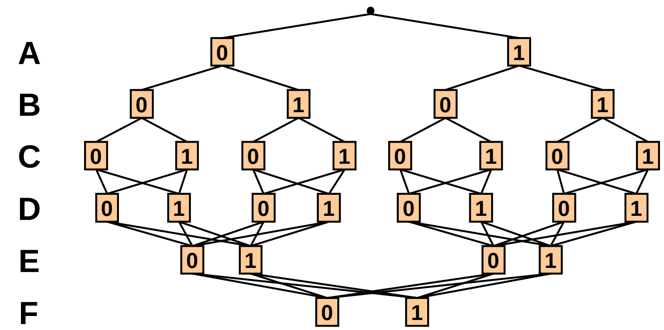
All Four Search Spaces





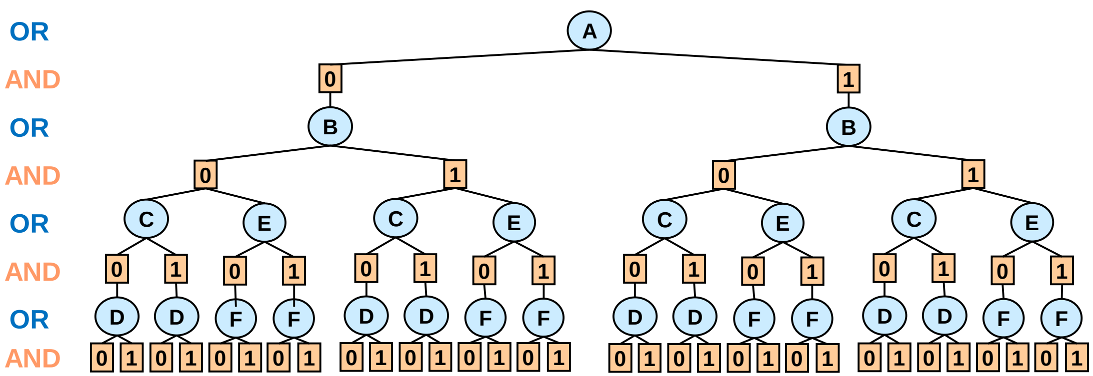
Full OR search tree

126 nodes



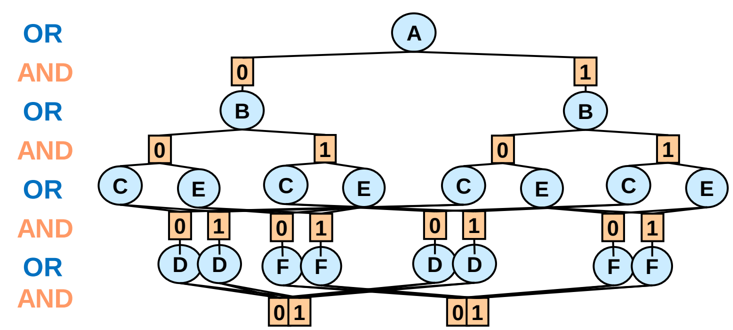
Context minimal OR search graph

28 nodes



Full AND/OR search tree

54 AND nodes



Context minimal AND/OR search graph

18 AND nodes

k = domain size

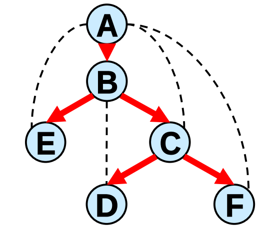
n = number of variables

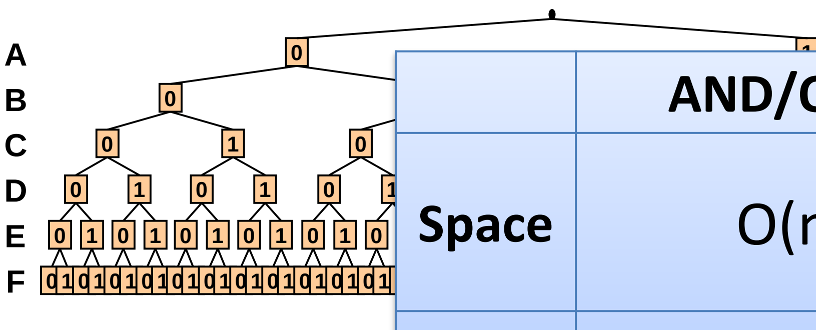
w*= treewidth

pw*= pathwidth

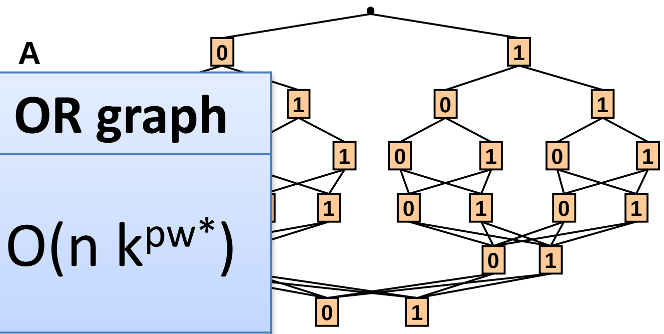
Any query is best computed over the context-minimal AO space

All Four Search Spaces





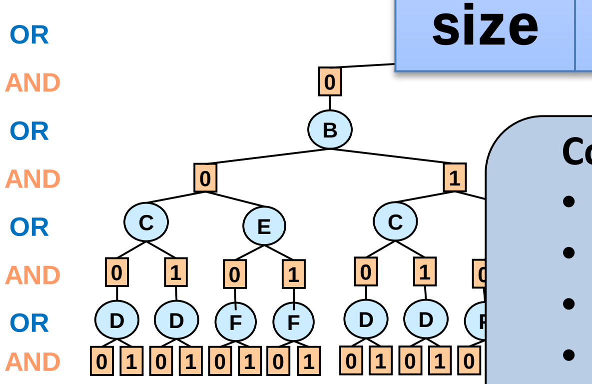
Fu



inimal OR search graph

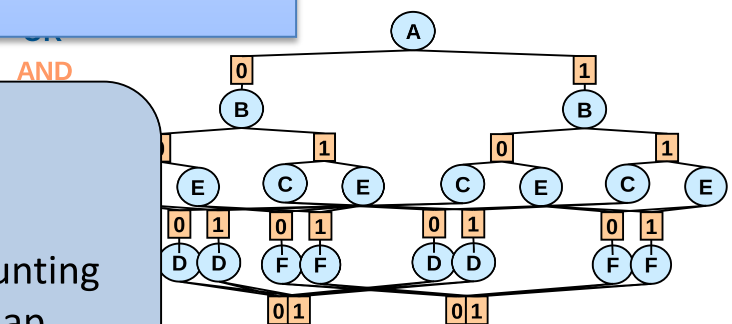
28 nodes

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



Full AND/O

54 AND nodes



Context minimal AND/OR search graph

18 AND nodes

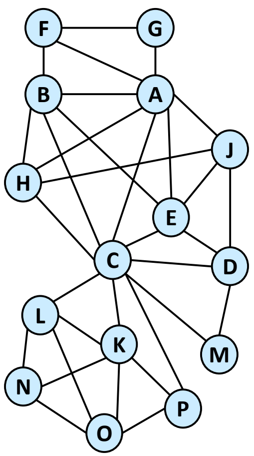
Computes any query:

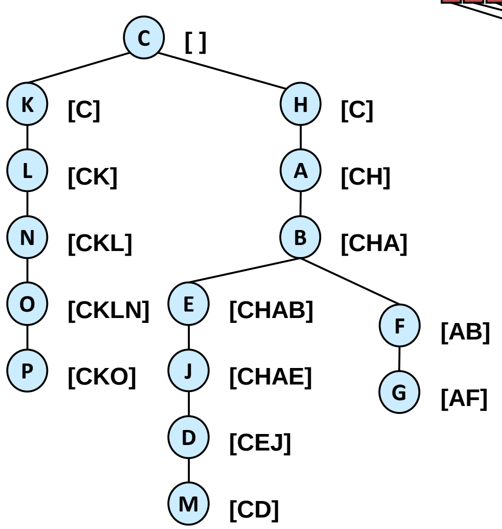
- Constraint satisfaction
- Max-Inference: Optimization
- Sum-Inference: Weighted counting
- Mixed-Inference: Marginal Map,
- Maximum expected utility

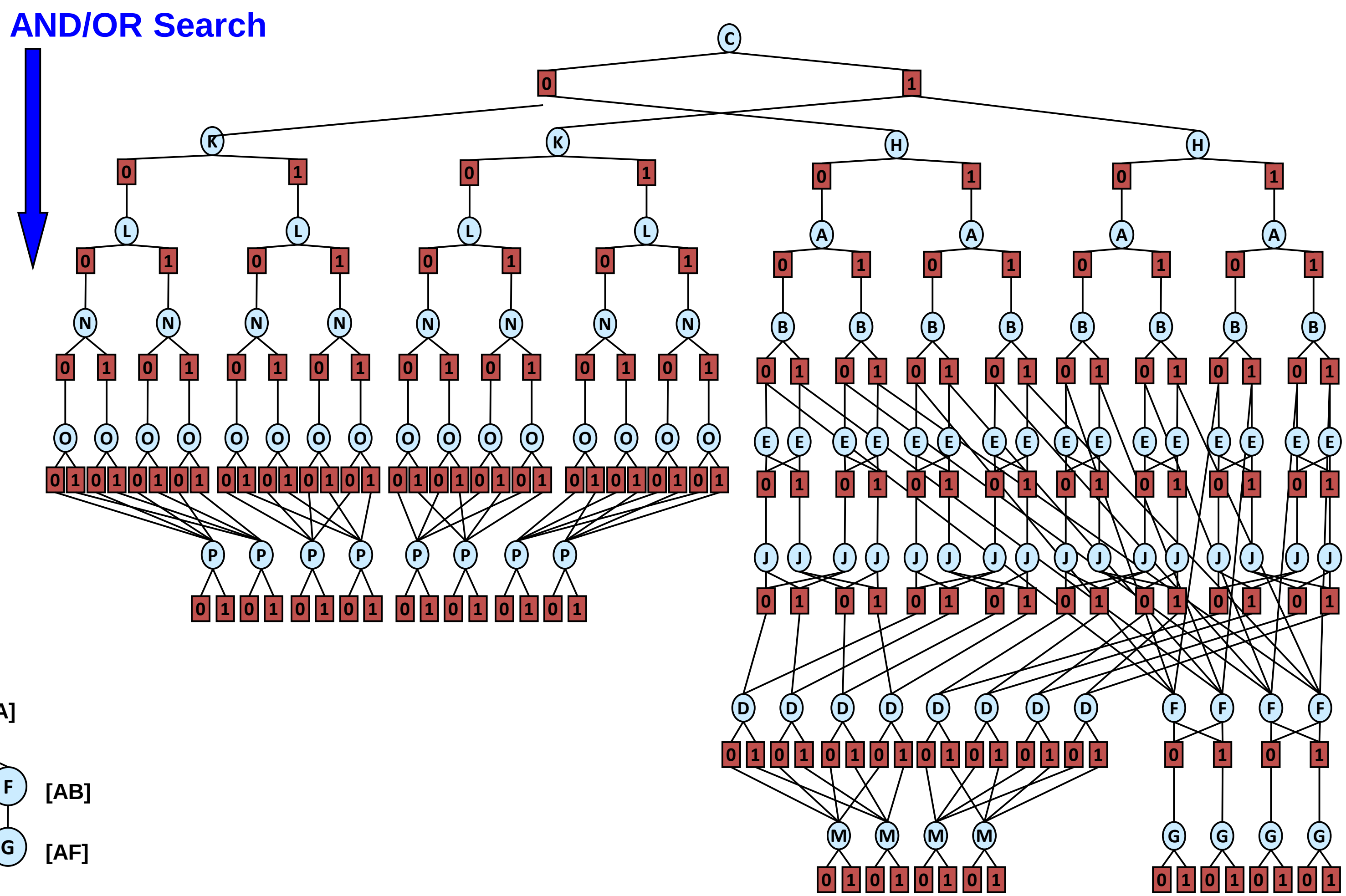
k = domain size
n = number of variables
w*= treewidth
pw*= pathwidth

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

AND/OR Search and Variable Elimination

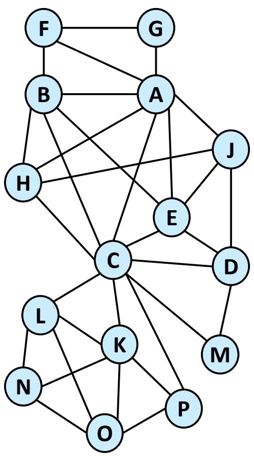


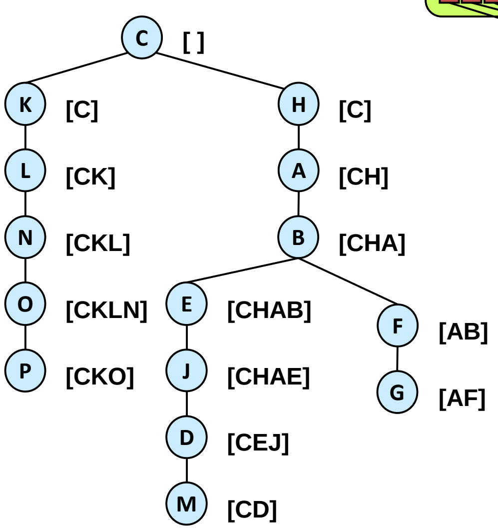


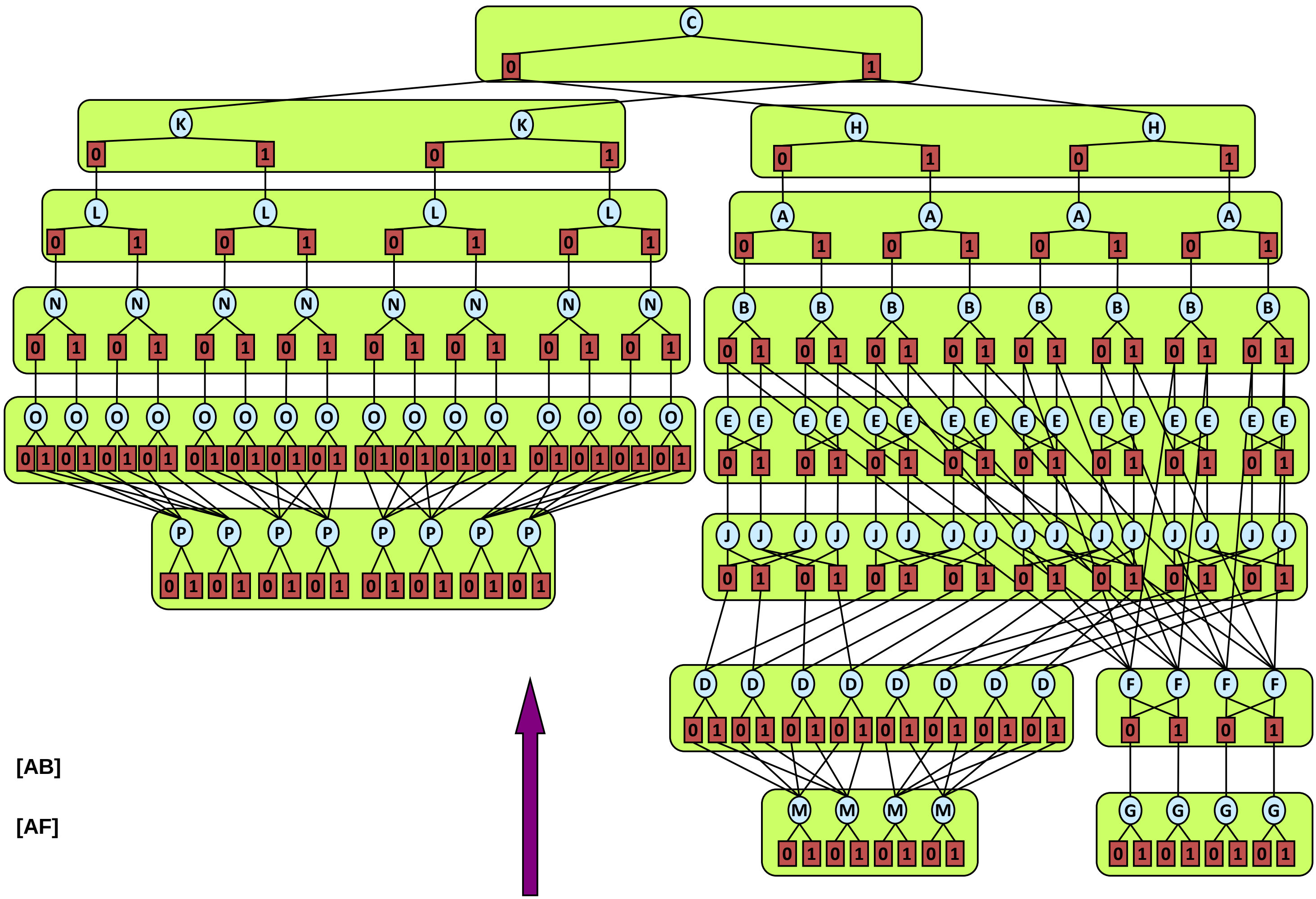


(C K H A B E J L N O D P M F G)

AND/OR Search and Variable Elimination



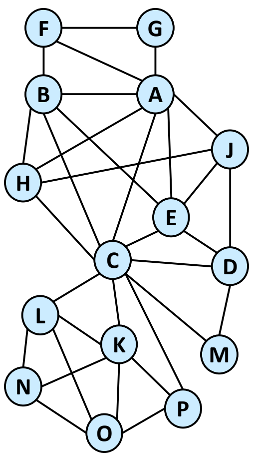


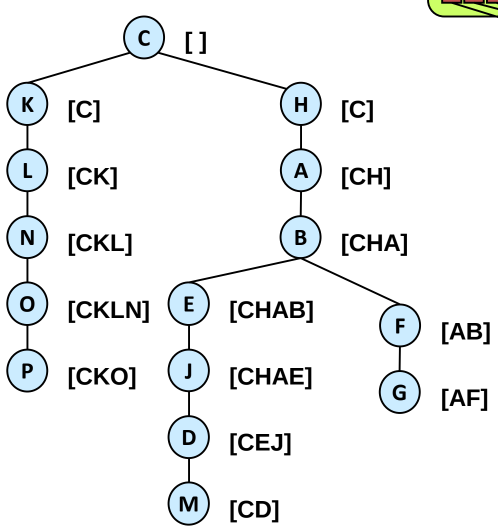


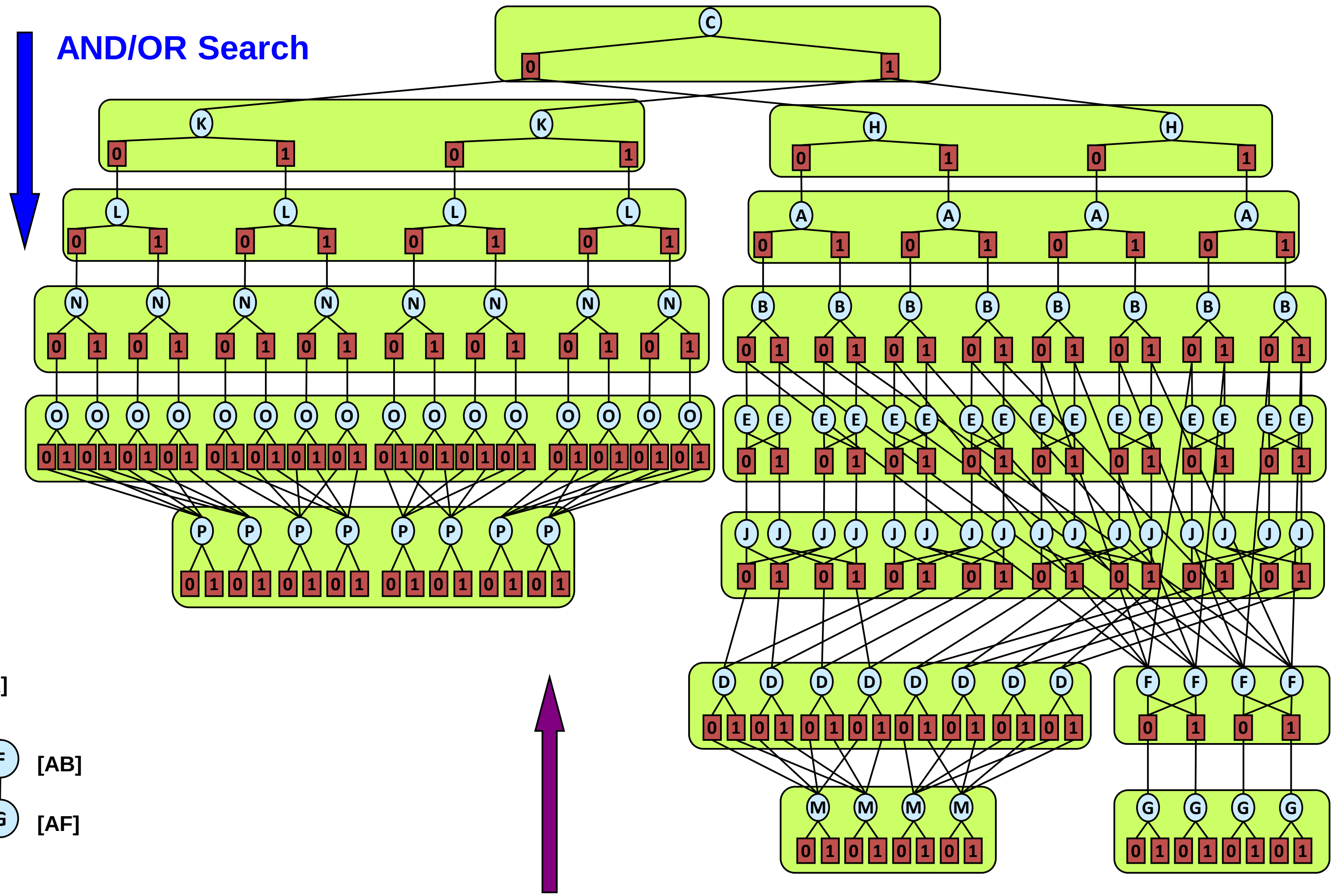
Variable Elimination

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

AND/OR Search and Variable Elimination





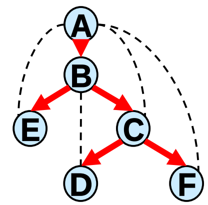


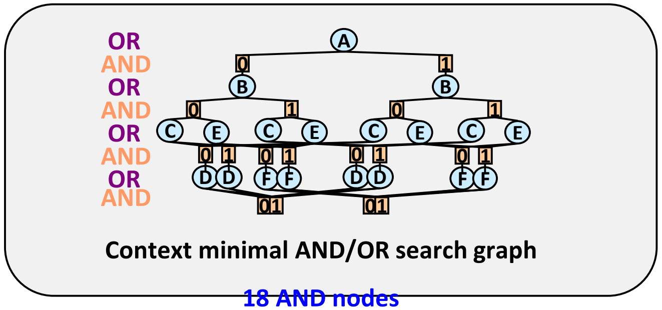
(C K H A B E J L N O D P M F G)

Road Map: Search

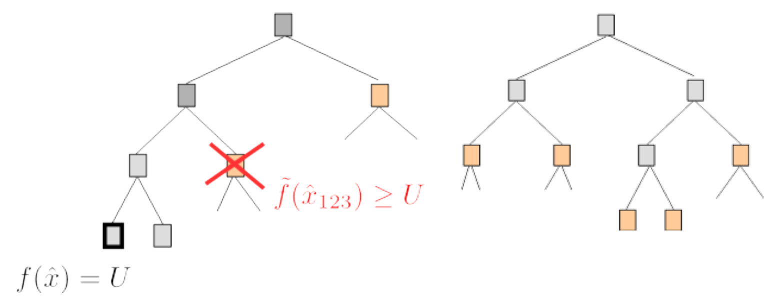
- Review Graphical Modes

- AND/OR search spaces, pseudo-trees
  - AND/OR search trees
  - AND/OR search graphs
  - **Generating good pseudo-trees**
  - Brute-force AND/OR

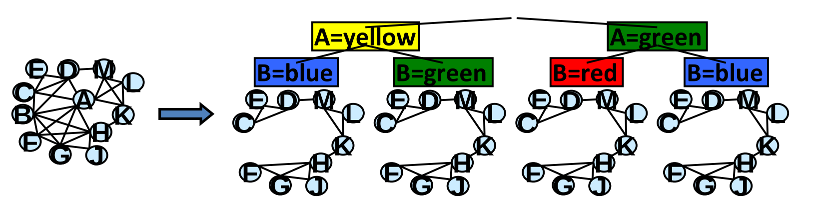




- Heuristic search for AND/OR spaces
  - Basic Heuristic search (Depth and Best)
  - Depth-first AND/OR branch and bound
  - Best-first AND/OR search
  - The Guiding MBE heuristic
  - Marginal Map (max-sum-product)



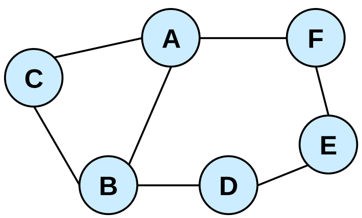
- Hybrids of search and Inference
- Summary and Class 2



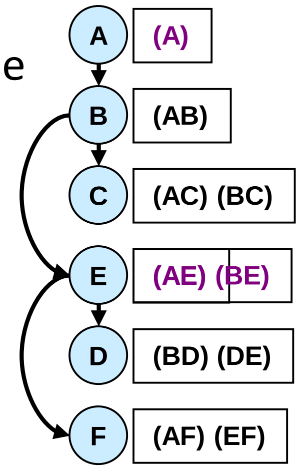
Variable Orderings and Pseudo-Trees

Note: we order from top to bottom here

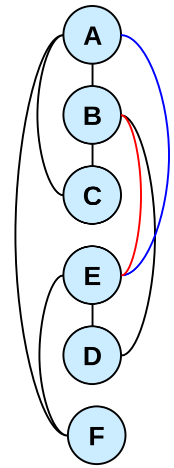
Bucket-tree = pseudo-tree



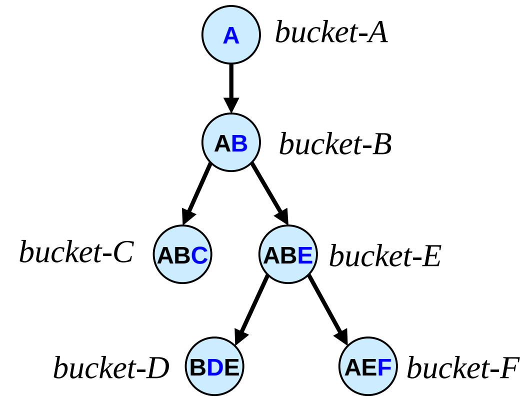
***d*: A B C E D F**



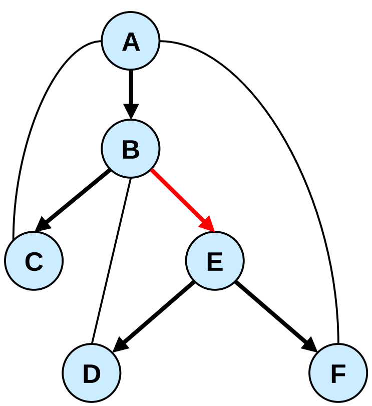
Bucket-tree based on *d*



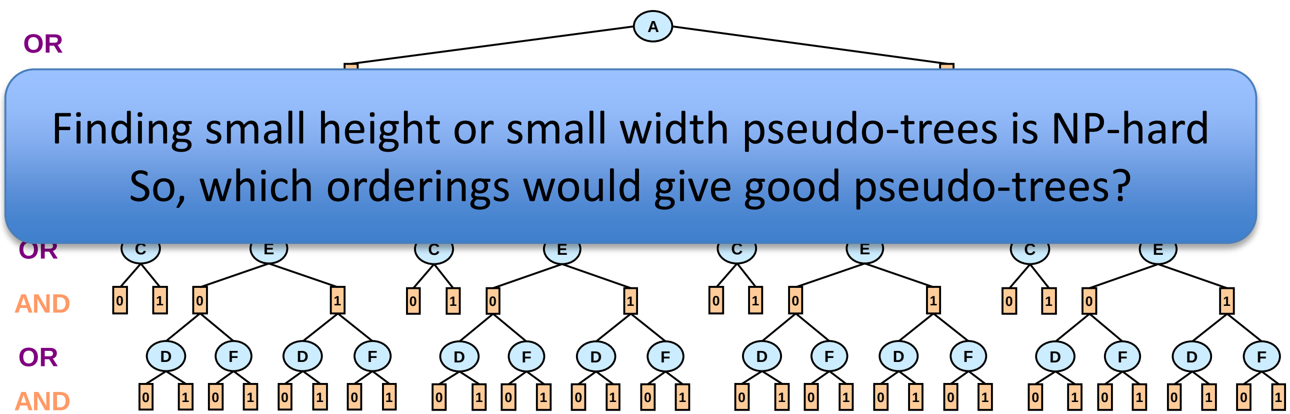
Induced graph



Bucket-tree



Bucket-tree used as pseudo-tree



Finding small height or small width pseudo-trees is NP-hard

So, which orderings would give good pseudo-trees?

AND/OR search tree

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

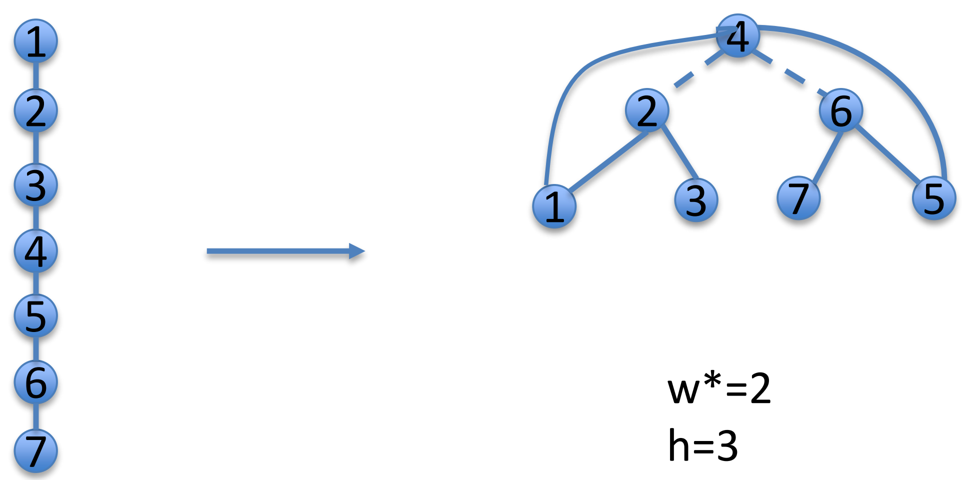
SPOT5 Benchmarks

For more see [Dechter 2003]

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 <= w* log n
- Optimality of h and w* cannot be achieved at once.

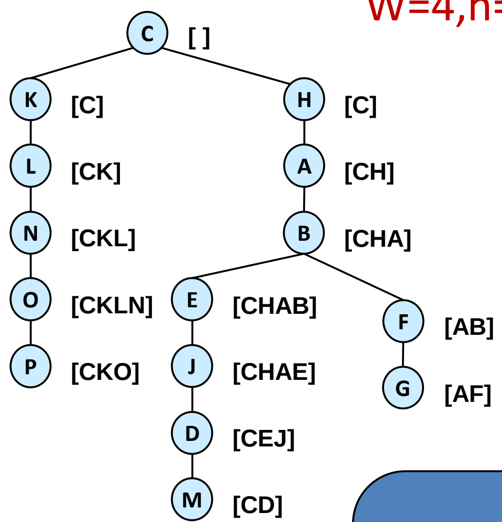
W*=1
h=7

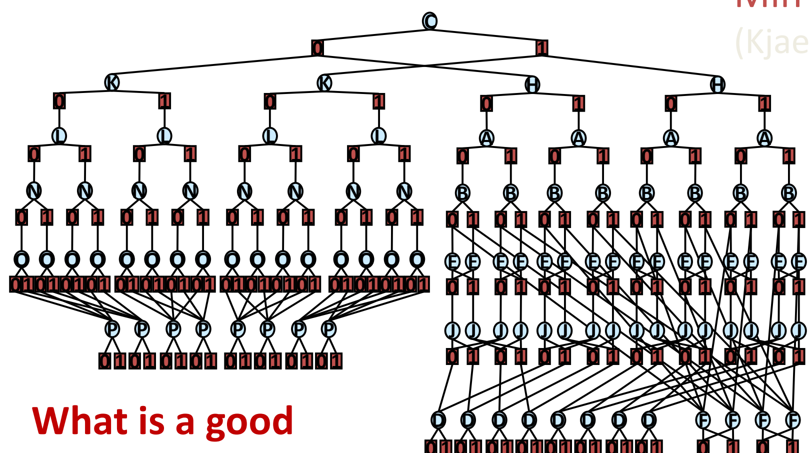


w*=2
h=3

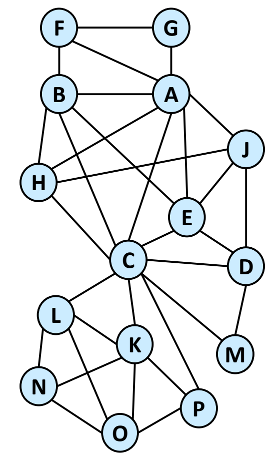
The Impact of the Pseudo-Tree

W=4,h=8





Min-Fill
(Kjaerulff90)



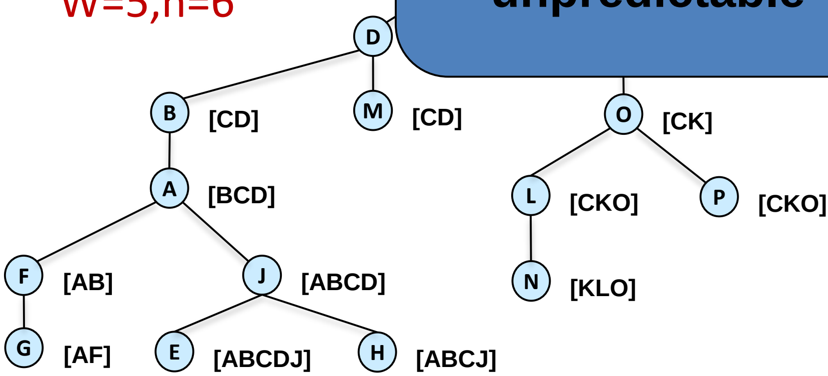
What is a good

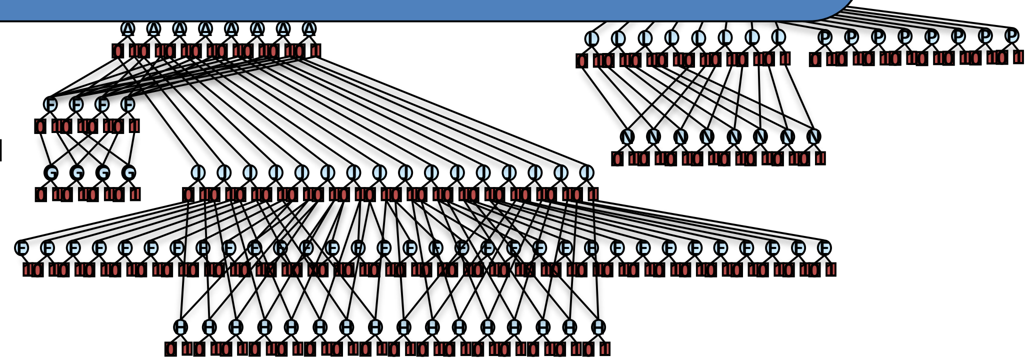
(C K H A B E J L N O

...graph
...oning
...is)

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

W=5,h=6

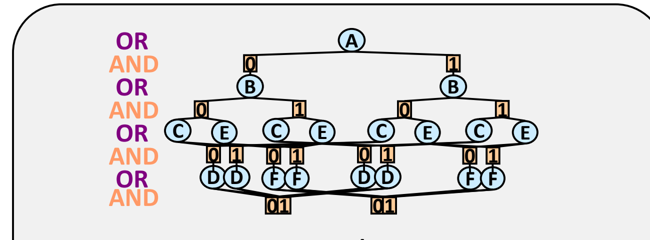




(C D K B A O M L N P J H E F G)

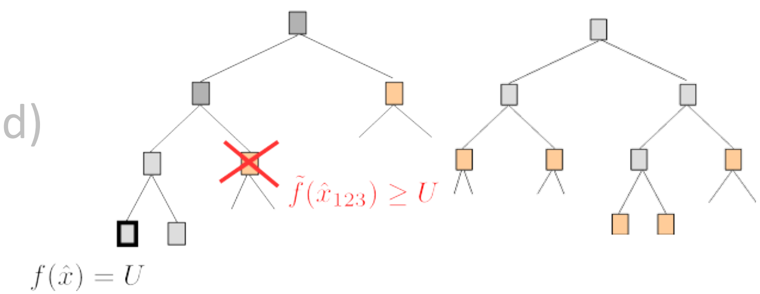
Road Map: Search

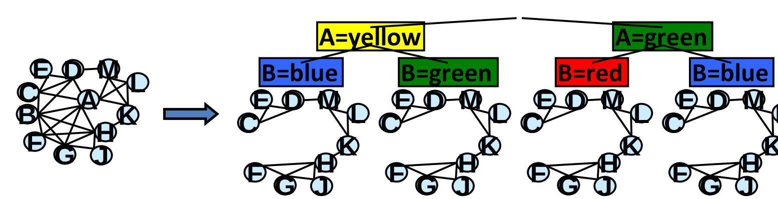
- Review Graphical Modes
- AND/OR search spaces, pseudo-trees
  - AND/OR search trees
  - AND/OR search graphs
  - Generating good pseudo-trees
  - **Brute-force AND/OR**
- Heuristic search (HS) for AND/OR spaces
  - Basic Heuristic search (Depth and Best)
  - AND/OR Depth-first HS (branch and bound)
  - AND/OR Best-first heuristic search
  - The Guiding MBE heuristic
  - Marginal Map (max-sum-product)
- Hybrids of search and Inference
- Summary and Class 2



Context minimal AND/OR search graph

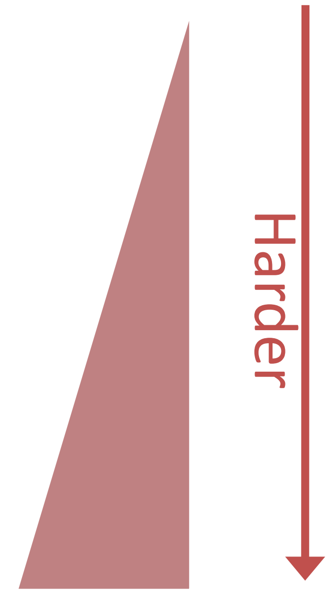
18 AND nodes





Types of queries

Max-Inference	$f(\mathbf{x}^*) = \max_{\mathbf{x}} \prod_{\alpha} f_\alpha(\mathbf{x}_\alpha)$
Sum-Inference	$Z = \sum_{\mathbf{x}} \prod_{\alpha} f_\alpha(\mathbf{x}_\alpha)$
Mixed-Inference	$f(\mathbf{x}_M^*) = \max_{\mathbf{x}_M} \sum_{\mathbf{x}_S} \prod_{\alpha} f_\alpha(\mathbf{x}_\alpha)$



- **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)

$P(E \mid A, B)$

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

$P(B \mid A)$

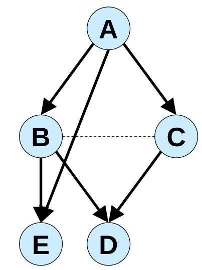
A	B=0	B=1
0	.4	.6
1	.1	.9

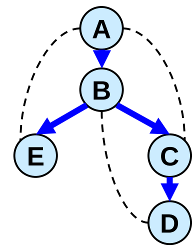
$P(C \mid A)$

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

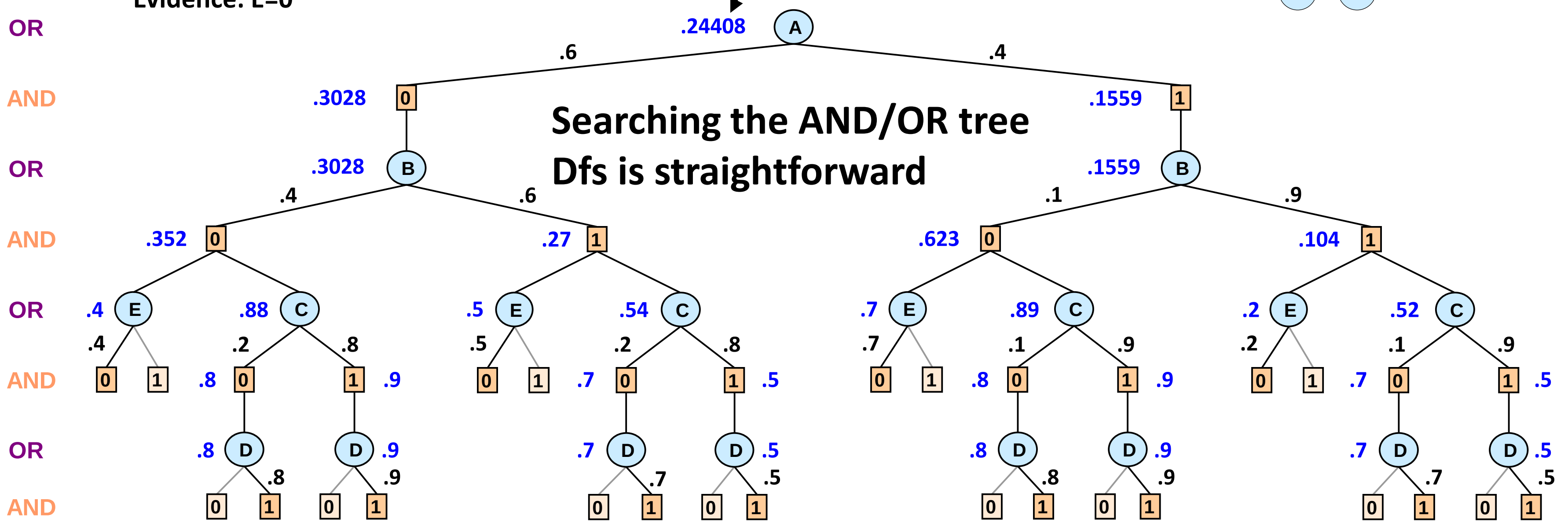
$P(A)$

A	P(A)
0	.6
1	.4





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



$P(D \mid B, C)$

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)

$P(E \mid A, B)$

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

$P(B \mid A)$

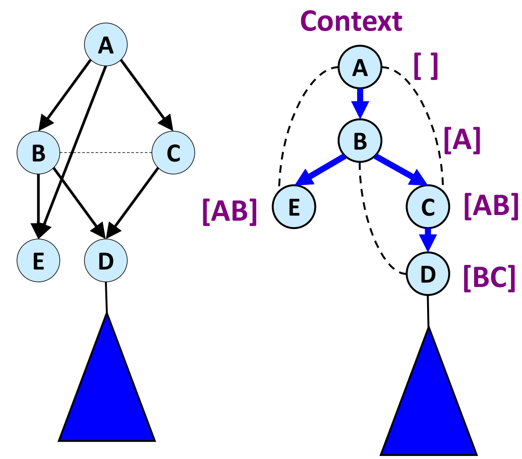
A	B=0	B=1
0	.4	.6
1	.1	.9

$P(C \mid A)$

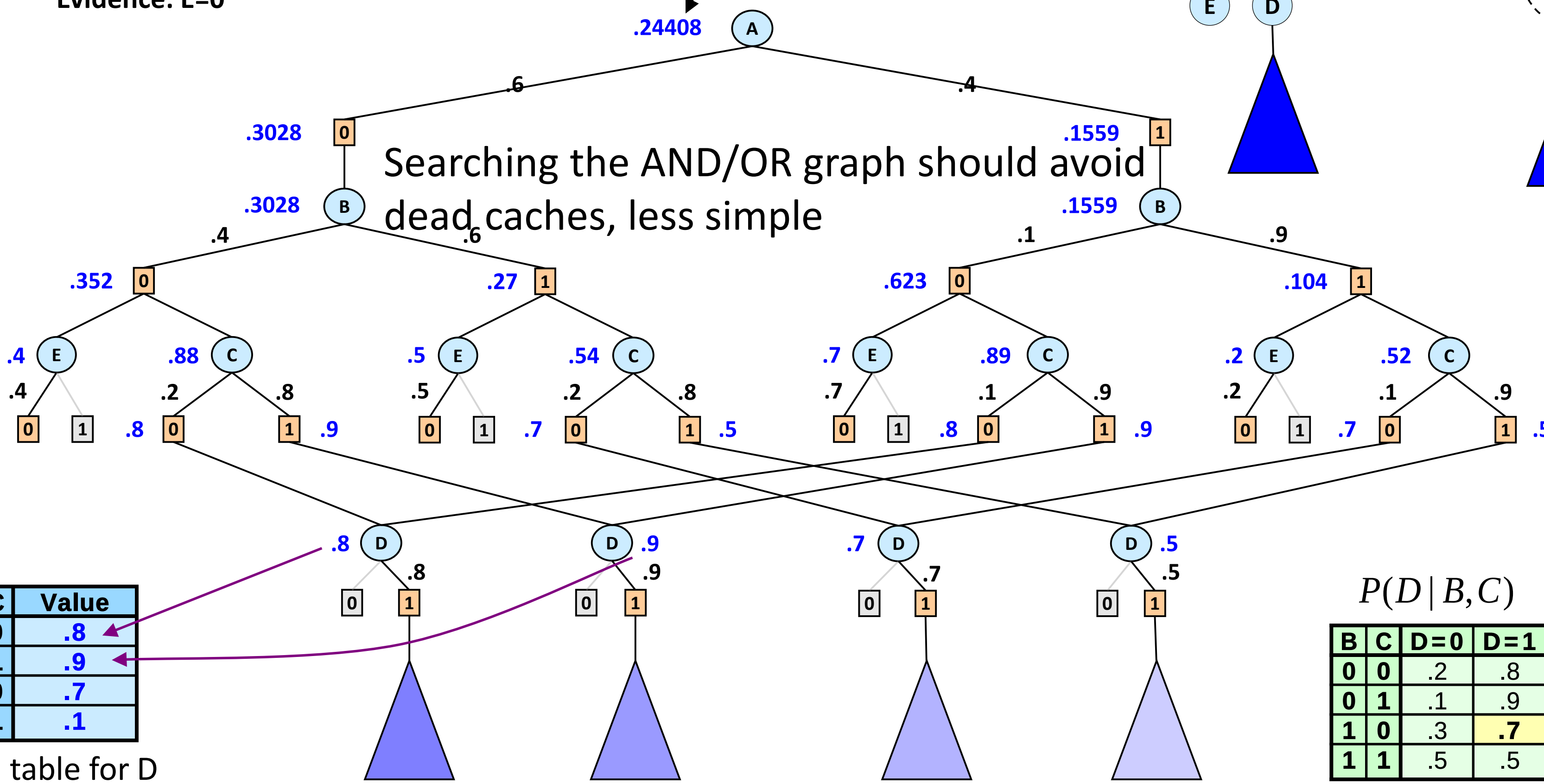
A	C=0	C=1
0	.2	.8
1	.7	.3

$P(A)$

A	P(A)
0	.6
1	.4



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



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

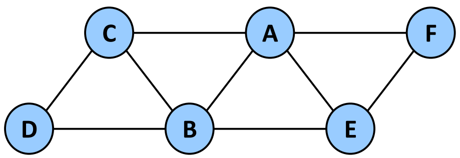
Cache table for D

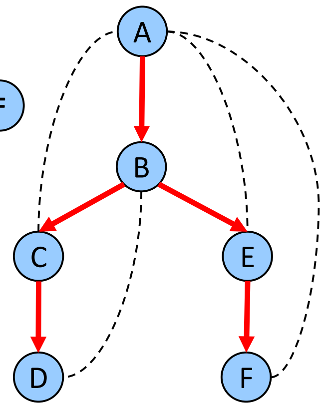
$P(D \mid B, C)$

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
0	0	2
0	1	0
1	0	1
1	1	4

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

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

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

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

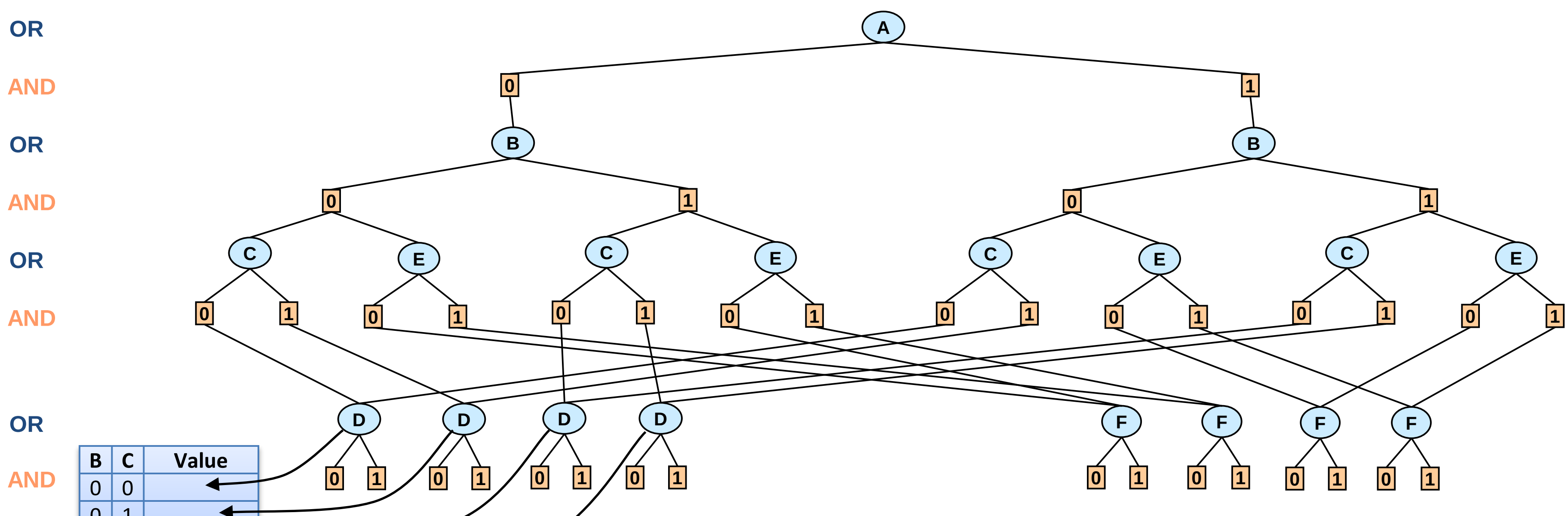
B	D	f_6
0	0	4
0	1	2
1	0	1
1	1	0

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

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

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

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



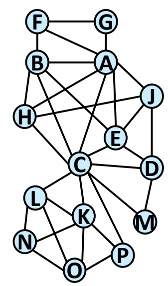
B	C	Value
0	0	
0	1	
1	0	
1	1	

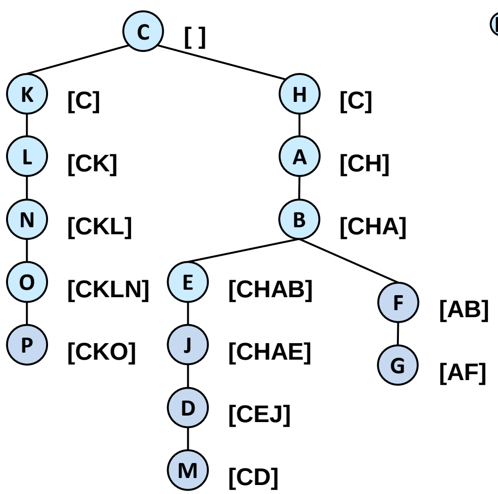
Context minimal AND/OR search graph

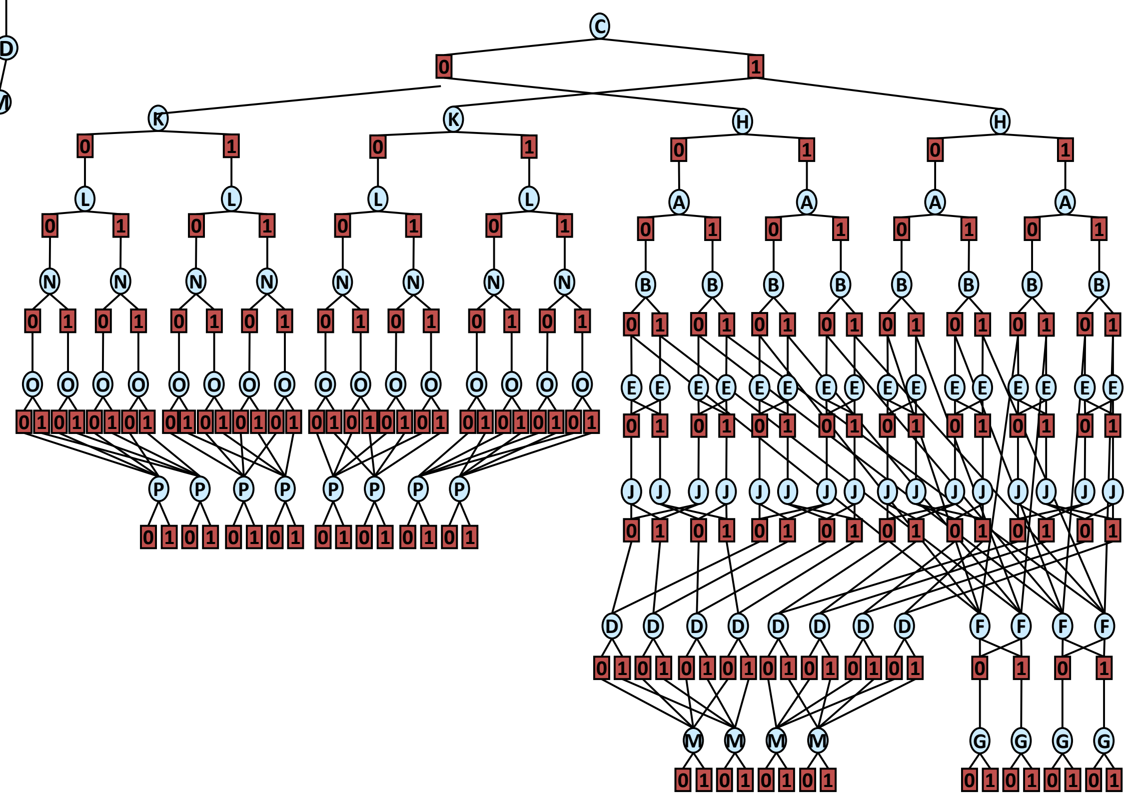
Cache table for **D**

Dead Caches

Definition 8.1.9 (dead cache) *If X is the parent of Y in pseudo-tree $\mathcal{T}$, and $context(X) \subset context(Y)$, then $context(Y)$ represents a* dead cache.





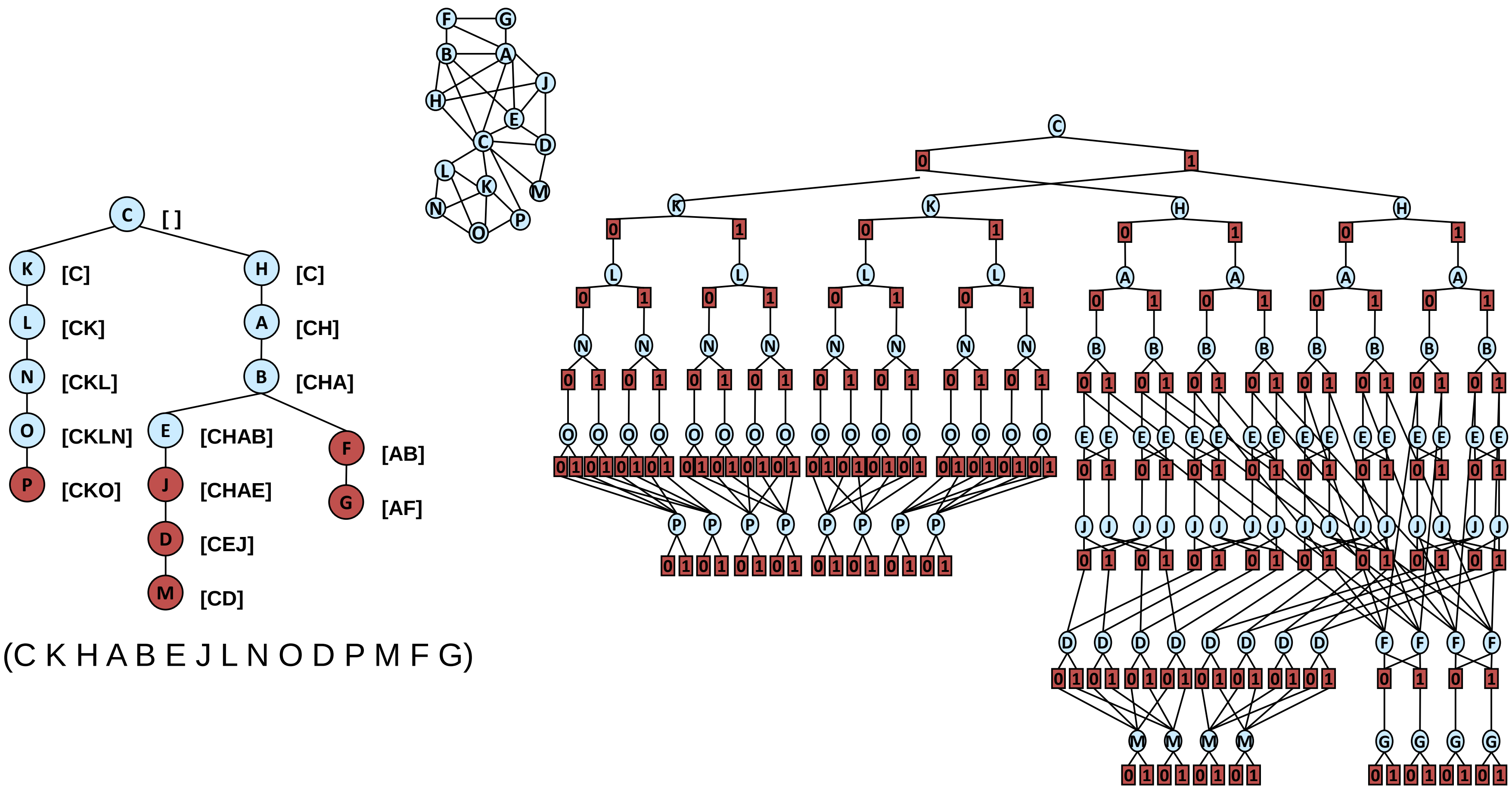


(C K H A B E J L N O D P M F G)

(Darwiche 2000)

Dead Caches

Definition 8.1.9 (dead cache) *If X is the parent of Y in pseudo-tree $\mathcal{T}$, and $context(X) \subset context(Y)$, then $context(Y)$ represents a* dead cache.



(Darwiche 2000)

Searching AND/OR Graphs

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

i=0 | i | i=w*

Space: O(n)

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

Space: O(exp(i))

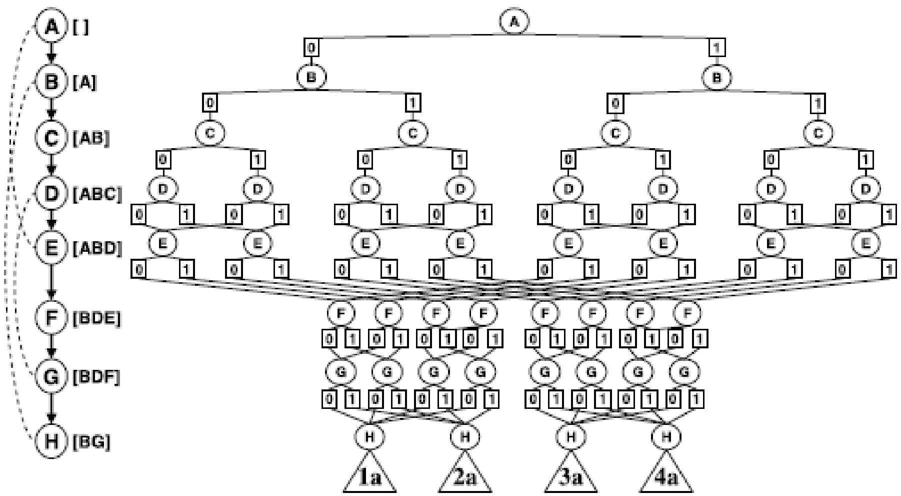
Time: O(exp(m_i+i)

Space: O(exp w*)

Time: O(exp w*)

m_i is related to the size of the i-cutset.

Different Levels of Caching



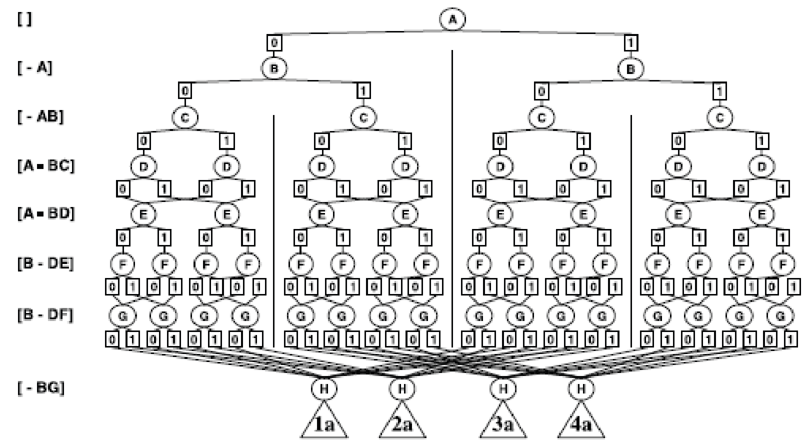


Figure 7.11:
AOC(2) graph (adaptive caching).

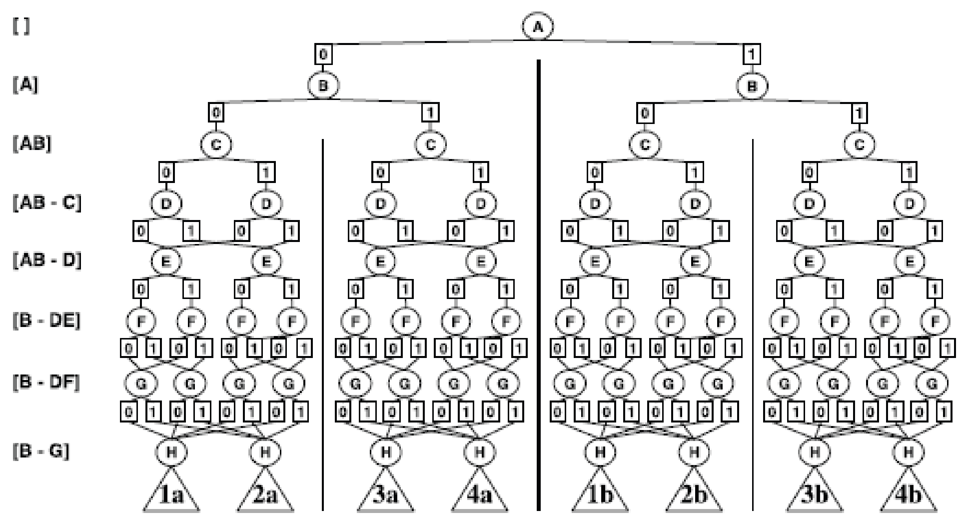
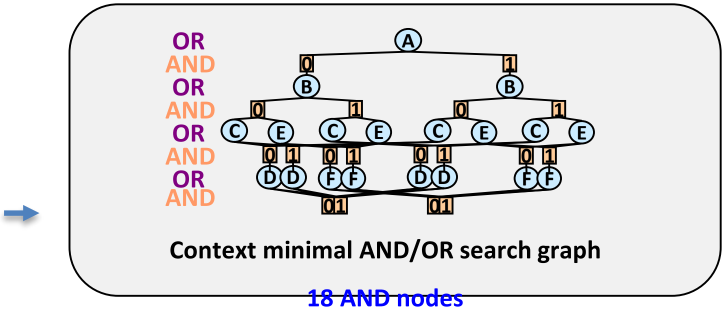


Figure 7.12: **AOCutset(2)** graph (AND/OR Cutset).

Road Map: Search

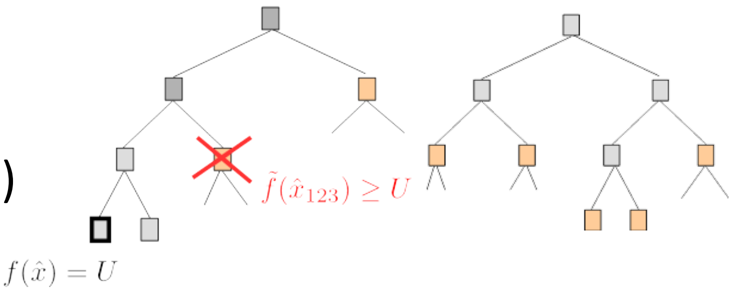
- Review Graphical Modes

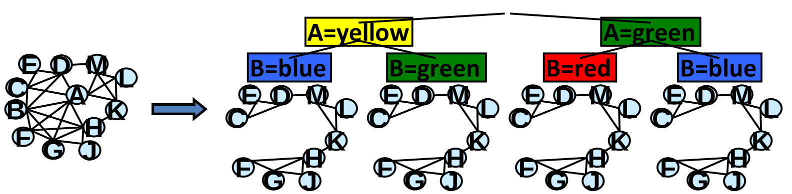
- AND/OR search spaces, pseudo-trees
  - AND/OR search trees
  - AND/OR search graphs
  - **Generating good pseudo-trees**
  - Brute-force search



Questions?

- Heuristic search (HS) for AND/OR spaces
  - **Basic Heuristic search (Depth and Best)**
  - AND/OR Depth-first HS (branch and bound)
  - AND/OR Best-first heuristic search
  - The Guiding MBE heuristic
  - Marginal Map (max-sum-product)





- Hybrids of search and Inference
- Summary and Class 2

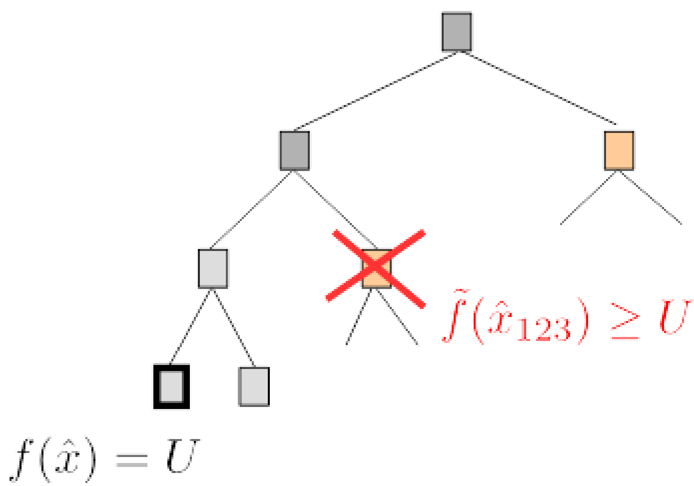
Basic Heuristic Search

We assume min-sum problems in the following

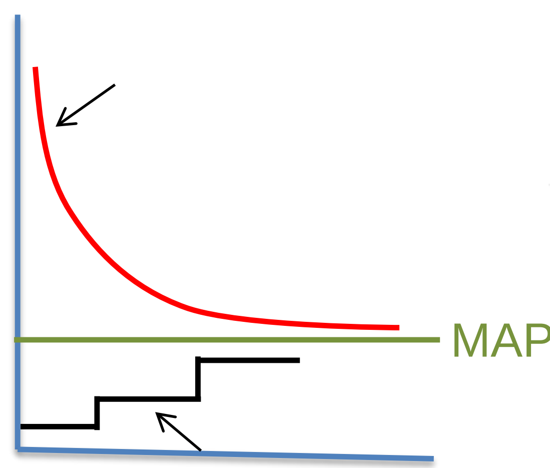
Heuristic function $\tilde{f}(\hat{x}_p)$ computes a lower bound on the best extension of partial configuration $\hat{x}_p$ and can be used to guide heuristic search.
We focus on:

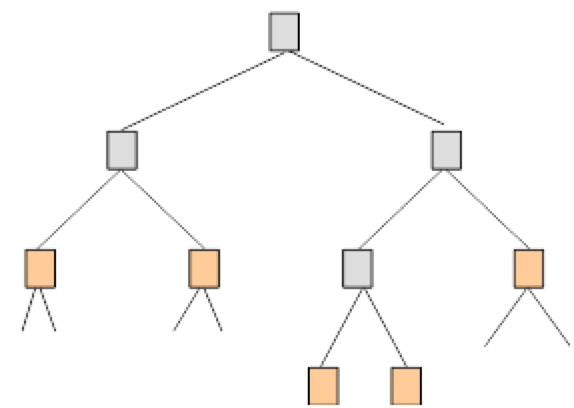
1. Branch-and-Bound
Use heuristic function $\tilde{f}(\hat{x}_p)$ to prune the depth-first search tree
Linear space

2. Best-First Search
Always expand the node with the lowest heuristic value $\tilde{f}(\hat{x}_p)$
Needs lots of memory



BnB is upper-bound anytime

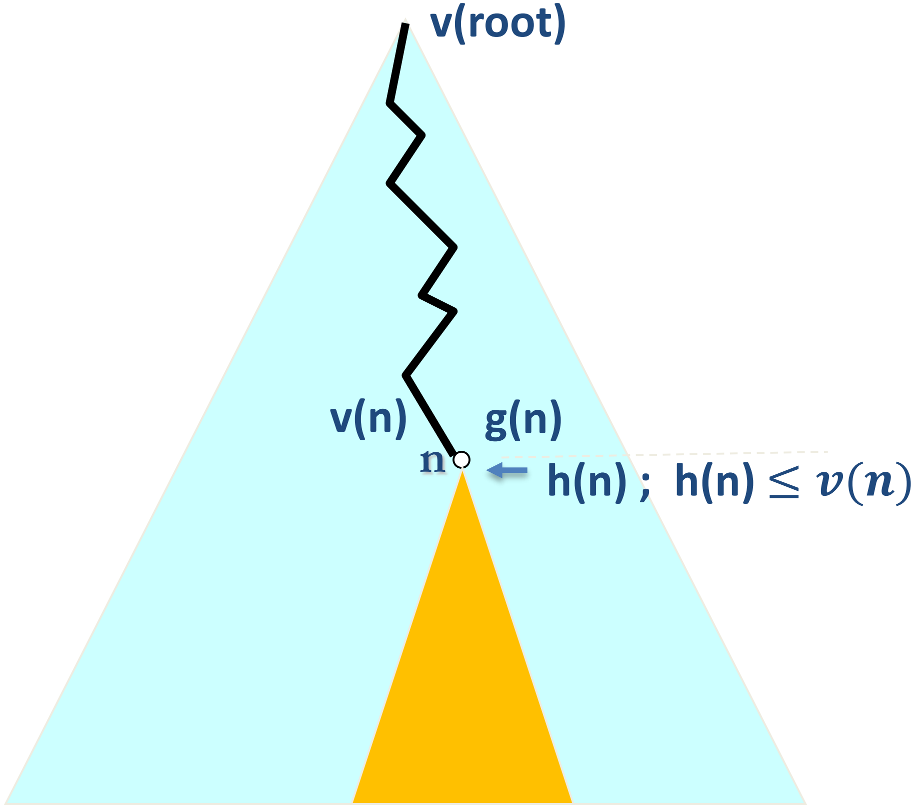




Basic Heuristic Search; **Best-First**

Task: compute v(root): MAP, Marginal ,MMAP

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

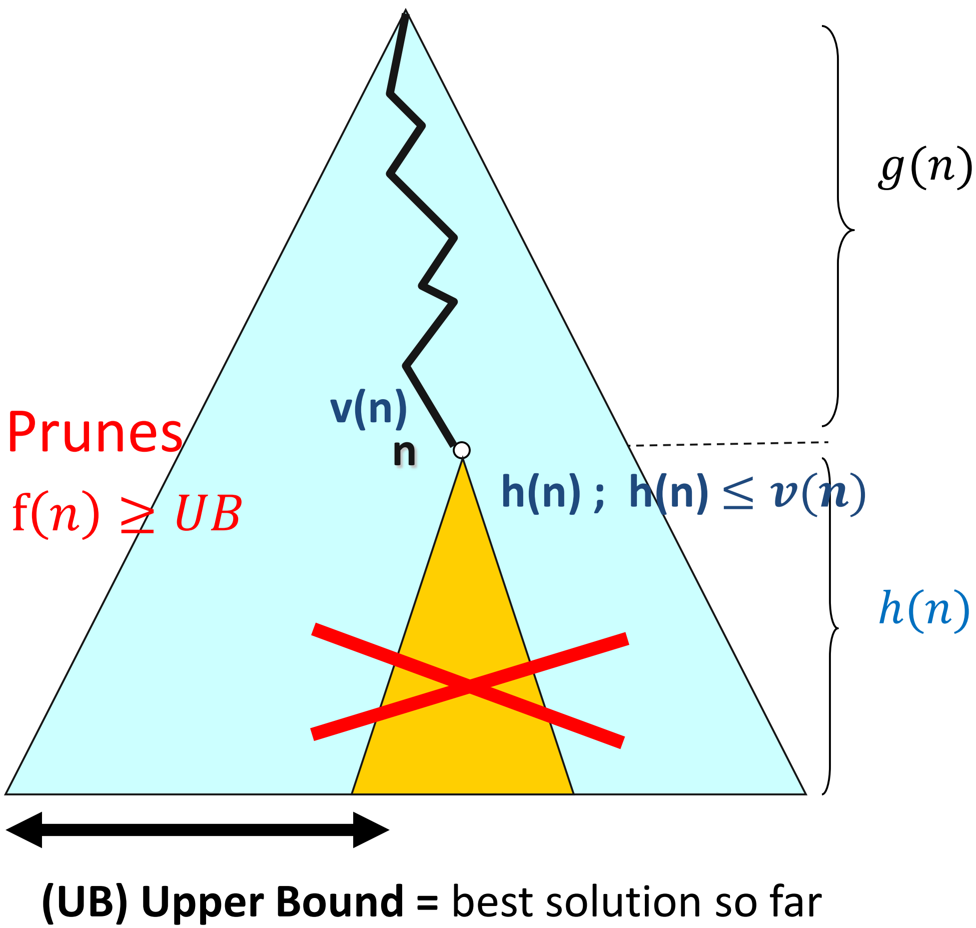


- **Best-First Algorithms, (A*)**
  - Expand nodes in OPEN list in order of min f(n)
  - Terminates with first full solution (for MAP)
- **Properties**
  - Optimal, if **h(n)** $\leq \boldsymbol{v(n)}$
  - Expands least set of nodes
  - exponential memory
  - **Not anytime solution for MAP**
  - **Yields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

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

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

Basic Heuristic Search; **Depth-First**

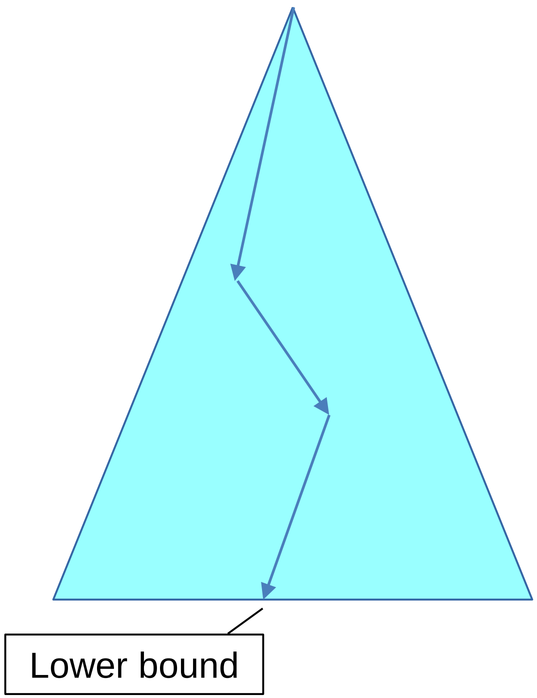


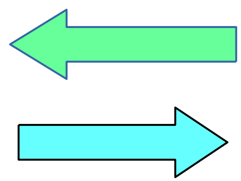
- **Depth-First (B&B for MAP)**
  - Expand in dfs order
  - Update UB with each solution
  - Prunes if f(n) ≥ UB
- **Properties**
  - Can use only linear memory
  - Yields upper bounds anytime

Best+Depth-First Search

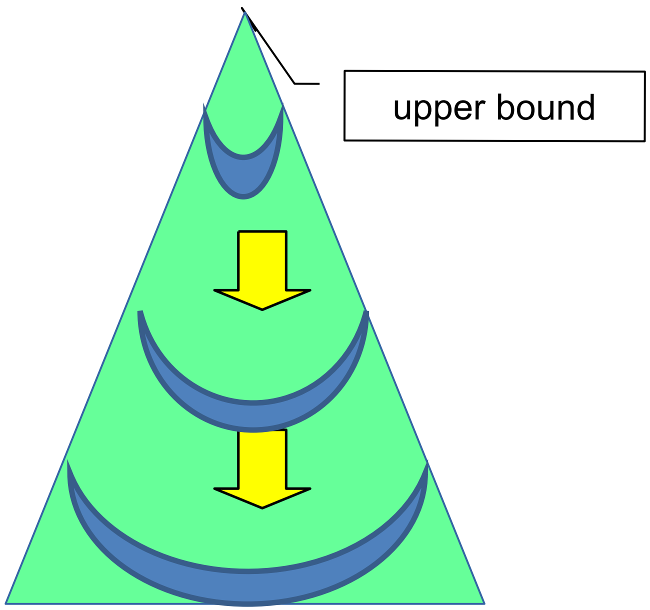
- Yields upper and lower bounds anytime

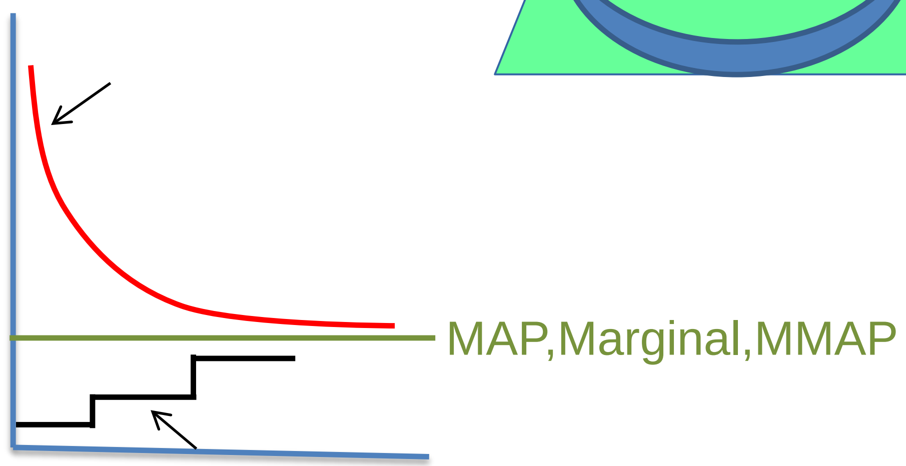
Depth-First search





Best-First search

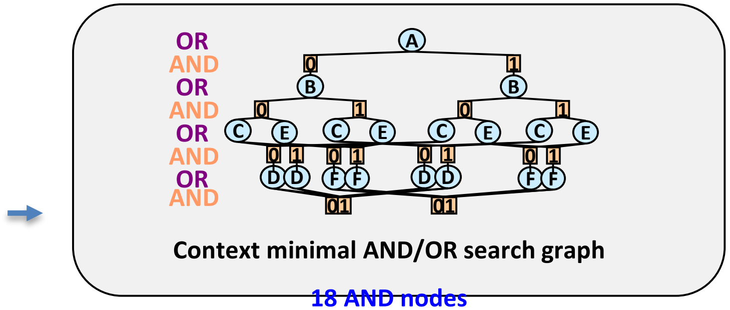




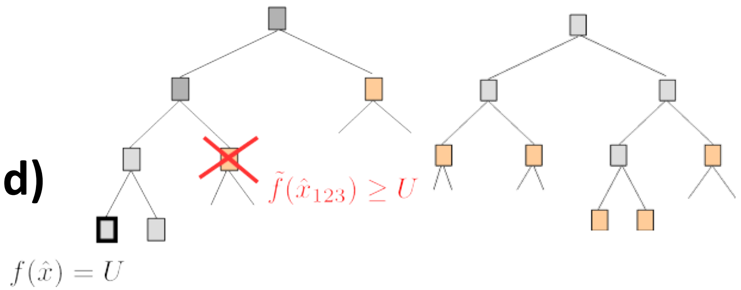
Road Map: Search

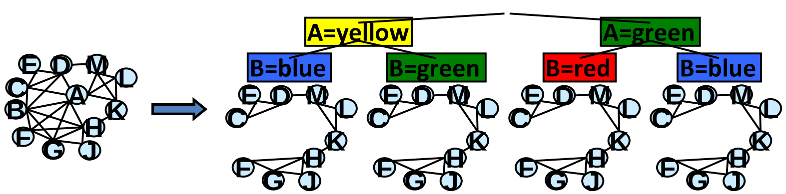
- Review Graphical Modes

- AND/OR search spaces, pseudo-trees
  - AND/OR search trees
  - AND/OR search graphs
  - **Generating good pseudo-trees**
  - Brute-force search



- Heuristic search (HS) for AND/OR spaces
  - Basic Heuristic search (Depth and Best)
  - **AND/OR Depth-first HS (branch and bound)**
  - AND/OR Best-first heuristic search
  - The Guiding MBE heuristic
  - Marginal Map (max-sum-product)



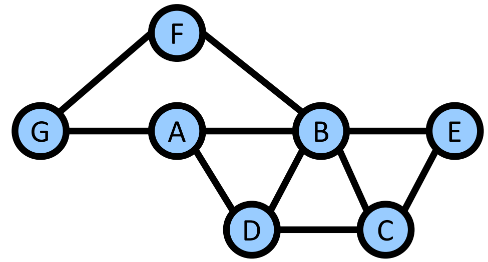


- Hybrids of search and Inference
- Summary and Class 2

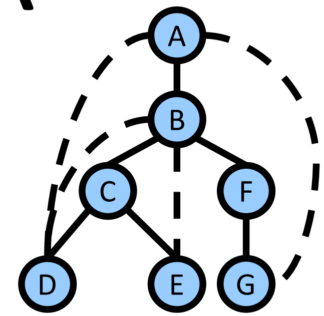
Value and Heuristic for AND/OR

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

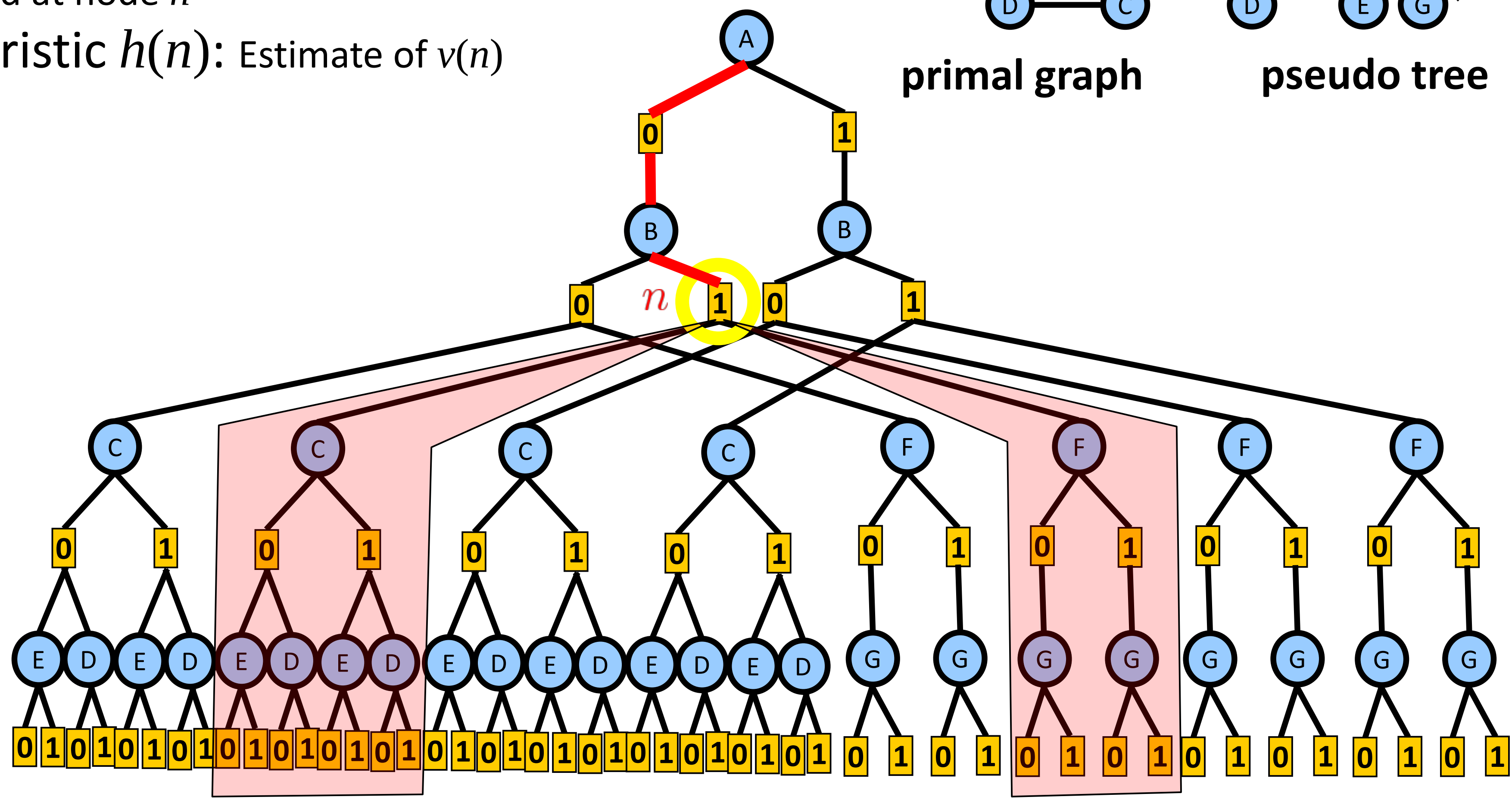
Heuristic $h(n)$: Estimate of $v(n)$



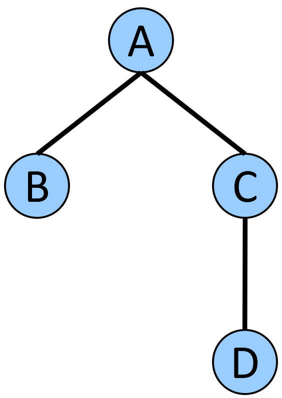
primal graph



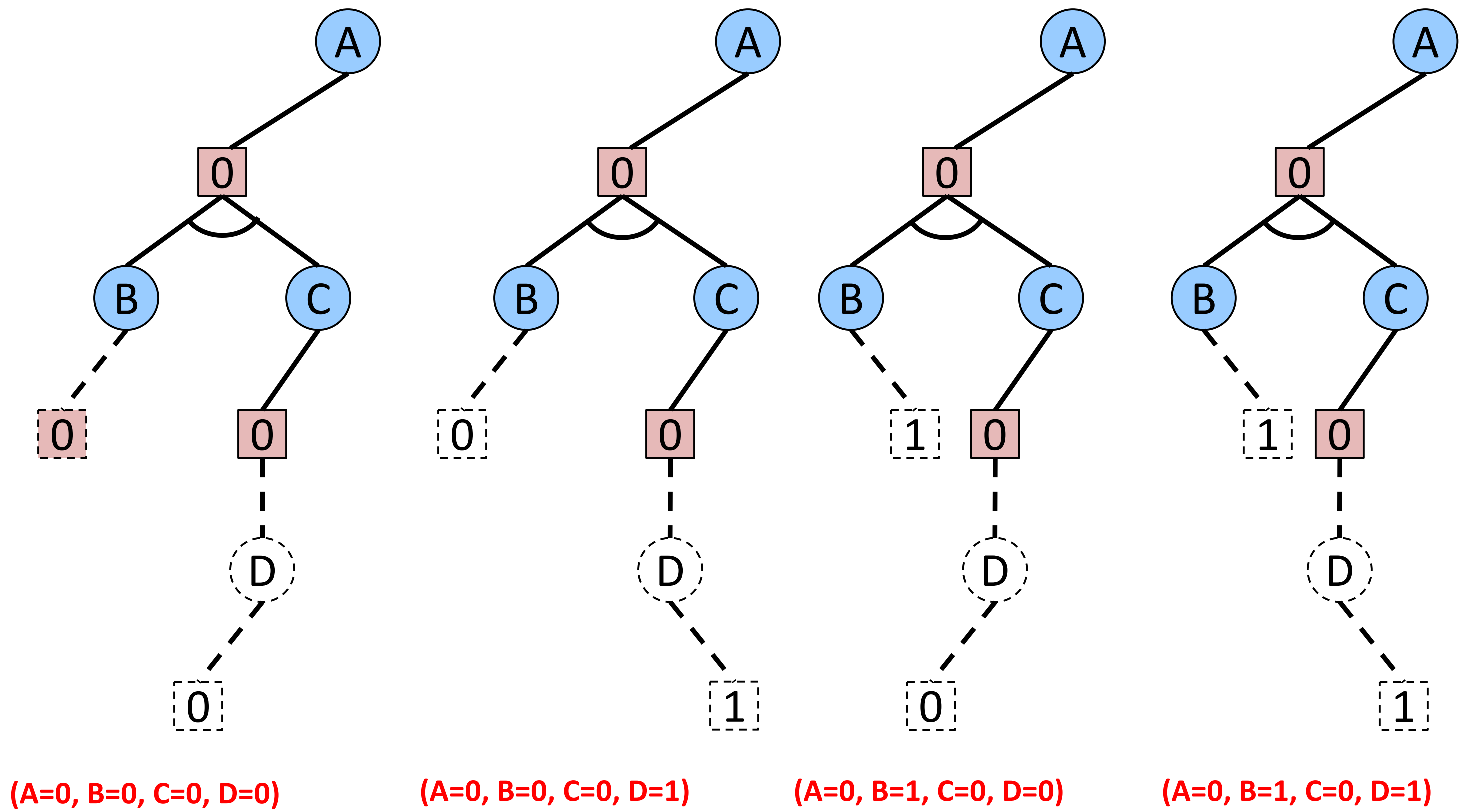
pseudo tree



Partial Solution Tree



Pseudo tree



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

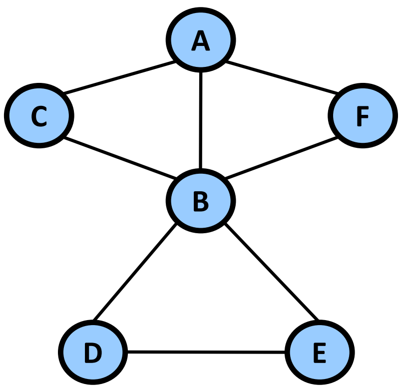
g(T’) = conditioned value of a node

V(T’) = the combined value below T’

f*(T’) = conditioned value through T’

Exact Evaluation Function

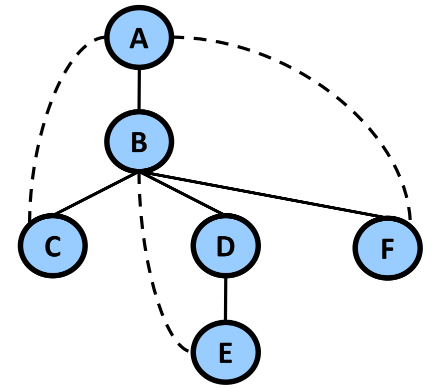
Conditioned value of a node

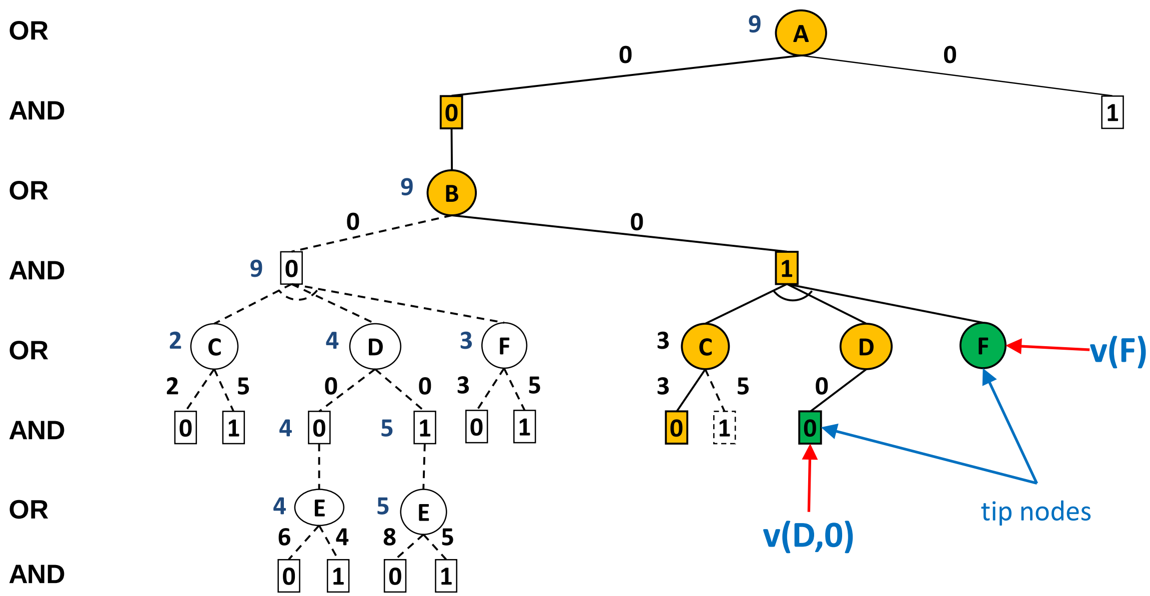


A	B	C	$f_1(ABC)$
0	0	0	2
0	0	1	5
0	1	0	3
0	1	1	5
1	0	0	9
1	0	1	3
1	1	0	7
1	1	1	2

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

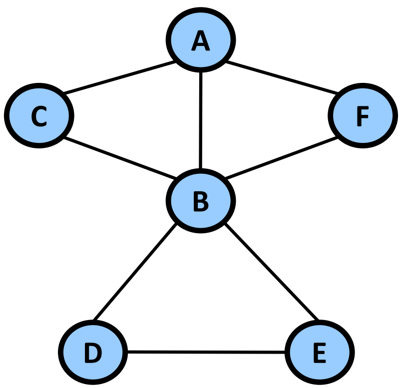
B	D	E	$f_3(BDE)$
0	0	0	6
0	0	1	4
0	1	0	8
0	1	1	5
1	0	0	9
1	0	1	3
1	1	0	7
1	1	1	4





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

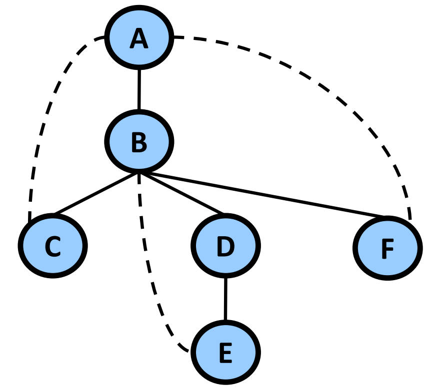
Heuristic Evaluation Function

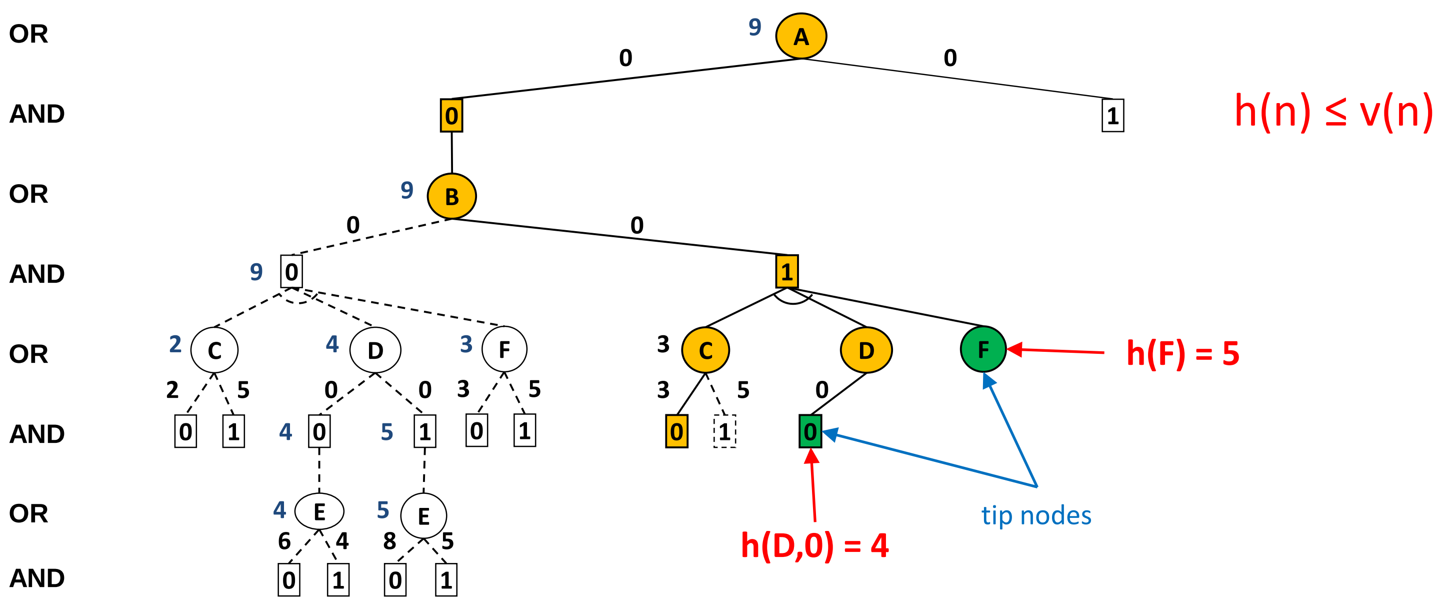


A	B	C	$f_1(ABC)$
0	0	0	2
0	0	1	5
0	1	0	3
0	1	1	5
1	0	0	9
1	0	1	3
1	1	0	7
1	1	1	2

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

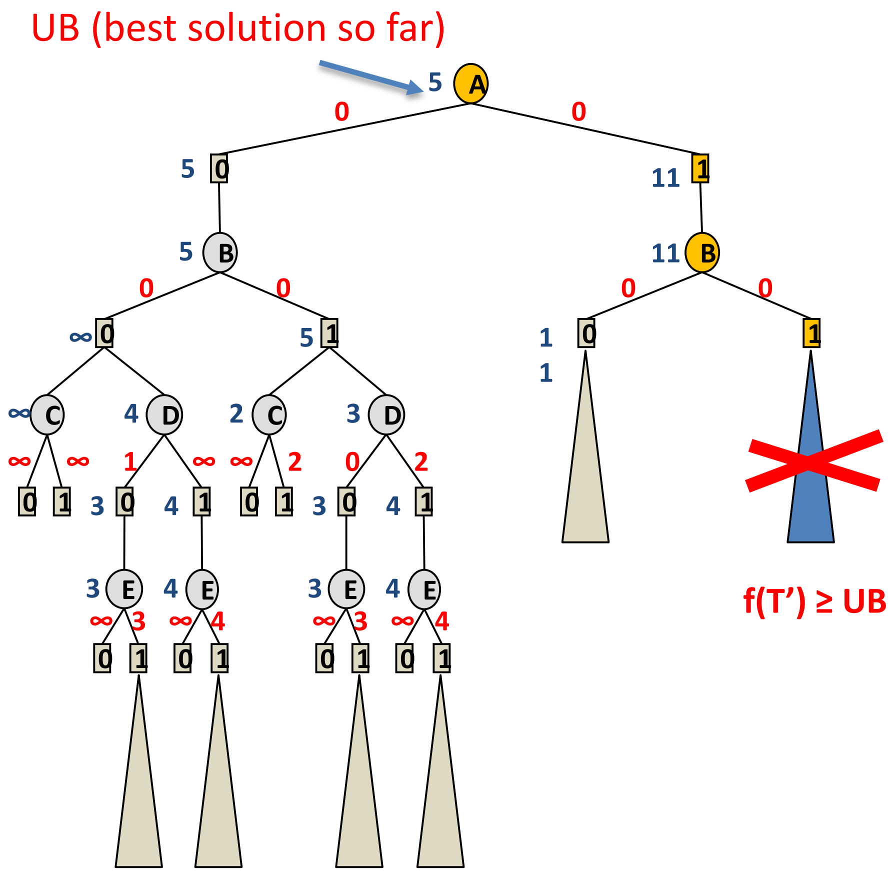
B	D	E	$f_3(BDE)$
0	0	0	6
0	0	1	4
0	1	0	8
0	1	1	5
1	0	0	9
1	0	1	3
1	1	0	7
1	1	1	4





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

Depth-First AND/OR Branch-and-Bound



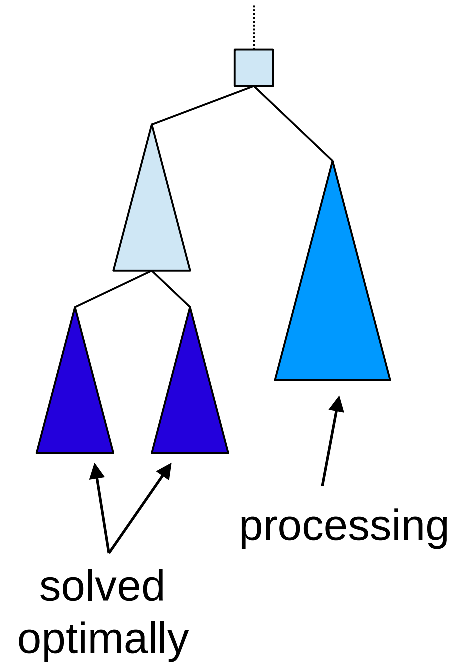
- Associate each node n with a heuristic lower bound h(n) on v(n)

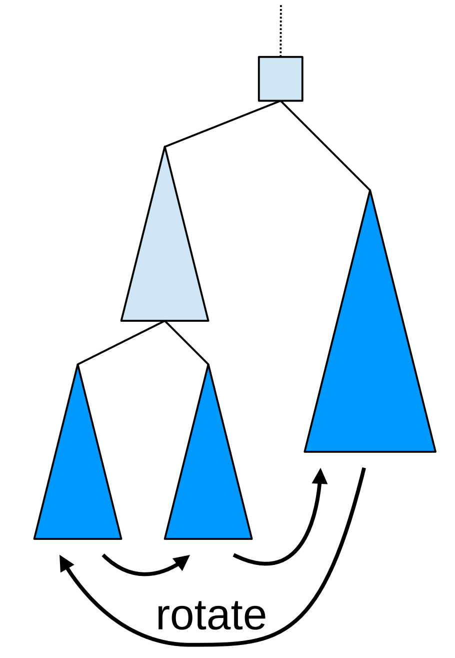
Algorithm AOBB:

- **EXPAND** (top-down)
  - Evaluate f(T') and prune search if f(T') ≥ 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

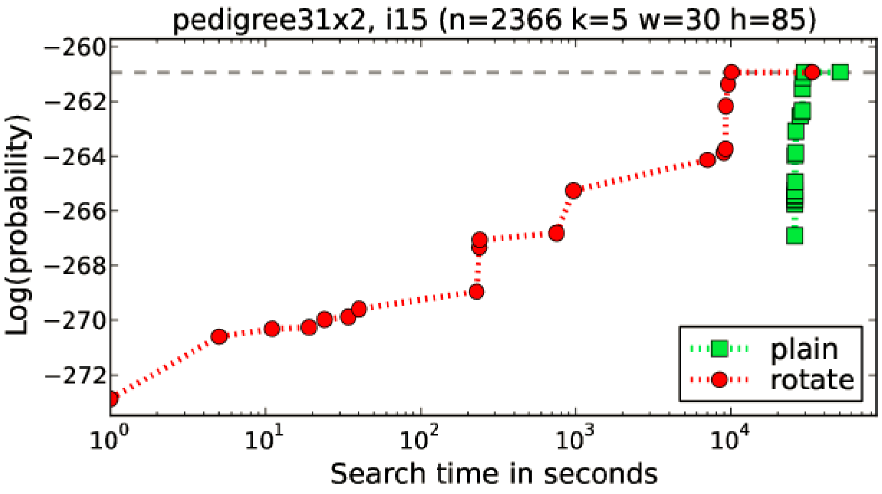
- OR Branch-and-Bound is anytime
- But AND/OR breaks anytime behavior of depth-first scheme:
  - First anytime solution delayed until last sub-problem starts processing
- **Breadth-Rotating AOBB:**
  - Take turns processing sub-problems
    - Limit number of expansions per visit
  - Solve each sub-problem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

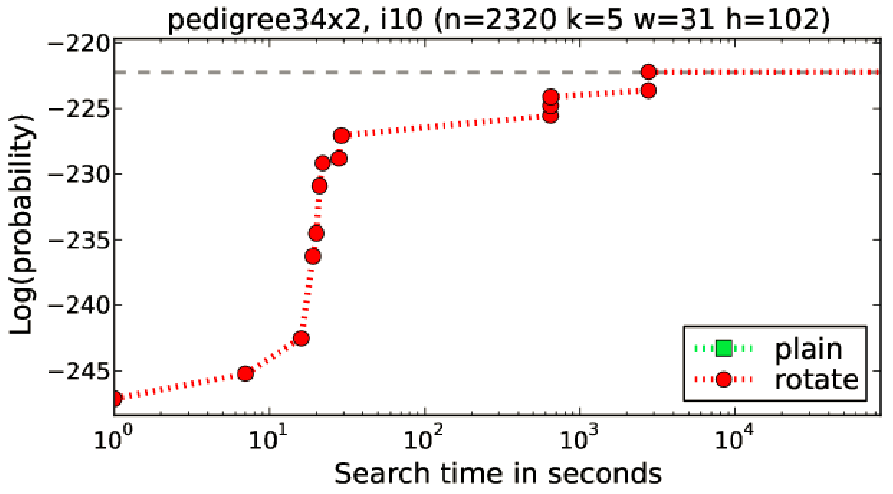




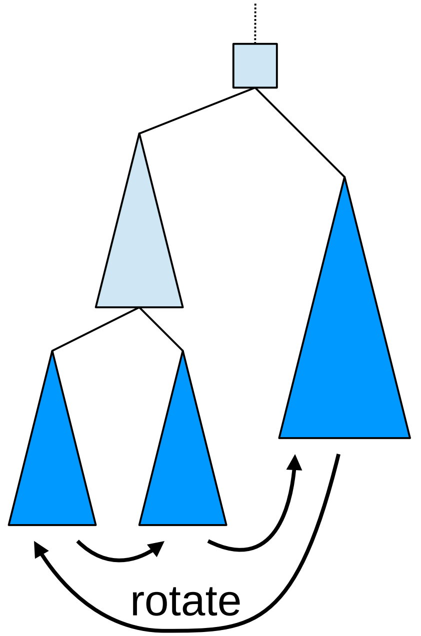
[Otten and Dechter, 2012]

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

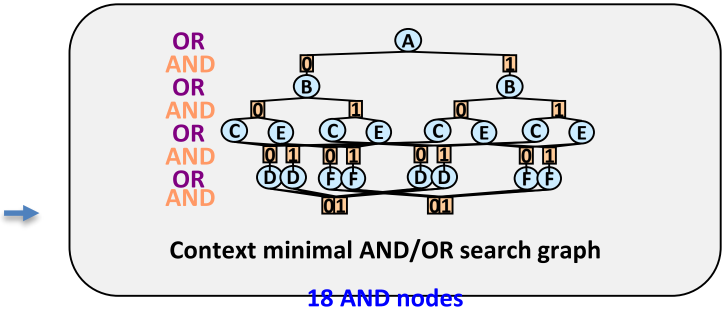


[Otten and Dechter, 2012]

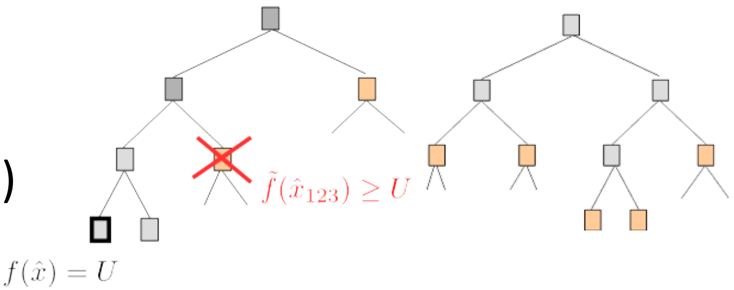
Road Map: Search

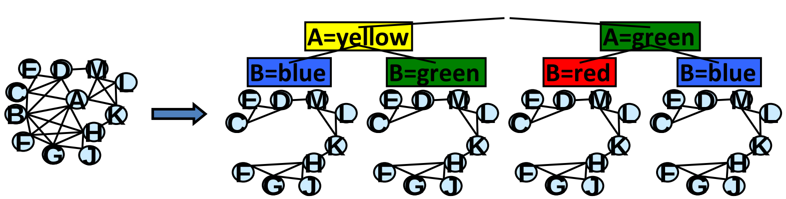
- Review Graphical Modes

- AND/OR search spaces, pseudo-trees
  - AND/OR search trees
  - AND/OR search graphs
  - **Generating good pseudo-trees**
  - Brute-force search



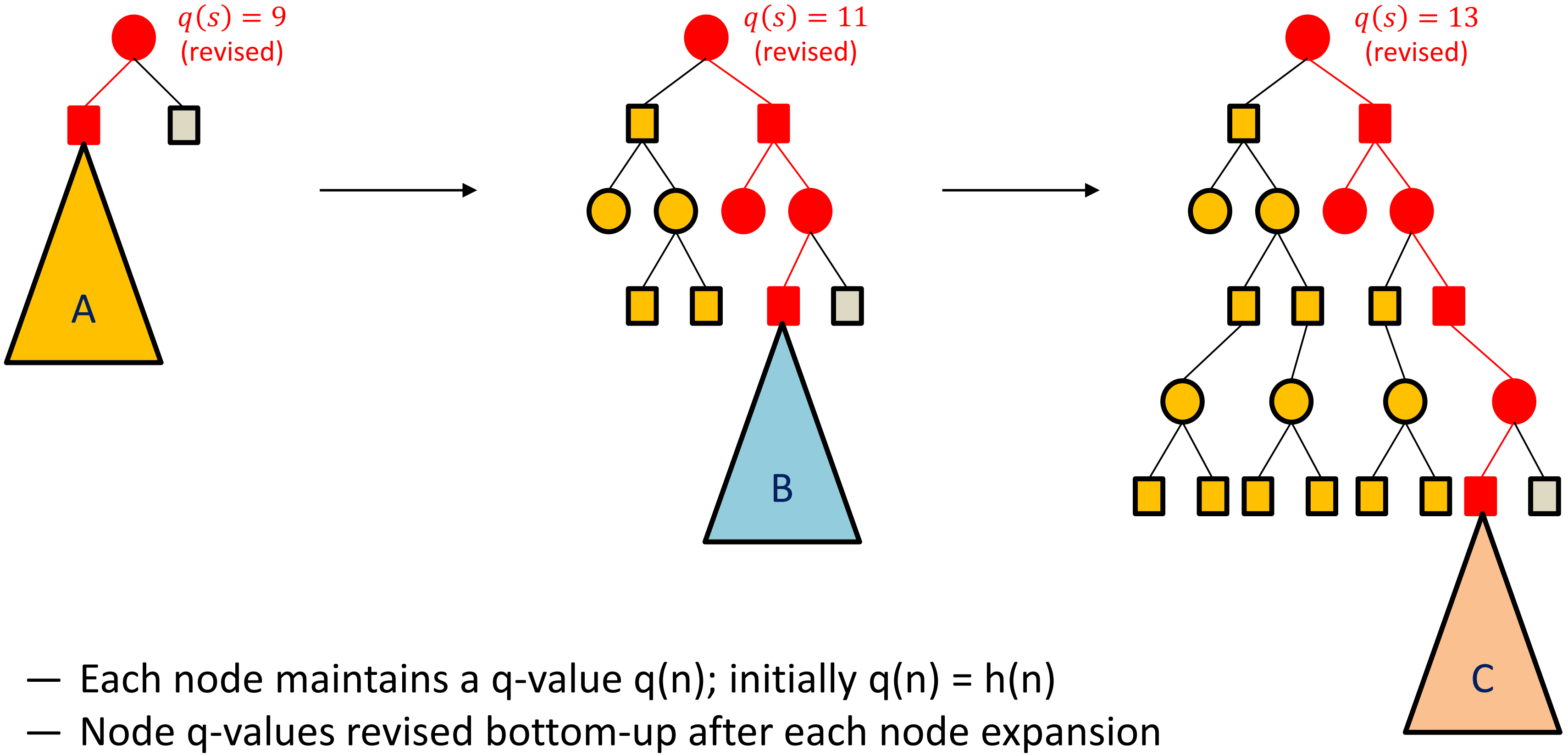
- Heuristic search (HS) for AND/OR spaces
  - Basic Heuristic search (Depth and Best)
  - AND/OR Depth-first HS (branch and bound)
  - **AND/OR Best-first heuristic search**
  - The Guiding MBE heuristic
  - Marginal Map (max-sum-product)





- Hybrids of search and Inference
- Summary and Class 2

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)

[Marinescu and Dechter, 2006; 2009]

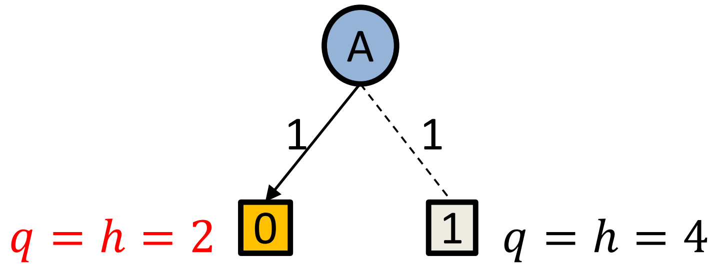
AOBF: Best-First AND/OR Search

- 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: $q(n) = \min_{n' \in succ(n)} \big(w(n,n') + q(n')\big)$
  - AND: summation: $q(n) = \Sigma_{n' \in succ(n)} q(n')$, (initially, q(n) = h(n))

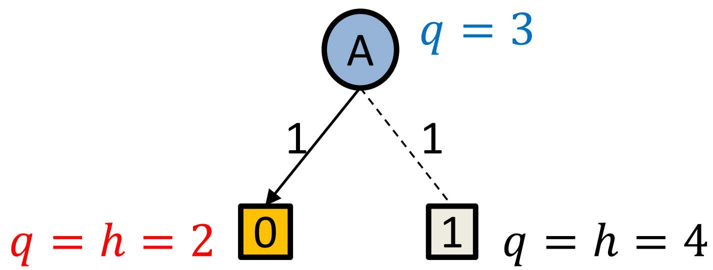
[Marinescu and Dechter, 2006; 2009]

AOBF – Expansion, Revision



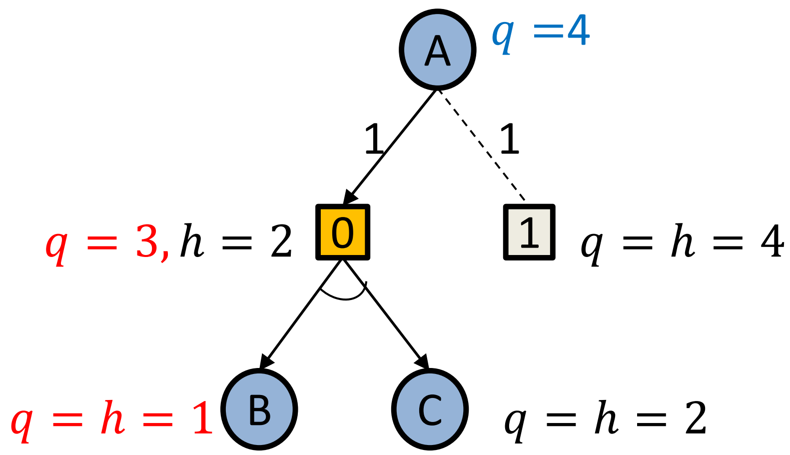
— Expand OR node A by generating its AND successors: (A,0) and (A,1)
— Initialize $q(A,0) = h(A,0) = 2; q(A,1) = h(A,1) = 4$
— Best successor is (A,0)

AOBF – Expansion, Revision



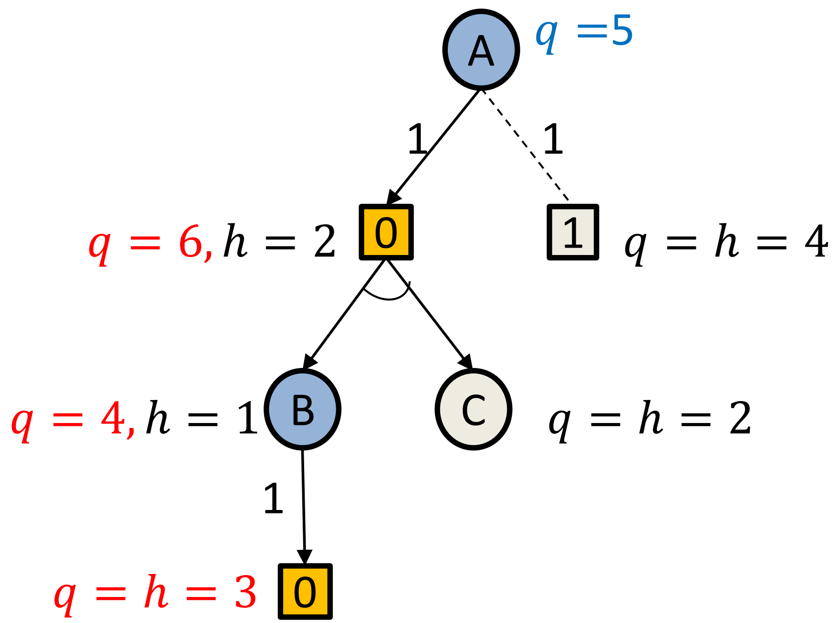
- Expand OR node A by generating its AND successors: (A,0) and (A,1)
- Initialize $q(A,0) = h(A,0) = 2; q(A,1) = h(A,1) = 4$
- Best successor is (A,0)
- Revise q-value of A: $q(A) = \min\big(q(A,0) + w_{(A,0)}, q(A,1) + w_{A,1}\big) = 3$

AOBF – Expansion, Revision



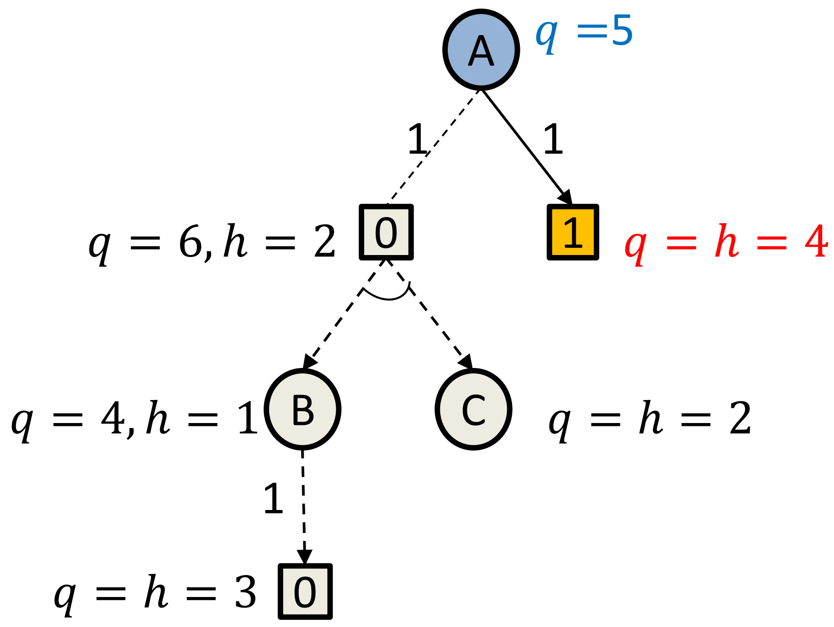
- Expand AND node (A,0) by generating its OR successors: B and C
- Revise node (A,0) q-value: $q(A,0) = q(B) + q(C) = 1 + 2 = 3$
- Revise node A q-value: $q(A) = \min(q(A,0) + w_{A,0}, q(A,1) + w_{A,1}) = 4$
- Best successor of A is (A,0)
- Expand next any of the tip nodes B or C

AOBF – Expansion, Revision



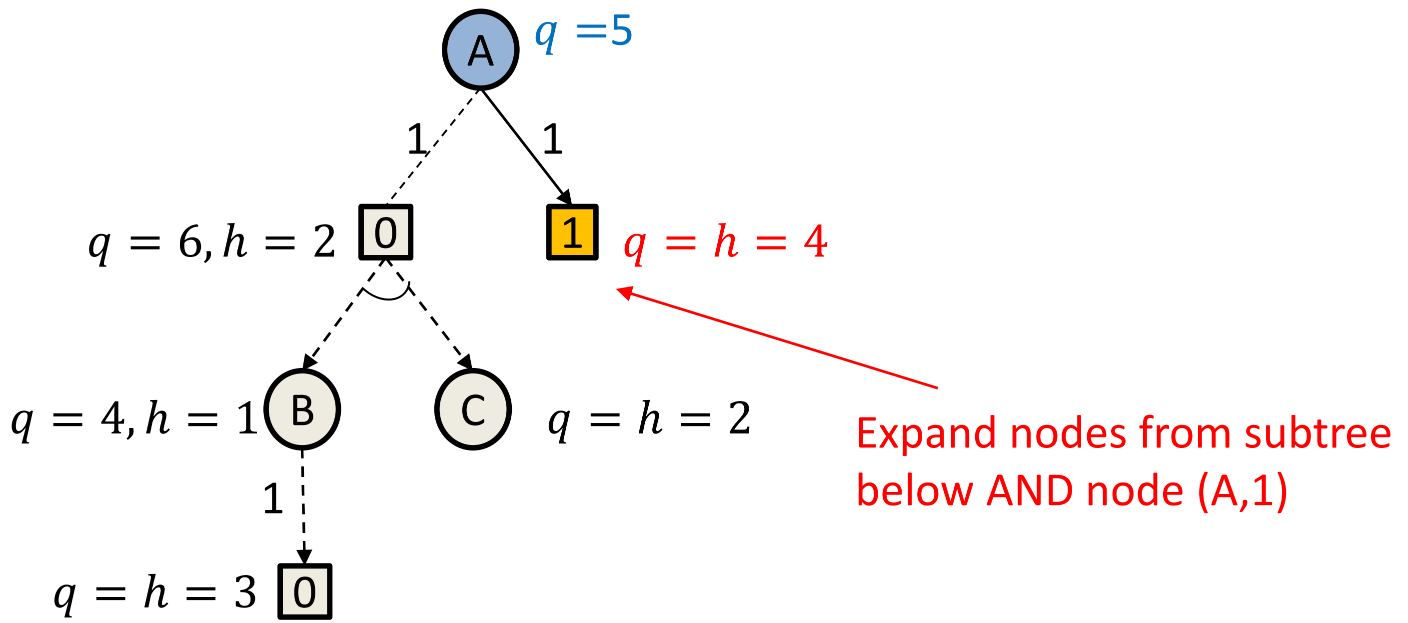
- Expand OR node B by generating its AND successor: (B,0)
- Revise node q-values: $q(B) = 4, q(A, 0) = 6$
- Revise node A q-value: $q(A) = \min\big(q(A, 0) + w_{A,0}, q(A, 1) + w_{A,1}\big) = 5$
- Best successor of A is now (A,1):
  $q(A, 1) + w_{A,1} = 5 < q(A, 0) + w_{A,0} = 7$

AOBF – Expansion, Revision



- Expand OR node B by generating its AND successor: (B,0)
- Revise node q-values: $q(B) = 4, q(A, 0) = 6$
- Revise node A q-value: $q(A) = \min\left(q(A, 0) + w_{A,0}, q(A, 1) + w_{A,1}\right) = 5$
- Best successor of A is now (A,1):
  $q(A, 1) + w_{A,1} = 5 < q(A, 0) + w_{A,0} = 7$

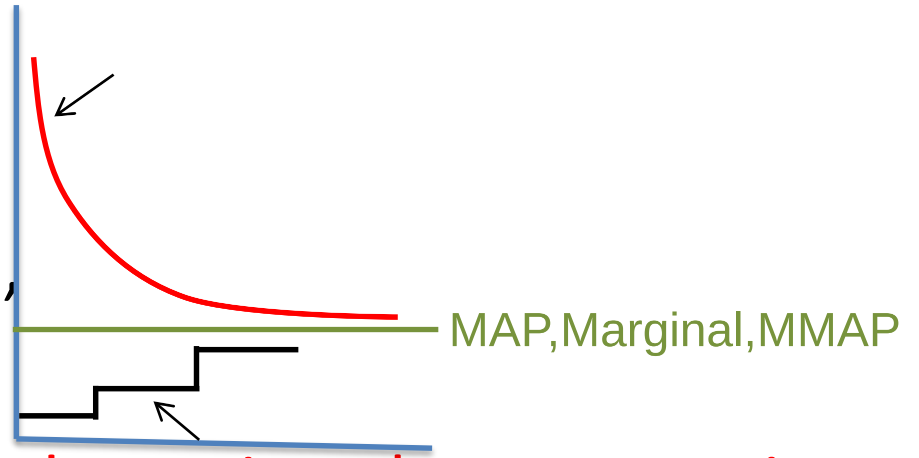
AOBF – Expansion, Revision



— Expand nodes below AND node (A,1)
— All other expanded nodes (i.e., left subtree) are kept in memory
— Needs a lot of memory!

AOBF versus AOBB

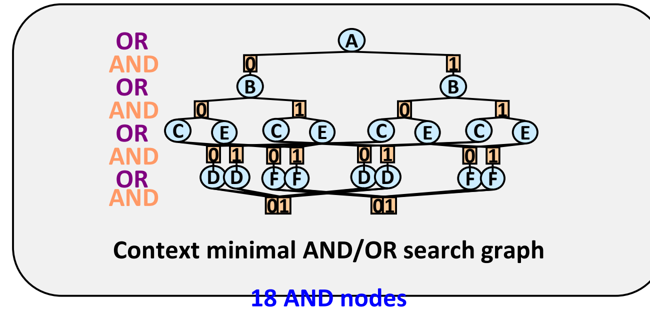
- **AOBF** expands a smallest subset of the AO search space
  - This translates into significant time savings
- **AOBB** can use far less memory by avoiding dead-caches, whereas **AOBF** keeps in memory the explicated search graph
- **AOBB** (**BRAOBB**) is anytime,



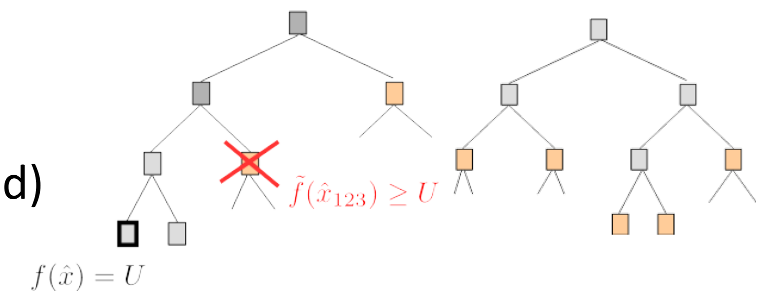
- **AOBF** generates lower bounds anytime, but not anytime solutions (configuration)

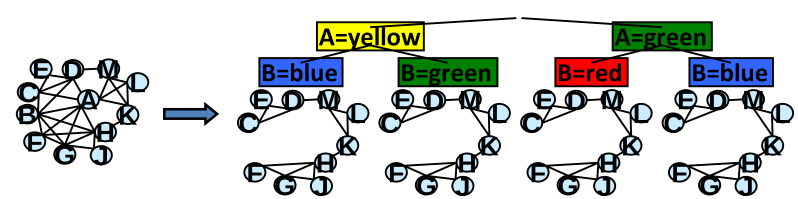
Road Map: Search

- Review Graphical Modes
- AND/OR search spaces, pseudo-trees
  - AND/OR search trees
  - AND/OR search graphs
  - **Generating good pseudo-trees**
  - Brute-force search



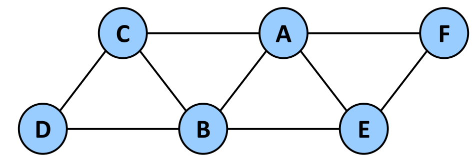
- Heuristic search (HS) for AND/OR spaces
  - Basic Heuristic search (Depth and Best)
  - AND/OR Depth-first HS (branch and bound)
  - AND/OR Best-first heuristic search
  - **The Guiding MBE heuristic**
  - Marginal Map (max-sum-product)





- Hybrids of search and Inference
- Summary and Class 2

Heuristics for Graphical Models



Given a cost function:

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

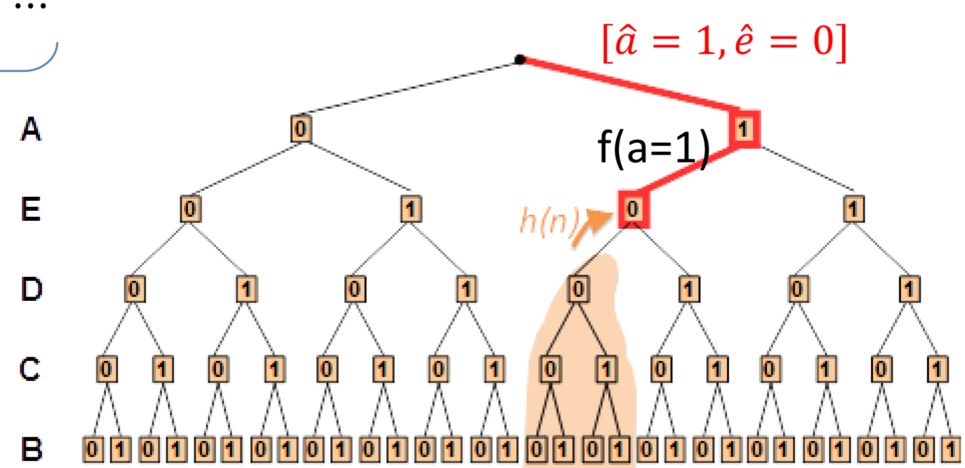
define an evaluation function over a partial assignment as the cost of its best extension:

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

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

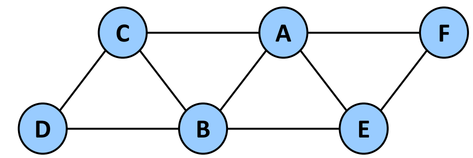
(h*=v)

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



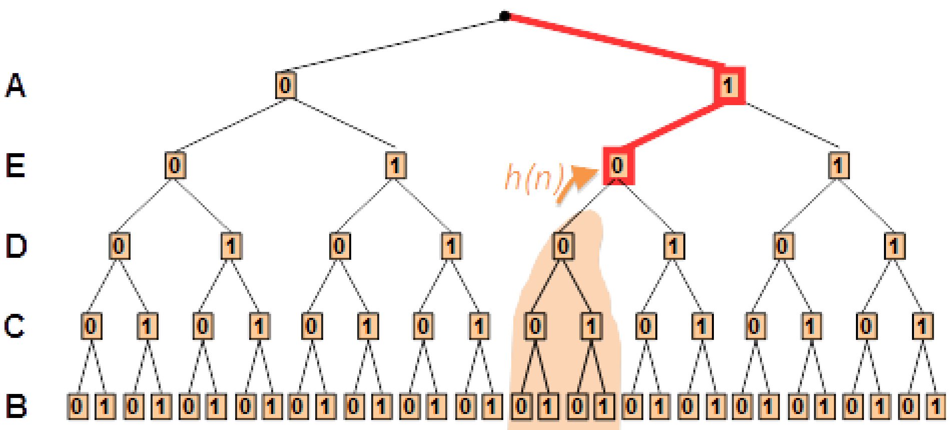
[Kask and Dechter, 2001]

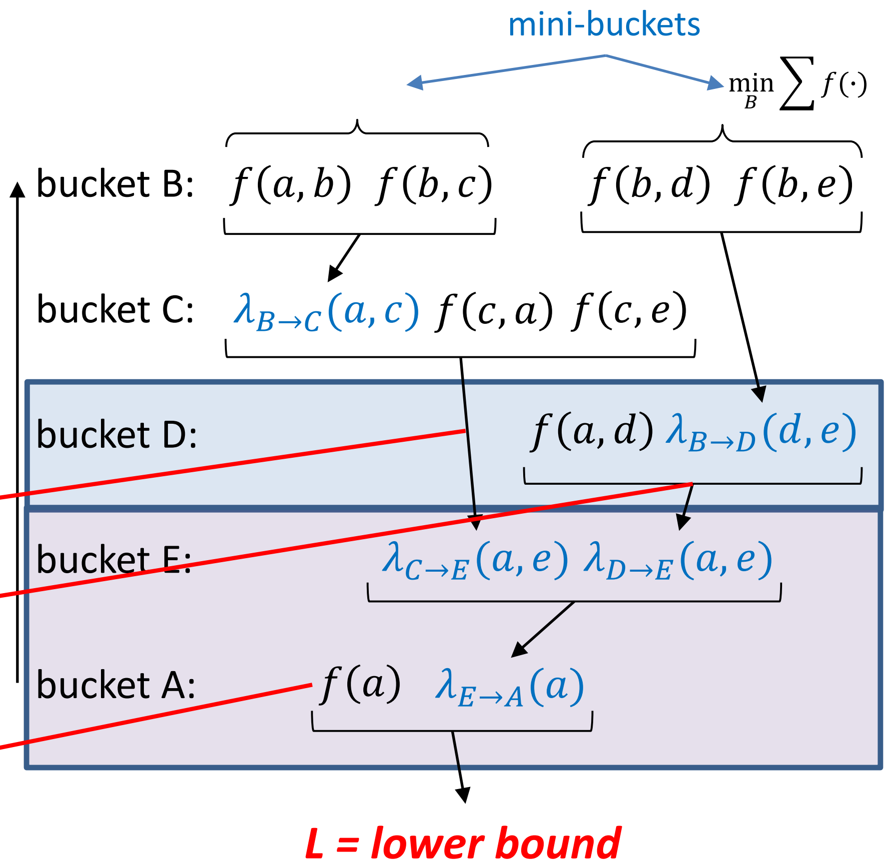
Static Mini-Bucket Heuristics



Given a partial assignment, $[\hat{a} = 1, \hat{e} = 0]$

(weighted) mini-bucket gives an admissible heuristic:





cost to go:

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

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

cost so far:

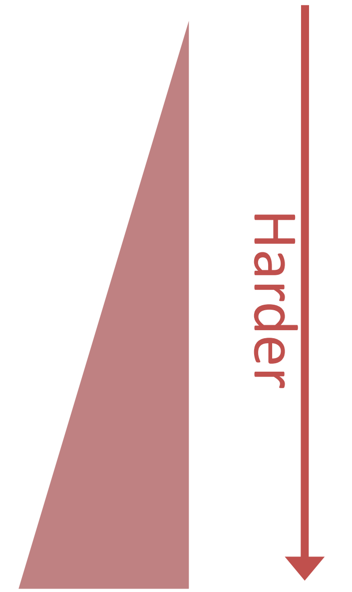
$$g(\hat{a},\hat{e},D) = f(A = \hat{a})$$

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 → more pre-processing → more accurate heuristic → less search
- Allows controlled trade-off between pre-processing and search
- Can be computed statically or dynamically during search

Heuristic for the Types of queries

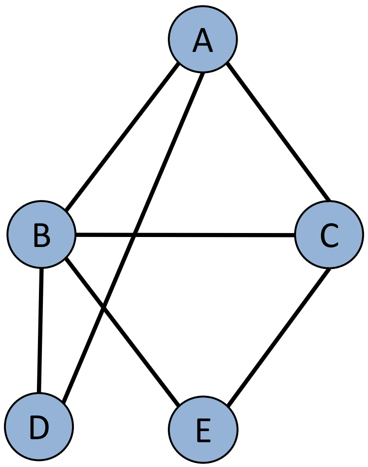
Max-Inference	$f(\mathbf{x}^*) = \max_{\mathbf{x}} \prod_{\alpha} f_\alpha(\mathbf{x}_\alpha)$
Sum-Inference	$Z = \sum_{\mathbf{x}} \prod_{\alpha} f_\alpha(\mathbf{x}_\alpha)$
Mixed-Inference	$f(\mathbf{x}_M^*) = \max_{\mathbf{x}_M} \sum_{\mathbf{x}_S} \prod_{\alpha} f_\alpha(\mathbf{x}_\alpha)$



- **NP-hard**: exponentially many terms
- We will focus on **approximation** algorithms
  - **Anytime**: very fast & very approximate ! Slower & more accurate

Review:Mini-bucket Elimination

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



mini-buckets

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

bucket C: $f(c,a)\ f(c,e)\ \lambda_{B\to C}(a,c)$

bucket D: $f(a,d)\ \ \lambda_{B\to D}(d,e)$

bucket E: $\lambda_{C\to E}(a,e)\ \lambda_{E\to E}(a,e)$

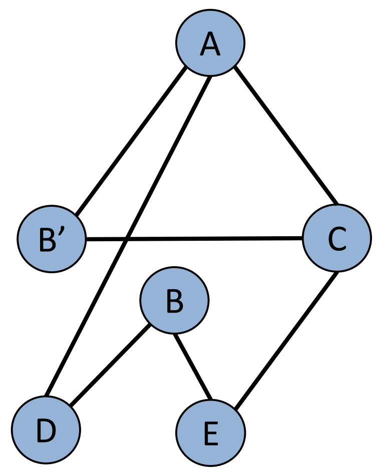
bucket A: $f(a)\ \lambda_{E\to A}(a)$

U = upper bound

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

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

$$\lambda_{C\to E}(a,e) = \max_c \ldots$$



Review: Weighted Mini-bucket

[Liu & Ihler 2011]

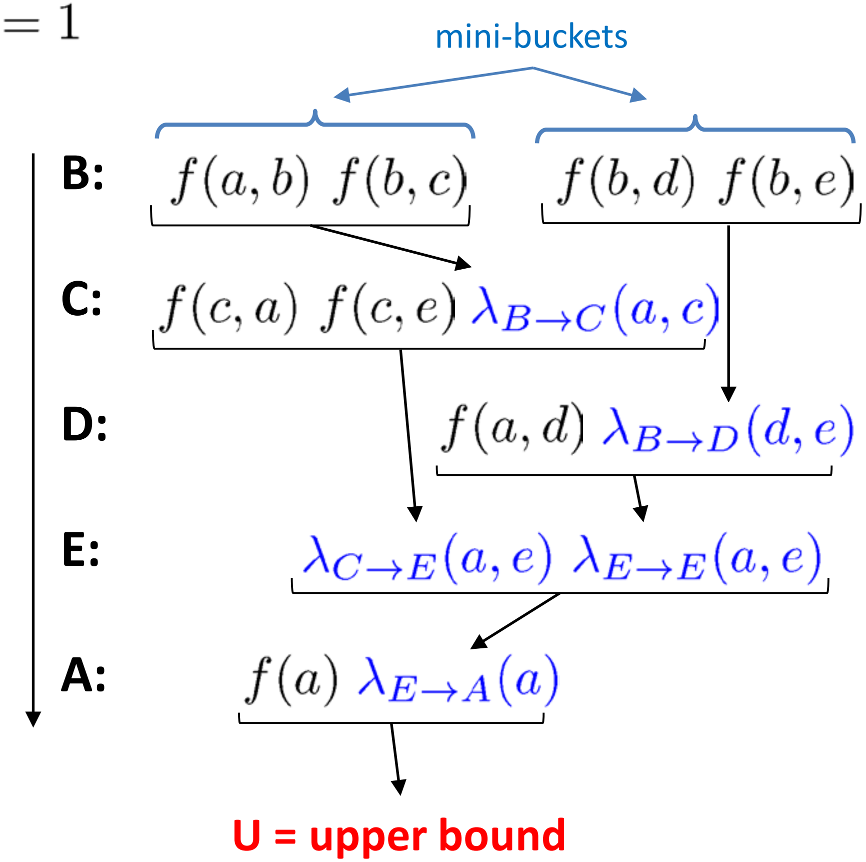
For Sum-Inference

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

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

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

⋮



Compute downward messages using weighted sum

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

Review: WMB for Marginal MAP

For Max-Inference: max-sum-product

$$\lambda_{B\to C}(a,c) = \sum_b^{w_1} f(a,b) f(b,c)$$

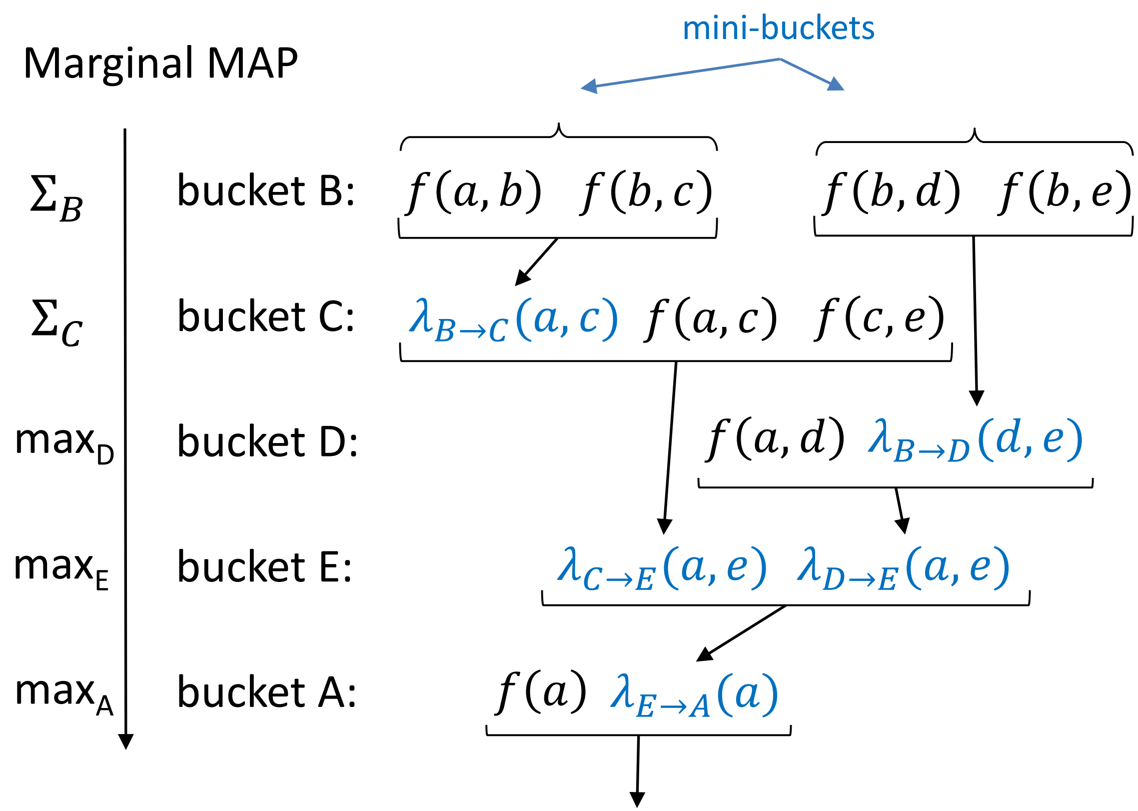
$$\lambda_{B\to D}(d,e) = \sum_b^{w_2} f(b,d) f(b,e)$$

$$(w_1 + w_2 = 1)$$

⋮

$$\lambda_{E\to A}(a) = \max_e \lambda_{C\to E}(a,e)\lambda_{D\to E}(a,e)$$

$$U = \max_a f(a)\lambda_{E\to A}(a)$$



U = upper bound

Can optimize over cost-shifting and weights
(single pass "MM" or iterative message passing)

[Liu and Ihler, 2011; 2013]
[Dechter and Rish, 2003]

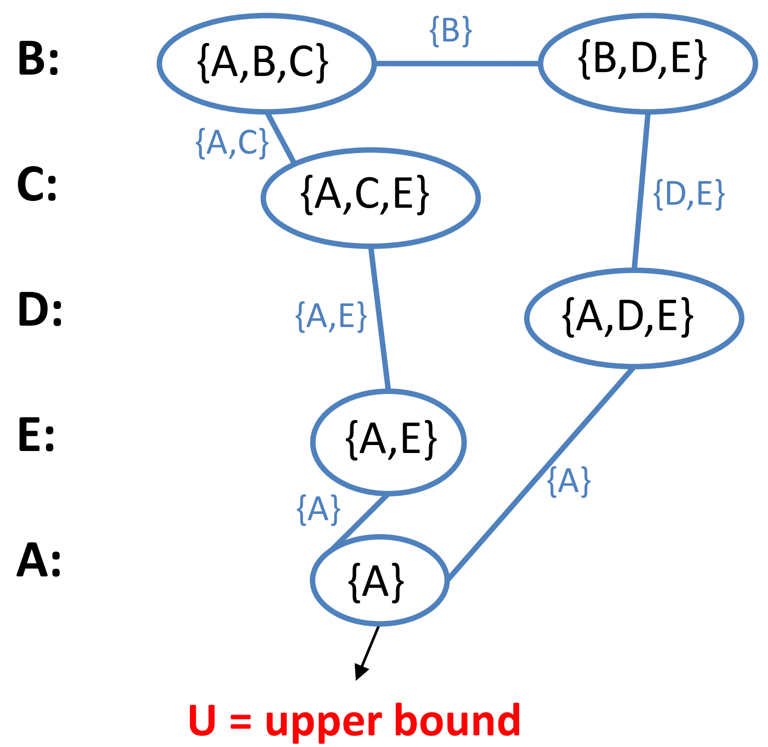
Review:MBE+Moment Matching

For all queries

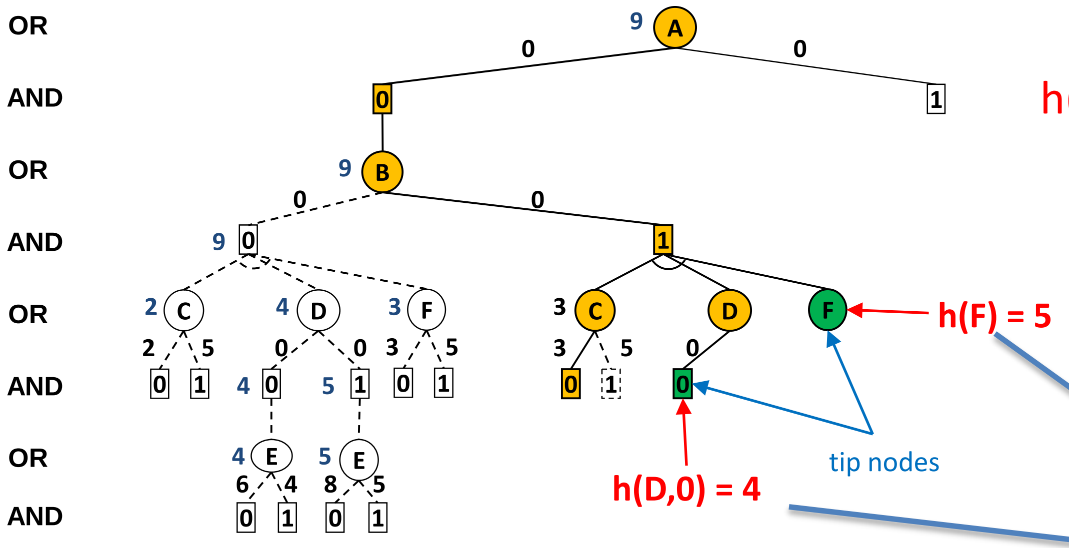
[Ihler et al. 2012]

- Downward pass as cost shifting
- Can also do cost shifting within mini-buckets:
  "Join graph" message passing
- "Moment-matching" version:
  One message exchange within each bucket, during downward sweep
- Optimal bound defined by cliques ("regions") and cost-shifting f'n scopes ("coordinates")

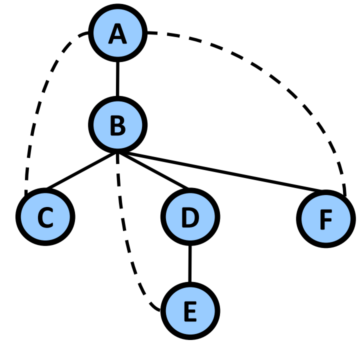
Join graph:



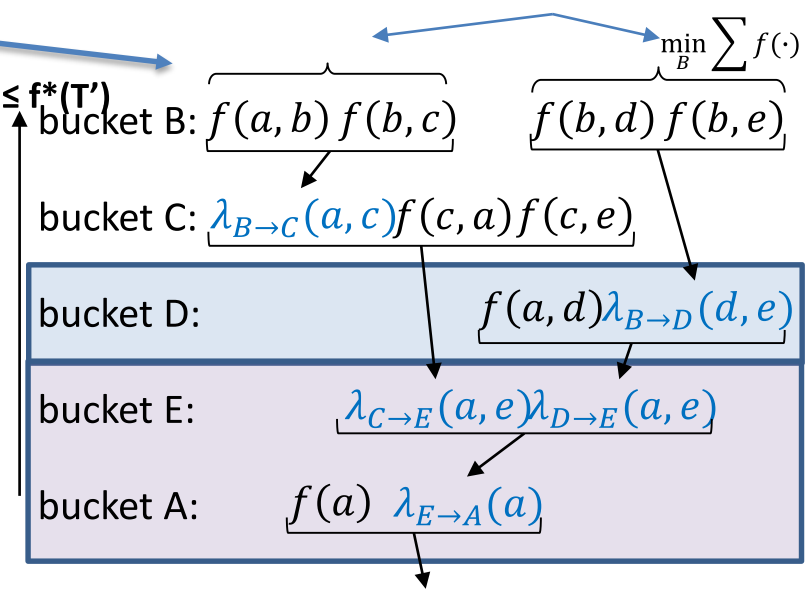
MBE Heuristic Guides AO Search



h(n) ≤ v(n)



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



L = lower bound

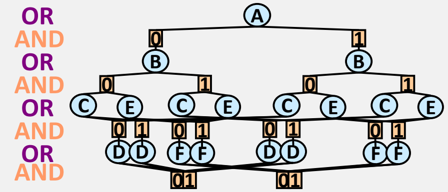
Heuristics for AND/OR Search

- In the AND/OR search space **h(n)** can be computed using any heuristic. Examples:
  - Static Mini-Bucket heuristics
    [Kask & Dechter, 2001], [Marinescu & Dechter, 2005; 2009]
  - Dynamic Mini-Bucket heuristics
    [Marinescu & Dechter, 2005; 2009]
  - Maintaining local consistency
    [Larrosa & Schiex, 2003], [de Givry et al., 2005]
  - LP relaxations
    [Nemhauser & Wosley, 1998]; [Marinescu & Dechter, 2010]

Road Map: Search

- Review Graphical Modes

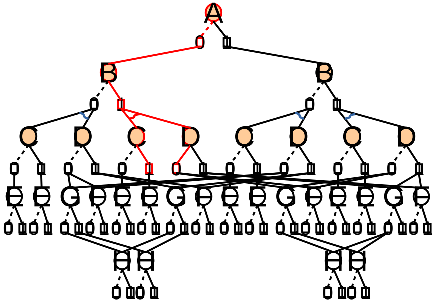
- AND/OR search spaces, pseudo-trees
  - AND/OR search trees
  - AND/OR search graphs
  - Generating good pseudo-trees
  - Brute-force traversal



Context minimal AND/OR search graph

18 AND nodes

- Heuristic search for AND/OR spaces
  - Basic heuristic search
  - Depth-first AND/OR branch and bound
  - Best-first AND/OR search
  - The Guiding MBE heuristic
  - **Marginal Map (max-sum-product)**

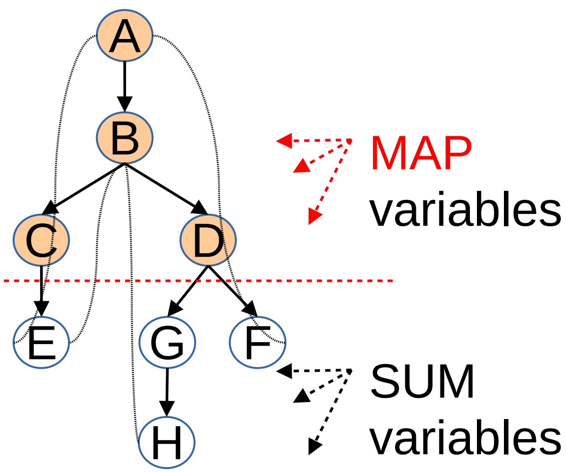


$$\boldsymbol{x}_B^* = \arg\max_{\boldsymbol{x}_B} \sum_{\boldsymbol{x}_A} \prod_{\alpha} \psi(\boldsymbol{x}_\alpha)$$

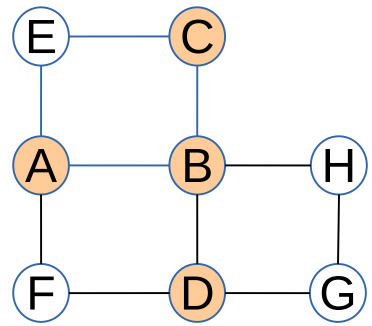
NP^{PP}-complete

- Hybrids of search and Inference

AND/OR Search for Marginal MAP



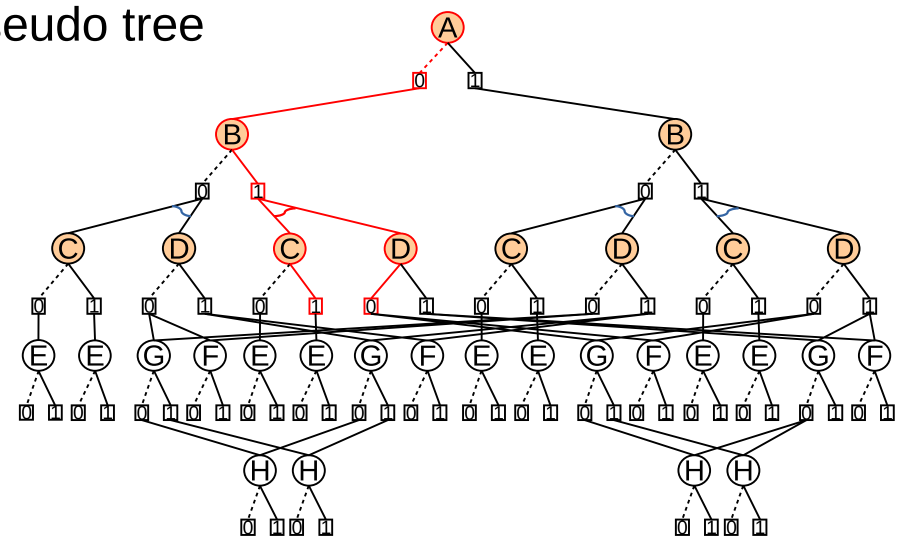
constrained
pseudo tree



primal
graph

$X_M = \{A, B, C, D\}$

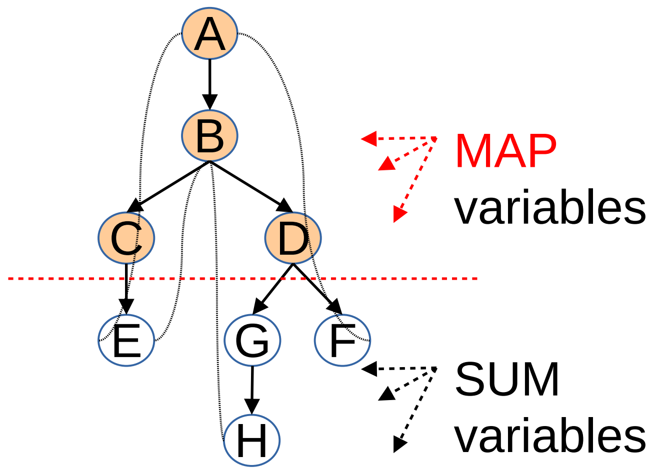
$X_S = \{E, F, G, H\}$



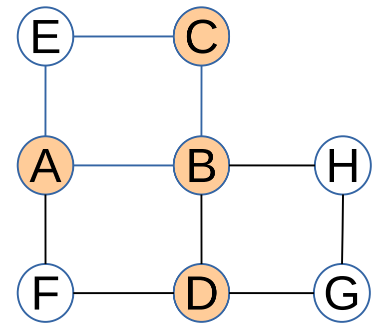
- Node types
  - OR (MAP): max
  - OR (SUM): sum
  - AND: multiplication

[Marinescu, Dechter and Ihler, 2014]

AND/OR Search for Marginal MAP

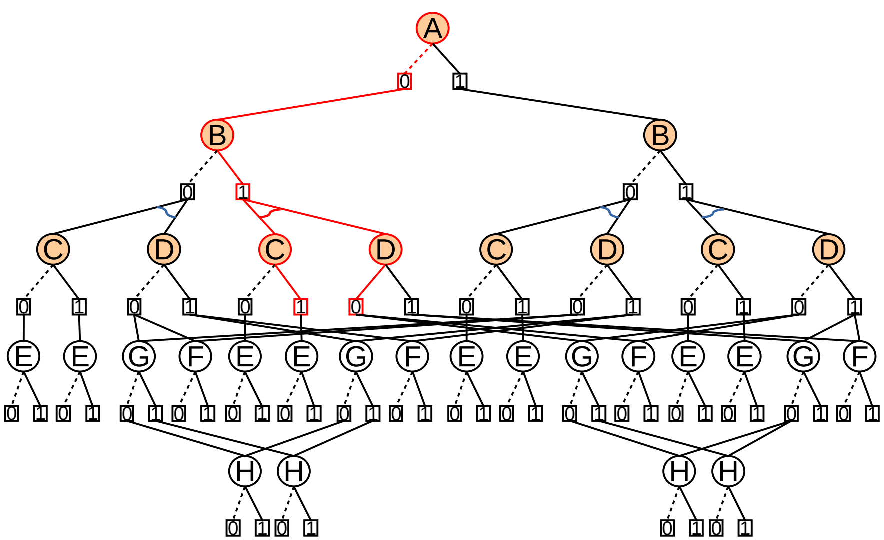


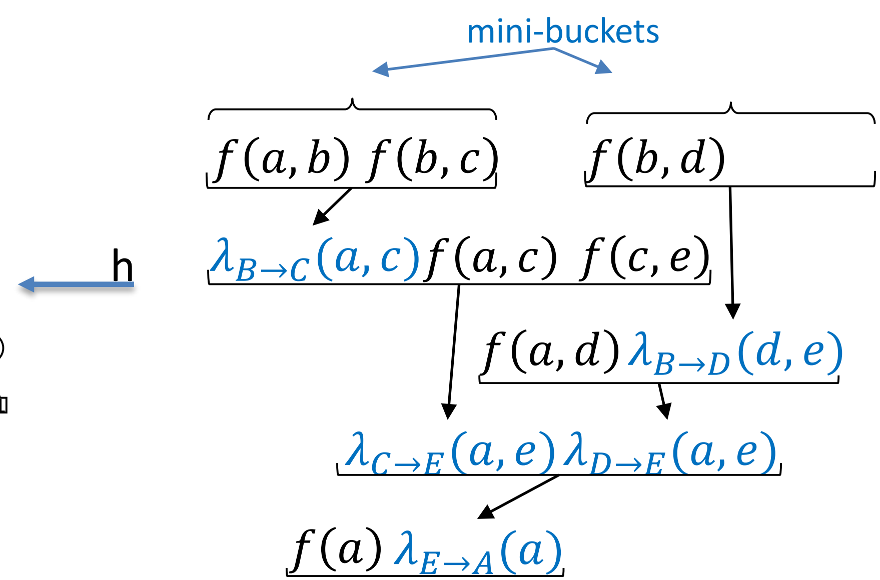
constrained
pseudo tree



primal graph

Anytime Depth+Best to yield upper and lower bounds





[Marinescu, Dechter and Ihler, 2014]

Mixed-Inference Search Algorithms

2003

Park, Darwiche
Depth-First BnB
Join-tree upper bound
(relaxed variable ordering)

- depth-first search
- dynamic heuristic

2009

Yuan, Hansen
Depth-First BnB
Incremental
Join-tree upper bound

- static heuristic

2014

Marinescu, Dechter, Ihler
Depth-First BnB
AND/OR Search
WMB Heuristic

- compact AND/OR search space
- more accurate WMB heuristics

2015

Marinescu, Dechter, Ihler
Best-First/Recursive BF
AND/OR Search
WMB heuristic

- BF avoids solving summation problems
- very memory intensive
- no anytime, return optimal solution or no solution

2016

Lee,Marinescu, Dechter, Ihler
Weighted Best-First
Anytime Depth-First AND/OR
WMB heuristic

- anytime solutions
- infrequent solution updates
- still memory intensive

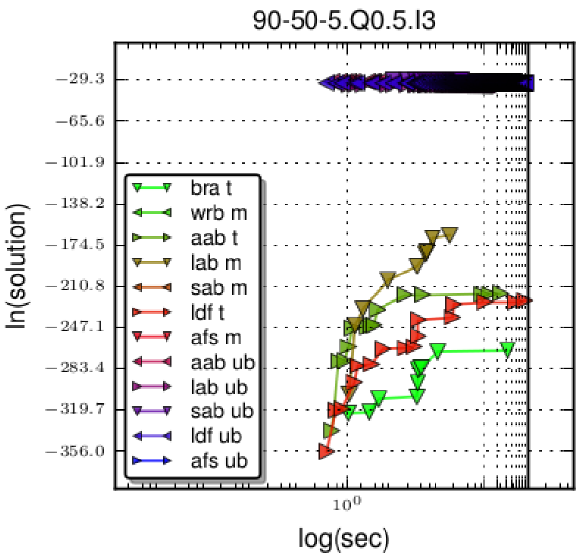
2017

Marinescu, Lee, Iihler, Dechter
Best+Depth-First
- high quality upper/lower bounds
- more frequent solution updates
- memory effective

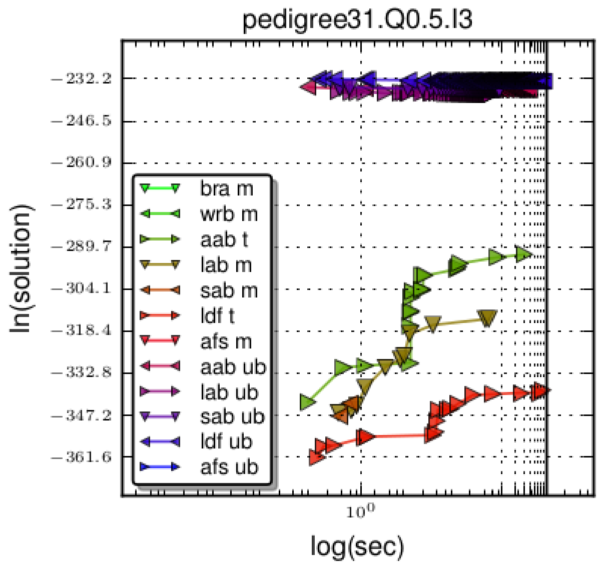
Anytime Bounding of Marginal MAP

(UAI'14, IJCAI'15, AAAI'16, AAAI'17, (Marinescu, Lee, Ihler, Dechter)

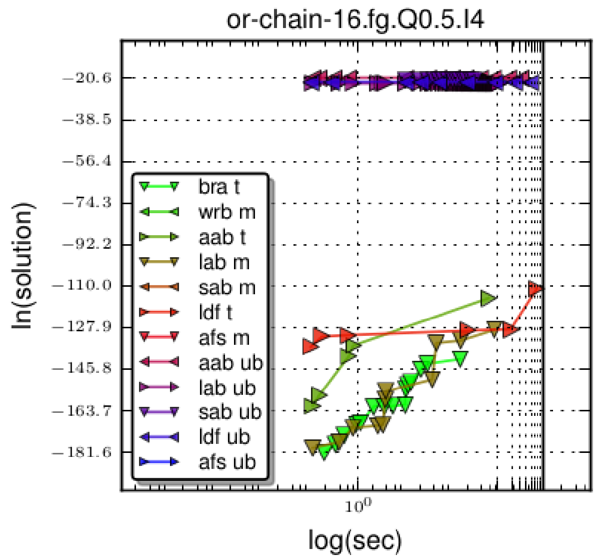
- Anytime search status for individual instances



N:2500 F:2500 K:2 S:3
W:788 H:817



N:1183 F:1183 K:5 S:5
W:272 H:290



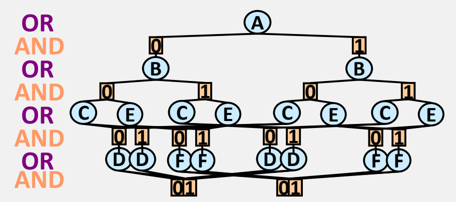
N:1675 F:1701 K:2 S:3
W:259 H:298

- search: LAOBF (lab), AAOBF (aab), LnDFS (ldt), BRAOBB (bra)
- heuristic: WMB-MM (20)
- memory: 24 GB

Other algorithms couldn't find any solution due to memory out

Road Map: Search

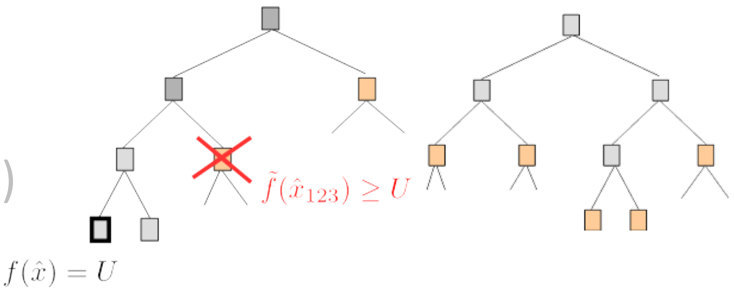
- Review Graphical Modes
- AND/OR search spaces, pseudo-trees
  - AND/OR search trees
  - AND/OR search graphs
  - Generating good pseudo-trees
  - Brute-force search



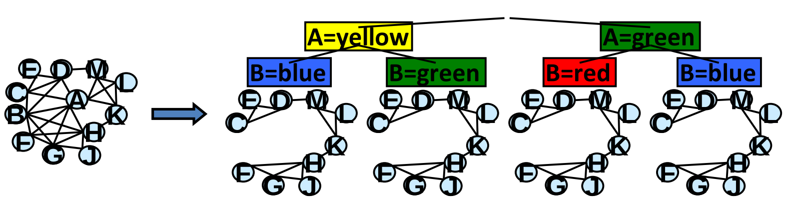
Context minimal AND/OR search graph

18 AND nodes

- Heuristic search (HS) for AND/OR spaces
  - Basic Heuristic search (Depth and Best)
  - AND/OR Depth-first HS (branch and bound)
  - AND/OR Best-first heuristic search
  - The Guiding MBE heuristic
  - Marginal Map (max-sum-product)

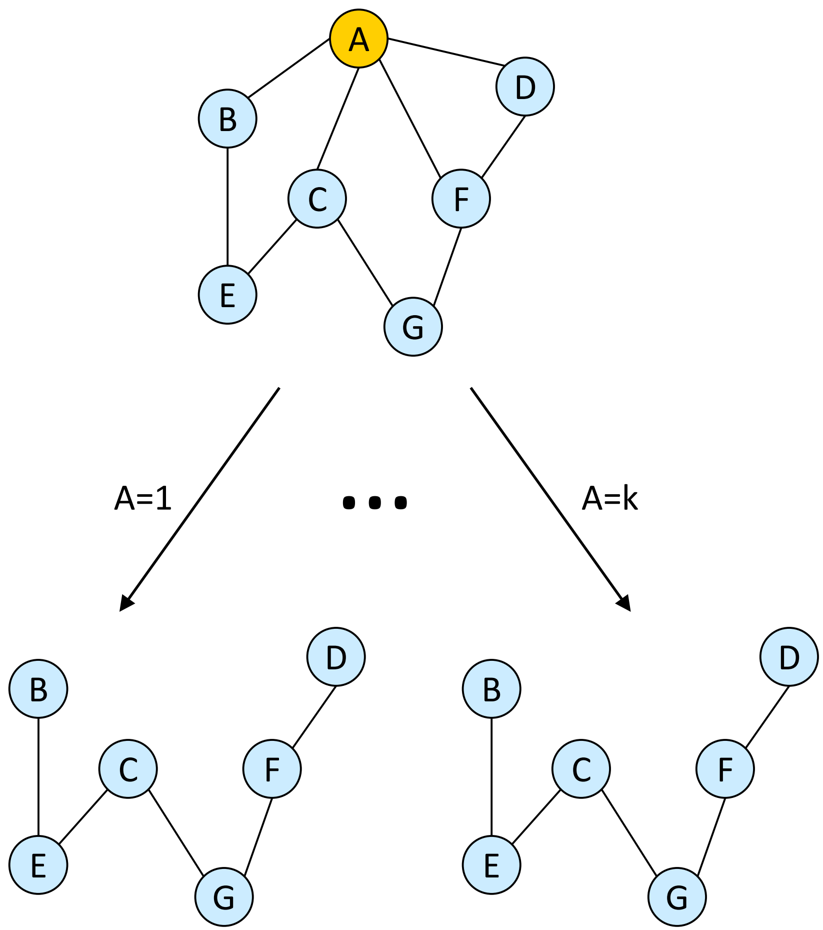


- **Hybrids of search and Inference**
- Summary and Class 2



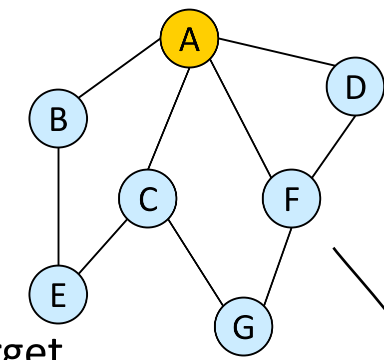
Conditioning versus Elimination

Conditioning (search)

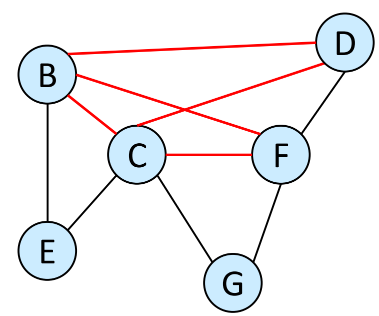


k “sparser” problems

Elimination (inference)



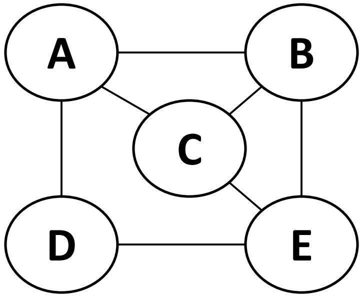
The main target
In conditioning is to
Trade memory for time.
It is not really possible to do better
Then elimination



1 “denser” problem

Hybrid: Cutset-Conditioning

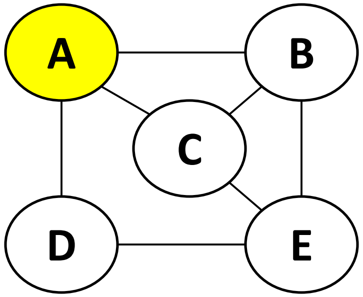
Variable Branching by Conditioning



Hybrid: Cutset-Conditioning

Variable Branching by Conditioning

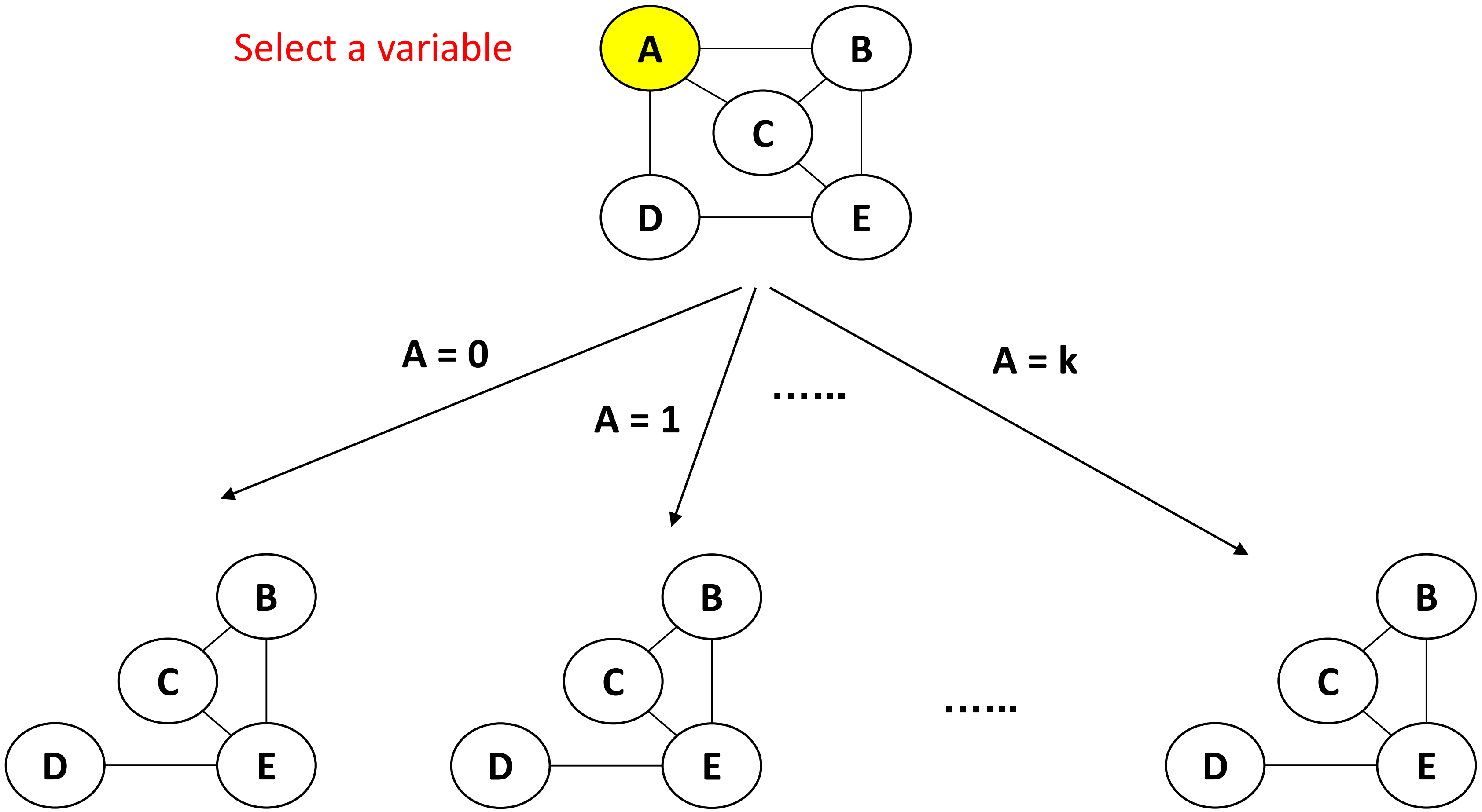
Select a variable



Hybrid: Cutset-Conditioning

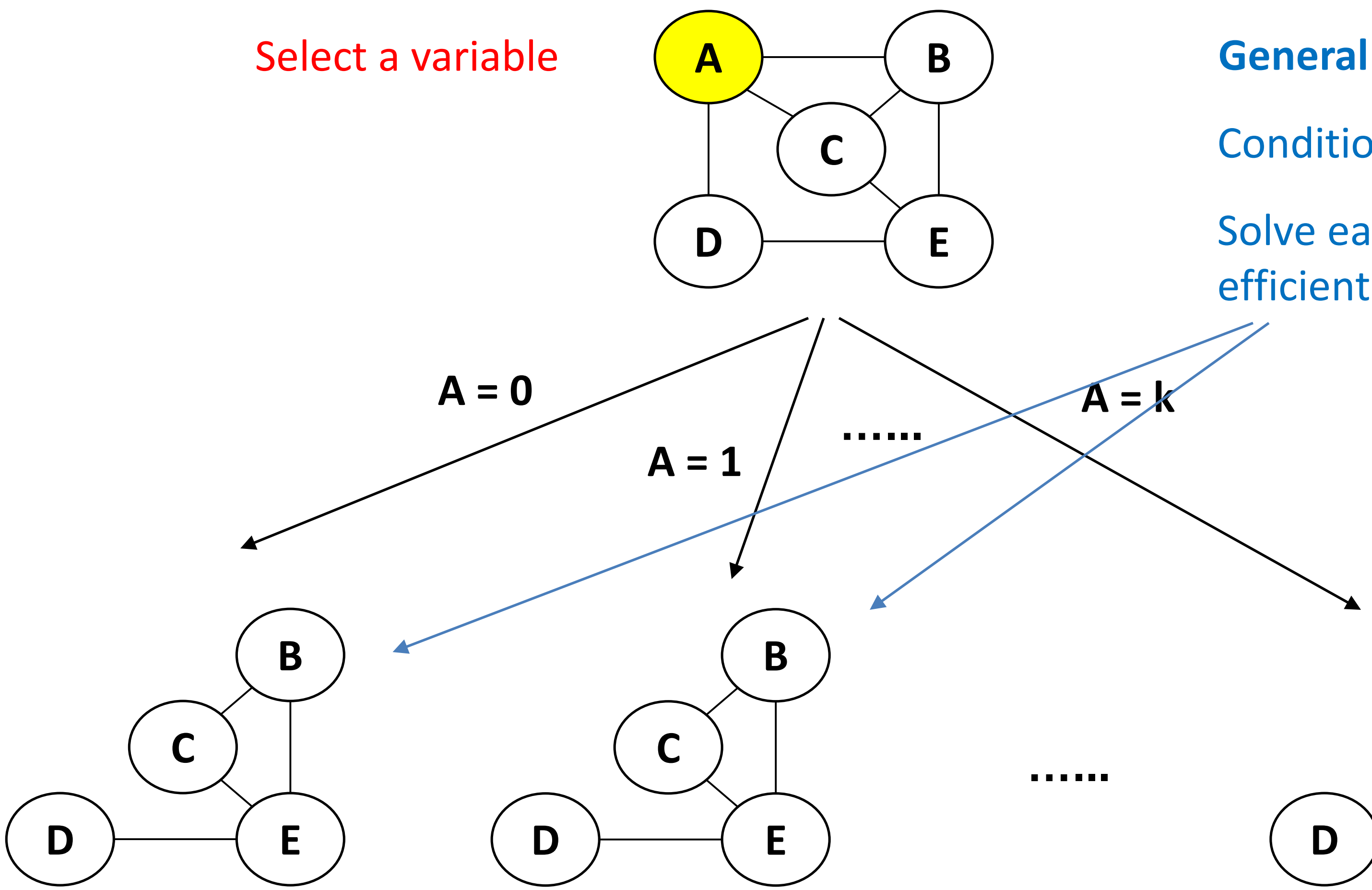
Variable Branching by Conditioning

Select a variable



Hybrid: Cutset-Conditioning

Variable Branching by Conditioning



General principle:

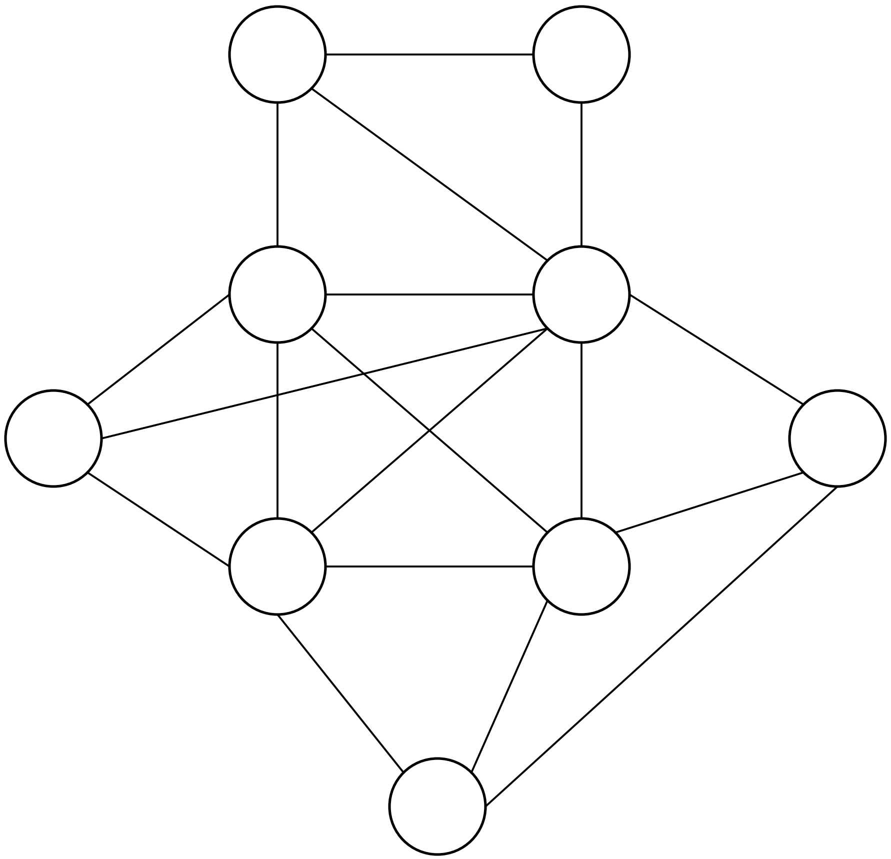
Condition until tractable

Solve each sub-problem efficiently

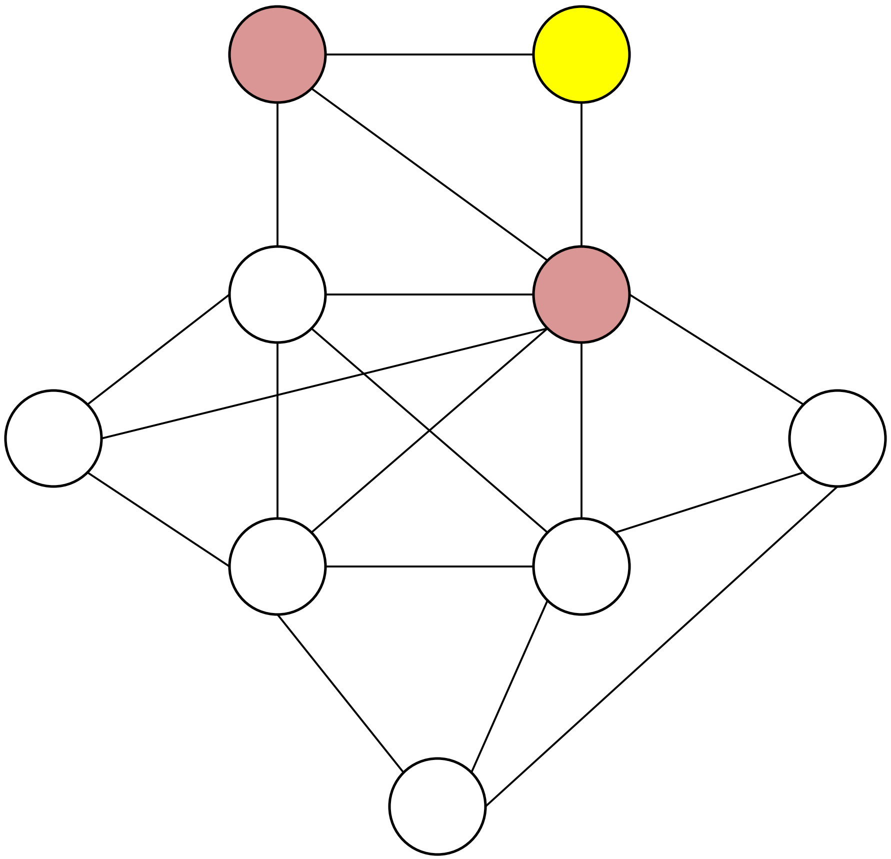
Hybrids Variants

- **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))

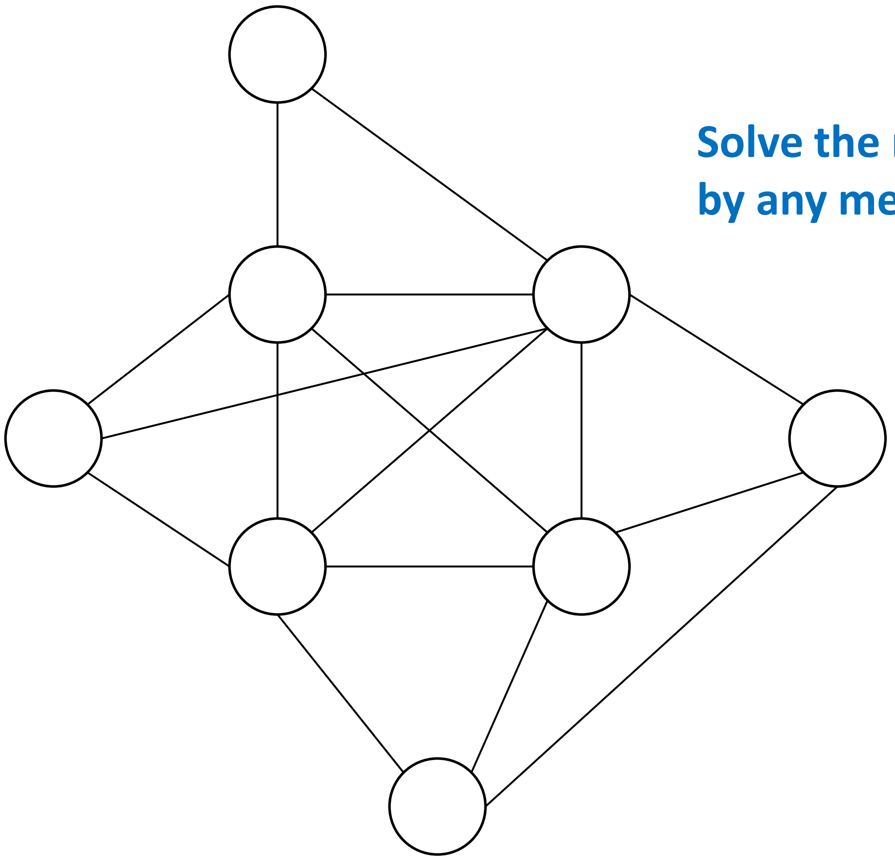
Eliminate First



Eliminate First

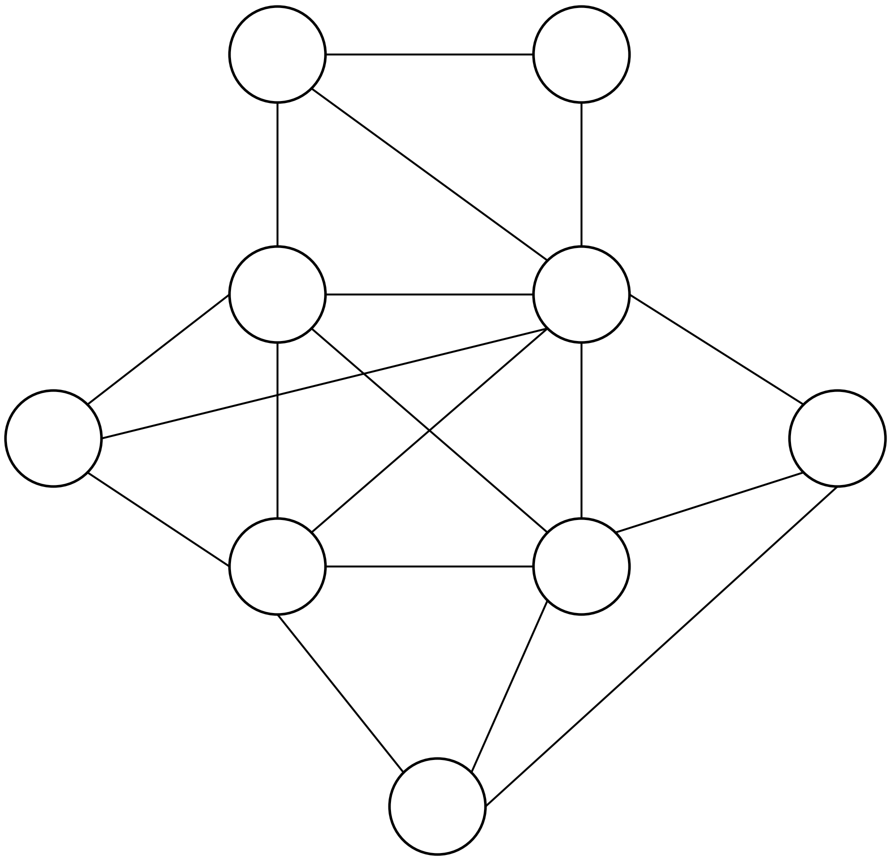


Eliminate First



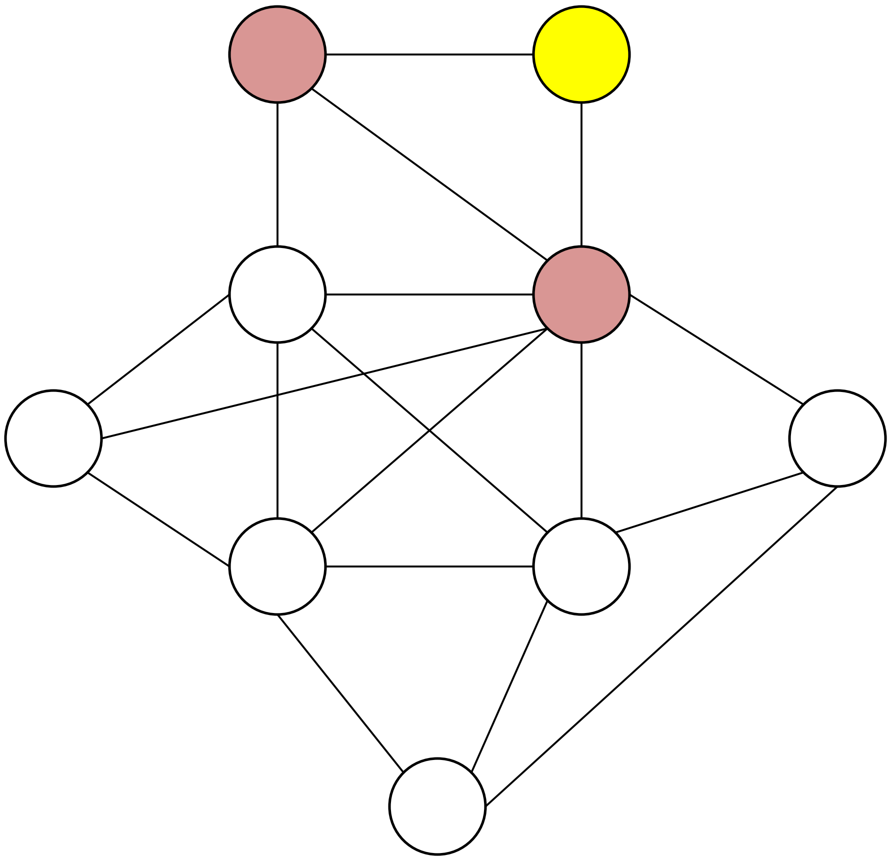
Solve the rest of the problem by any means

Alternate Conditioning and Elimination



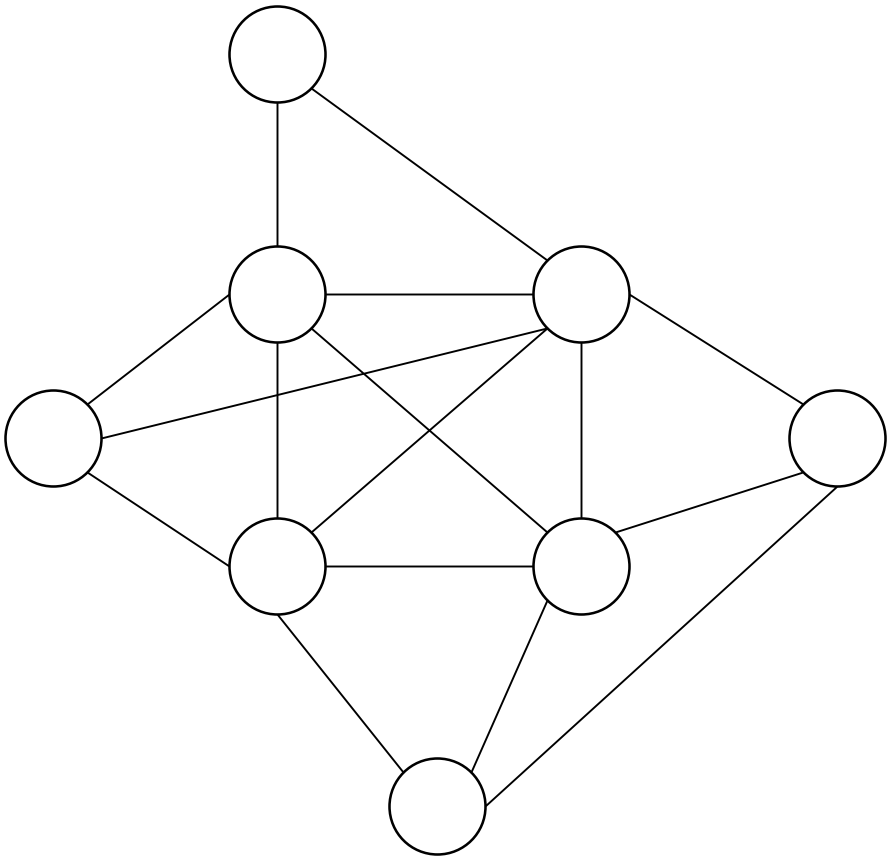
[Larrosa and Dechter, 2002]

Alternate Conditioning and Elimination



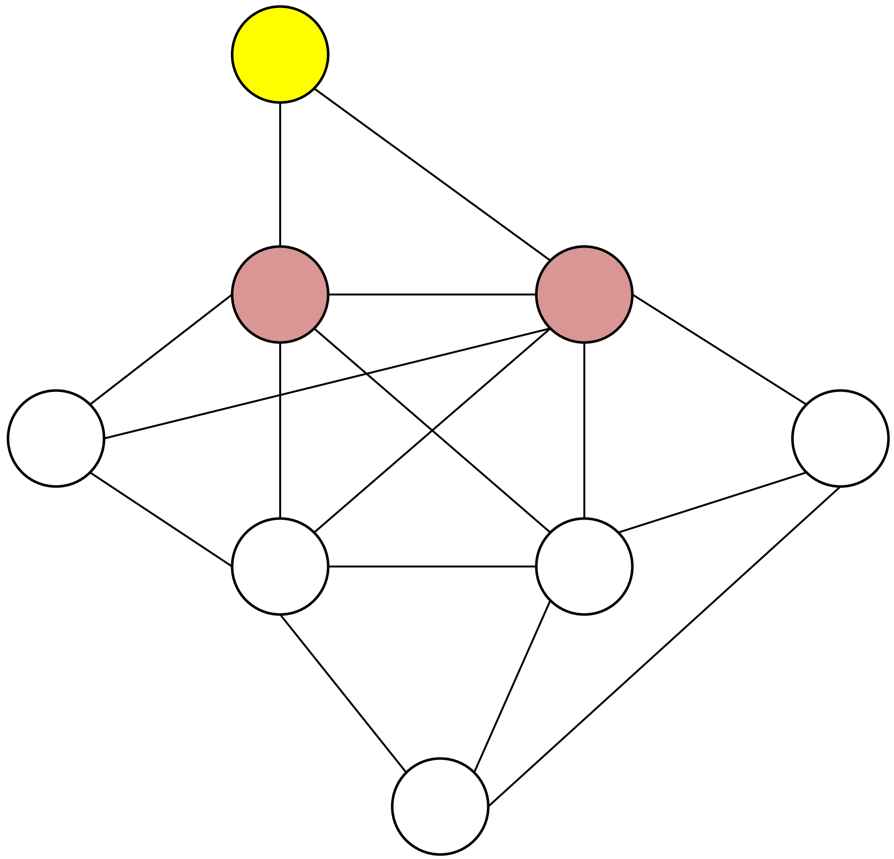
[Larrosa and Dechter, 2002]

Alternate Conditioning and Elimination



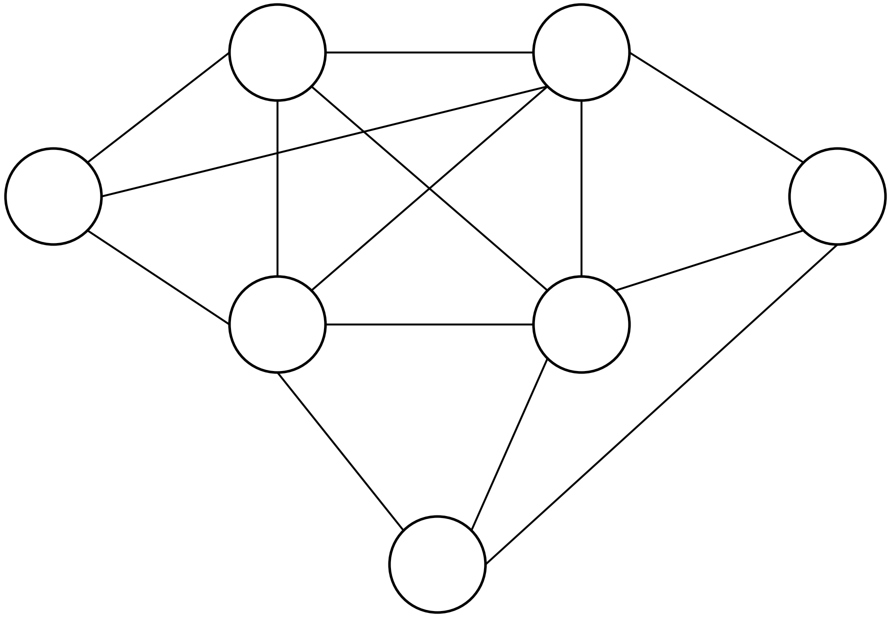
[Larrosa and Dechter, 2002]

Alternate Conditioning and Elimination



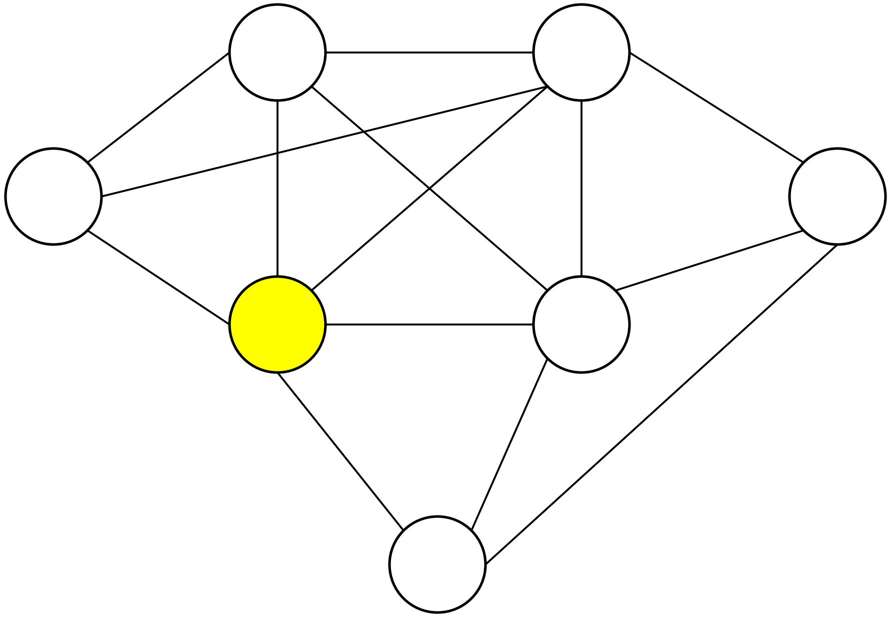
[Larrosa and Dechter, 2002]

Alternate Conditioning and Elimination

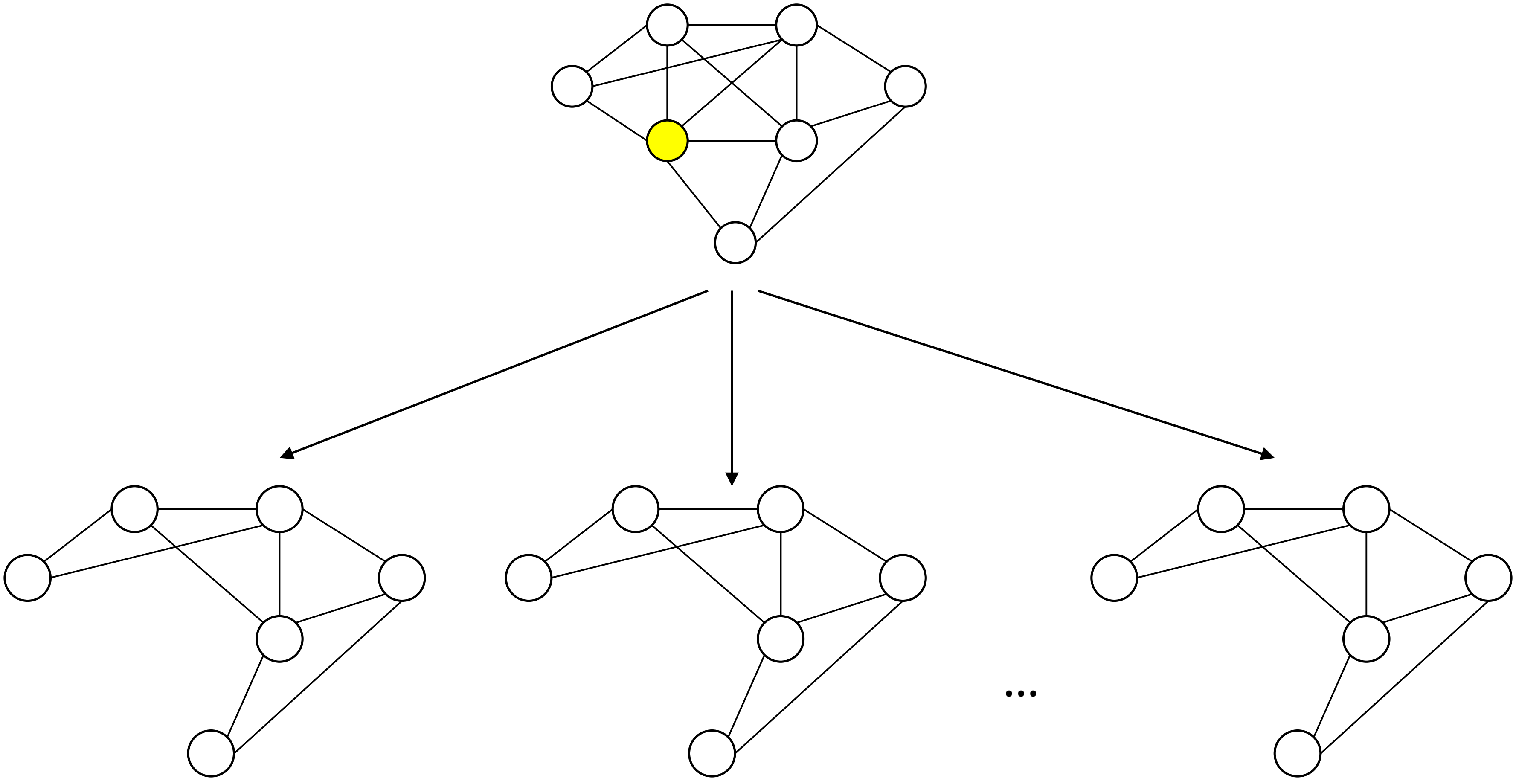


[Larrosa and Dechter, 2002]

Alternate Conditioning and Elimination

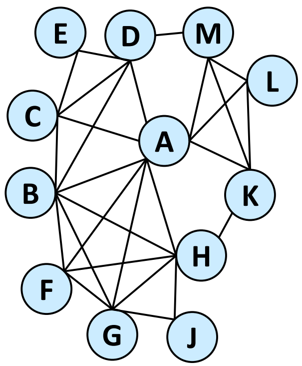


Alternate Conditioning and Elimination

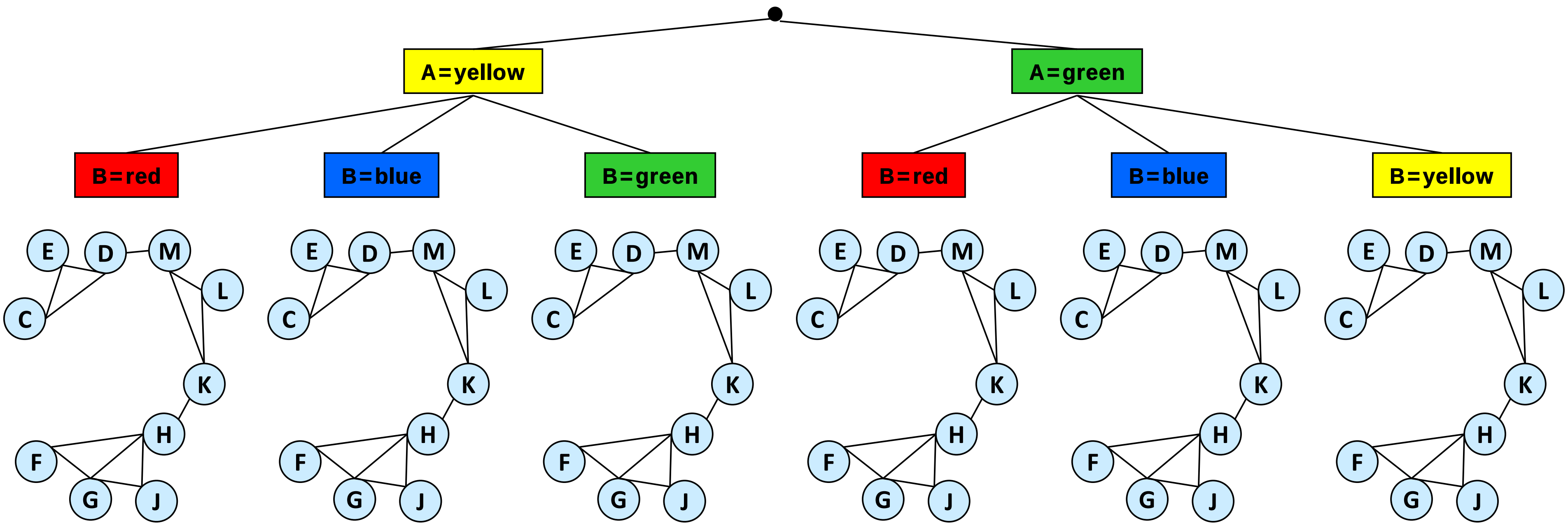


OR w-Cutset

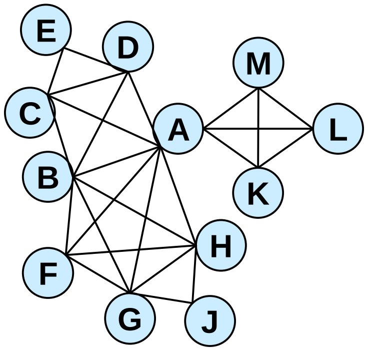
Graph Coloring problem



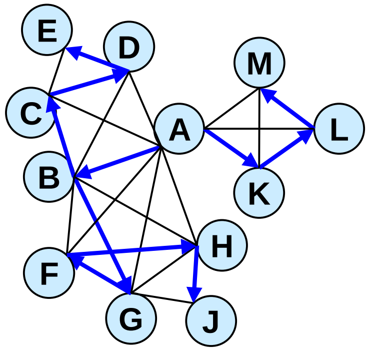
- Inference may require too much memory
- **Condition** on some of the variables



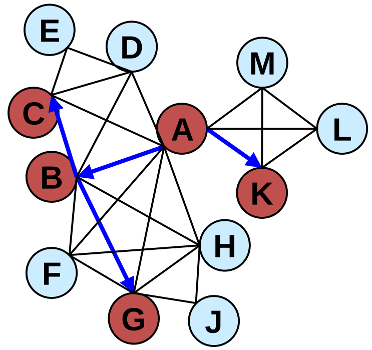
AND/OR w-cutset



grahpical model

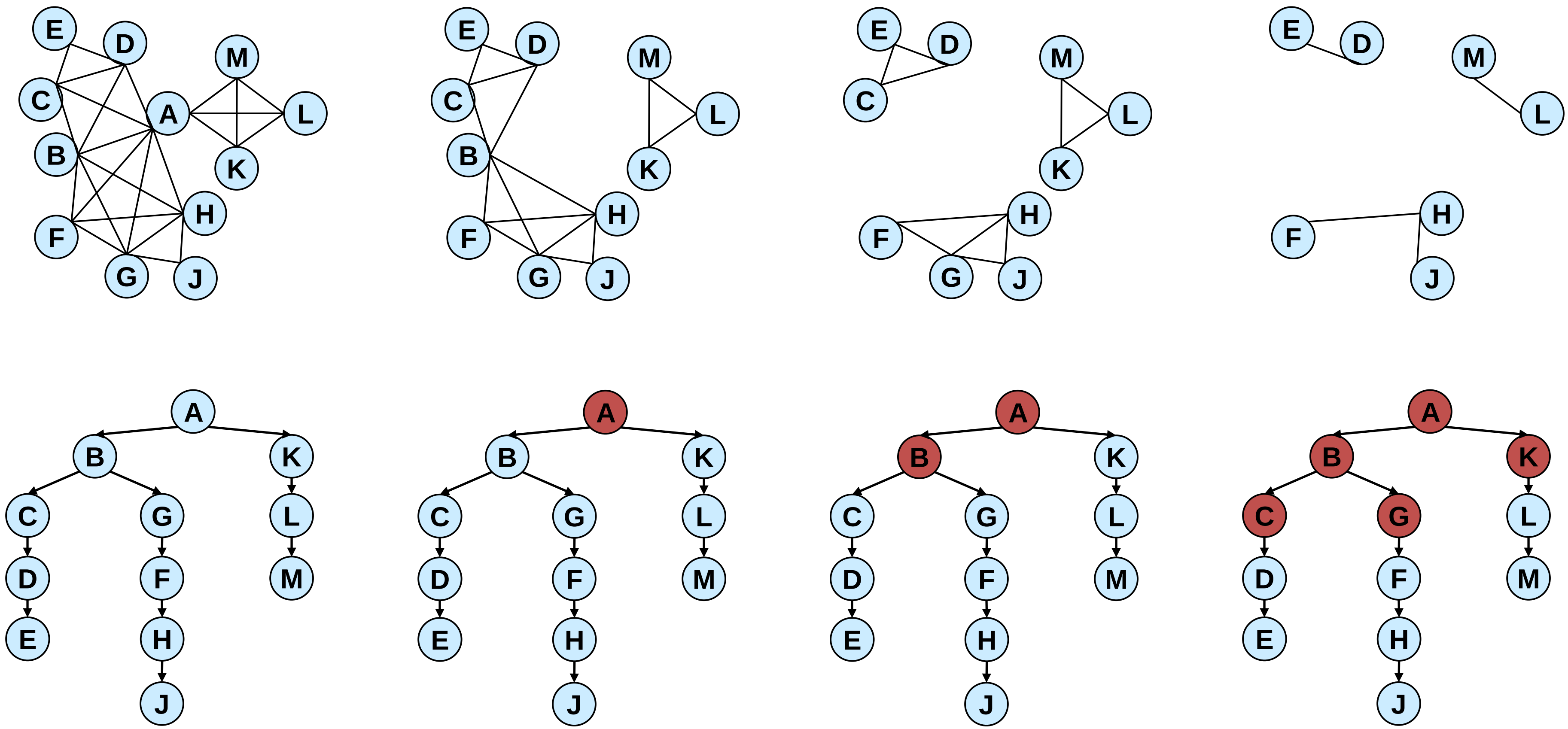


pseudo tree



1-cutset tree

AND/OR w-cutset



3-cutset

2-cutset

1-cutset

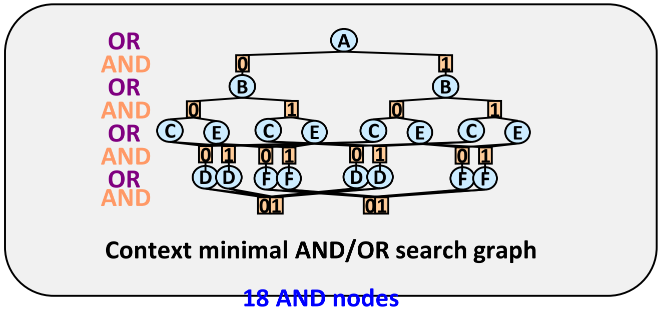
Summary: AND/OR Cutset-Conditioning

- Trade memory for time.
- We never improve time: cycle-cutset size is larger of equal to treewidth+1
- Sometime we do not worsen the time and memory can be much better (e.g., when the induced-width is high)

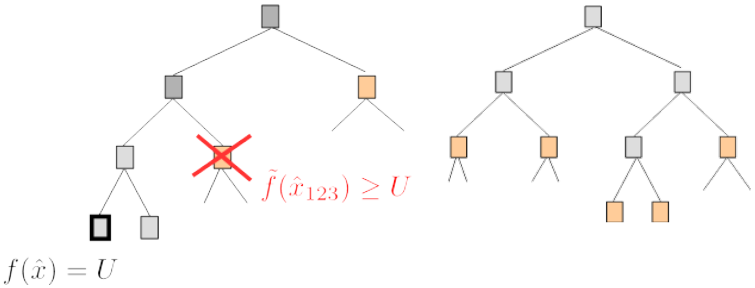
Road Map: Search

- Review Graphical Modes

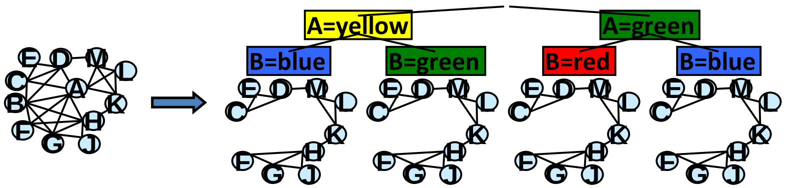
- AND/OR search spaces, pseudo-trees
  - AND/OR search trees
  - AND/OR search graphs
  - Generating good pseudo-trees
  - Brute-force search



- Heuristic search for AND/OR spaces
  - Depth-first AND/OR branch and bound
  - Best-first AND/OR search
  - The Guiding MBE heuristic
  - Marginal Map (max-sum-product)



- Hybrids of search and Inference
- **Summary and Class 2**



Software

- **aolib**
  - http://graphmod.ics.uci.edu/group/Software

  (standalone AOBB, AOBF solvers)
- **daoopt**
  - https://github.com/lotten/daoopt

  (distributed and standalone AOBB solver)
- **merlin**
  - https://developer.ibm.com/open/merlin

  (standalone WMB, AOBB, AOBF, RBFAOO solvers)

  open source, BSD license

UAI Probabilistic Inference Competitions

- **2006**

(aolib)

- **2008**

(aolib)

- **2012**

(daoopt)

- **2014**

(daoopt)

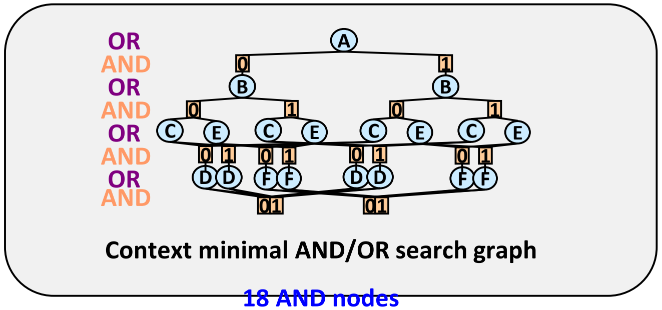
(daoopt)

(merlin)

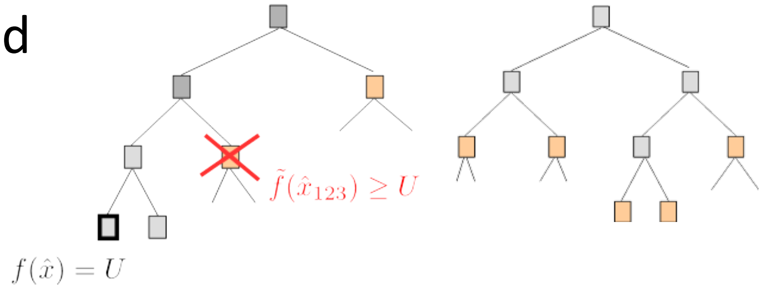
Marginal Map

Summary of Search

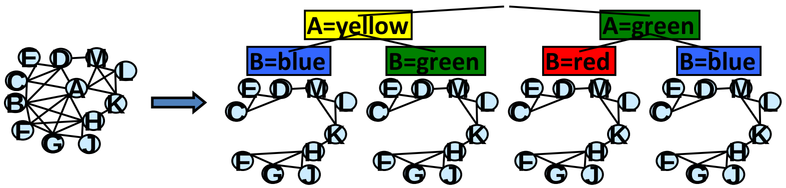
- AND/OR search spaces, pseudo-trees
  - AND/OR search trees
  - AND/OR search graphs
  - Generating good pseudo-trees
  - Brute-force search



- Heuristic search for AND/OR spaces
  - Depth-first AND/OR branch and bound
  - Best-first AND/OR search
  - The Guiding MBE heuristic
  - Marginal Map (max-sum-product)

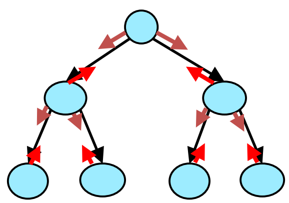


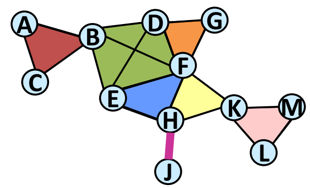
- Hybrids of search and Inference
- **Summary and Class 2**

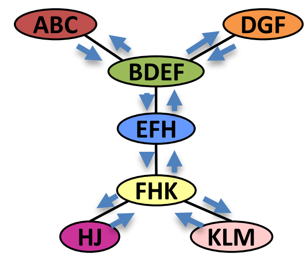


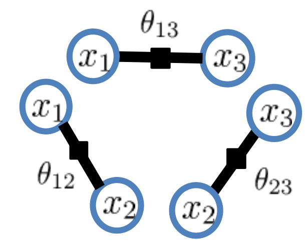
Preview of Class 3:

- Class 1: Introduction and Inference

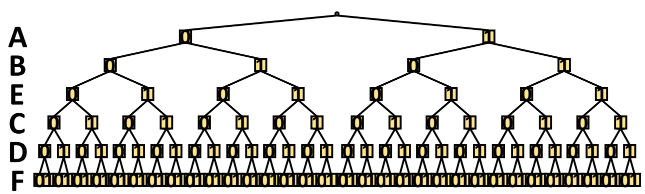


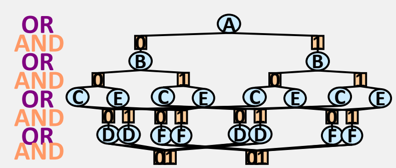






- Class 2: Search





Context minimal AND/OR search graph

- Class 3: Variational Methods and Monte-Carlo Sampling