Hybrid of search and inference: time-space tradeoffs

chapter 10

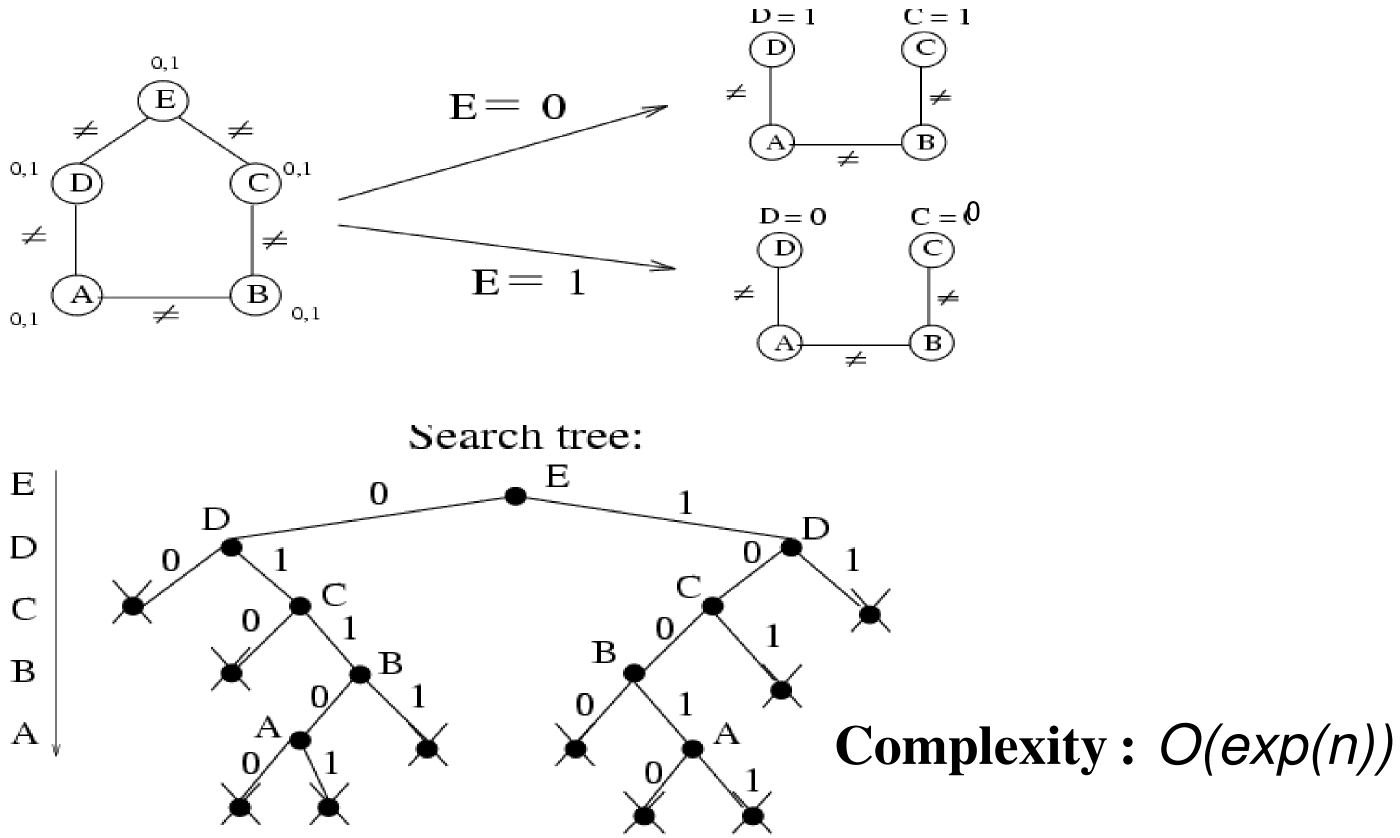
Reasoning Methods

- Our focus - search and elimination
- Search

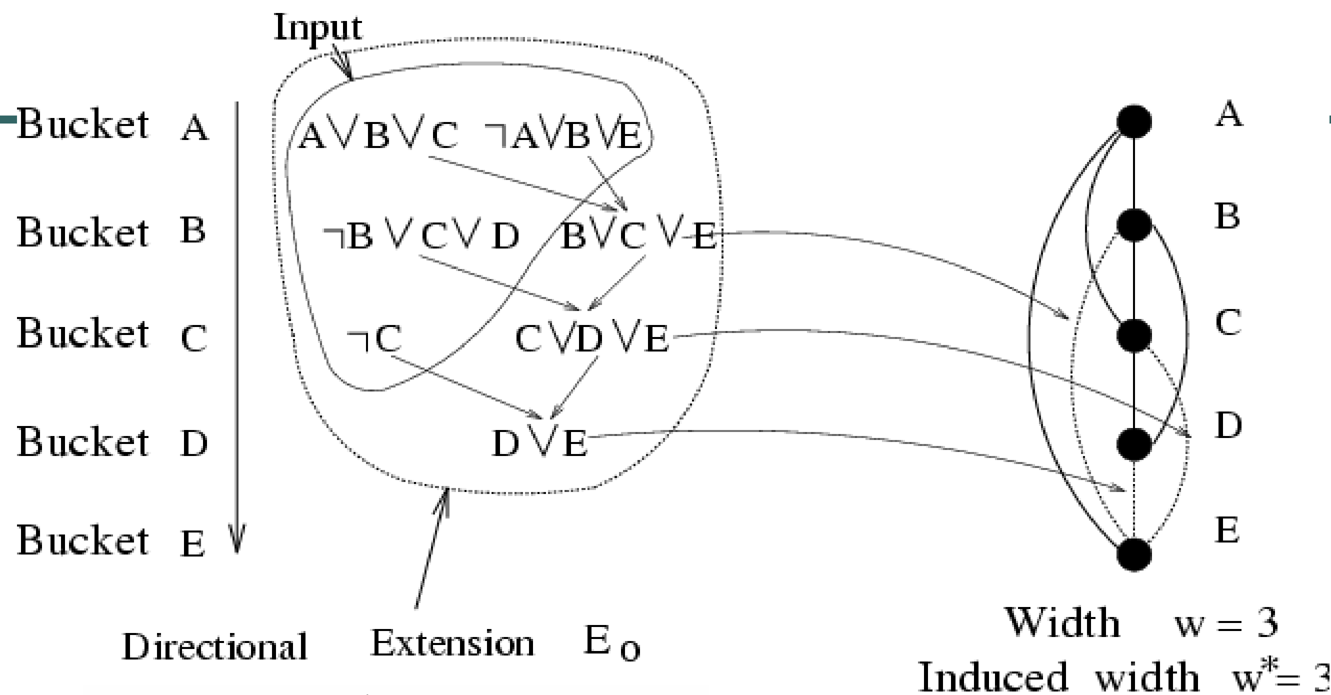
  (“guessing” assignments, reasoning by assumptions)
  - Branch-and-bound (optimization)
  - Backtracking search (CSPs)
  - Cycle-cutset (CSPs, belief nets)
- Variable elimination

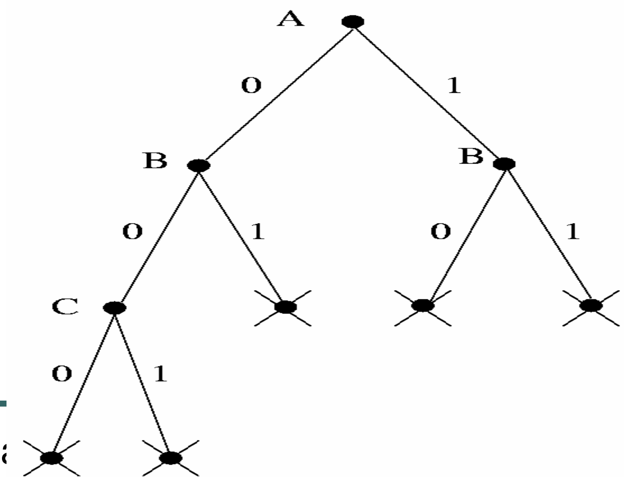
  (inference, “propagation” of constraints, probabilities, cost functions)
  - Dynamic programming (optimization)
  - Adaptive consistency (CSPs)
  - Joint-tree propagation (CSPs, belief nets)

Search:
Backtracking Search



Satisfiability: Inference vs search



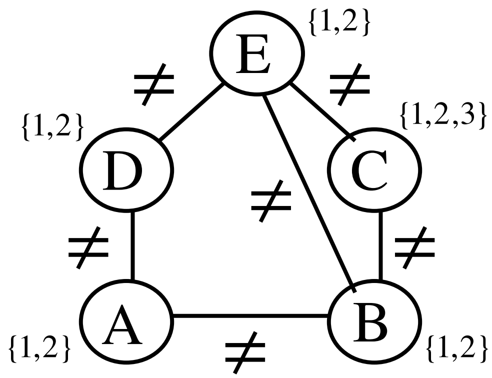


$$|bucket_i| = O(\exp(w^*))$$

$$\text{DR time and space}: O(n\exp(w^*))$$

Search = O(exp(n))

Bucket Elimination



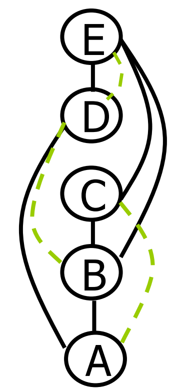
$Bucket(E)$: $\mathrm{E} \neq \mathrm{D}$, $\mathrm{E} \neq \mathrm{C}$, $\mathrm{E} \neq \mathrm{B}$

$Bucket(D)$: $\mathrm{D} \neq \mathrm{A}$ || R_{DCB}

$Bucket(C)$: $\mathrm{C} \neq \mathrm{B}$ || R_{ACB}

$Bucket(B)$: $\mathrm{B} \neq \mathrm{A}$ || R_{AB}

$Bucket(A)$: R_A



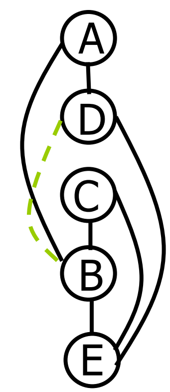
$Bucket(A)$: $\mathrm{A} \neq \mathrm{D}$, $\mathrm{A} \neq \mathrm{B}$

$Bucket(D)$: $\mathrm{D} \neq \mathrm{E}$ || R_{DB}

$Bucket(C)$: $\mathrm{C} \neq \mathrm{B}$, $\mathrm{C} \neq \mathrm{E}$

$Bucket(B)$: $\mathrm{B} \neq \mathrm{E}$ || $R^{D}{}_{BE}$, $R^{C}{}_{BE}$

$Bucket(E)$: || R_E

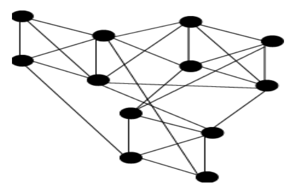


Complexity : $O(n\, exp(w^*(d)))$,

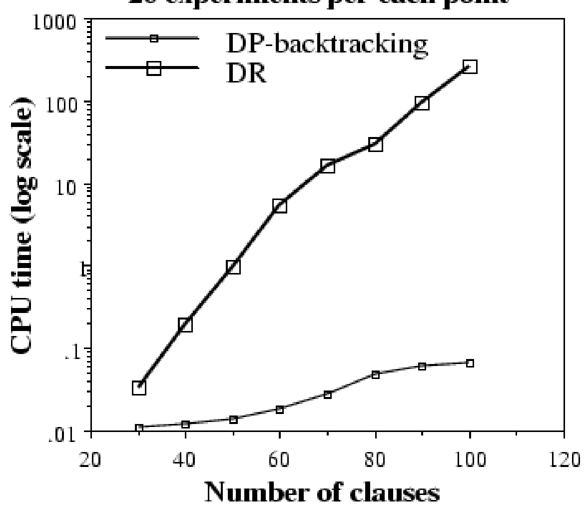
$w^*(\mathrm{d})$ - ***induced width along ordering d***

DR versus DPLL: complementary properties

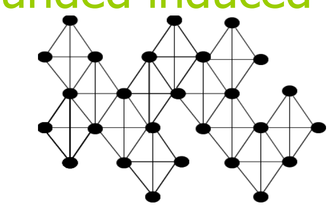
Uniform random 3-CNFs
(large induced width)



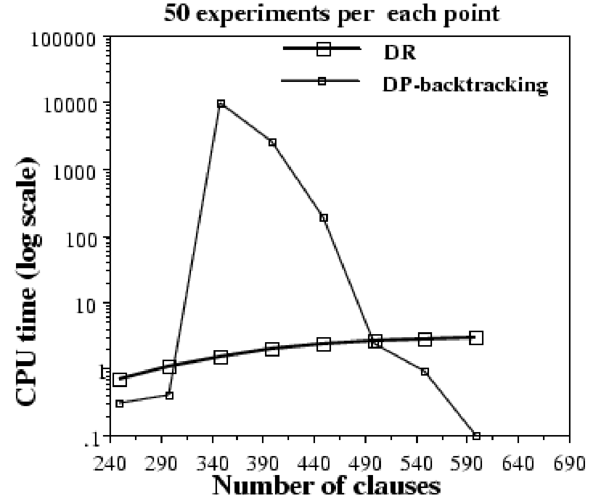
UNIFORM 3-CNFS
20 variables
20 experiments per each point



(k,m)-tree 3-CNFs
(bounded induced width)



DR vs. DP-backtracking
3-CNF CHAINS
25 subtheories, 5 variables in each
50 experiments per each point

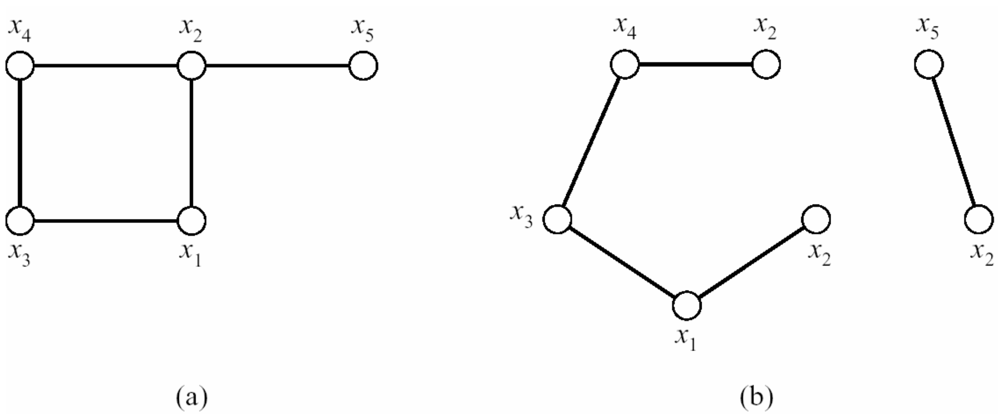


Exact CSP techniques: complexity

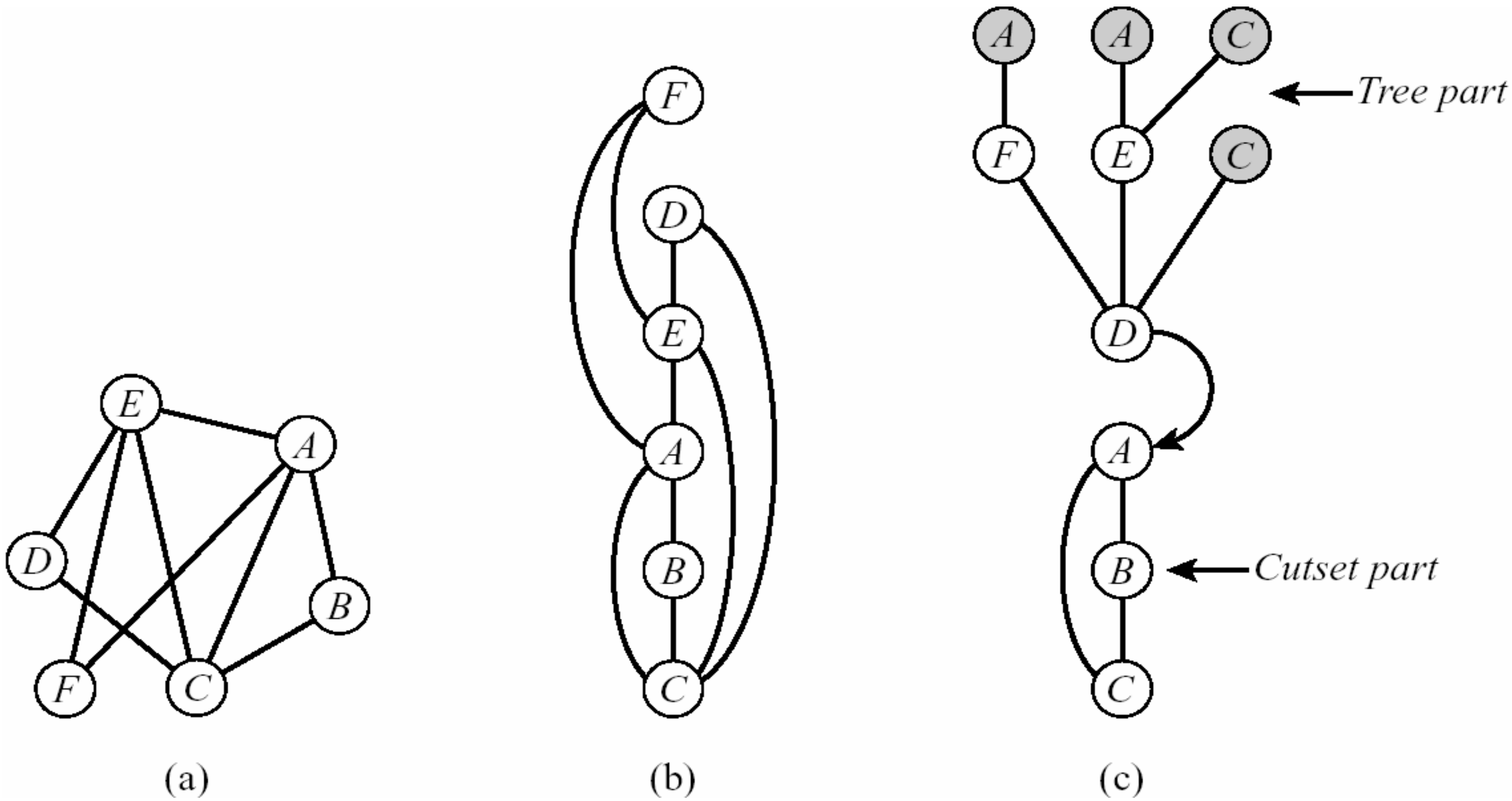
	Backtracking	Elimination
Worst-case time	O(exp(n))	O(n exp(w*)) $w^* \leqslant n$
Average time	better than worst-case	same as worst-case
Space	O(n)	O(n exp(w*)) $w^* \leqslant n$
Output	one solution	knowledge compilation

The cycle-cutset effect

- A cycle-cutset is a subset of nodes in an undirected graph whose removal results in a graph with no cycles
- An instantiated variable cuts the flow of information: cuts a cycle.
- If a cycle-cutset is instantiated the remaining problem is a tree and can be solved efficiently



Demonstrating cycle-cutset scheme



Complexity of the cycle-cutset scheme

Theorem: Algorithm cycle-cutset decomposition has time complexity of $O((n-c)k^{(c+2)})$ where n is the number of variables, c is the cycle-cutset size and k is the domain size. The space complexity of the algorithm is linear.

Recursive-search:
a linear space search guided by a tree-decomposition

- Given a tree network, we identify a node x_1 which, when removed, generates two subtrees of size $n/2$ (approximately).

- T_n is the time to solve a binary tree starting at x_1. T_n obeys recurrence
  - $T_n = k\, 2\, T_{n/2}$, $T_1 = k$
- We get:
  - $T_n = n\, k^{\log n + 1}$

- Given a tree-decomposition having induced-width w* this generalizes to recursive conditioning of tree-decompositions:
  - $T_n = n\, k^{(w^*+1) \log n}$
- because the number of values k is replaced by the number of tuples k^{w^*}

Alternative views of recursive-search

- **Proposition 1**: Given a constraint network R= (X,D,C), having graph G, a tree-decomposition T = (X, chi,Psi) that has induced-width w*, having diameter r (the longet path from cluster leaf to cluster leaf, then there exists a DFS tree dfs(T) whose depth is bounded by O(log r w*).
- **Proposition 2:** Recursive-conditioning along a tree-decomposition T of a constraint problem *R= (X,D,C),* having induced-width w*, is identical to backjumping along the DFS ordering of its corresponding dfs(T).
- **Proposition 3:** Recursive-conditioning is a depth-first search traversal of the AND/OR search tree relative to the DFS spanning tree dfs(T).

Example

- Consider a chain graph or a k-tree.

Hybrid: conditioning first

- Generalize cycle-cutset: condition of a subset that yield a bounded inference problem, not necessarily linear.

- **b-cutset**: a subset of nodes is called a b-cutset iff when the subset is removed the resulting graph has an induced-width less than or equal to b. A **minimal b-cutset** of a graph has a smallest size among all b-cutsets of the graph. A cycle-cutset is a 1-cutset of a graph.

- Adjusted induced-width: The adjusted induced-width with of G respect to V is the induced-width of G after the variable set V is removed.

Elim-cond(b)

- **Idea:** runs backtracking search on the b-cutset variables and bucket-elimination on the remaining variables.
- **Input:** A constraint network R = (X,D,C), Y a b-cutset, d an ordering that starts with Y whose adjusted induced-width, along d, is bounded by b, Z = X-Y.
- **Output:** A consistent assignment, if there is one.
- 1. **while** {y} ← next partial solution of Y found by backtracking, **do**
  - a) z ← solution found by adaptive-consistency(R_y).
  - B) **if** z is not *false*, return solution (y,z).
- 2. endwhile.
- **return**: the problem has no solutions.

Complexity of elim-cond(b)

- **Theorem:** Given *R*= (X,D,C), if elim-cond(b) is applied along ordering d when Y is a b-cutset then the space complexity of elim-cond(b) is O(n exp(b)), and its time complexity is O(n exp (|Y|+b)).

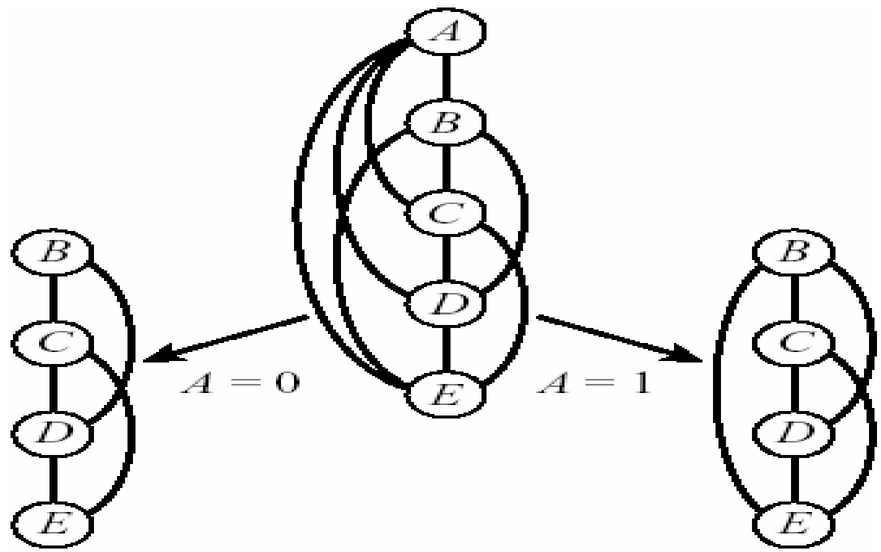
Finding a b-cutset

- Verifying a b-cutset can be done in polynomial time
- A simple greedy: use a good induced-width ordering and starting at the top add to the b-cutset any variable with more than b parents.
- Alternative: generate a tree-decomposition, then select a b-cutset that reduce each cluster below b.

Time-space tradeoff using b-cutset

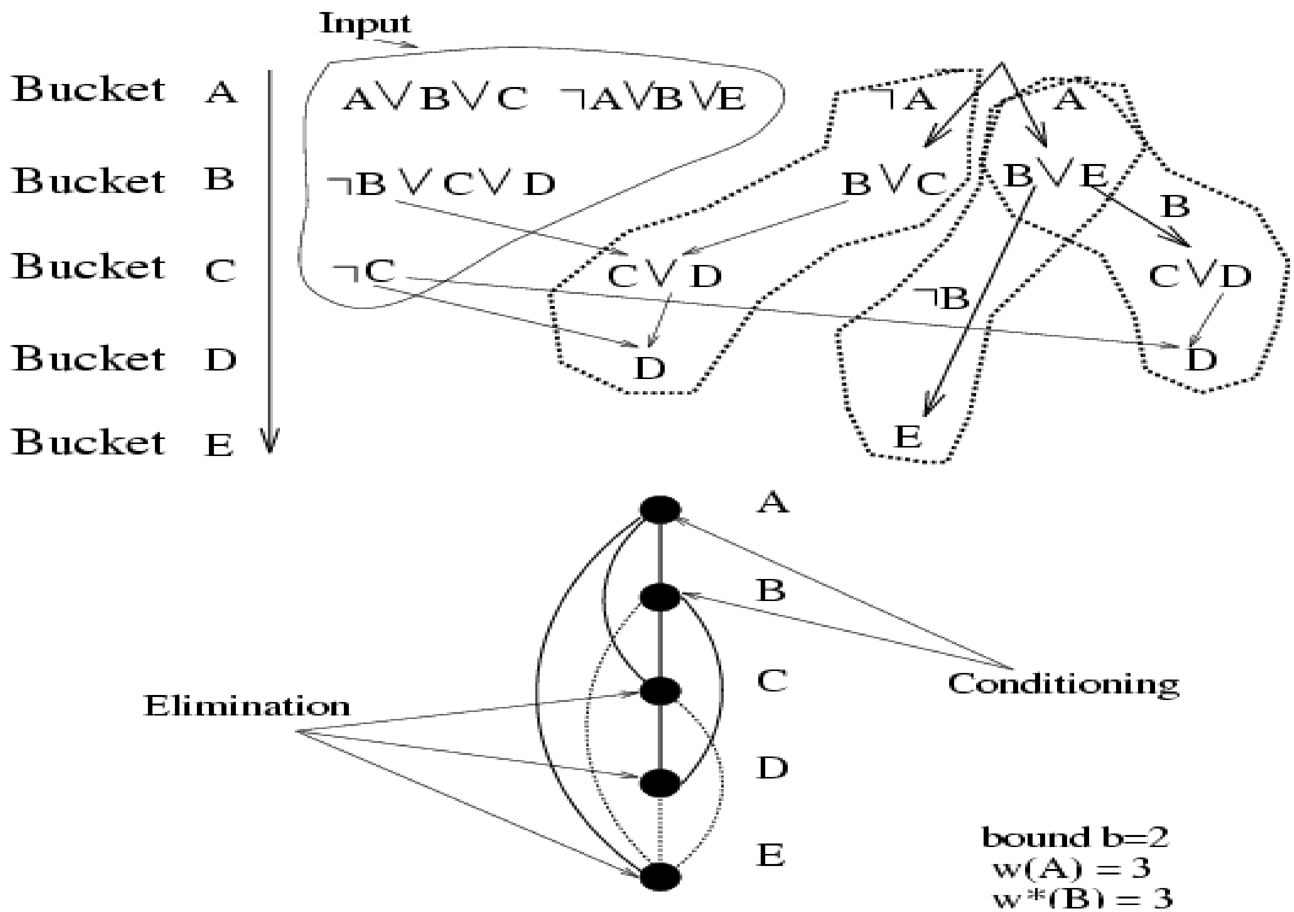
- There is no guaranteed worst-case time improvement of elim-cond(b) over pure bucket-elimination.
- The size of the smallest cycle-cutset (1-cutset), c_1 and the smallest induced width, w^*, obey:
  - $c_1 \geq w^* - 1$. Therefore, $1 + c_1 \geq w^*$, where the left side of this inequality is the exponent that determines time complexity of elim-cond(b=1), while w^* governs the complexity of bucket-elimination.
- $c_i - c_{(i+1)} \geq 1$
- $1+c_1 \geq 2+c_2 \geq \ldots b+c_b, \ldots \geq w^*+c_{w^*} = w^*$
- We get a hybrid scheme whose time complexity decreases as its space increases until it reaches the induced-width.

Example of conditioning on A



- Consider the theory:
- (~C v E)(A v B v C v D)(~A v B v E v D)(B v C v D)

Resolve if $w^*(x_i) < b$, condition otherwise

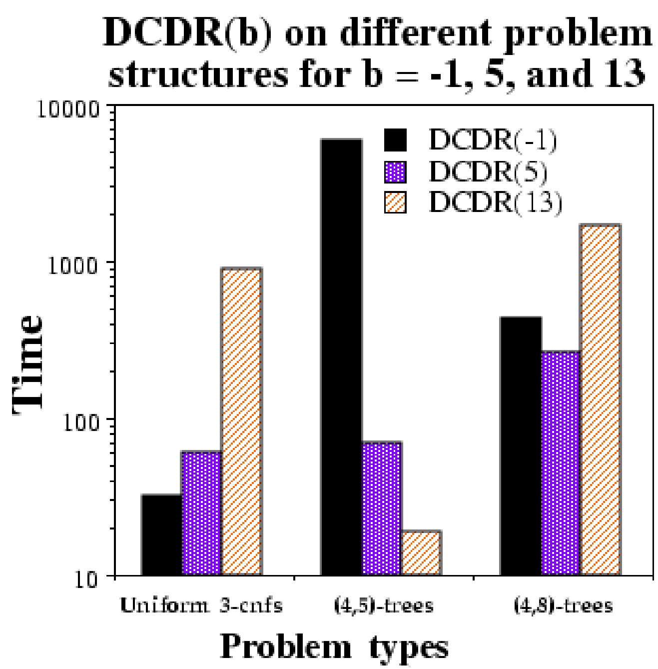


DCDR(b): empirical results

$$\text{Adjustable trade-off}:$$

$$b < 0: \text{ pure DPLL}, \ b \geq w^*: \text{ pure DR}, 0 \leq b \leq w^*: \text{hybrid}$$

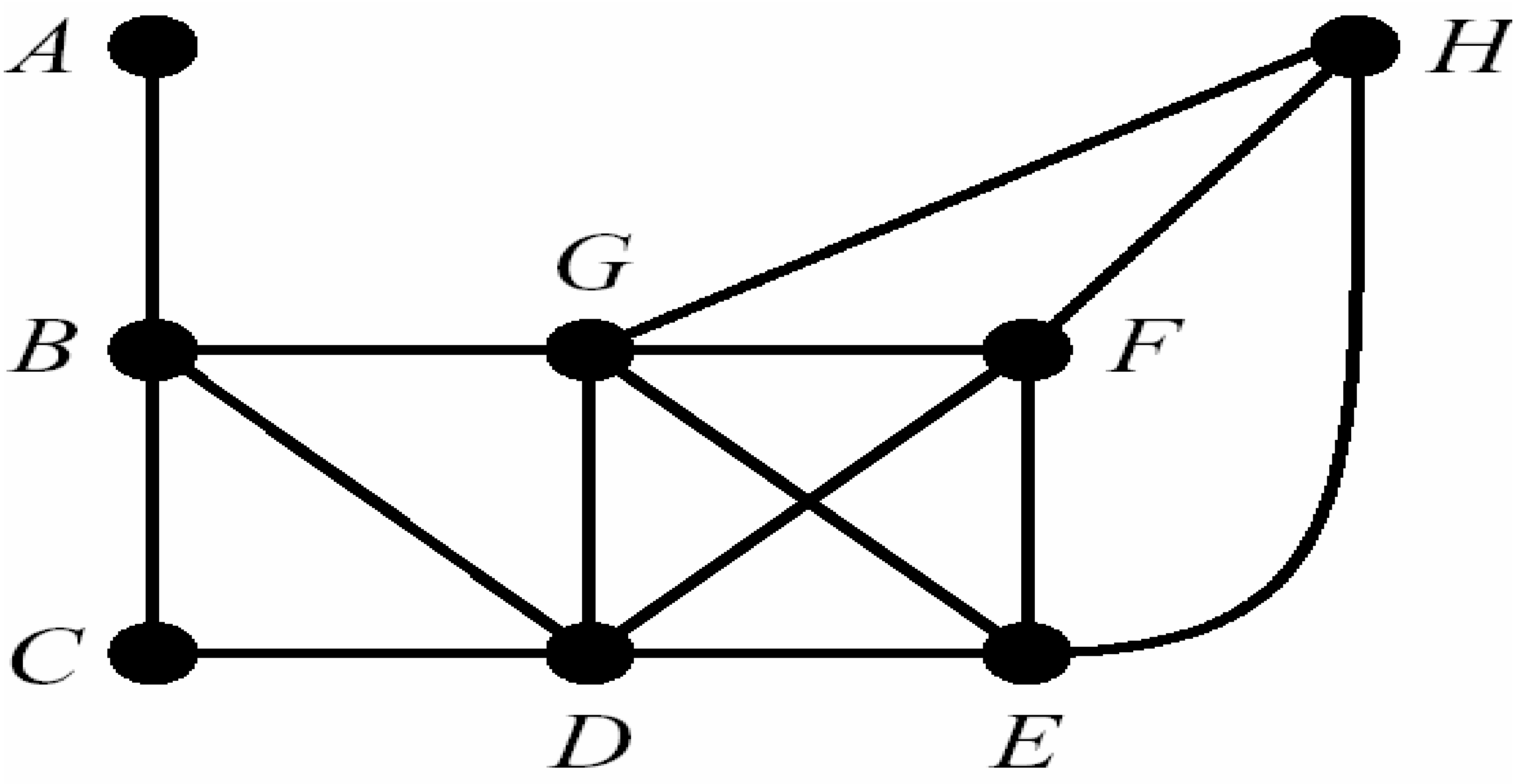
$$\text{Time } \exp(b + c\_b), \text{space } \exp(b)$$



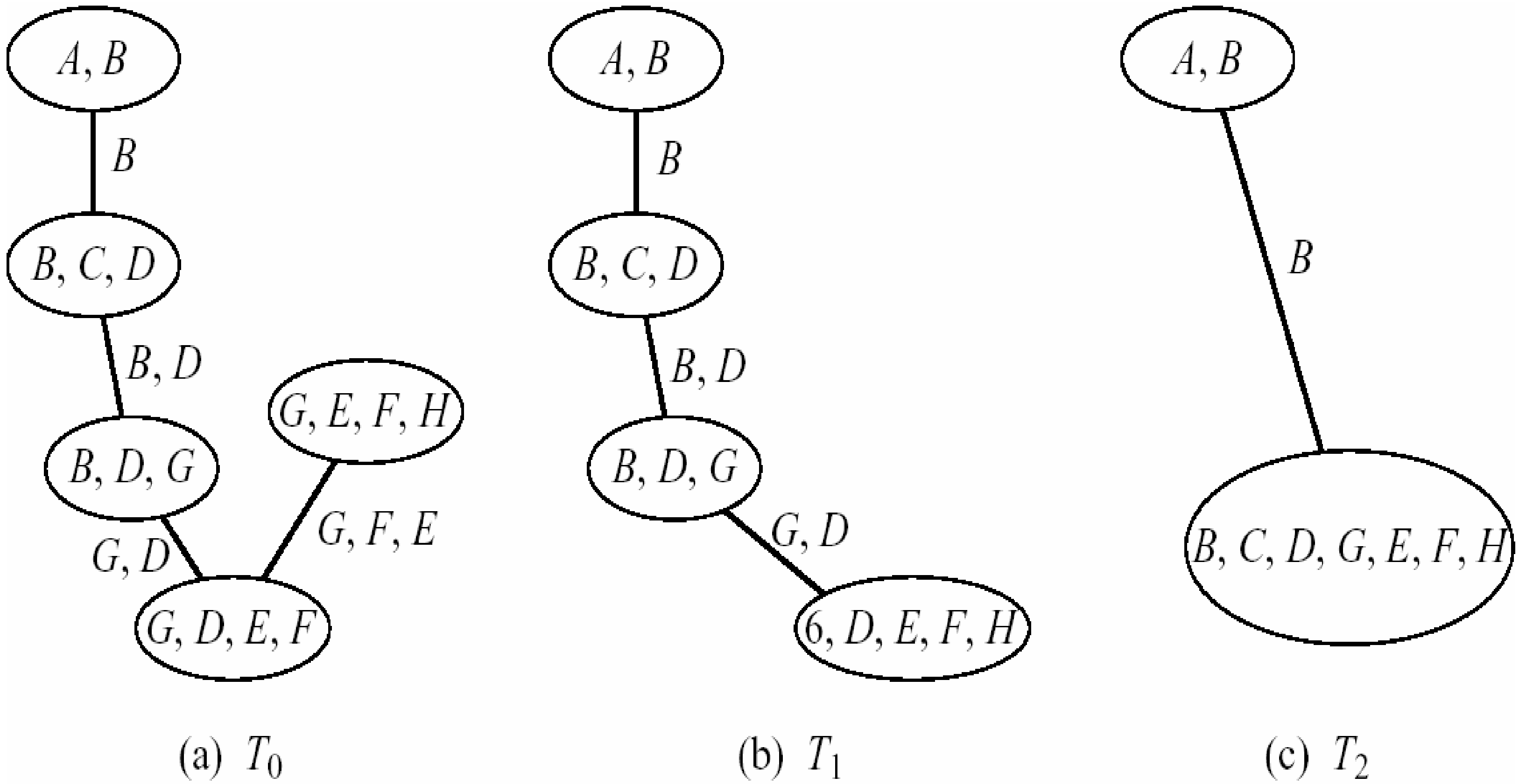
Hybrid, inference first: The super cluster tree elimination

- Algorithm CTE is time exponential in the cluster size and space exponential in the separator size.
- Trade space for time by increasing the cluster size and decreasing the separator sizes.
- Join clusters with fat separators.

Example



A primary and secondary tree-decompositions



Sep-based time-space tradeoff

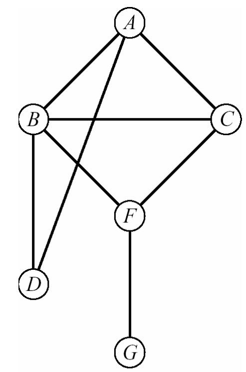
- Let *T* be a tree-decomposition of hypergraph *H*. Let $s_0, s_1, \cdots, s_n$ be the sizes of the separators in *T*, listed in strictly descending order. With each separator size s_i we associate a secondary tree decomposition T_i, generated by combining adjacent nodes whose separator sizes are strictly greater than s_j.

- Let r_i the largest set of variables in any cluster of .
- Note that as s_i decreases, r_i increase.

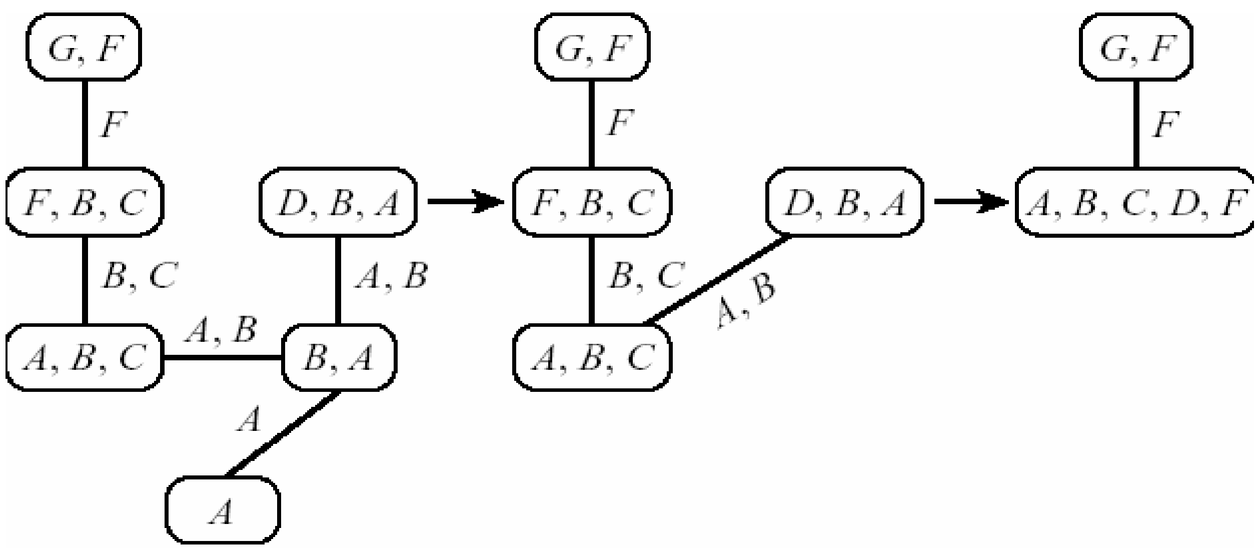
- **Theorem:** The complexity of CTE when applied to each T_i is O(n exp(r_i)) time, and O(n exp(s_i)) space.

Super-buckets

From a bucket-tree to a join-tree to a super-bucket tree



(a)



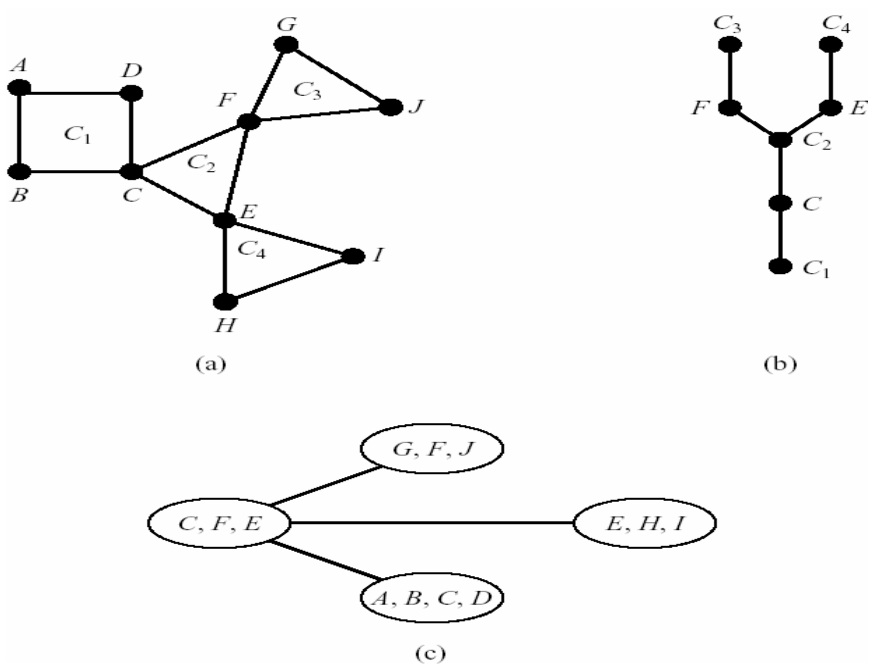
(a) (b) (c)

Nonseparable components: a special case of tree-decomposition

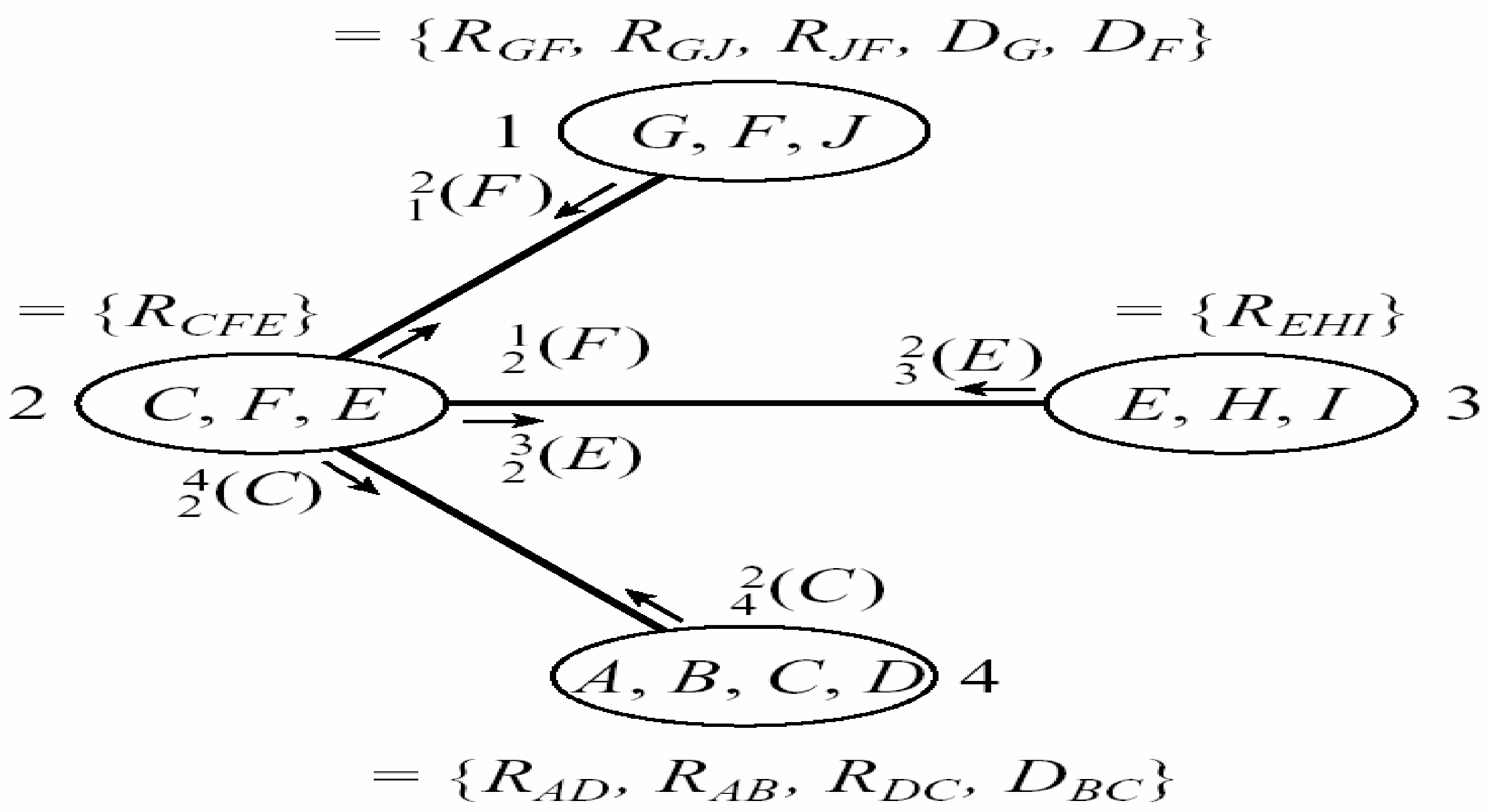
- A connected graph G=(V,E) has a separation node v if there exist nodes a and b such that all paths connecting a and b pass through v.
- A graph that has a separation node is called **separable**, and one that has none is called **non-separable**. A subgraph with no separation nodes is called a **non-separable component** or a bi-connected component.
- A dfs algorithm can find all non-separable components and they have a tree structure

Decomposition into nonseparable components

- Assume a constraint network having unary, binary and ternary constraints :*R* = { R_AD,R_AB, R_DC,R_BC, R_GF,D_G,D_F,R_EHI,R_CFE }.



Executing ATC (Adaptive tree consistency)



Complexity

- **Theorem:** If R = (X,D,C) is a constraint network whose constraint graph has nonseparable components of at most size r, then the super-bucket elimination algorithm, whose buckets are the nonseparable components, is time exponential O(n exp(r)) and is linear in space.

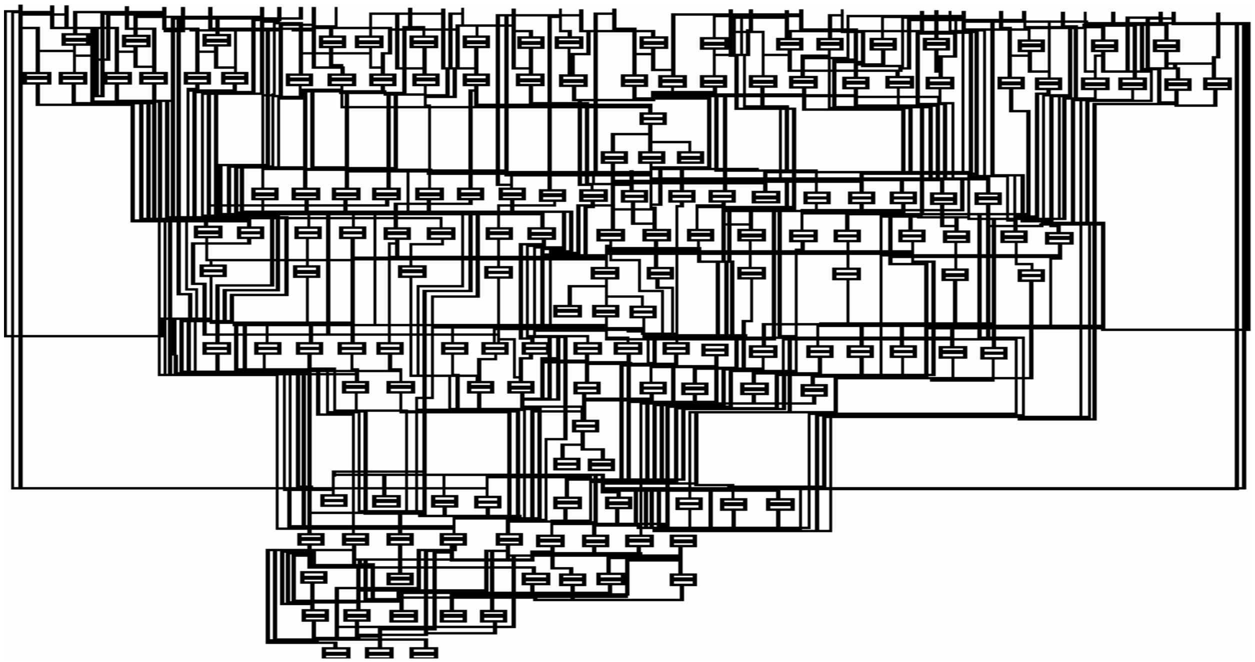
Hybrids of hybrids

- **hybrid(b_1,b_2):**
- First, a tree-decomposition having separators bounded by b_1 is created, followed by application of the CTE algorithm, but each clique is processed by elim-cond(b_2). If c^*_{b_2} is the size of the maximum b_2-cutset in each clique of the b_1-tree-decomposition, the algorithm is space exponential in b_1 but time exponential in c^*_{b_2}.
- Special cases:
  - hybrid(b_1,1): Applies cycle-cutset in each clique.
  - b_1 = b_2. For b=1, hybrid(1,1) is the non-separable components utilizing the cycle-cutset in each component.
- The space complexity of this algorithm is linear but its time complexity can be much better than the cycle-cutsets scheme or the non-separable component approach alone.

Case study: combinatorial circuits: benchmark used for fault diagnosis and testing community

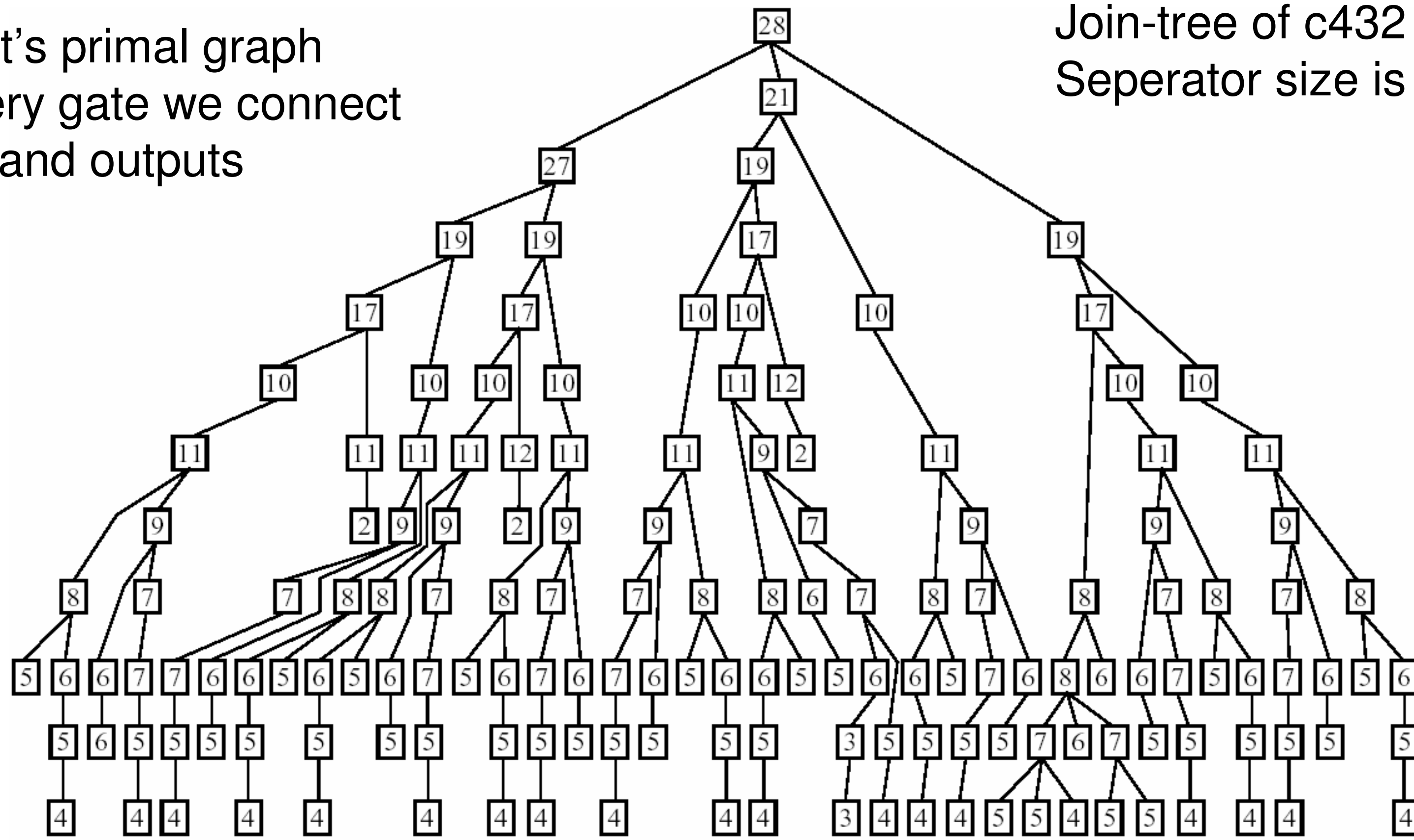
Problem: Given a circuit and its unexpected output, identify faulty components.

The problem can be modeled as a constraint optimization problem and solved by bucket elimination.



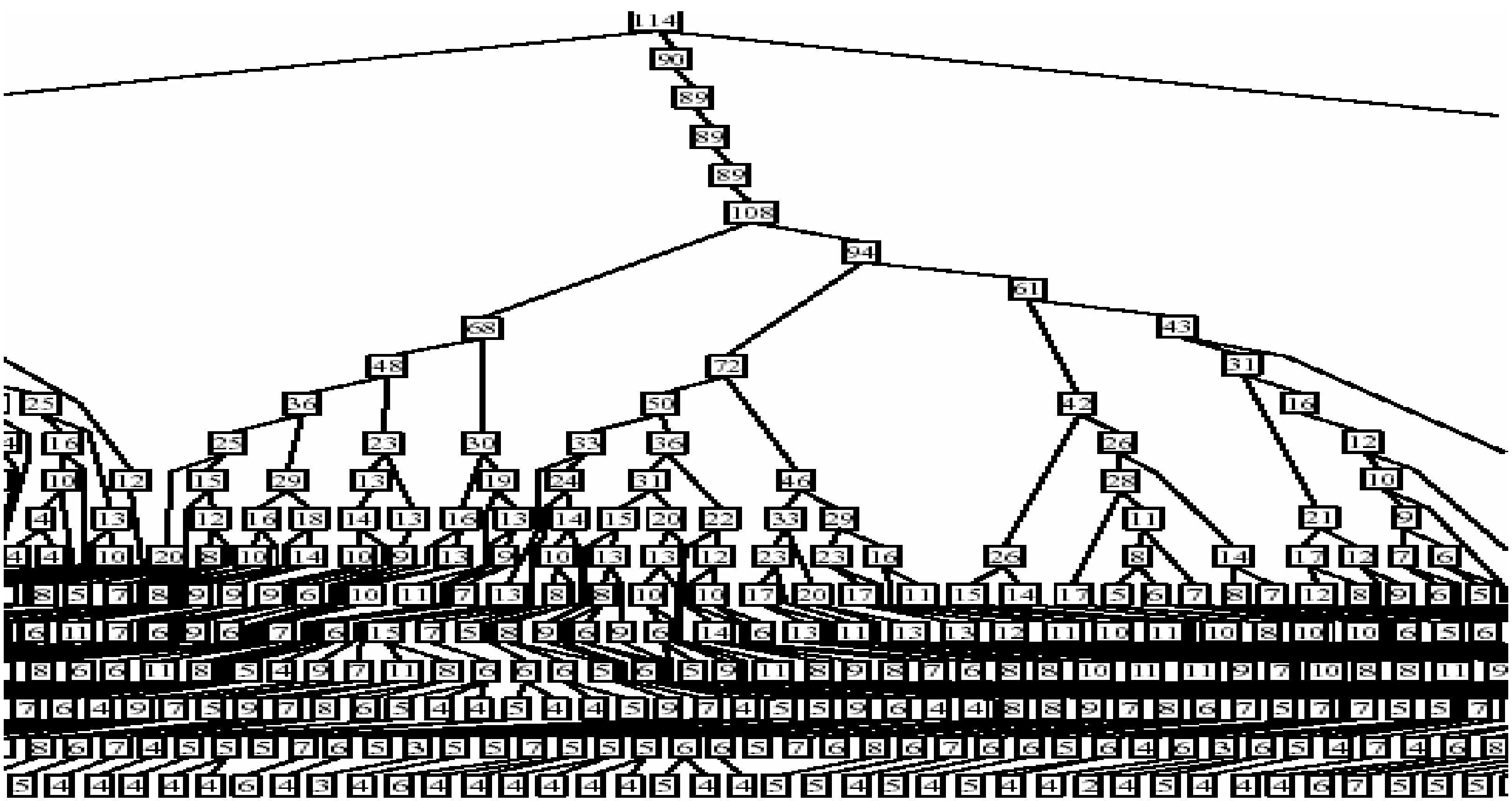
Case study: C432

A circuit's primal graph
For every gate we connect
inputs and outputs

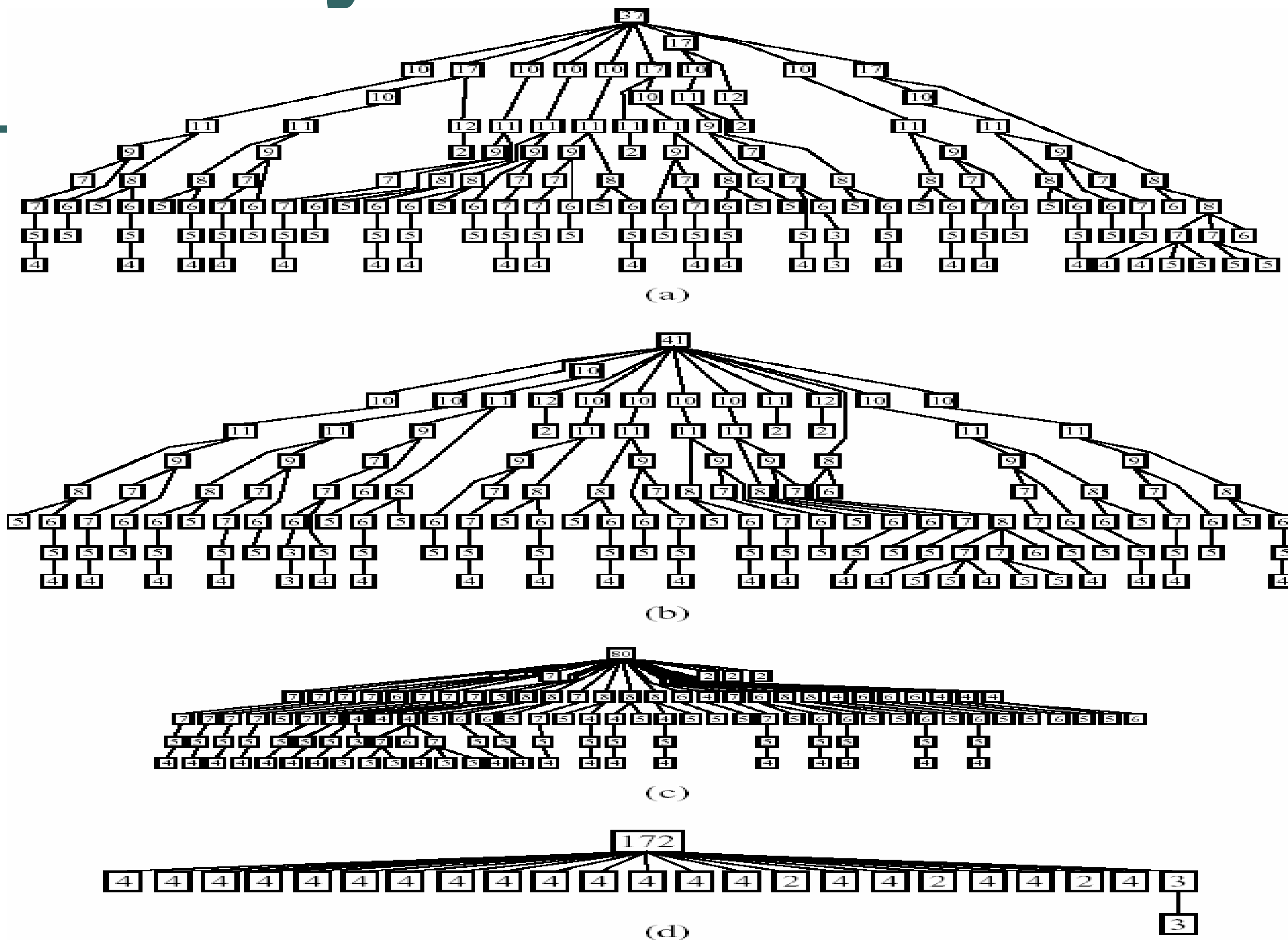


Join-tree of c432
Seperator size is 23

Join-tree of C3540 (1719 vars) max sep size 89



Secondary trees for C432



Time-space tradeoffsTime/Space tradeoff Time is measured by the maximum of the separator size and the cutset size and space by the maximum separator size.