Systematic vs non-systematic algorithms for constraint optimization

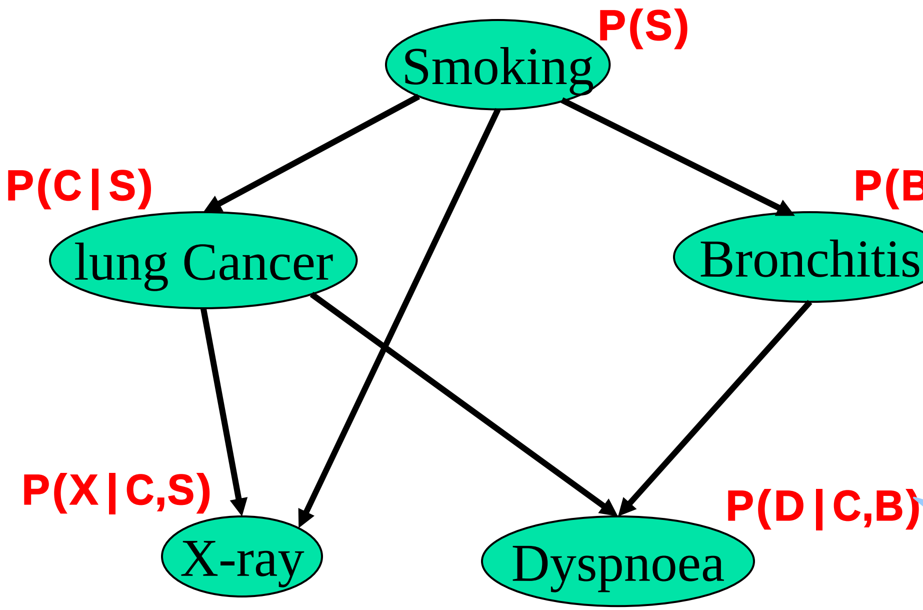
Rina Dechter
University of California
Irvine

Collaborators:
Kalev Kask
Radu Marinescu,
Robert mateescu

Overview

- **Introduction and background:**
  - **Combinatorial optimization tasks: CSP, Max-CSP, belief updating, MPE**
- Bounded inference: mini-bucket and mini-clustering
- Heuristic generation for Brunch and Bound
  - BBMB(i), BBBT(i)
- Empirical evaluation on Max-CSPs, MPE, CSP
- Conclusions

Probabilistic Networks



BN = (G, Θ)

CPD:

C	B	D=0	D=1
0	0	0.1	0.9
0	1	0.7	0.3
1	0	0.8	0.2
1	1	0.9	0.1

$$P(S, C, B, X, D) = P(S)\, P(C|S)\, P(B|S)\, P(X|C,S)\, P(D|C,B)$$

P(S|d)= ?

MPE: argmax ***P(S) P(C|S) P(B|S) P(X|C,S) P(D|C,B)***

Belief networks

A *belief network* is a quadruple

$BN = <X, D, G, P>$ where :

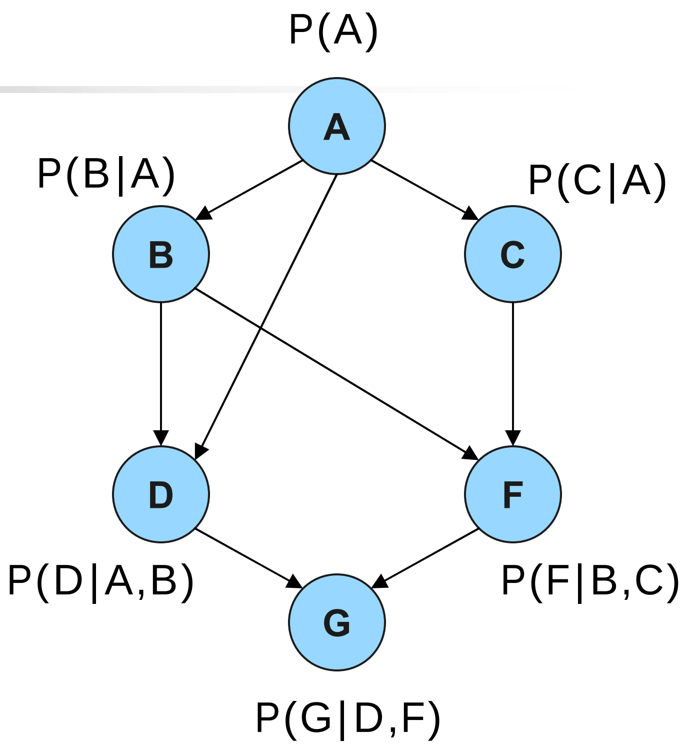
$X = \{X_1, ..., X_n\}$ is a set of random variables

$D = \{D_1, ..., D_n\}$ is the set of their domains

G is a DAG (directed acyclic graph) over X

$P = \{p_1, ..., p_n\}, \; p_i = P(X_i \mid pa_i)$ are CPTs

(conditional probability tables)



- The belief updating problem is the task of computing the posterior probability *P(X/e)* of query node *X* given evidence *e*.

.

Constraint Satisfaction

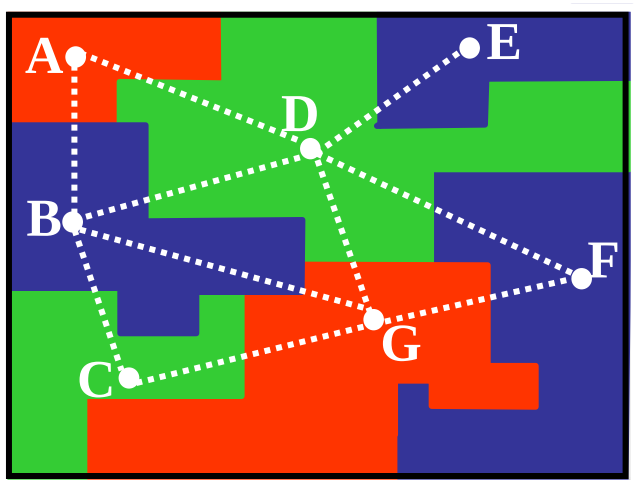
Example: map coloring

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

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

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

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

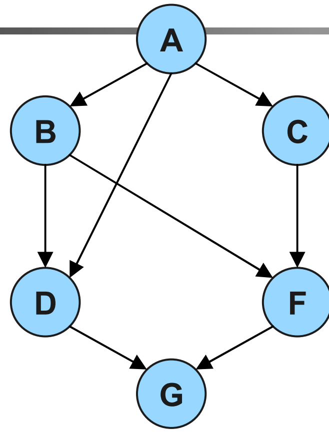


Semantics: set of all solutions

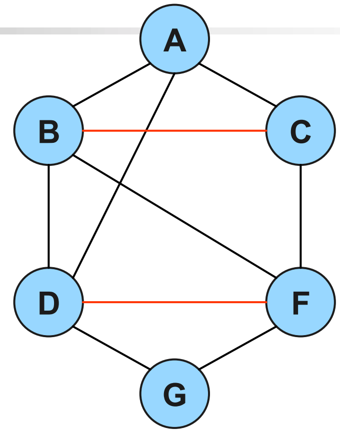
Primary task: find a solution

Belief and constraint networks

P(A)
P(B|A)
P(C|A)
P(D|A,B)
P(F|B,C)
P(G|D,F)

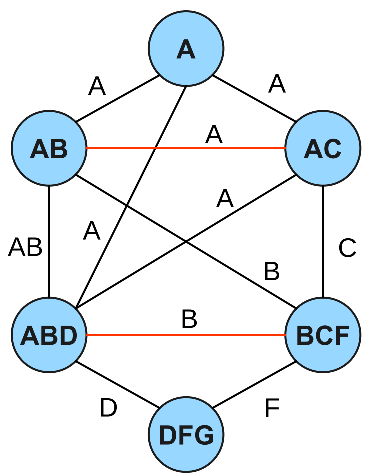


a) Belief network



R(A)
R(A,B)
R(A,C)
R(A,B,D)
R(B,C,F)
R(D,F,G)

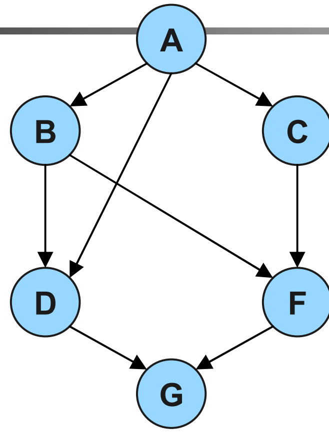
b) Constraint network



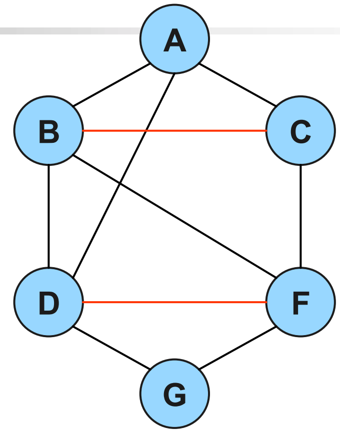
c) Dual graph

Belief and constraint networks

P(A)
P(B|A)
P(C|A)
P(D|A,B)
P(F|B,C)
P(G|D,F)

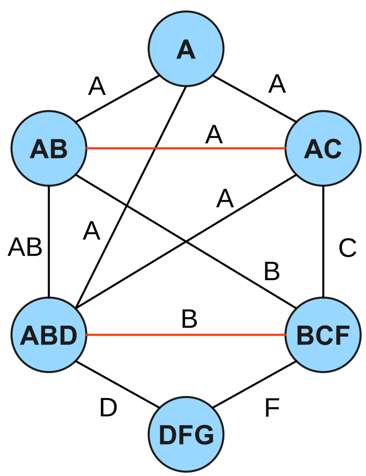


a) Belief network



R(A)
R(A,B)
R(A,C)
R(A,B,D)
R(B,C,F)
R(D,F,G)

b) Constraint network

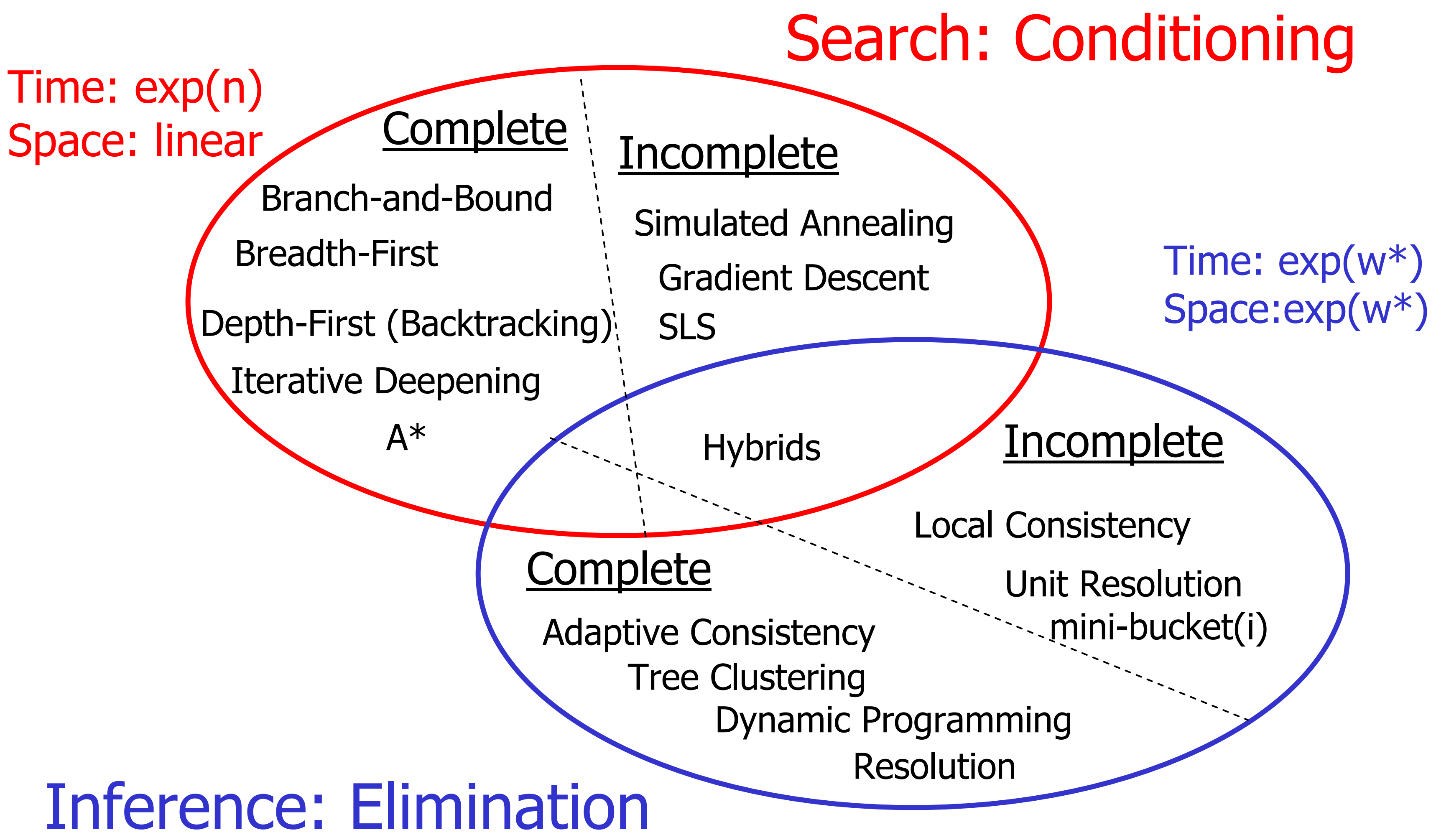


c) Dual graph

- Belief updating: $\Sigma_{X-y} \prod_j P_j$
- MPE: $\max_X \prod_j P_j$
- CSP: $\prod_X \times_j C_j$
- Max-CSP: $\min_X \Sigma_j F_j$

all are np-hard,
Also hard to approximate

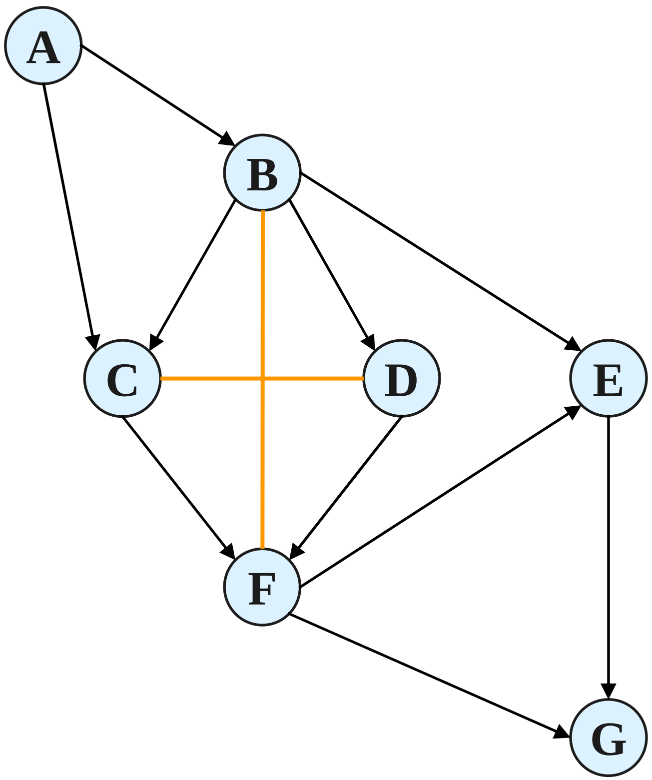
Solution Techniques

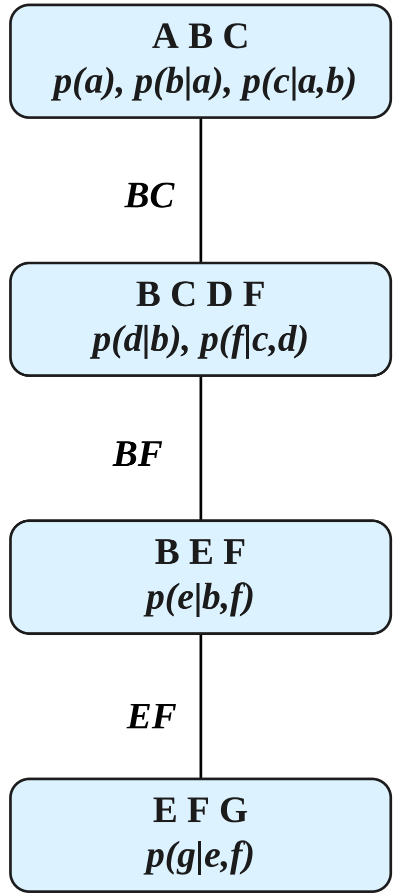


Overview

- Introduction and Background
- **Bounded inference:**
  - mini-bucket and mini-clustering
  - Belief propagation: IJGP(i)
- Heuristic generation for Brunch and Bound
  - BBMB(i), BBBT(i)
- Empirical evaluation on Max-CSPs, MPE, CSP
- Conclusions

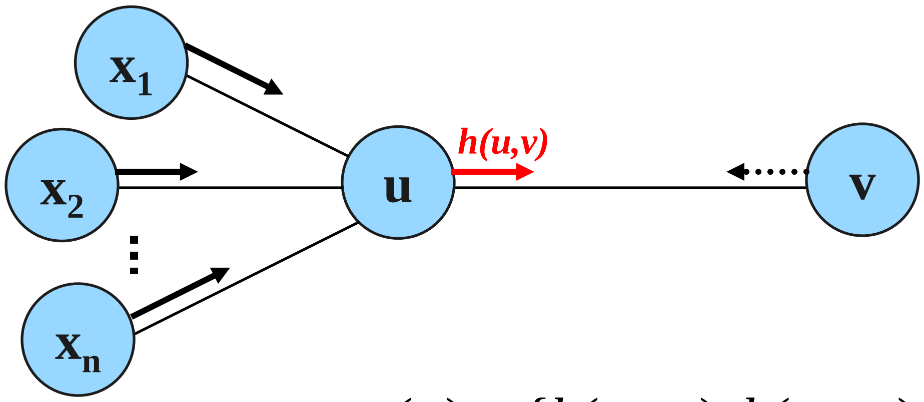
Tree decomposition





- Each function in a cluster
- Satisfy running intersection property

Belief Propagation

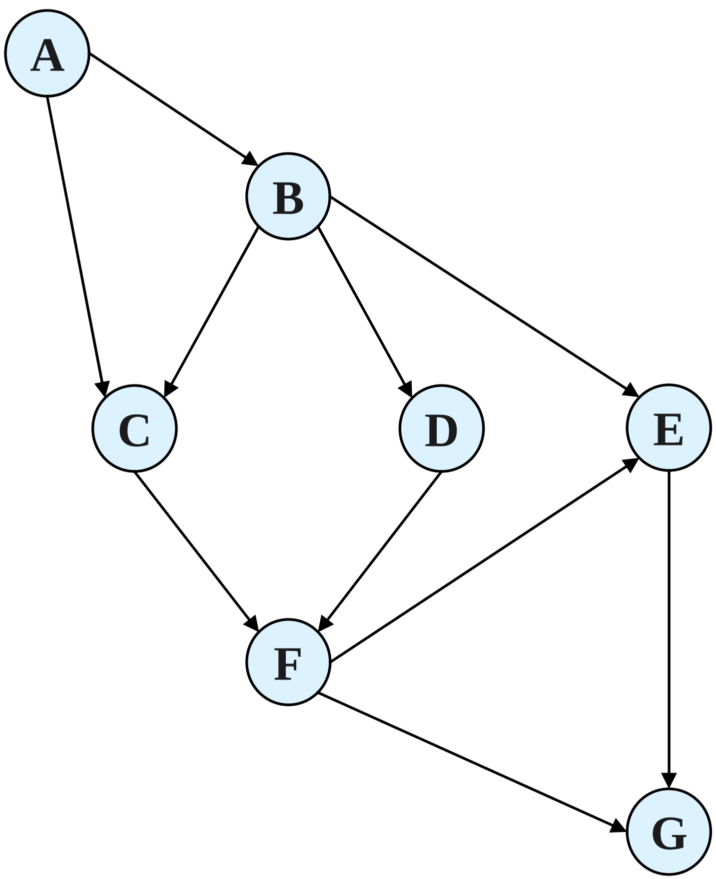


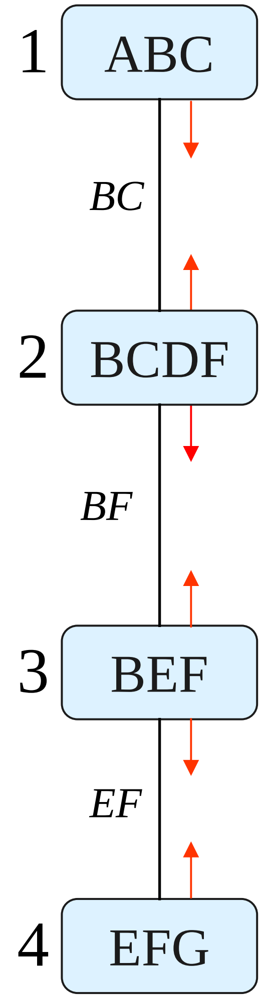
$$\psi(u) \cup \{h(x_1,u), h(x_2,u), \ldots, h(x_n,u), h(v,u)\}$$

Compute the message:

$$h(u,v) = \sum_{elim(u,v)} \prod_{f \in cluster(u) - \{h(v,u)\}} f$$

CTE: Cluster Tree Elimination





$$h_{(1,2)}(b,c) = \sum_{a} p(a) \cdot p(b|a) \cdot p(c|a,b)$$

$$h_{(2,1)}(b,c) = \sum_{d,f} p(d|b) \cdot p(f|c,d) \cdot h_{(3,2)}(b,f)$$

$$h_{(2,3)}(b,f) = \sum_{c,d} p(d|b) \cdot p(f|c,d) \cdot h_{(1,2)}(b,c)$$

$$h_{(3,2)}(b,f) = \sum_{e} p(e|b,f) \cdot h_{(4,3)}(e,f)$$

$$h_{(3,4)}(e,f) = \sum_{b} p(e|b,f) \cdot h_{(2,3)}(b,f)$$

$$h_{(4,3)}(e,f) = p(G = g_e | e,f)$$

Time: *O (exp(w*+1))*

Space: *O (exp(sep))*

For each cluster P(X|e) is computed

Bucket elimination

Algorithm Elim-*MPE* (Dechter 1996)

$\max_b \prod$ ← Elimination operator

bucket B: P(b|a) P(d|b,a) P(e|b,c)

bucket C: P(c|a) $h^B(a, d, c, e)$

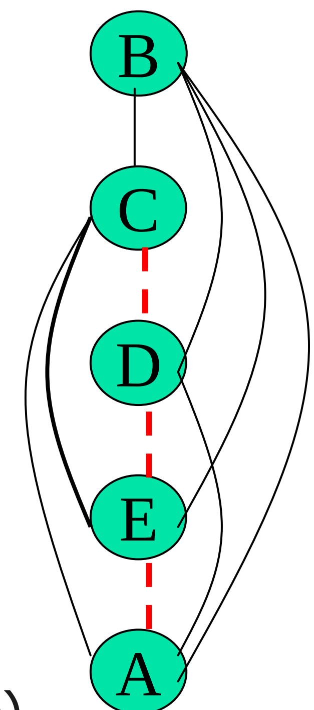
bucket D: $h^C(a, d, e)$

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

bucket A: P(a) $h^E(a)$

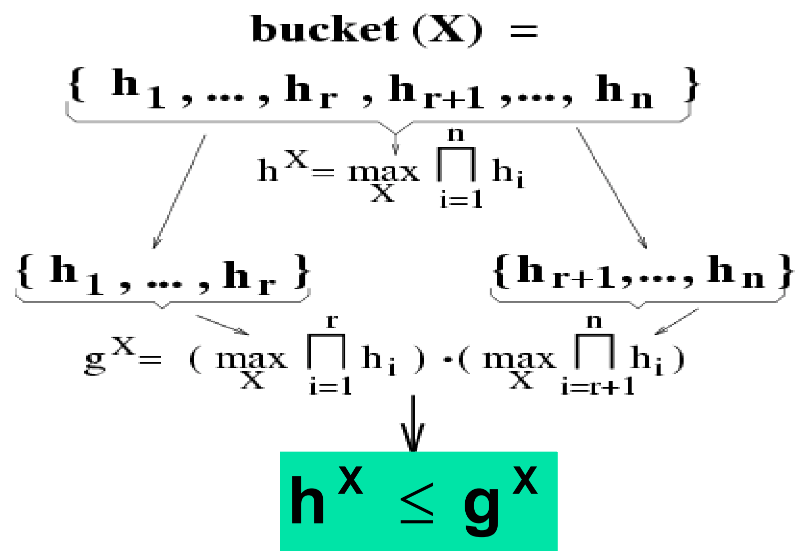
mpe

W*=4
"induced width"
(max clique size)



Two Principles for Bounded Inference

- **Bounded-Partitioning**
  - **mini-bucket(i), MC(i)**
  - **Computes a bound**
  - **Exp(i) time space**

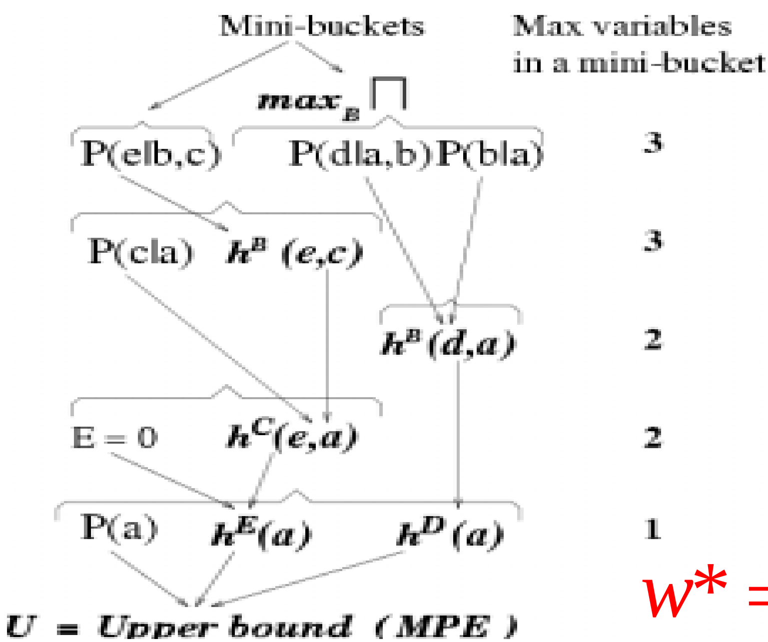


Approx-mpe(i)

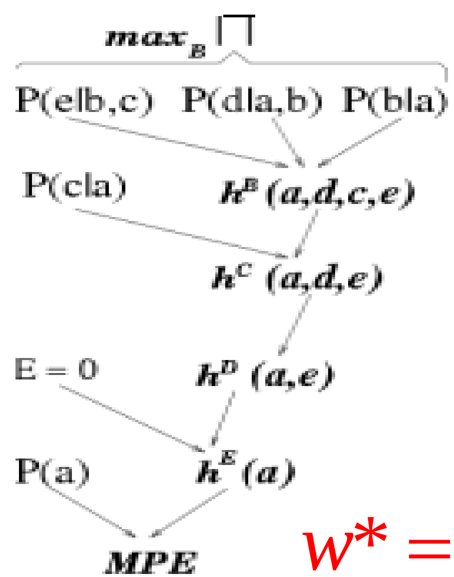
Algorithm Approx-*MPE* (Dechter&Rish 1997)

- Input: i – max number of variables allowed in a mini-bucket
- Output: [lower bound (P of a sub-optimal solution), upper bound]

Example: approx-mpe(3) versus elim-mpe

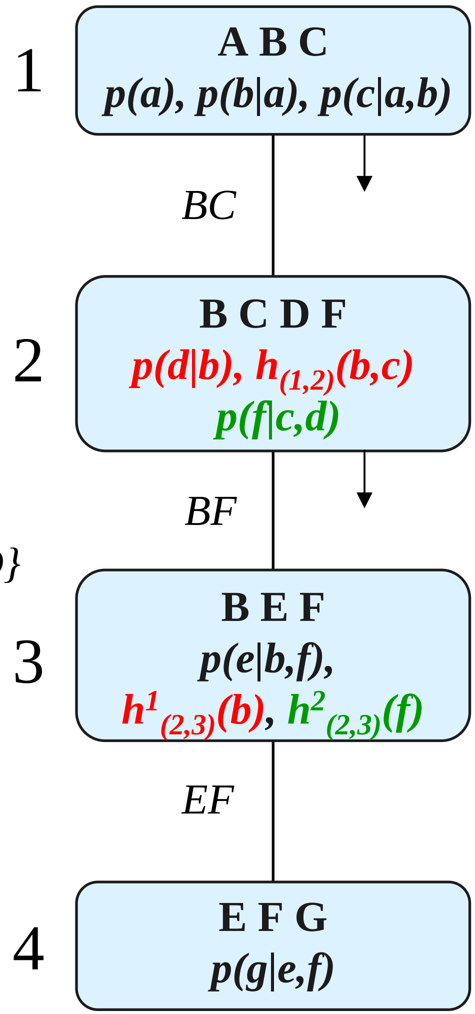


$w^* = 2$



$w^* = 4$

Mini-Clustering idea



$$h^1_{(1,2)}(b,c) = \sum_a p(a) \cdot p(b \mid a) \cdot p(c \mid a,b)$$

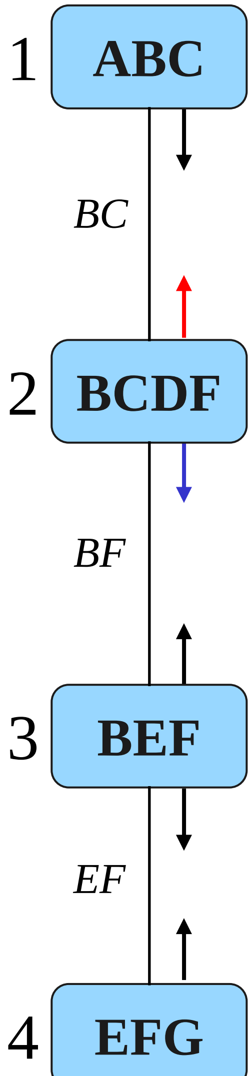
sep(2,3)={B,F}

elim(2,3)={C,D}

$$h^1_{(2,3)}(b) = \sum_{c,d} p(d \mid b) \cdot h^1_{(1,2)}(b,c)$$

$$h^2_{(2,3)}(f) = \max_{c,d} p(f \mid c,d)$$

Mini-Clustering – MCTE(i)



$H_{(1,2)}$: $$h^1_{(1,2)}(b,c) := \sum_a p(a) \cdot p(b \mid a) \cdot p(c \mid a,b)$$

$H_{(2,1)}$: $$h^1_{(2,1)}(b) := \sum_{d,f} p(d \mid b) \cdot h^1_{(3,2)}(b,f)$$ $$h^2_{(2,1)}(c) := \sum_{d,f} p(f \mid c,d)$$

$H_{(2,3)}$: $$h^1_{(2,3)}(b) := \sum_{c,d} p(d \mid b) \cdot h^1_{(1,2)}(b,c)$$ $$h^2_{(2,3)}(f) := \sum_{c,d} p(f \mid c,d)$$

$H_{(3,2)}$: $$h^1_{(3,2)}(b,f) := \sum_e p(e \mid b,f) \cdot h^1_{(4,3)}(e,f)$$

$H_{(3,4)}$: $$h^1_{(3,4)}(e,f) := \sum_b p(e \mid b,f) \cdot h^1_{(2,3)}(b) \cdot h^2_{(2,3)}(f)$$

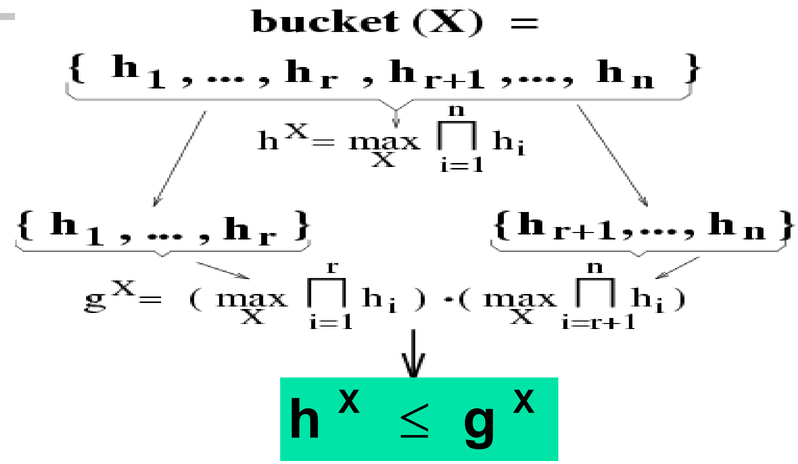
$H_{(4,3)}$: $$h^1_{(4,3)}(e,f) := p(G = g_e \mid e,f)$$

Properties of MC(i)

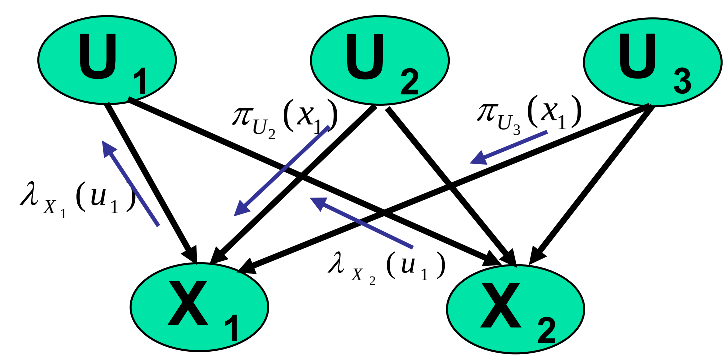
- MCTE(i) computes a bound on the exact value : $\otimes_{f \in M_{(u,v)}} f$ is an approximation of $m_{(u,v)}$.
- Time & space complexity: $O(N \times hw^* \times d^i)$
- Approximation improves with i but takes more time

Two Principles for Bounded Inference

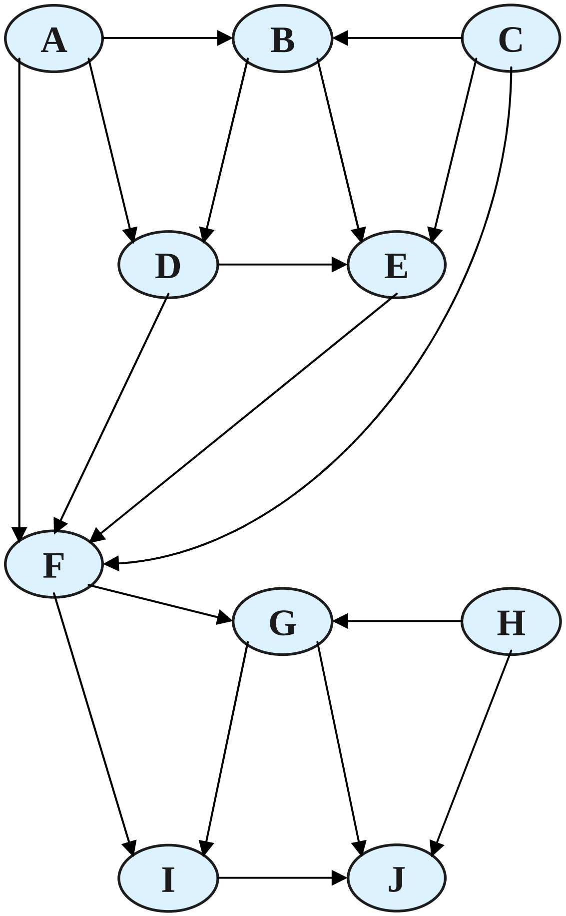
- **Bounded-Partitioning**
  - **mini-bucket(i), MC(i)**
  - **Computes a bound**
  - **Exp(i) time space**



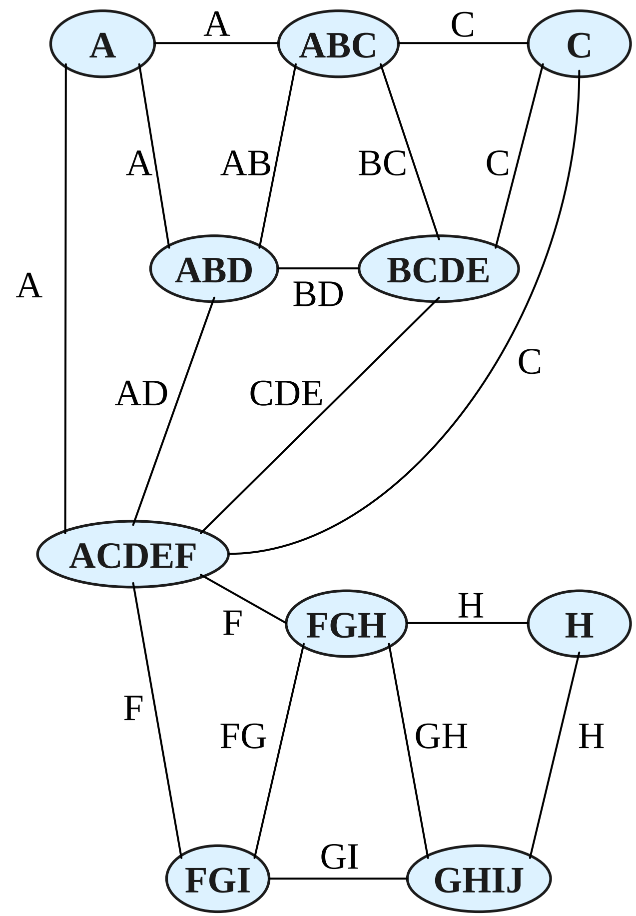
- **Belief propagation on join-graphs**
  - **IBP, IJGP(i)**
  - **No guuarantees**
  - **Each iteration is exp(i)**



Iterative Join-Graph Propagation

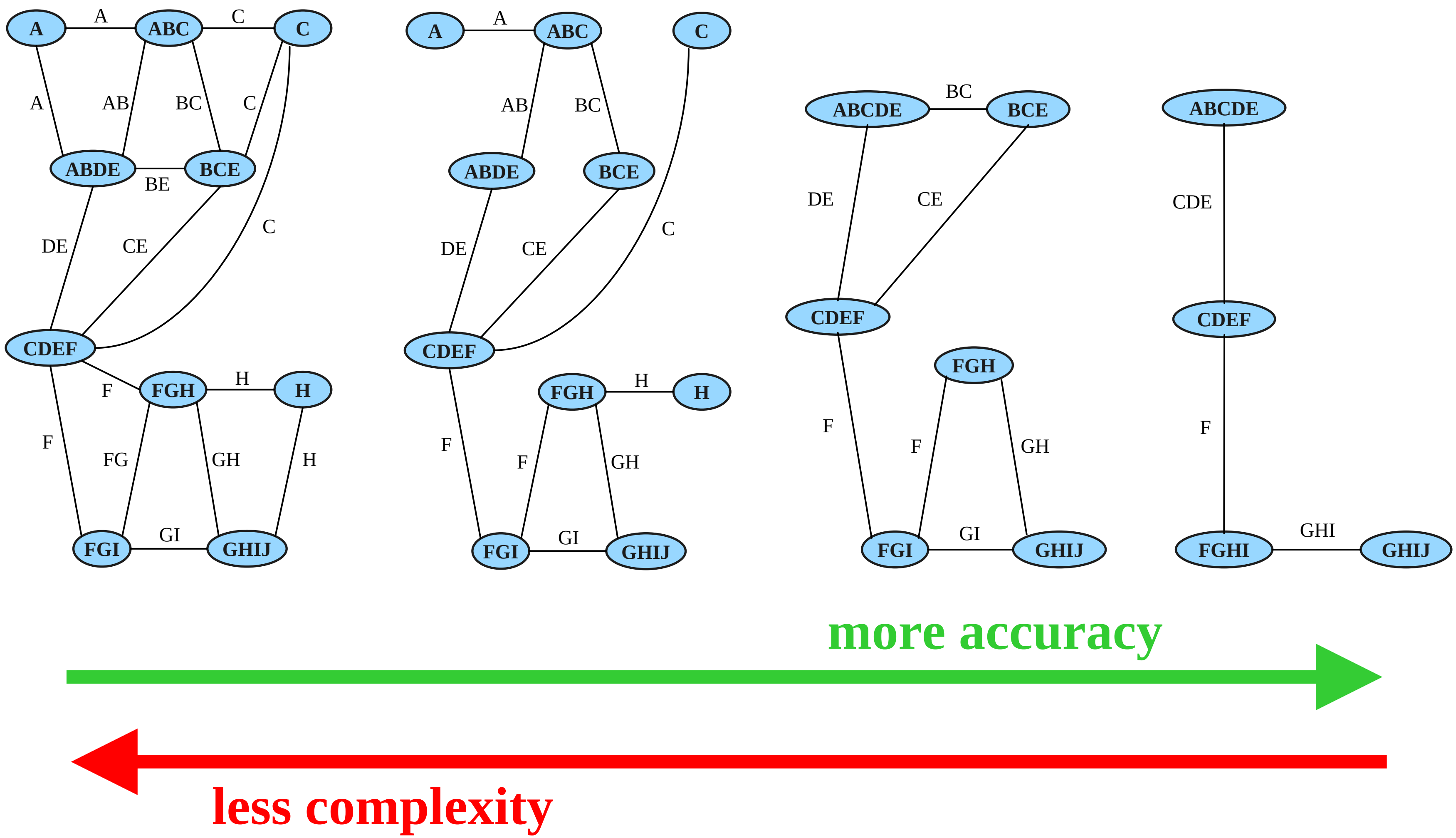


a) Belief network



a) The graph IBP works on

Join-graphs


A
A
ABC
C
C
A
AB
BC
C
ABDE
BE
BCE
DE
CE
C
CDEF
F
FGH
H
H
F
FG
GH
H
FGI
GI
GHIJ
A
A
ABC
C
AB
BC
ABDE
BCE
DE
CE
C
CDEF
FGH
H
H
F
F
GH
FGI
GI
GHIJ
ABCDE
BC
BCE
DE
CE
CDEF
FGH
F
F
GH
FGI
GI
GHIJ
ABCDE
CDE
CDEF
F
FGHI
GHI
GHIJ
more accuracy
less complexity

Empirical results showed:

- **Mini-bucket(i) and MC(i)**
  - Accuracy/time increase with i-bound
  - Compute bounds.
  - demonstrate impressive performance for many problem classes for both optimization and belief updating.
- **IJGP(i) is generally superior to MC for belief updating. But no bound.**

BnB for Constraint Optimization

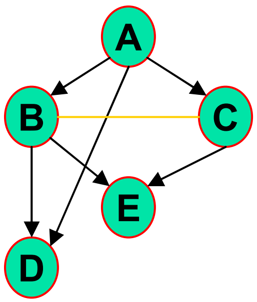
- Max-CSP
- MPE
- CSP

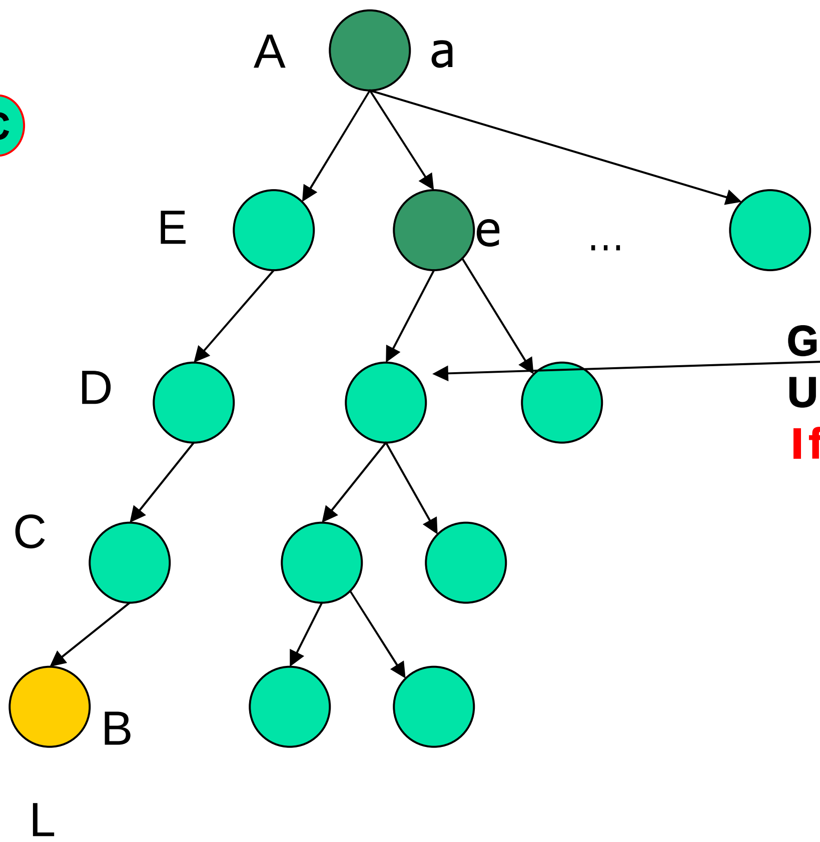
Overview

- **Introduction and background:**
  - **Combinatorial optimization tasks: CSP, Max-CSP, belief updating, MPE**

- Bounded inference: mini-bucket and mini-clustering
- Heuristic generation for Brunch and Bound
  - BBMB(i), BBBT(i)
- Empirical evaluation on Max-CSPs, MPE, CSP
- Conclusions

BnB with inferred heuristics



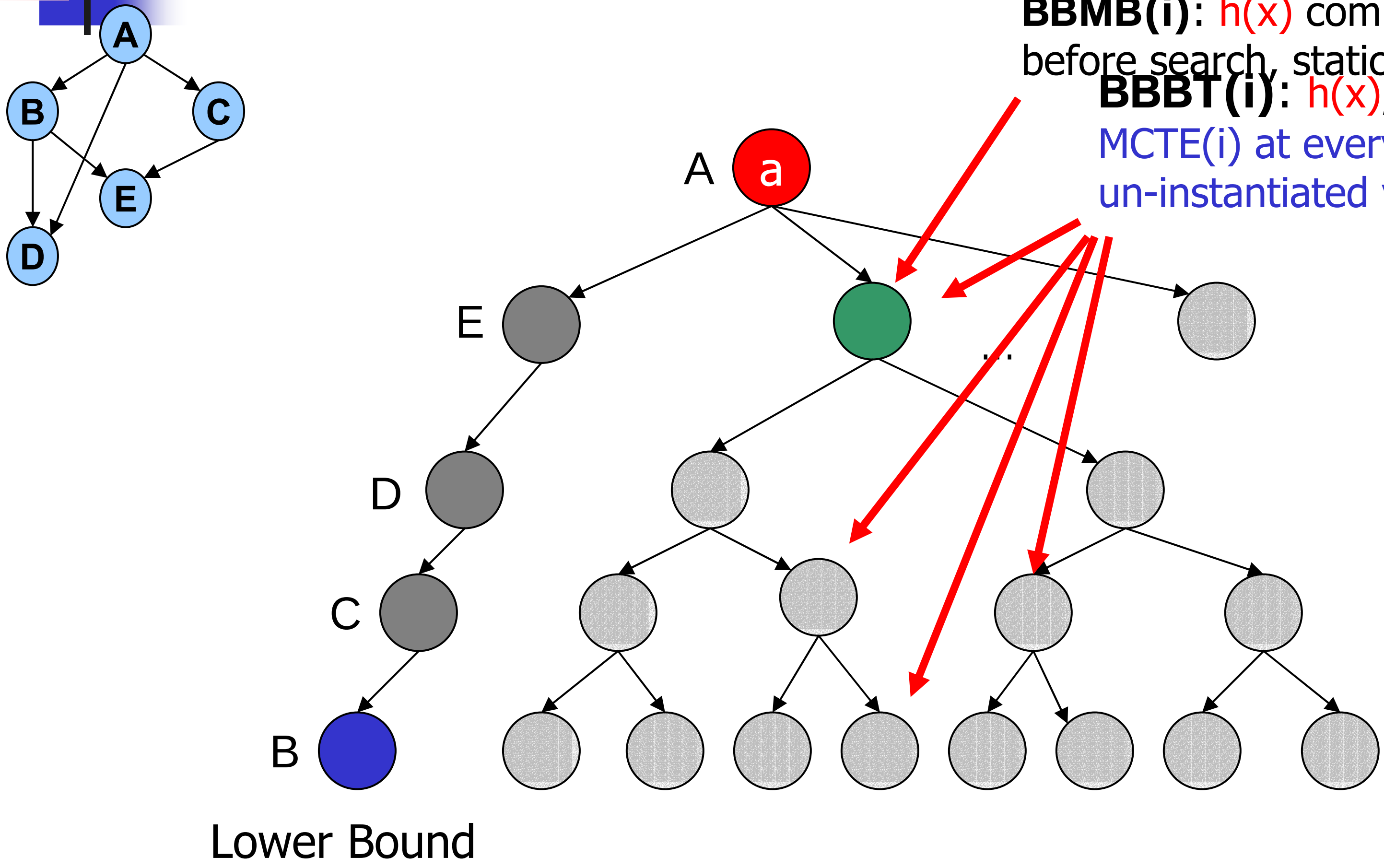


Guiding heuristic evaluation function
Upper-bound h(x).
If h < L search is pruned.

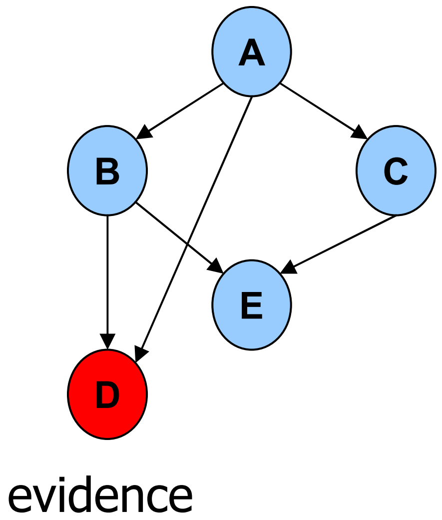
Two BnB schemes

BBMB(i): h(x) computed by MB(i), before search, static ordering

BBBT(i): h(x), computed via MCTE(i) at every node for every un-instantiated variable



Optimization Task



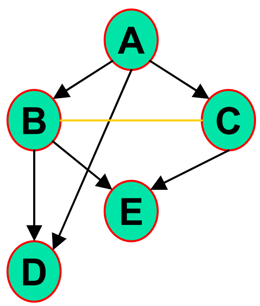
- The *Most Probable scenario* problem is to find a most probably complete assignment that is consistent with the evidence *e*.

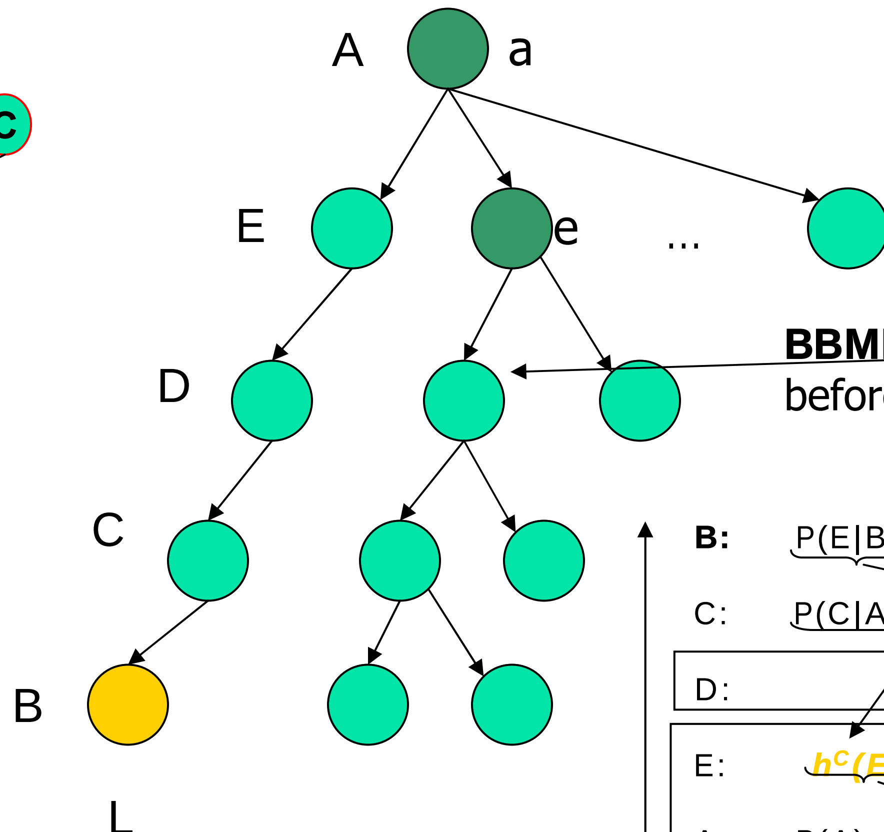
- **Systematic Search** (BnB)
- **Non-Systematic Search** (SLS, BP)

The main idea

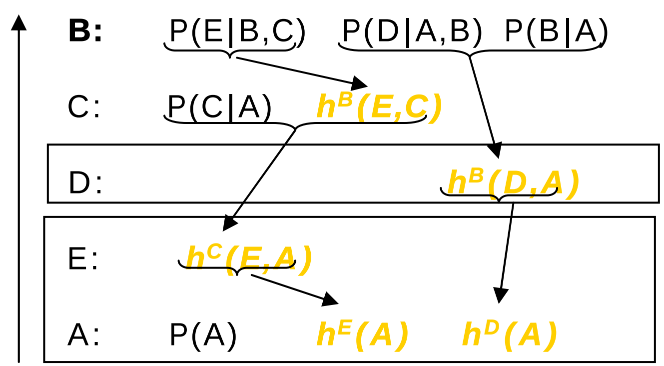
- BnB with exact heuristic:
  - h(x1,…,xi) if equals max-cost extenion of partial solution, (x1,…,xi) will yield backtrack-free search
- Idea:
  - mini-bucket(i) compiles an upper bound h(x1,…,xi) for max-cost extensions of (x1,….,x_i) for every partial solution along the fixed ordering.
- → Run MB(i), then run BnB in reverse order using the mini-bucket heuristics

BBMB





BBMB(i): h(x) computed by MB(i), before search, static ordering



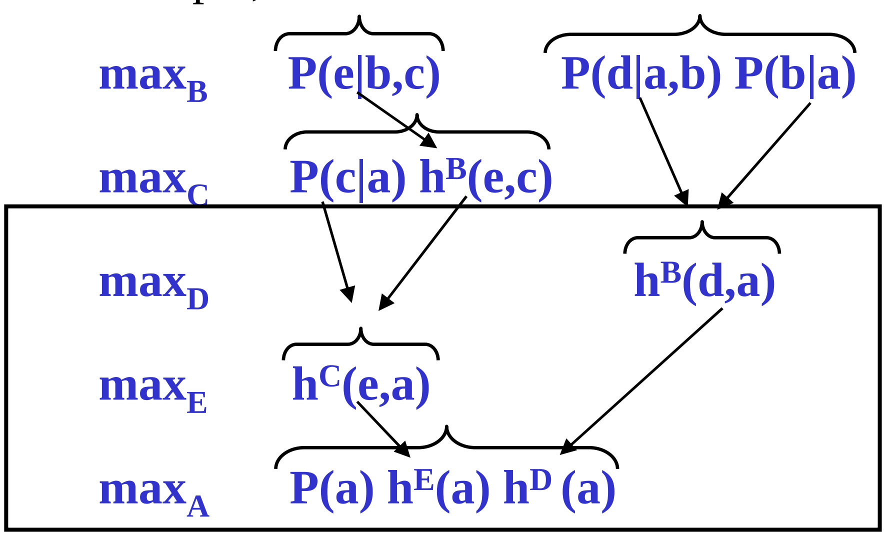
$$f(a,e,D) = P(a) \cdot h^{B}(D,a) \cdot h^{C}(e,a)$$

Heuristic Function

The evaluation function $f(x^p)$ can be computed using function recorded by the Mini-Bucket scheme and can be used to estimate the probability of the best extension of partial assignment $x^p=\{x_1, \ldots, x_p\}$,

$$f(x^p)=g(xp) \bullet H(x^p)$$

For example,



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

$$g(a,e,d) = P(a)$$

Properties

- Heuristic is monotone, admissible
- Heuristic is computed in linear time
- IMPORTANT:
  - Mini-buckets generate heuristics of varying strength using i.
  - Higher i-bound $\Rightarrow$ more pre-processing $\Rightarrow$ stronger heuristic $\Rightarrow$ less search.
  - **Allows controlled trade-off between preprocessing and search**

Experimental Methodology

Test networks:

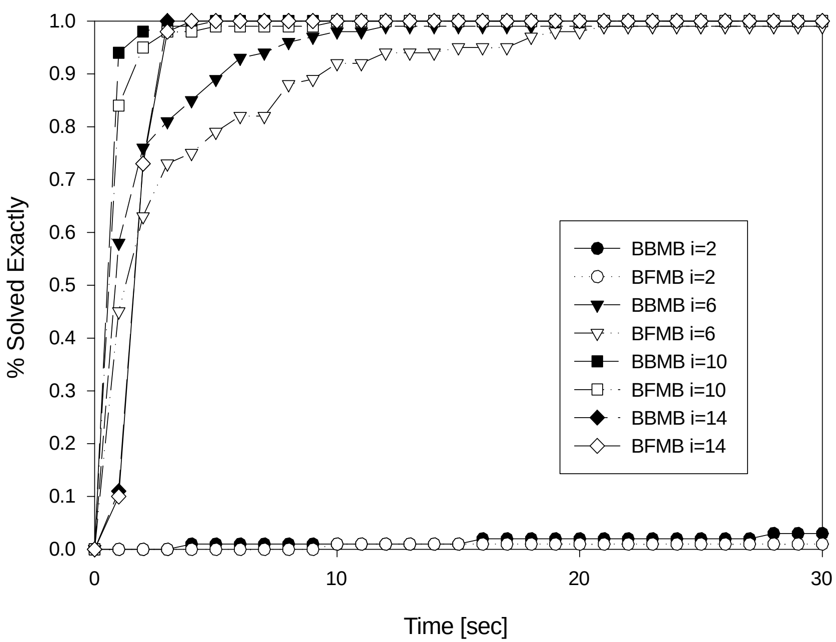
- Random Coding (Bayesian)
- CPCS (Bayesian)
- Random (CSP)

Measures of performance

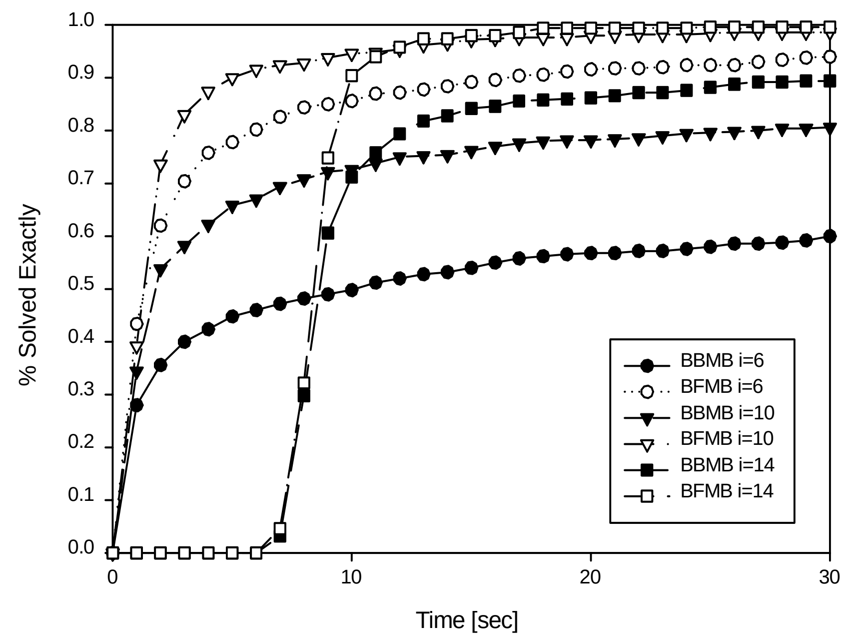
- Compare in terms of accuracy given a fixed amount of time - how close is the probability/cost of the assignment they find to the probability/cost of the optimal solution
- Compare trade-off performance as a function of time

Empirical Evaluation of mini-bucket heuristics

Random Coding, K=100, noise=0.28



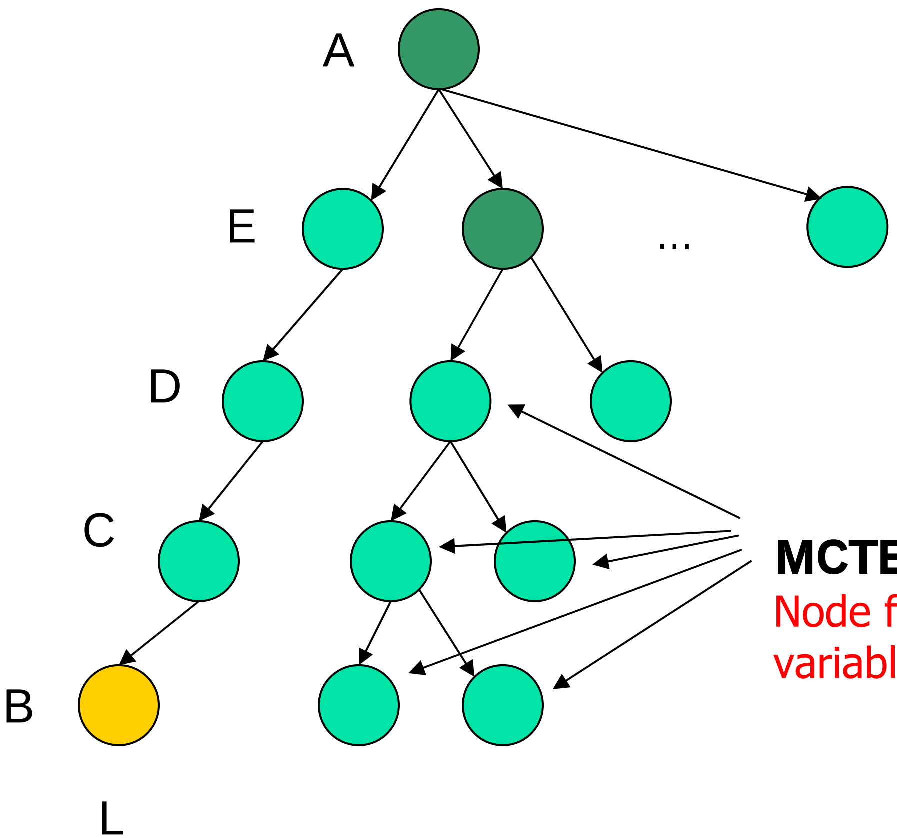
Random Coding, K=100, noise=0.32



Max-CSP experiments
(Kask and Dechter, 2000)

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

BBBT(i) – Search Space



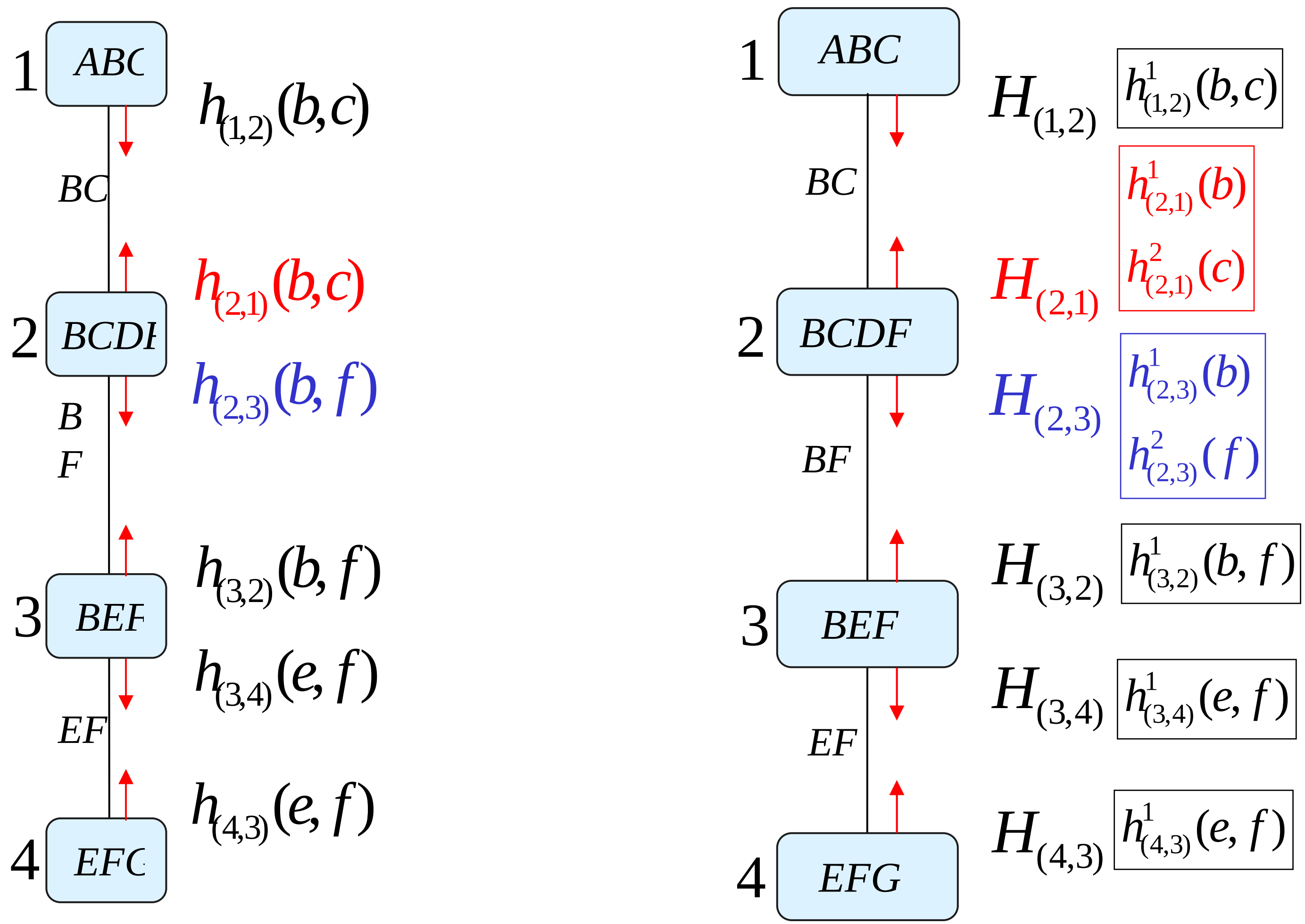
MCTE(i) computes h(x), at every Node for every uninstantiated variable.

- Branch-and-Bound search where MCTE(i) is executed at each visited node
  - Domain pruning
  - Dynamic variable ordering
  - Dynamic value ordering

BBBT and the singleton optimality task

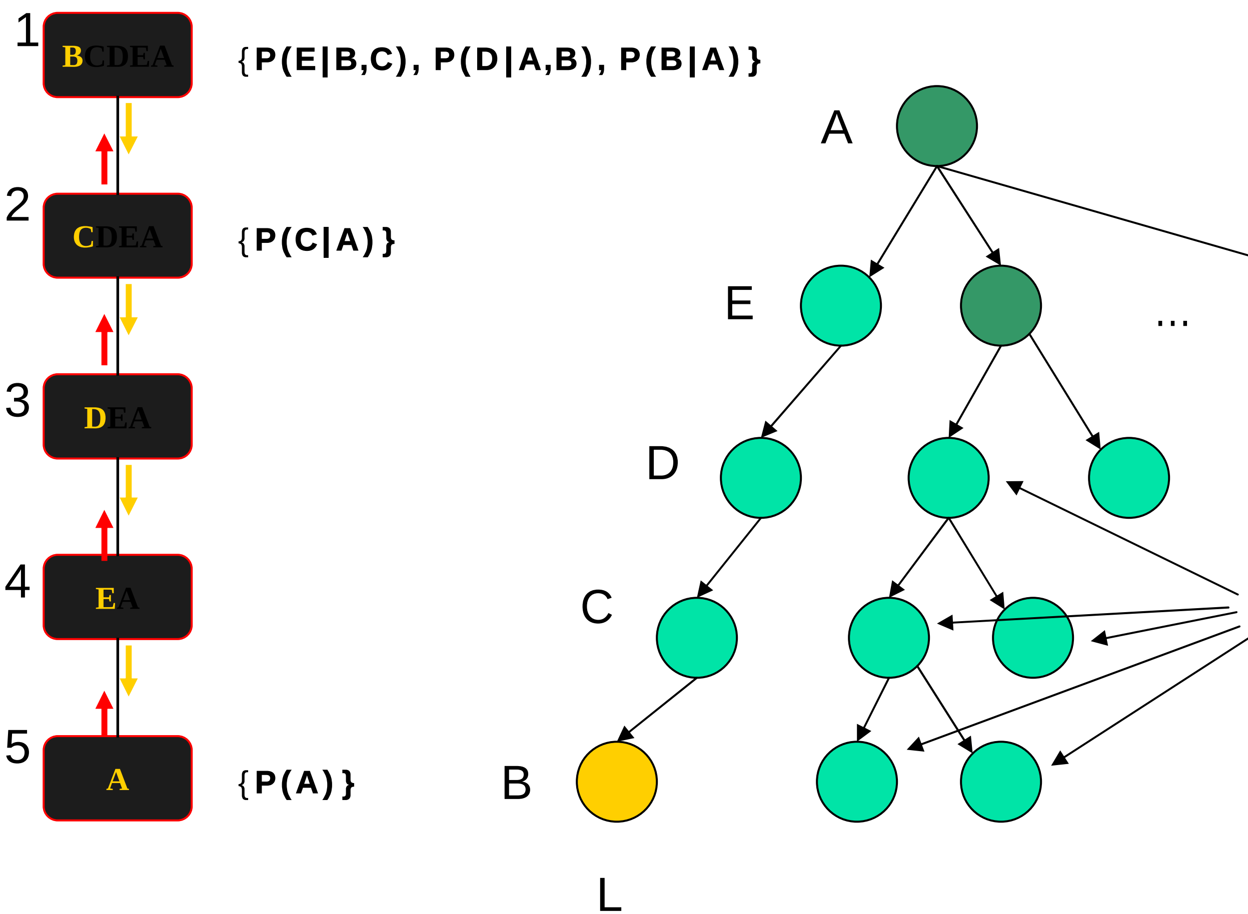
- Computes for each variable and each value the max-cost extension for the rest of the free variable given the instantiated ones.

Cluster Tree Elimination vs. Mini Clustering



Compute bounds to singleton optimality task for every var-val

BBBT – Search Space



BBBT(i): h(x), computed via MC(i) at every Node for every uninstantiated variable. Dynamic, Variable and value ordering

BBBT: BnB Search with MBTE(*i*) Heuristics

- Main Idea:
  - During search, maintain Lower Bound L (the best MPE cost so far).
  - When processing variable X_p:
    - Compute heuristic estimates mZ_i for all un-instantiated variables (using MBTE(i)).
    - Use the costs to prune the domains of un-instantiated variables.
    - Backtrack when an empty domain occurs, otherwise expand the current assignment (smallest domain variable).

BnB with Lower Bound Heuristics

- BBMB(i), the earlier algorithm:
  - Heuristic, computed by MB(i), is static, variable ordering fixed.
- BBBT(i), the new algorithm:
  - Lower bound is computed at each node of the search by MCTE(i).
  - Used for dynamic variable and value ordering.
  - Domain pruning.

Non-Systematic Algorithms

- Stochastic Local Search [Park, 2002]
  - Guided Local Search (GLS) [Mills and Tsang,2000]
  - Discrete Lagrange Multipliers (DLM) [Wah and Shang,1997]
  - Stochastic Local Search (SLS) [Kask and Dechter,1999]
- Iterative Belief Propagation
  - Iterative Join Graph Propagation (*max*-IJGP)

Stochastic Local Search (I)

- **Discrete Lagrange Multipliers (DLM)** [Wah and Shang,1997]
  - For each clause *C:* weight w_c , Lagrange multiplier λ_c Cost function: $sum(w_c + \lambda_c)$.
  - At local maxima, increase λs of all unsatisfied clauses.

- **Guided Local Search (GLS)** [Mills and Tsang,2000]
  - For each clause *C:* weight w_c , Lagrange multiplier λ_c
  - Cost function: $sum(\lambda_c)$.
  - At local maxima, increase λs of the unsatisfied clauses with maximum utility.

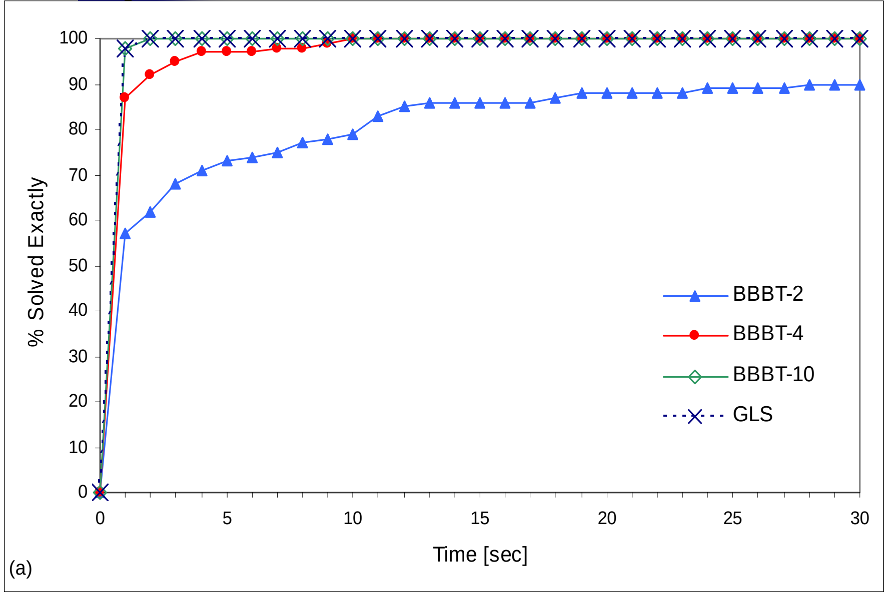
Stochastic Local Search (II)

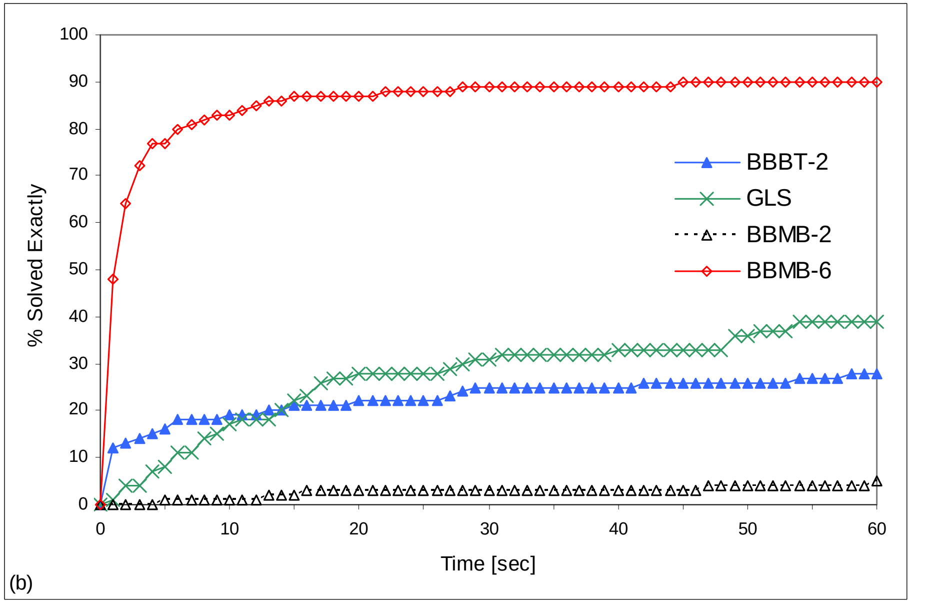
- **Stochastic Local Search (SLS)** [Kask and Dechter, 1999]
  - At each step performs either a hill climbing or a stochastic variable change.
  - Periodically, the search is restarted in order to escape local maxima.

Experimental Results

- Algorithms:
  - Complete
    - BBBT(i)
    - BBMB(i)
  - Incomplete, competing methods
    - DLM
    - GLS
    - SLS
    - IJGP
    - IBP (coding)
- Benchmarks:
  - Coding networks
  - Bayesian Network Repository
  - Grid networks (N-by-N)
  - Random noisy-OR networks
  - Random networks
- Measures:
  - Time
  - Accuracy (% exact)
  - #Backtracks
  - Bit Error Rate (coding)

Random Networks – Average Accuracy





Average Accuracy. Random Bayesian (N=100, C=90, P=2), w* = 17
100 samples, 10 observations
(a) K = 2, (b) K = 3.

We see: GLS is good for small domains,
GLS is poor for large domains
BBMB is best for strong heuristics
BBBT exploit better weak heuristics

Grid Networks – Accuracy and Time

K	BBBT / BBMB i=2 %[time]	BBBT / BBMB i=4 %[time]	BBBT / BBMB i=6 %[time]	BBBT / BBMB i=8 %[time]	BBBT / BBMB i=10 %[time]	GLS %[time]	DLM %[time]	SLS %[time]
2	51[17.7] 1[29.9]	99[2.62] 13[23.7]	100[0.66] 93[2.16]	100[0.48] 92[0.08]	**100[0.42]** 95[0.02]	100[1.54]	0[30.01]	0[30.01]
3	3[58.7] 0[60.01]	28[47.4] 1[58.9]	80[19.5] 25[50.9]	93[14.8] 89[8.63]	**94[23.2]** 92[0.73]	4[58.7]	0[60.01]	0[60.01]
4	1[118.8] 0[120]	12[108.3] 0[120]	46[78.4] 6[113.4]	61[88.5] 72[46.4]	33[136] **85[9.91]**	0[120]	0[120]	0[120]

Average Accuracy and Time. Random Grid (N=100), w*=15, 100 samples, 10 observations

Random Coding Networks – Bit Error Rate

σ	BBBT BBMB IJGP	BBBT BBMB IJGP	BBBT BBMB IJGP	BBBT BBMB IJGP	BBBT BBMB IJGP	IBP GLS SLS
	i=2 BER[time]	i=4 BER[time]	i=6 BER[time]	i=8 BER[time]	i=10 BER[time]	BER[time]
0.32	0.0056[3.18] 0.0034[0.07] 0.0034[0.16]	0.0104[2.87] 0.0034[0.08] 0.0034[0.18]	0.0072[1.75] 0.0034[0.03] 0.0034[0.33]	0.0034[0.72] 0.0034[0.01] 0.0034[0.92]	0.0034[0.59] 0.0034[0.02] 0.0034[3.02]	**0.0034[0.01]** 0.2344[60.01] 0.4980[60.01]
0.40	0.0642[19.4] 0.0114[0.63] 0.0114[0.16]	0.0400[12.8] 0.0114[0.53] 0.0138[0.18]	0.0262[6.96] 0.0114[0.12] 0.0118[0.33]	0.0148[4.52] 0.0114[0.05] 0.0116[0.91]	0.0190[4.34] 0.0114[0.04] 0.0120[3.02]	**0.0108[0.01]** 0.2084[60.01] 0.5128[60.01]
0.52	0.1920[48.1] 0.0948[1.35] 0.1224[0.08]	0.1790[42.0] 0.0948[1.47] 0.1242[0.09]	0.1384[31.3] 0.0948[0.36] 0.1256[0.16]	0.1144[21.4] 0.0948[0.11] 0.1236[0.47]	0.1144[19.7] 0.0948[0.05] 0.1132[1.54]	**0.0894[0.01]** 0.2462[60.02] 0.5128[60.01]

Average BER. Random Coding (N=200, P=4), w*=22,
100 samples, 60 seconds

Real World Benchmarks

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

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

Empirical Results: Max-CSP

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

BBBT(i) vs BBMB(i), N=50

$N = 50$, $K = 5$, $C = 150$. $w^* = 17.6$. 10 instances. time = 600sec.

T	BBMB i=2	BBMB i=3	BBMB i=4	BBMB i=5	BBMB i=6	BBBT i=2	PFC-MPRDAC
	# solved time backtracks	# solved time backtracks	# solved time backtracks	# solved time backtracks	# solved time backtracks	# solved time backtracks	# solved time backtracks
5	6 45 1.11M	7 54 1.51M	6 6.2 177K	9 75 2.29M	10 6.2 123K	10 1.9 55	10 0.01 436
7	4 134 5.86M	5 150 4.62M	7 213 5.3M	8 208 5.14M	9 97 2.1M	10 2.5 94	10 1.7 15K
9	-	-	1 325 7.4M	3 227 4.97M	3 229 4.85M	10 14.3 2.1K	10 27.3 242K

BBBT(i) vs. BBMB(i)

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

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

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

A new BnB search algorithm for solving CSPs

- Method
  - Solution Counting heuristic is computed by Iterative Join Graph Propagation (IJGP)
- Hypothesis
  - Better scalability that competing BnB for CSP
- Results
  - Competitive with CSP, best algorithm in practice for random CSPs; SLS is incomplete, our new algorithm is complete.
  - Strength: quickly finding a solution if one exists

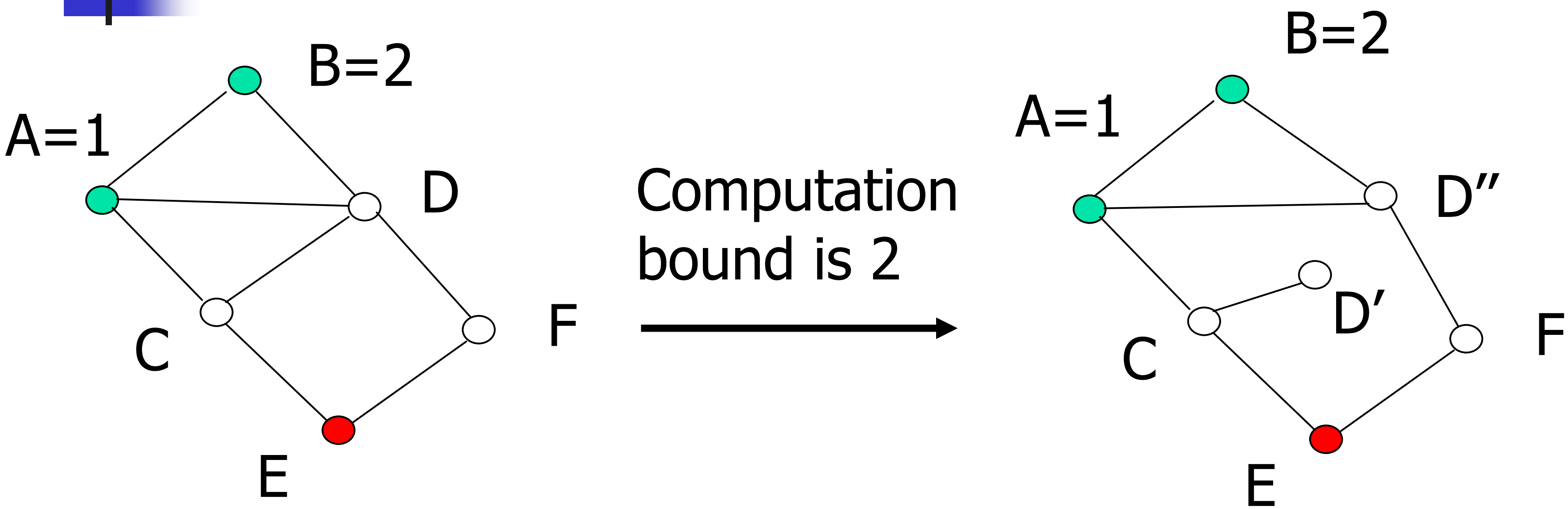
Min-Conflict Heuristic

- Each constraint $C(X_i)$ is represented by a cost function:
  - $f_C(X_i)=0$, iff $C(X_i)$ not violated, 1 otherwise
  - Solution is when $\sum f_j = 0$
- Basic function of interest used to guide BnB
  - MC = min ($\sum f_j$ | E, X_i) = lowest cost over assignments that agree with evidence E and assignment X_i
- BnB Search: Compute mc heuristic, lower bound on MC
  - Prune nodes S_i whose $mc(S_i)>0$
  - Allows dynamic variable ordering
  - Does not allow value ordering (all legal nodes have mc=0)

Solution-Count Heuristic

- Each constraint $C(X_i)$ is represented by a solution count function:
  - $f_C(X_i)=1$, iff $C(X_i)$ not violated, 0 otherwise
  - Solution is when $\prod f_j = 1$
- Basic function of interest used to guide BnB
  - $SC = \text{sum}\ (\prod f_j \mid E, X_i)$ = number of solutions that agree with evidence E and assignment X_i
- BnB Search
  - Compute sc heuristic, lower bound on SC
  - Prune nodes S_i whose $mc(S_i)=0$
  - Allows dynamic variable and value ordering

Example



	E=1	E=2	E=3
MC:	0	1	0
SC:	1	0	2

	E=1	E=2	E=3
mc:	0	0	0
sc:	.4	.2	.4

Heuristics & Algorithms

- Need lower bounds for BnB

- Compute
  - min $(\sum f_j \mid E, X_i)$ – min conflicts
  - sum $(\prod f_j \mid E, X_i)$ – solution count

- Using
  - MC(i)
  - IJGP(i)

BnB Search

- IJGP [MC, SC] + MBTE [MC, SC]
  - **IJGP-SC**
  - IJGP-MC
  - **MBTE-SC**
  - **MBTE-MC**
  - **MBTE-MC + IJGP-SC**
  - etc.

N=200,K=4,C=760,T=4, 5 min

	Time	w*	# cons	# incons	# nodes	# BTs
BnB IJGP-SC w-pruning 2	56	81	60	0	744	406
BnB IJGP-SC w-pruning 3	28	81	60	0	300	71
BnB MBTE-MC 0 IJGP-SC 2	83	81	30	0	311	290
BnB MBTE-MC 0 IJGP-SC 3	116	81	40	0	325	301
SLS	25	81	70		350238	
BnB MBTE-SC 2	75	81	10	0	453	210
BnB MBTE-SC 3	183	81	10	0	762	464

N=200,K=4,C=765,T=4, 5 min

		Time	w*	# cons	# incons	# nodes	# BTs
IJGP-SC	BnB IJGP-SC w-pruning 2	29	82.8	30	0	424	167
	BnB IJGP-SC w-pruning 3	21	82.8	20	0	278	59
	BnB IJGP-SC w-pruning 4	31	82.8	30	0	309	88
MBTE-MC	BnB MBTE-MC 0 IJGP-SC 2	38	82.8	10	0	200	200
	BnB MBTE-MC 0 IJGP-SC 3	67	82.8	10	0	231	224
	BnB MBTE-MC 0 IJGP-SC 4						
	SLS	39	82.8	30		525176	
MBTE-SC	BnB MBTE-SC 2	95	82.8	20	0	500	240
	BnB MBTE-SC 3	85	82.8	20	0	358	131
	BnB MBTE-SC 4	91	82.8	20	0	272	54

N=300,K=4,C=1125,T=4 10 min

	w*	time	solved	nodes		
IJGP-SC 2	112	24	4			
IJGP-SC 3	112	30	3			
SLS	112	28	4	450K		

Summary

- A new algorithm for solving CSP, based on Solution Counting heuristic computed by IJGP
- Competitive with CSP, best algorithm in practice for random CSPs; SLS is incomplete, our new algorithm is complete
- Strength: good for consistent problems
- Weakness: not as good for inconsistent problem

Conclusions

- We introduce two general BnB schemes that generate bounding heurisics automatically.
- BBBT and BBMB do not dominate each other.
  - When large *i-bounds* are effective, BBMB is more powerful.
  - However, when space is at issue, BBBT with small *i-bound* is often more powerful.
- BBBT/BBMB together are often superior to stochastic local search, except in cases when the domain size is small, in which case they are competitive.
- BBBT/BBMB as complete algorithms can prove optimality if given enough time, unlike SLS.
- BBBT can be extended to CSPs: heuristics based on approximate counting are very promising.