On The Feasibility Of Distributed Constraint Satisfaction

Rina Dechter, UC-Irvine

- Z. Collin and S. Katz, Technion, Israel
- R. Mateescu, K. Kask, UC-Irvine

Overview

- Distributed network consistency; the self-stabilization framework, revisited (Collin, Dechter and Katz, 1991, 1995)
- Distributed view of structured constraint propagation and tree-clustering
- Distributing hybrids of arc-consistency and stochastic local consistency

The network consistency problems: Outline

- The problem and the distributed model
- Feasibility
- The network consistency (NC) protocol
- The tree consistency (TC) protocol
- Conclusions

Constraint networks

A *constraint network* is a triple

$R = \langle X, D, C \rangle$ where:

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

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

$C = \{C_1, ..., C_t\}, C_i = (S_i, R_i)$ are the constraints,

S_i being the *scope* of the *relation* R_i.

The main task:

- Determine if the problem has a solution (an assignment that satisfies all the relations);
- If yes, find one or all of them.

R_1

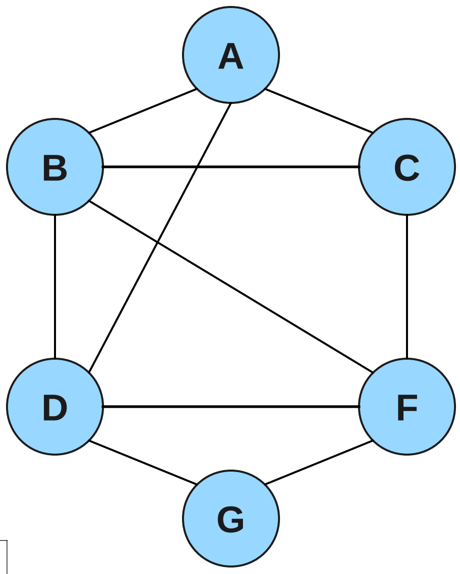
A
1
2
3

R_2

A	B
1	2
1	3
2	1
2	3
3	1
3	2

R_3

A	C
1	2
3	2



R_4

A	B	D
1	2	3
1	3	2
2	1	3
2	3	1
3	1	2
3	2	1

R_5

B	C	F
1	2	3
3	2	1

R_6

D	F	G
1	2	3
2	1	3

The Distributed Model

- Constraint-based connectionist model:
  - Node $i \Leftrightarrow$ processor P_i
  - edge $(i, j) \Leftrightarrow$ communication link between i and j
- The Network-Consistency (NC) Problem
  - Given a communication network, each processor should select a value from its domain that is compatible with its neighbor's values (a binary csp)
- Two scheduling policies:
  - Distributed, allows Parallelism – A subset of processors activated simultaneously
  - Central scheduler: One processor is activated at a time

Motivation and Assumptions of the Distributed Model

Applications: Communication radio networks, Multi-agents

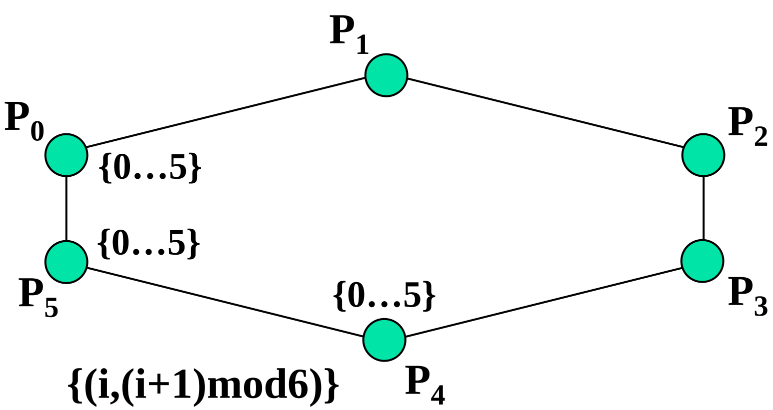
- **Limited computation power –** (cannot move all information to one node.) Pi is a finite state machine.
- **Computation is only with neighbors** (communicaion network mirrors the constraint graph)
- **Self stabilizing protocol:** Execution starts from any initial configuration, good for error correction
- **All processors are identically programmed**

Feasibility

- **Question:** Can we define a decision function so that the network will converge to a consistent solution from any initial configuration? For any schedule?
- **Answer: NO!** – not in a uniform model, even when one processor is activated each time.
- **Yes!** – in an almost uniform model, where all processors, but one, are identical.
- **Result:** we present a self-stabilizing, asynchronous, almost uniform NC-protocol.

Feasibility

- **Theorem:** No uniform self stabilizing protocol can solve the networks consistency problem even when a single processor is activated at a time (under the central demon).
- **Proof:** ring ordering problem



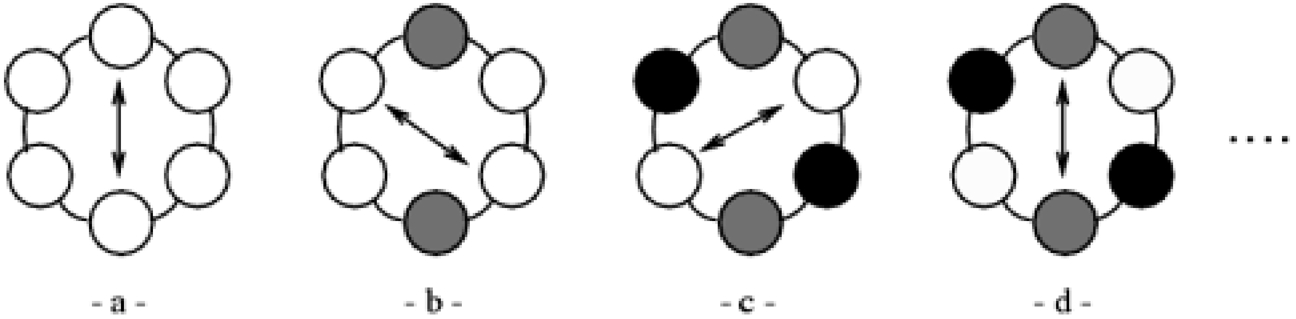
$n = 6 = 3 \cdot 2$

…

P_0	P_q	P_{2q}		$P_{(r-1)q}$
P				$P_{(r-1)q+1}$
P_{q-1}	P_{2q-1}			P_{rq-1}

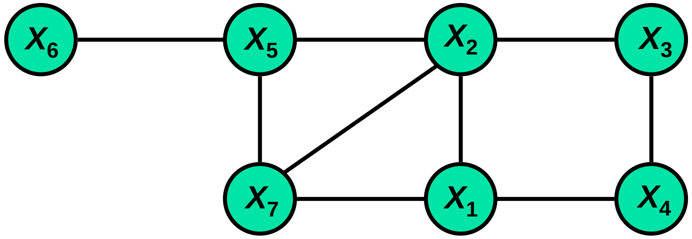
Feasibility

- **Theorem:** No uniform self stabilizing protocol can solve the networks consistency problem even when a single processor is activated at a time.
- **Proof:** ring ordering problem (n = 2x3= 6). Schedule (1,4,2,5,3,6). Initial states are identical.



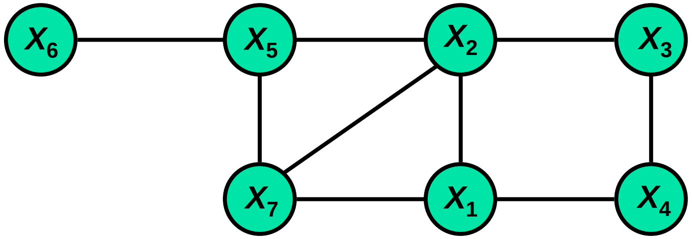
From Sequential to Distributed

- The network consistency protocol is based on sequential Backjumping with DFS ordering.
- The network

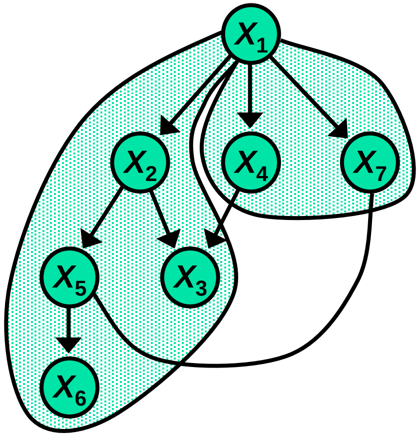


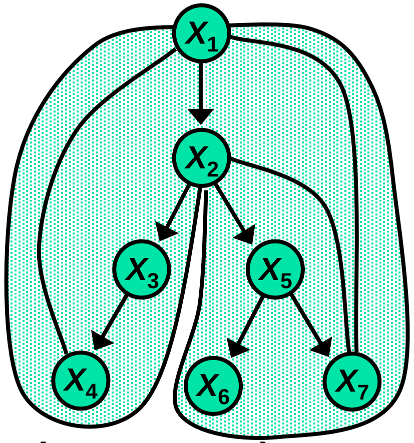
From Sequential to Distributed

- Based on sequential Backjumping with DFS ordering.
- The network



- BFS ordering vs DFS ordering



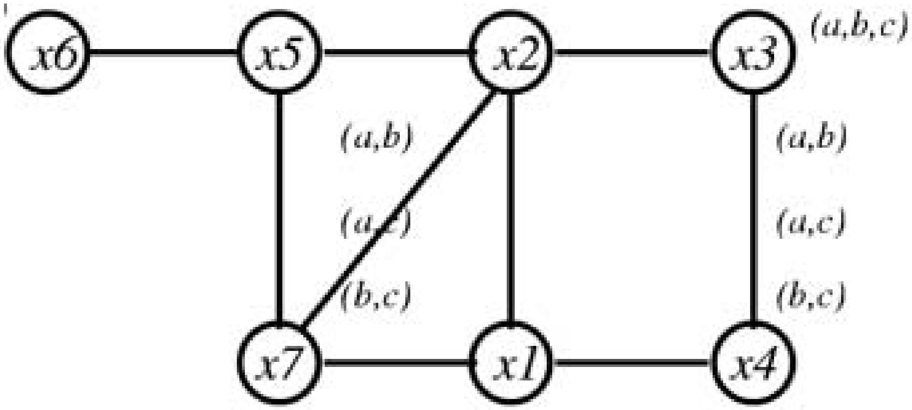


- DFS orderings allows parallelism (Freuder & Quinn, 1987)

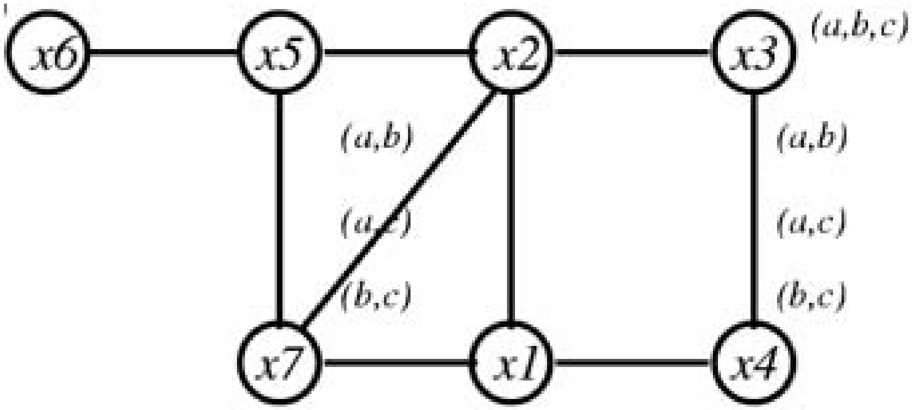
The Network Consistency protocol, Almost uniform, stochastic

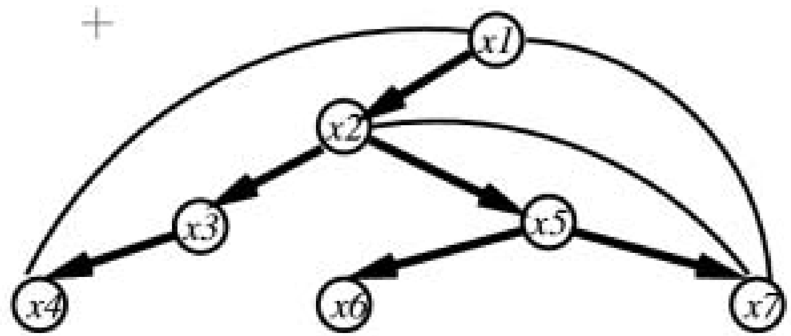
- Three sub-protocols
  - DFS spanning tree generation
  - Activation control mechanism
  - Consistent assignment generation

The DFS search tree protocol

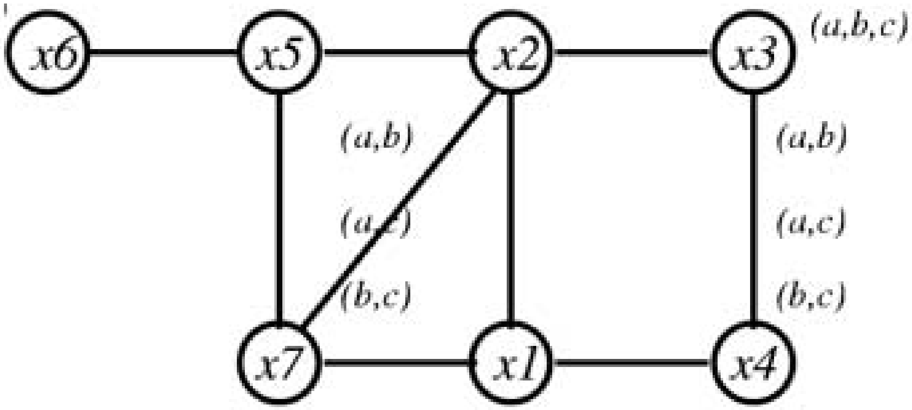


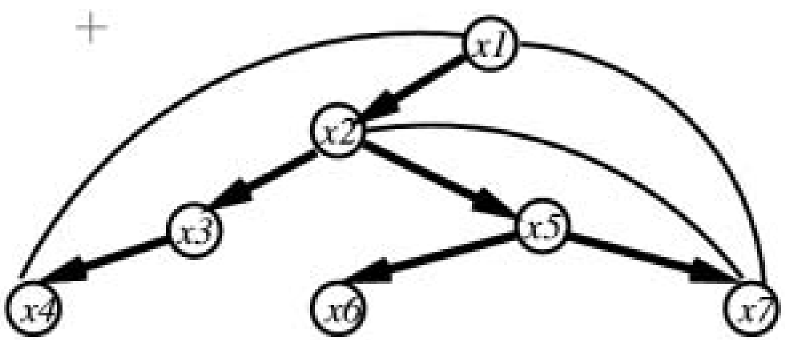
The DFS search tree protocol



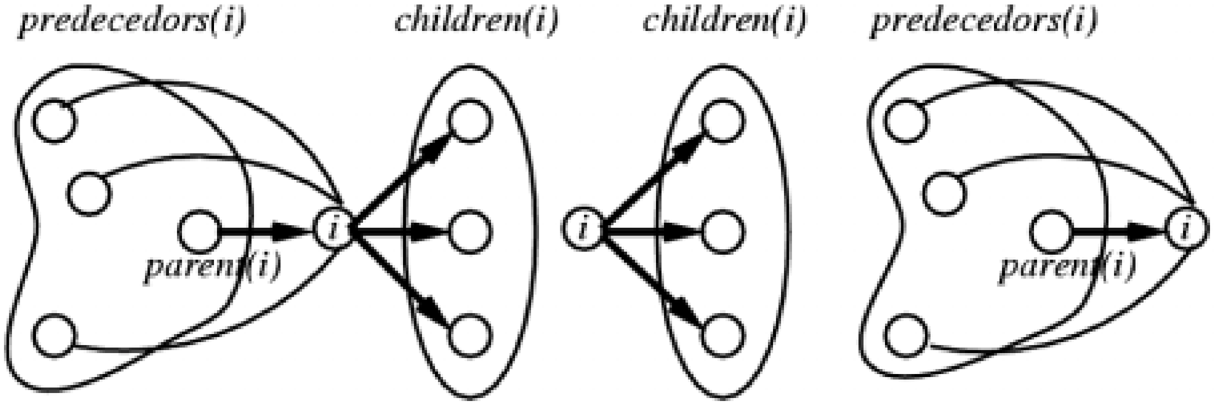


The DFS spanning tree generation protocol





The protocol creates a tree expressed by:



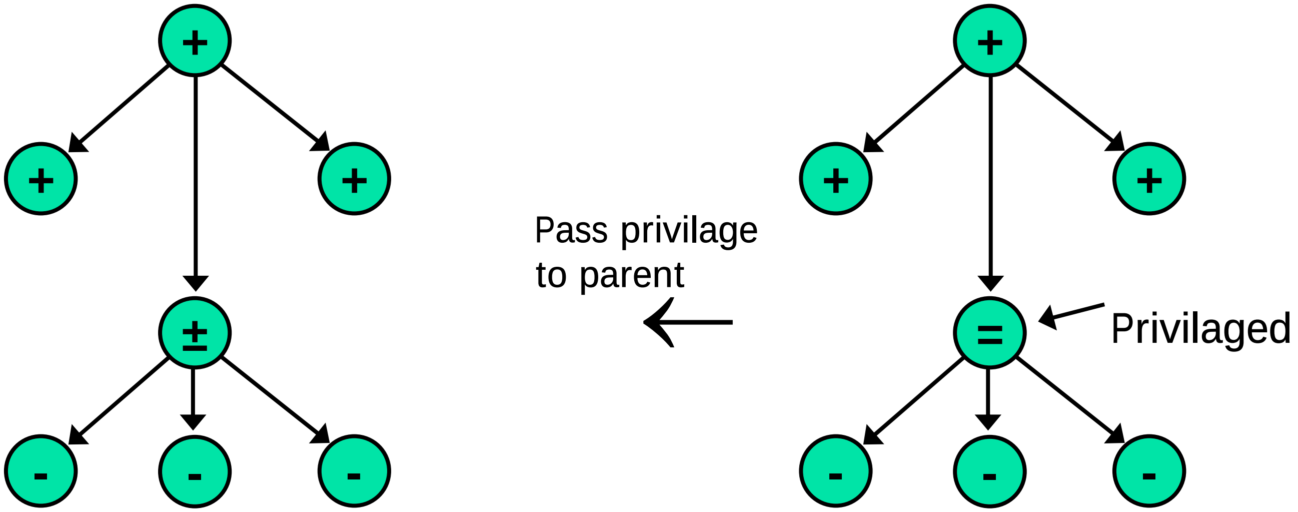
Control Activation Mechanism

- The activation mechanism extends Dijkstra’s balance/unbalance scheme for two processors.
- A processor changes its state when it is privileged.

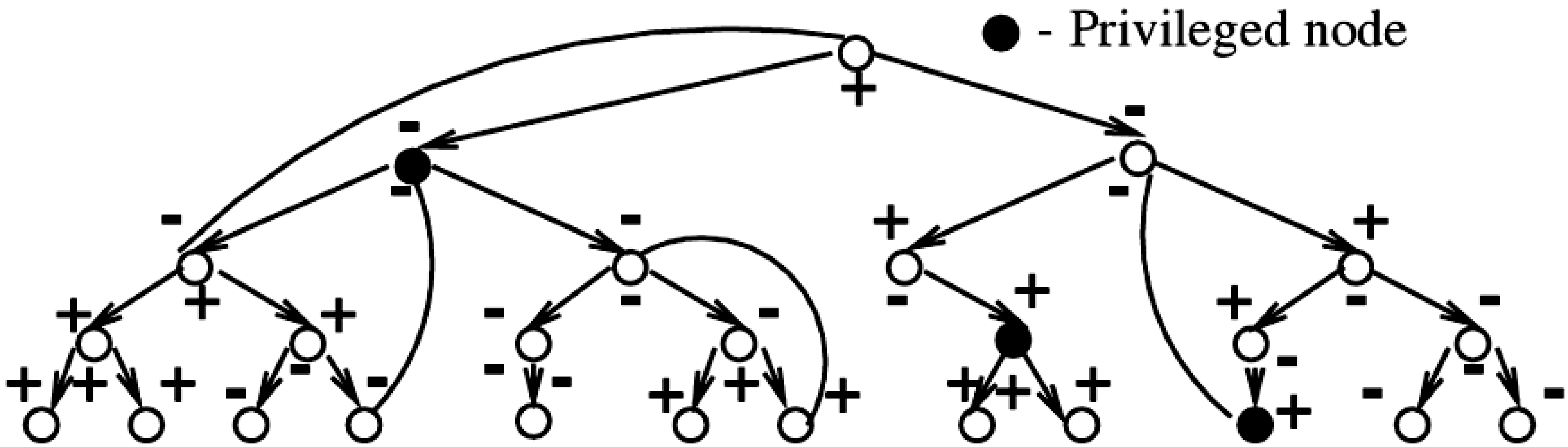
Privilege

- **Control activation**: a processor changes state when privilaged

- Every node has two boolean fields:
  - **parent-tag** referring to its inlink,
  - **children-tag** referring to its outlink.
- A processor is privileged when its inlink is **unbalanced** and its outlinks are **balanced.**



Privilage passing



The NC Protocol

- **When the privilege comes from the parent**
  - Assign a consistent value, if possible, and pass privilege to the children; or else, assign a "dead-end" and return the privilege to the parent.
- **If privilege came from children**
  - Try another value, or else return privilege to parent.

Properties of NC Protocol

- Every node verifies consistency only against its relevant ancestors.
- Only privileged nodes change their states (i.e. reassign value + pass privilege).
- Eventually there is no more than one privileged node on every path from the root to a leaf.
- The privileges travel along the tree backwards and forwards from the root to the leaves.

Theorem

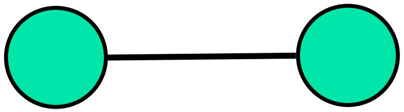
- The NC protocol converges to a solution if one exists from any initial configuration.
- Complexity: exponential in depth of DFS tree.

The network consistency problems: Outline

- The problem and the distributed model
- Feasibility
- The network consistency (NC) protocol
- The tree consistency (TC) protocol
- Conclusions

Trees

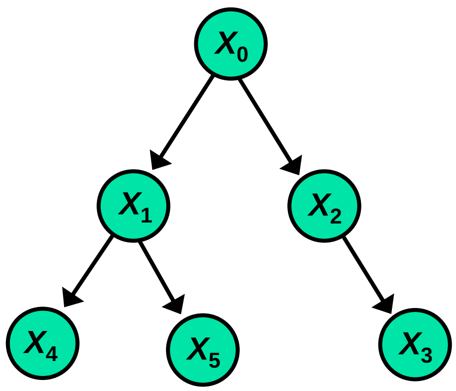
- No uniform, self stabilizing protocol can solve the *tree-consistency* problem under the **distributed scheduler**

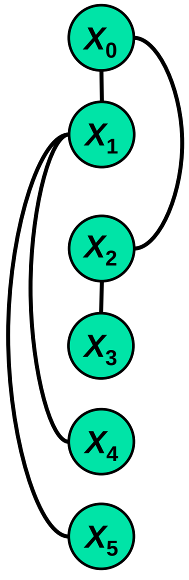


- If start from identical states and activated simultaneously
- **TC Uniform Protocol under central demon**
  - Generating directed tree
  - Arc-consistency
  - Directed value assignment

Arc Consistency

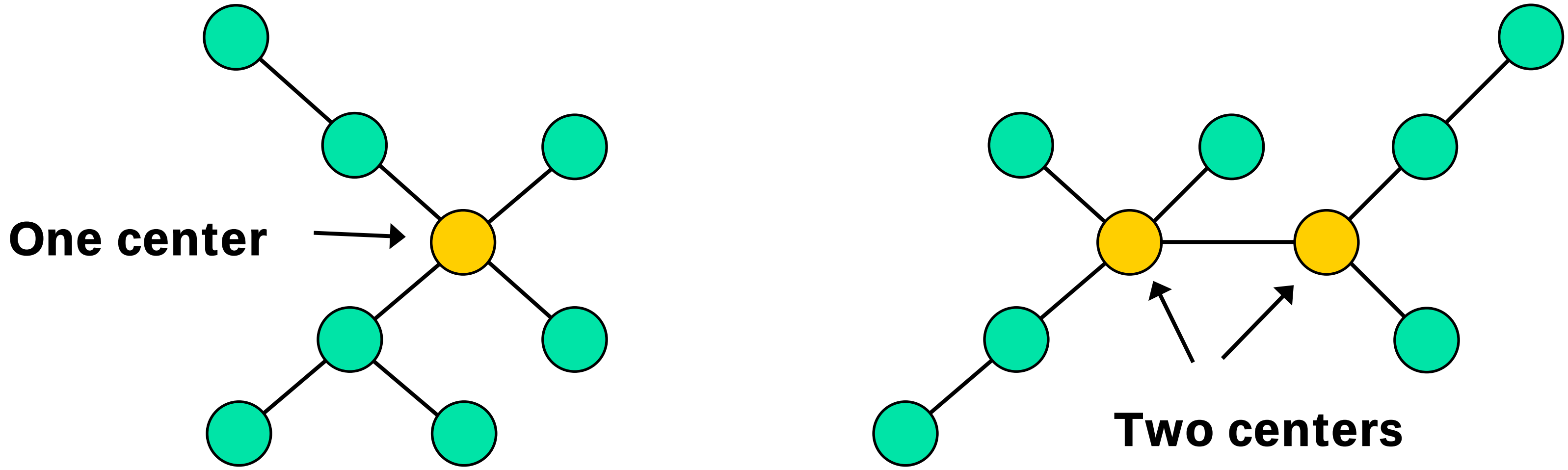
- Remove from *Di* all values inconsistent with any of Xi's neighbors





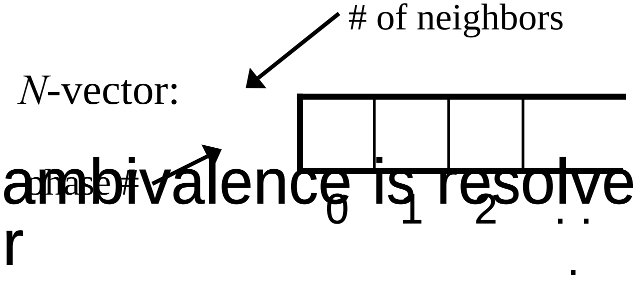
Generating a Pseudo Tree

- **Center of the tree** – a node whose maximal distance from a leaf is minimal



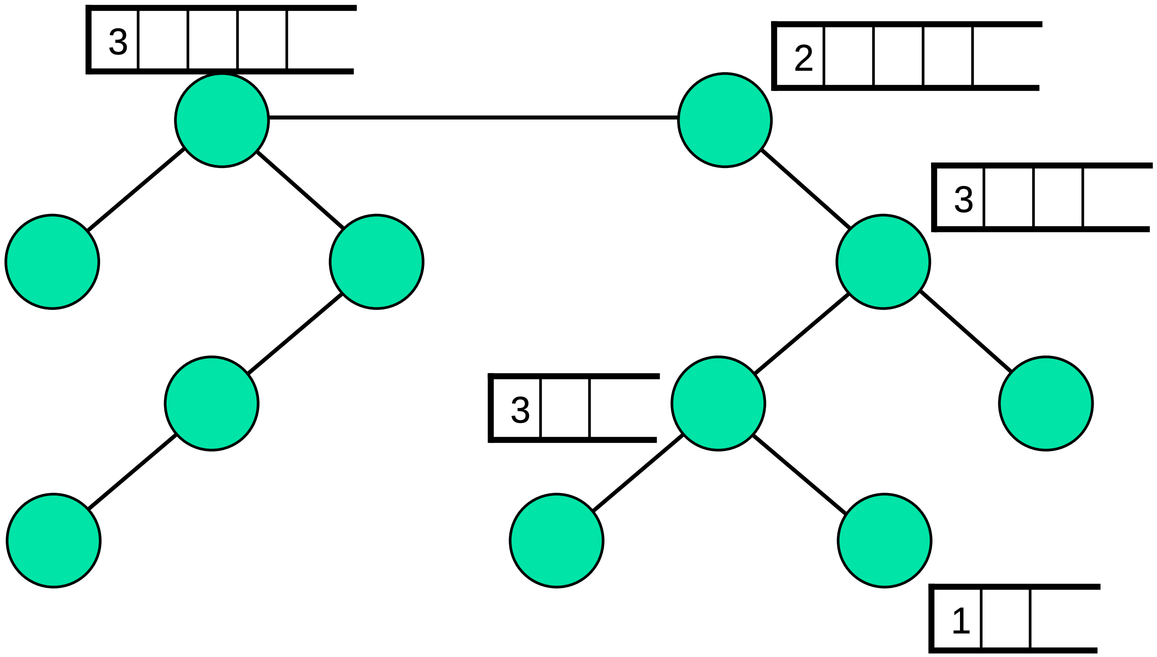
Finding the Centers

- **Sequentially:**
  - Remove the leaves of the tree in phases.
  - The remaining node(s) (1 or 2) are (is) the center(s).
- **In parallel:**
  - Simulate the sequential algorithm by using N-vectors to represent phases.

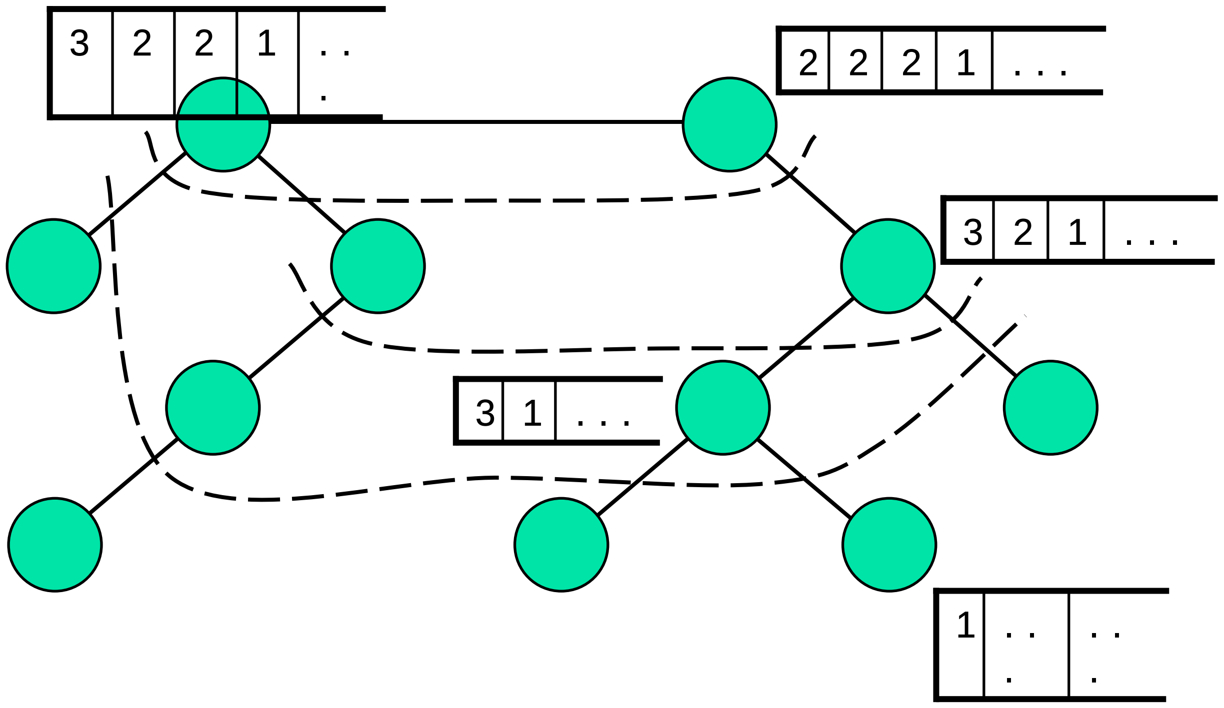


  - The ambivalence is resolved by the scheduling order

Resolving Ambiguity by Scheduling Order



Resolving Ambiguity by Scheduling Order



Summary

- Feasibility:
  - Trees:
    - Uniform and single node activation (synchronous),
    - Almost uniform and asynchronous
  - Networks:
    - almost uniform and asynchronous
- Complexity:
  - NC protocol is exponentially in the depth of the DFS tree.
  - TC protocol is linear

Conclusions

- A uniform, self-stabilizing algorithm for solving the NC problem is not possible.
- A uniform self –stabilizing protocol is realizable under the distributed demon. Its complexity is exp in depth of dfs tree.
- The TC protocol is uniform and works under a central demon.
- Both protocols are **self stabilizing**, where any consistent solution is a **stable pattern**.
- Question: under what scheduling policies a uniform self stabilizing protocol do exists?

Overview

- Distributed network consistency; the self-stabilization framework, revisited
- Distributed structured constraint propagation
- Distributing hybrids of arc-consistency and stochastic local consistency

Constraint networks

A *constraint network* is a triple

$R = < X, D, C >$ where:

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

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

$C = \{C_1, ..., C_t\}, C_i = (S_i, R_i)$ are the constraints,

S_i being the *scope* of the *relation* R_i.

R_2

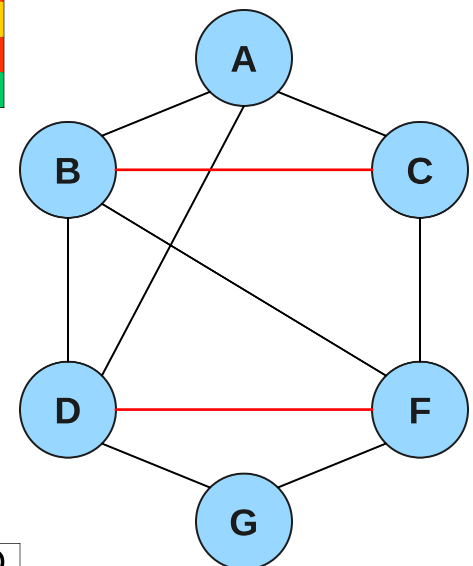
A	B
1	2
1	3
2	1
2	3
3	1
3	2

R_1

A
1
2
3

R_3

A	C
1	2
3	2



R_5

B	C	F
1	2	3
3	2	1

R_4

A	B	D
1	2	3
1	3	2
2	1	3
2	3	1
3	1	2
3	2	1

R_6

D	F	G
1	2	3
2	1	3

The main task:

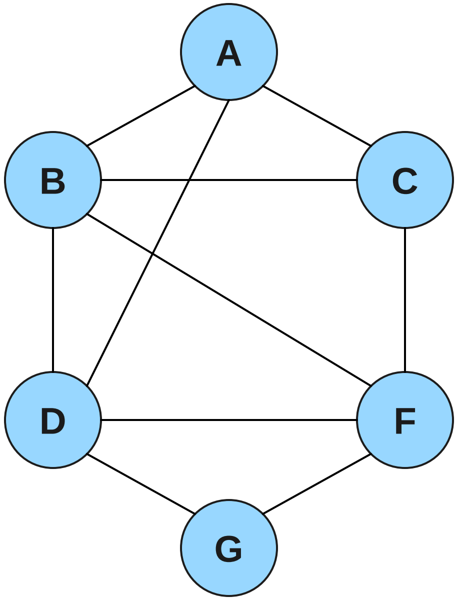
- Determine if the problem has a solution (an assignment that satisfies all the relations);
- If yes, find one or all of them.

Distributed Arc-Consistency

- Arc-consistency can be formulated as a distributed algorithm:

$$D_i^j \leftarrow \pi_j(R_{ij} \bowtie D_i) \qquad (1)$$

$$D_i \leftarrow D_i \cap (\bowtie_{k\in ne(i)} D_k^i) \qquad (2)$$



a Constraint network

DRAC on the dual join-graph

R_1

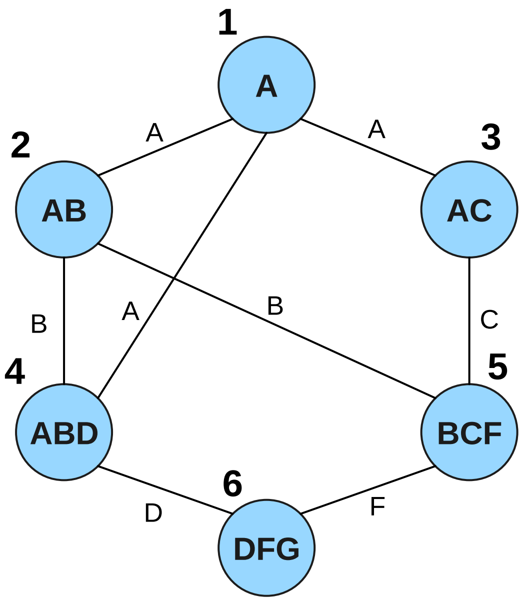
A
1
2
3

R_2

A	B
1	2
1	3
2	1
2	3
3	1
3	2

R_3

A	C
1	2
3	2



R_4

A	B	D
1	2	3
1	3	2
2	1	3
2	3	1
3	1	2
3	2	1

R_5

B	C	F
1	2	3
3	2	1

R_6

D	F	G
1	2	3
2	1	3

Distributed Relational Arc-Consistency

- DRAC can be applied to the dual problem of any constraint network:

$$h_i^j \leftarrow \pi_{l_{ij}}(R_i \bowtie (\bowtie_{k \in ne(i)} h_k^i)) \qquad (1)$$

$$R_i \leftarrow R_i \cap (\bowtie_{k \in ne(i)} h_k^i) \qquad (2)$$

DR-AC on the dual graph

R_1

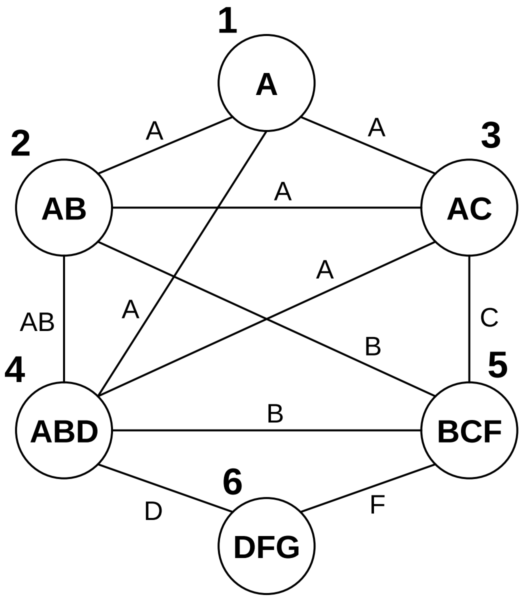
A
1
2
3

R_2

A	B
1	2
1	3
2	1
2	3
3	1
3	2

R_3

A	C
1	2
3	2



R_4

A	B	D
1	2	3
1	3	2
2	1	3
2	3	1
3	1	2
3	2	1

R_5

B	C	F
1	2	3
3	2	1

R_6

D	F	G
1	2	3
2	1	3

Iteration 1

R_1

A
1
2
3

h_2^1

A
1
2
3

h_3^1

A
1
3

h_4^1

A
1
2
3

h_5^2

B
1
3

h_4^2

A	B
1	2
1	3
2	1
2	3
3	1
3	2

h_3^2

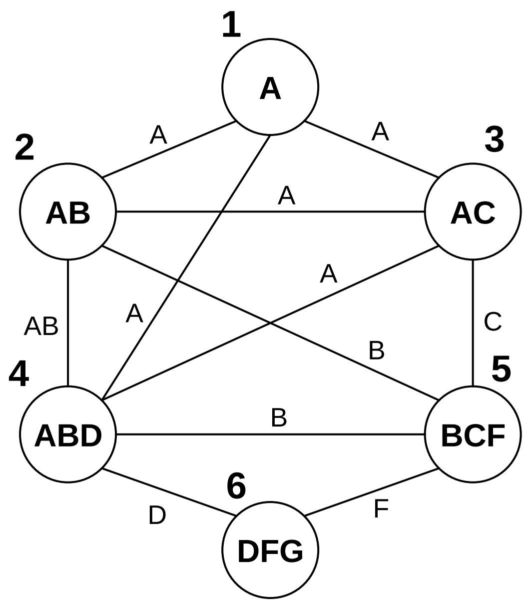
A
1
3

h_1^2

A
1
2
3

R_2

A	B
1	2
1	3
2	1
2	3
3	1
3	2



R_3

A	C
1	2
3	2

h_1^3

A
1
2
3

h_2^3

A
1
2
3

h_4^3

A
1
2
3

h_5^3

C
2

h_6^4

D
1
2

h_5^4

B
1
3

h_3^4

A
1
3

h_2^4

A	B
1	2
1	3
2	1
2	3
3	1
3	2

h_1^4

A
1
2
3

R_4

A	B	D
1	2	3
1	3	2
2	1	3
2	3	1
3	1	2
3	2	1

R_5

B	C	F
1	2	3
3	2	1

h_2^5

B
1
2
3

h_3^5

C
2

h_4^5

B
1
2
3

h_6^5

F
1
3

R_6

D	F	G
1	2	3
2	1	3

h_4^6

D
1
2
3

h_5^6

F
1
3

$$h_i^j \leftarrow \pi_{l_{ij}}(R_i \bowtie (\bowtie_{k \in ne(i)} h_k^i)) \qquad (1)$$

Iteration 1

R_1

A
1
2
3

h_2^1

A
1
2
3

h_3^1

A
1
3

h_4^1

A
1
2
3

h_5^2

B
1
3

h_4^2

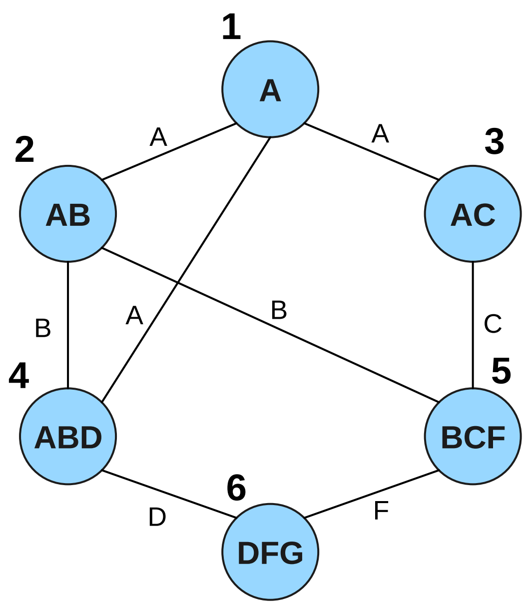
B
1
2
3

h_1^2

A
1
2
3

R_2

A	B
1	2
1	3
2	1
2	3
3	1
3	2



R_3

A	C
1	2
3	2

h_1^3

A
1
2
3

h_5^3

C
2

h_6^4

D
1
2

h_2^4

B
1
2
3

h_1^4

A
1
2
3

R_4

A	B	D
1	2	3
1	3	2
2	1	3
2	3	1
3	1	2
3	2	1

R_5

B	C	F
1	2	3
3	2	1

h_2^5

B
1
2
3

h_3^5

C
2

h_6^5

F
1
3

R_6

D	F	G
1	2	3
2	1	3

h_4^6

D
1
2
3

h_5^6

F
1
3

$$R_i \leftarrow R_i \cap (\bowtie_{k \in ne(i)} h_k^i) \quad (2)$$

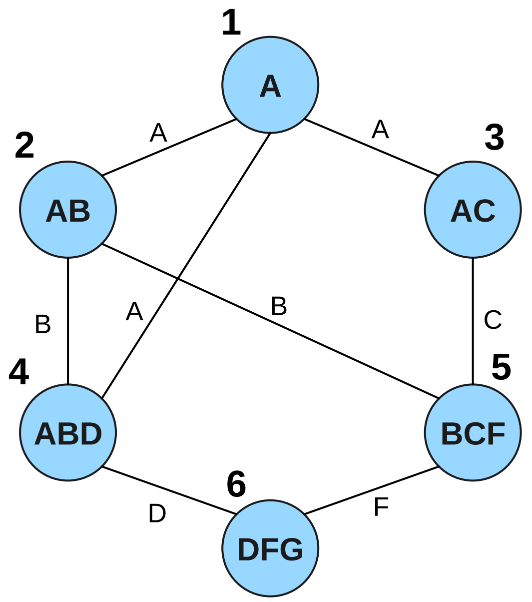
Iteration 1

R_1

A
1
3

R_2

A	B
1	3
2	1
2	3
3	1



R_3

A	C
1	2
3	2

R_4

A	B	D
1	3	2
2	3	1
3	1	2
3	2	1

R_5

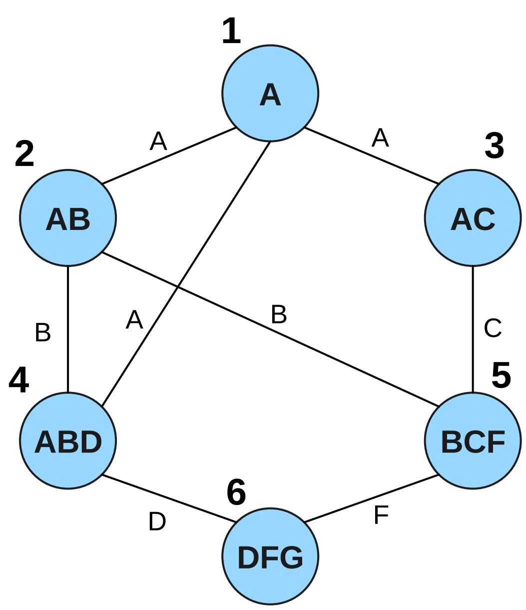
B	C	F
1	2	3
3	2	1

R_6

D	F	G
2	1	3

$$h_i^j \leftarrow \pi_{l_{ij}}(R_i \bowtie (\bowtie_{k \in ne(i)} h_k^i)) \qquad (1)$$

Iteration 2



R_1

A
1
3

h_2^1

A
1
2
3

h_3^1

A
1
3

h_4^1

A
1
2
3

h_5^2

B
1
3

h_4^2

B
1
2
3

h_1^2

A
1
3

R_2

A	B
1	3
2	1
2	3
3	1

R_3

A	C
1	2
3	2

h_1^3

A
1
3

h_5^3

C
2

h_6^4

D
2

h_2^4

B
1
3

h_1^4

A
1
3

R_4

A	B	D
1	3	2
2	3	1
3	1	2
3	2	1

R_5

B	C	F
1	2	3
3	2	1

h_2^5

B
1
3

h_3^5

C
2

h_6^5

F
1

R_6

D	F	G
2	1	3

h_4^6

D
1
2

h_5^6

F
1
3

$$R_i \leftarrow R_i \cap (\bowtie_{k \in ne(i)} h_k^i) \quad (2)$$

Iteration 2

R_1

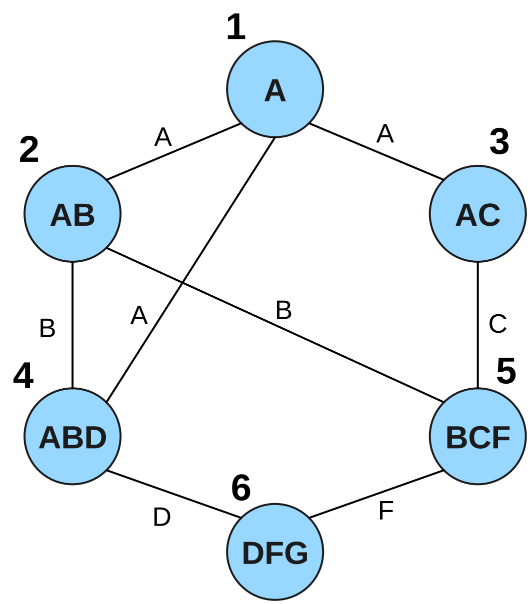
A
1
3

R_2

A	B
1	3
3	1

R_3

A	C
1	2
3	2



R_4

A	B	D
1	3	2
3	1	2

R_5

B	C	F
3	2	1

R_6

D	F	G
2	1	3

$$h_i^j \leftarrow \pi_{l_{ij}}(R_i \bowtie (\bowtie_{k \in ne(i)} h_k^i)) \quad (1)$$

Iteration 3

R_1

A
1
3

h_2^1

A
1
3

h_3^1

A
1
3

h_4^1

A
1
3

h_5^2

B
3

h_4^2

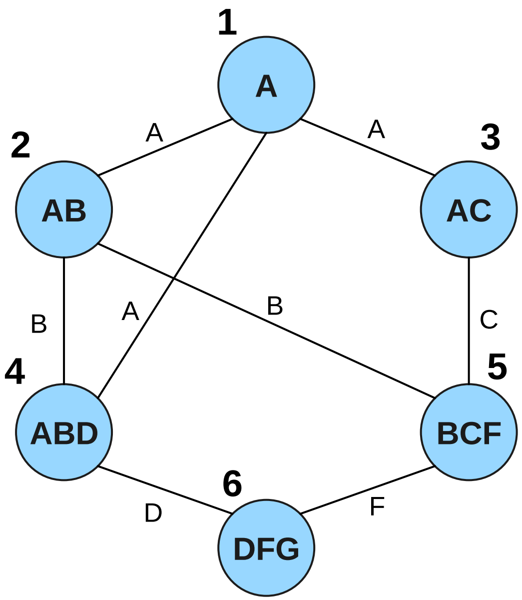
B
1
3

h_1^2

A
1
3

R_2

A	B
1	3
3	1



R_3

A	C
1	2
3	2

h_1^3

A
1
3

h_5^3

C
2

h_6^4

D
2

h_2^4

B
1
3

h_1^4

A
1
3

R_4

A	B	D
1	3	2
3	1	2

R_5

B	C	F
3	2	1

h_2^5

B
1
3

h_3^5

C
2

h_6^5

F
1

R_6

D	F	G
2	1	3

h_4^6

D
2

h_5^6

F
1

$$R_i \leftarrow R_i \cap \left(\bowtie_{k \in ne(i)} h_k^i \right) \quad (2)$$

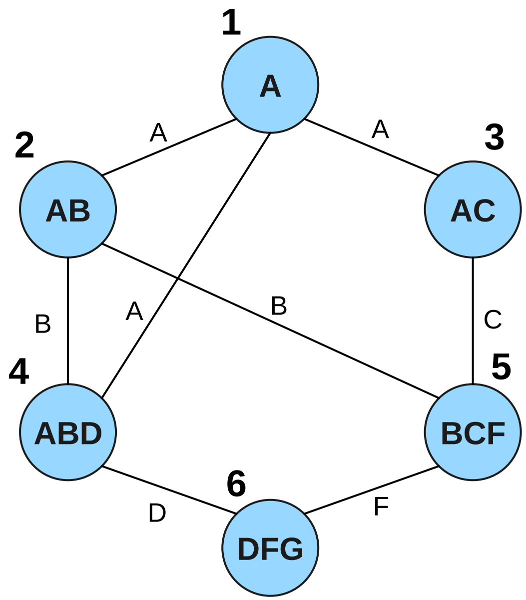
Iteration 3

R_1

A
1
3

R_2

A	B
1	3



R_3

A	C
1	2
3	2

R_4

A	B	D
1	3	2
3	1	2

R_5

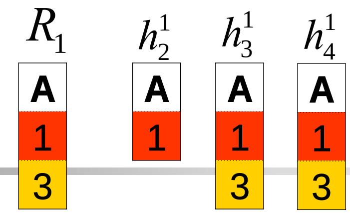
B	C	F
3	2	1

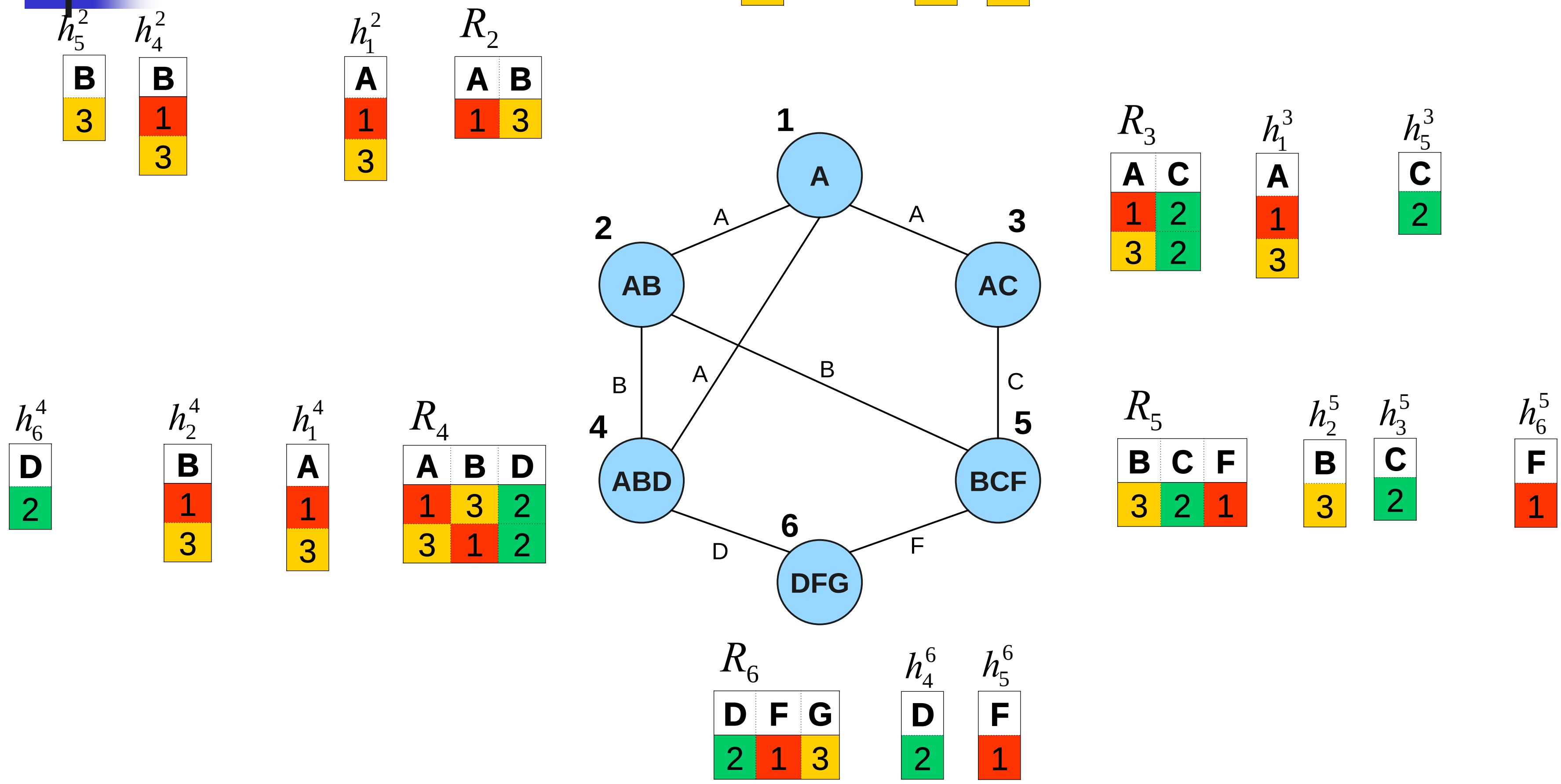
R_6

D	F	G
2	1	3

$$h_i^j \leftarrow \pi_{l_{ij}}(R_i \bowtie (\bowtie_{k \in ne(i)} h_k^i)) \quad (1)$$

Iteration 4





$$R_i \leftarrow R_i \cap (\bowtie_{k \in ne(i)} h_k^i) \qquad (2)$$

Iteration 4

R_1

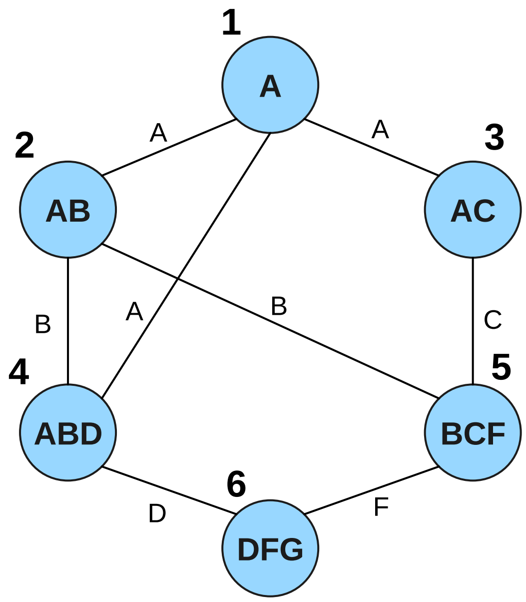
A
1

R_2

A	B
1	3

R_3

A	C
1	2
3	2



R_4

A	B	D
1	3	2

R_5

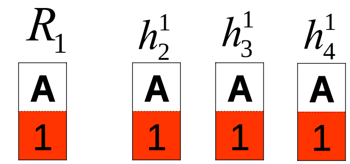
B	C	F
3	2	1

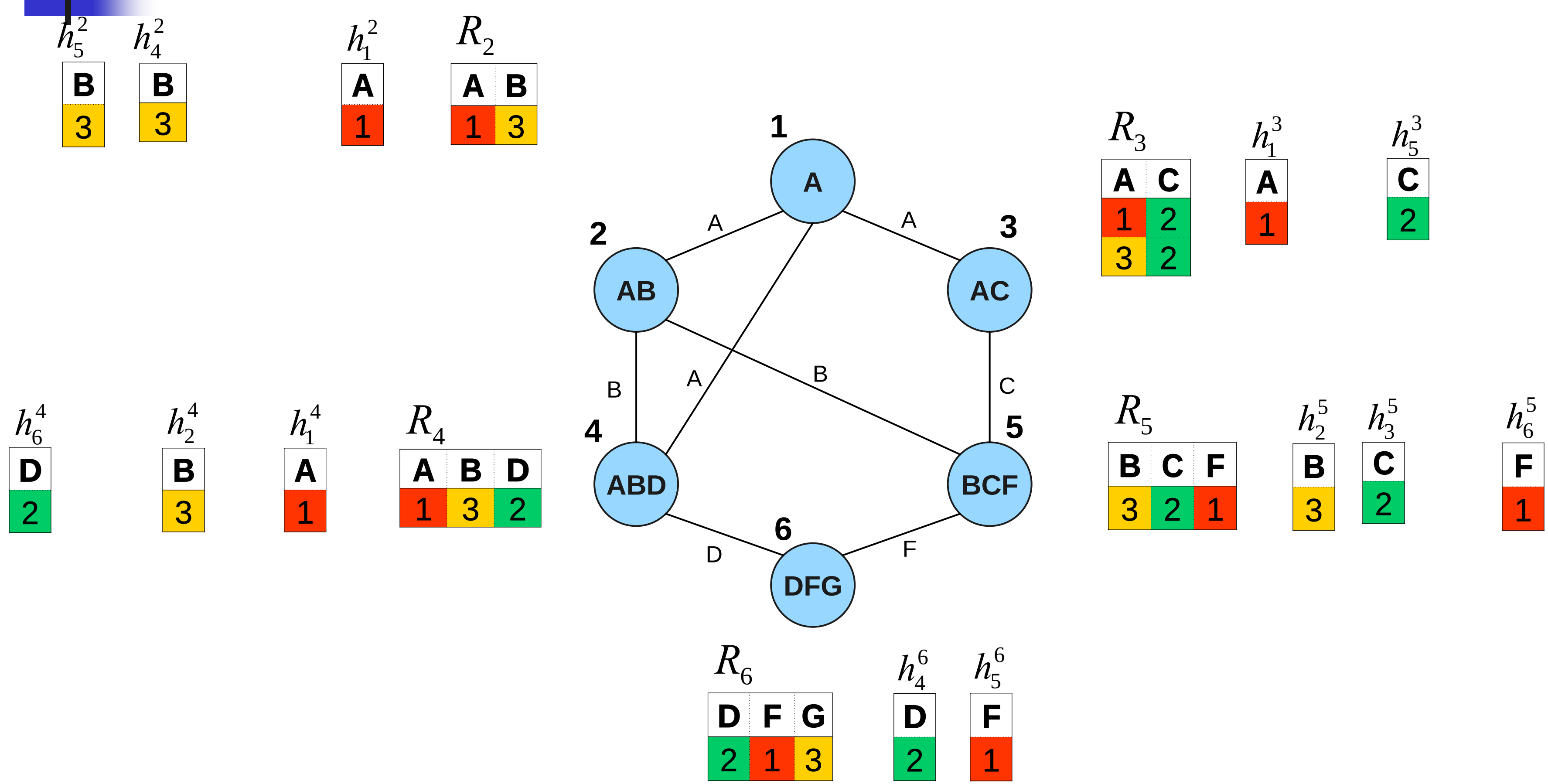
R_6

D	F	G
2	1	3

$$h_i^j \leftarrow \pi_{l_{ij}}(R_i \bowtie (\bowtie_{k \in ne(i)} h_k^i)) \qquad (1)$$

Iteration 5





$$R_i \leftarrow R_i \cap (\bowtie_{k \in ne(i)} h_k^i) \qquad (2)$$

Iteration 5

R_1

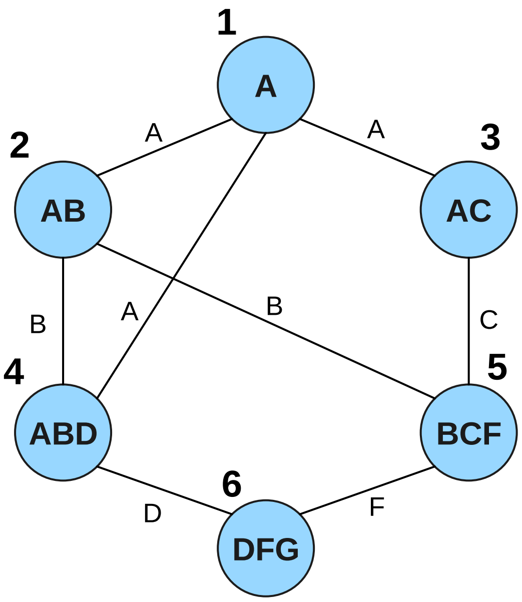
A
1

R_2

A	B
1	3

R_3

A	C
1	2



R_4

A	B	D
1	3	2

R_5

B	C	F
3	2	1

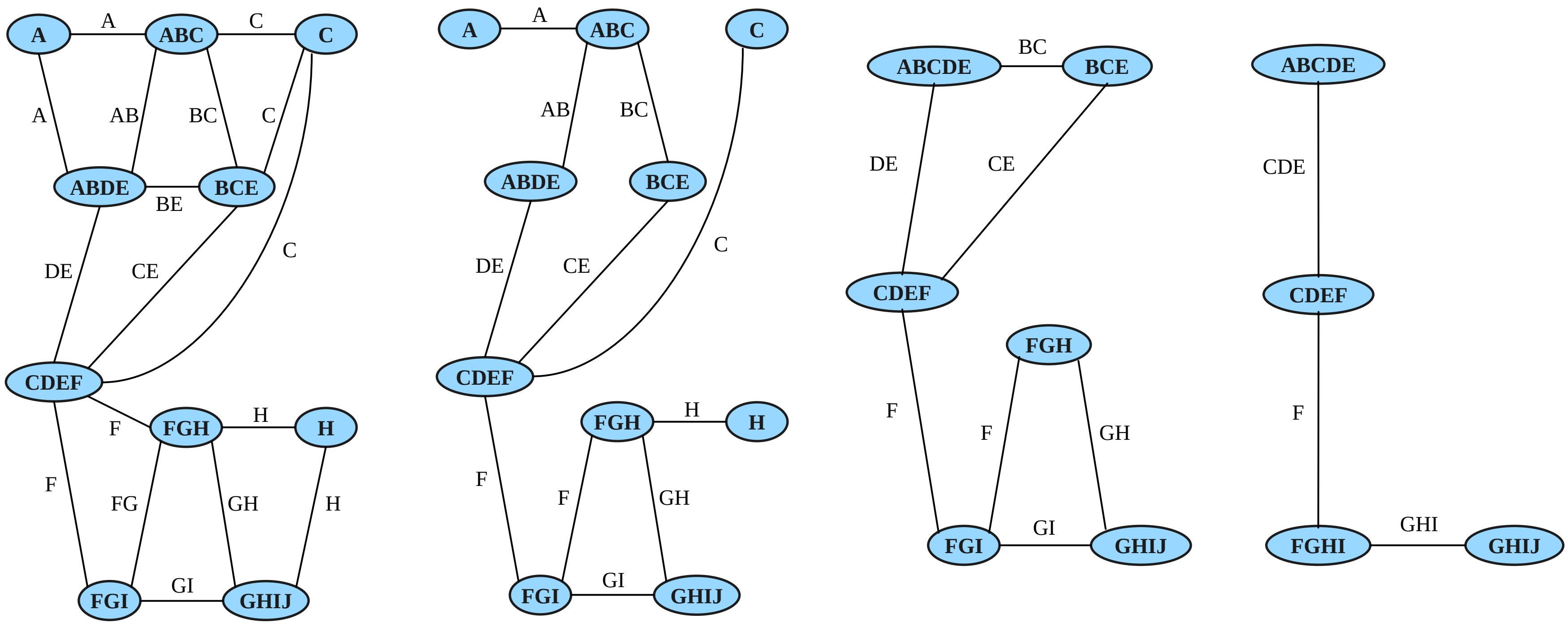
R_6

D	F	G
2	1	3

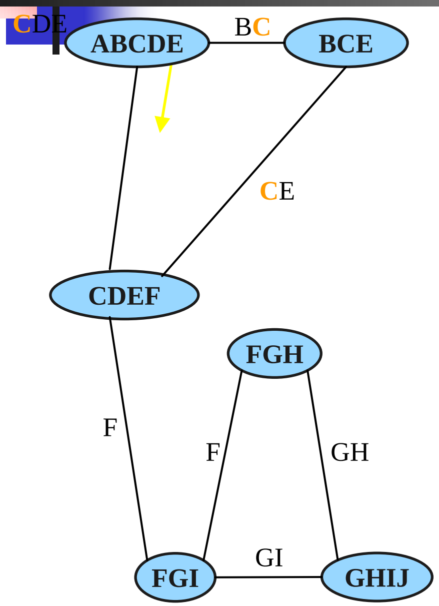
Extending DRAC to cluster join-graphs

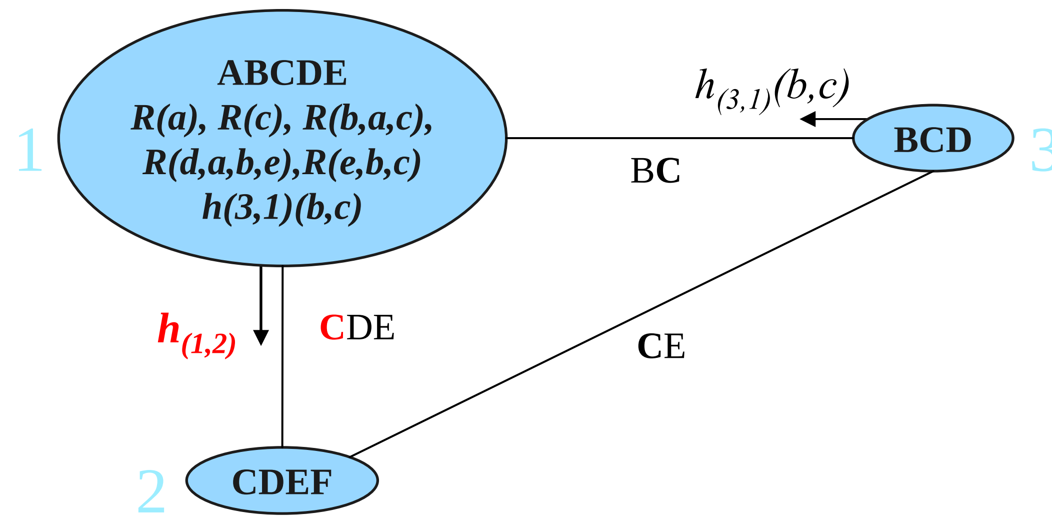
- Creates join-graphs where each cluster contains several constraints
- The graph satisfies the running intersection proprty: an equivalent problem
- Send the messages between connected clusters in the graph

Join-graphs


A
A
ABC
C
C
A
AB
BC
C
ABDE
BE
BCE
C
DE
CE
CDEF
F
FGH
H
H
F
FG
GH
H
FGI
GI
GHIJ
A
A
ABC
C
AB
BC
ABDE
BCE
C
DE
CE
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

Message propagation





Minimal arc-labeled:

sep(1,2)={D,E}

elim(1,2)={A,B,C}

$$h_{(1,2)}(de) = \sum_{a,b,c} p(a)p(c)p(b|ac)p(d|abe)p(e|bc)h_{(3,1)}(bc)$$

Non-minimal arc-labeled:

sep(1,2)={C,D,E}

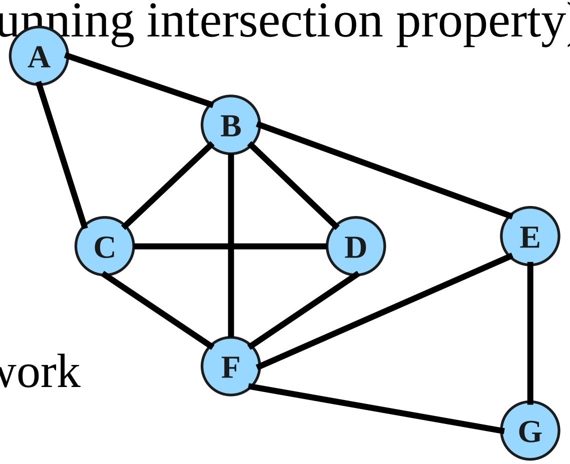
elim(1,2)={A,B}

$$h_{(1,2)}(cde) = \sum_{a,b} p(a)p(c)p(b|ac)p(d|abe)p(e|bc)h_{(3,1)}(bc)$$

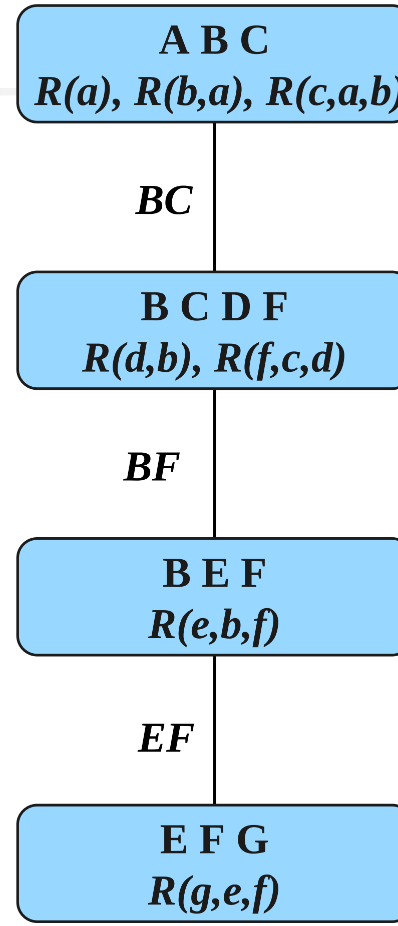
Tree decompositions

A *tree decomposition* for a belief network $BN = \langle X, D, G, P \rangle$ is a triple $\langle T, \chi, \psi \rangle$, where $T = (V, E)$ is a tree and χ and ψ are labeling functions, associating with each vertex $v \in V$ two sets, $\chi(v) \subseteq X$ and $\psi(v) \subseteq P$ satisfying :

1. For each function $p_i \in P$ there is exactly one vertex such that $p_i \in \psi(v)$ and $scope(p_i) \subseteq \chi(v)$
2. For each variable $X_i \in X$ the set $\{v \in V | X_i \in \chi(v)\}$ forms a connected subtree (running intersection property)

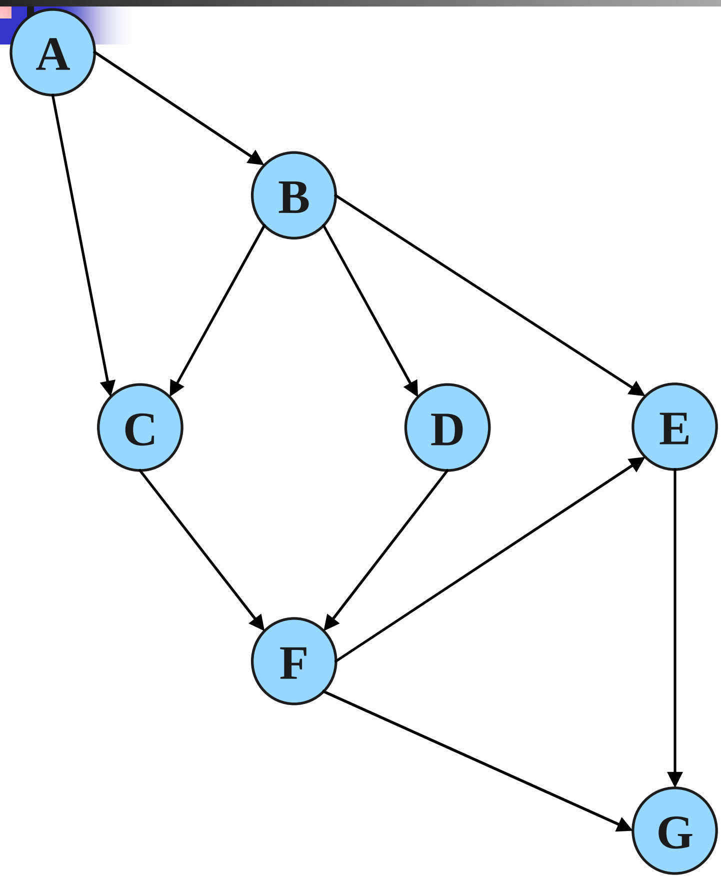


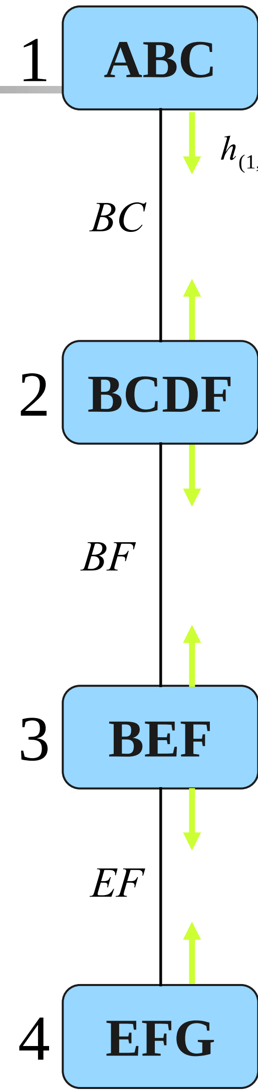
Belief network



Tree decomposition

Join-tree clustering - Example





$$h_{(1,2)}(B,C) = \prod_{BC} R(A) \times R(B,A) \times R(C,A,B)$$

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

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

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

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

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

Tree-decomposition: an architecture for distributed CSPs

- All the join-graphs, with/without min-arc labels represent equivalent problems.
- Each cluster is a subproblem solved centrally in one node. Each link corresponds to Relational arc-consistency.
- Only when the join-graph is a tree relational arc-consistency solved the problem exactly.
- For join-graphs we get an approximation that can be followed by the NC protocol over the arc-consistent join-graph

Tree-decomposition: an architecture for distributed CSPs

- Once message-passing is accomplished a solution can be accomplished by the tree-consistency (TC) protocol over the join-tree.
- Yields a uniform self-stabilizing algorithm when each cluster is a centralized computation.
- Exponential in cluster size

DRAC over join-trees

- Create a minimal arc join-tree dictating which constraints cooperate and what are the neighbors of clusters.
- Apply DRAC between clusters (uniform self-stabilizing, stochastic)
- Generate a solution using a TC protocol
- Complexity: exponential in number of variables in each cluster. Message passing like DRAC.
- Approximation: apply DRAC to a join-graph which is not a tree.

Overview

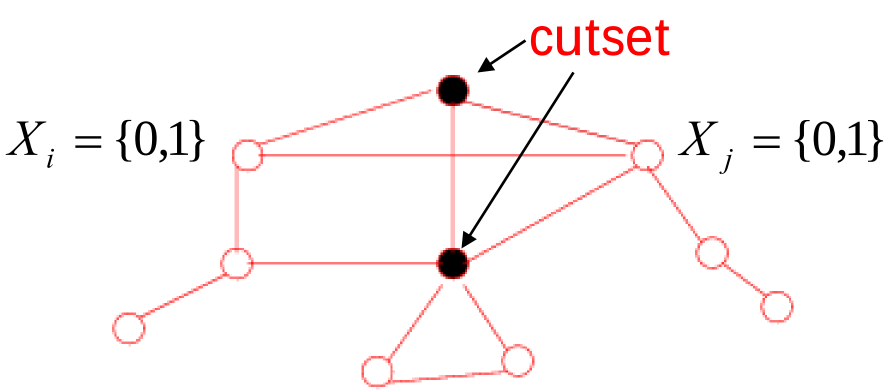
- Distributed network consistency; the self-stabilization framework, revisited
- Distributed structured constraint propagation
- Distributing hybrids of arc-consistency and stochastic local consistency

Approximating conditioning with elimination

Energy minimization in neural networks

(Pinkas and Dechter, 1995)

For **cycle-cutset nodes**, use the greedy update function (relative to neighbors).
For the rest of nodes, run the arc-consistency algorithm followed by value assignment.



GSAT with Cycle-Cutset

(Kask and Dechter, 1996)

Input: a CSP, a partition of the variables into **cycle-cutset** and **tree variables**

Output: an assignment to all the variables

Within each try:

Generate a random initial asignment,
and then alternate between the two steps:

1. Run **Tree algorithm** (arc-consistency+assignment) on the problem with fixed values of cutset variables.
2. Run GSAT on the problem with fixed values of tree variables.

Conslusions

- Distributed network consistency; the self-stabilization framework, revisited
- Distributed structured constraint propagation: complete and approximate
- Distributing hybrids of arc-consistency and stochastic local consistency