2011

Reformulating R(*,*m*)C with Tree Decomposition

Shant Karakashian
University of Nebraska - Lincoln, shantk@cse.unl.edu

Robert J. Woodward
University of Nebraska - Lincoln, rwoodwar@cse.unl.edu

Berthe Y. Choueiry
University of Nebraska - Lincoln, choueiry@cse.unl.edu

Reformulating R(*,*m*)C with Tree Decomposition

Shant Karakashian, Robert J. Woodward, Berthe Y. Choueiry

Constraint Systems Laboratory

University of Nebraska-Lincoln

Acknowledgements

- Experiments conducted at UNL's Holland Computing Center
- NSF Grant No. RI-111795

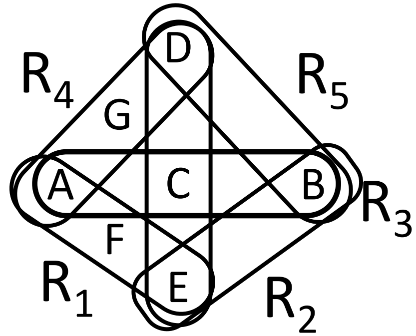
Outline

- Introduction
- R(*,*m*)C Property & Algorithm
- Exploit Tree Decomposition to
  - Avoid useless update & reduce propagation effort
    - ↪ Update queue: PROCESSQ ⇝ PROCESSMQ
    - ↪ The two algorithms *yield the same filtering*
  - Synthesize & add new constraints to improve propagation
    - ↪ Property enforced: R(*,*m*)C ⇝ T-R(*,*m*,*z*)C
    - ↪ The same algorithm *yields stronger filtering*
- Experimental Results
- Conclusion

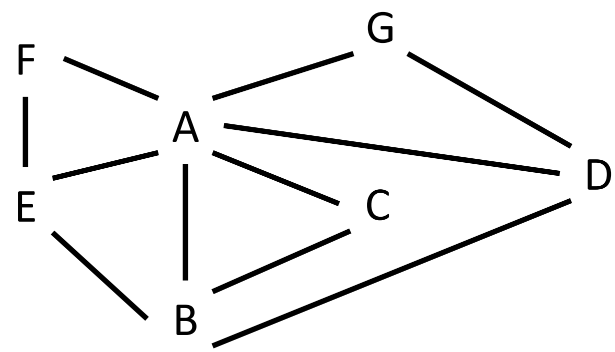
Constraint Satisfaction Problem

- CSP
  - Variables ($\mathcal{V}$), domains
  - Constraints: relations ($\mathcal{R}$), scope
- Representation
  - Hypergraph
  - Primal graph
  - Dual graph
- Solved with
  - Search
  - Enforcing consistency
- Warning
  - Consistency property vs. algorithms

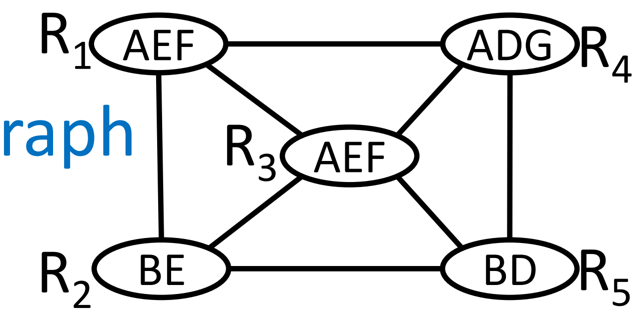
Hypergraph



Primal graph

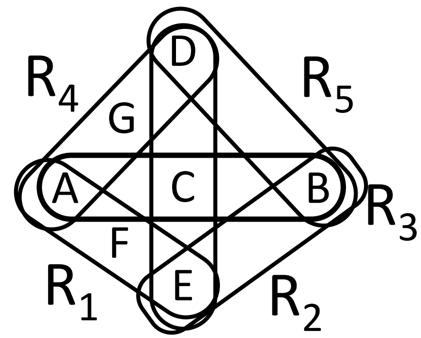


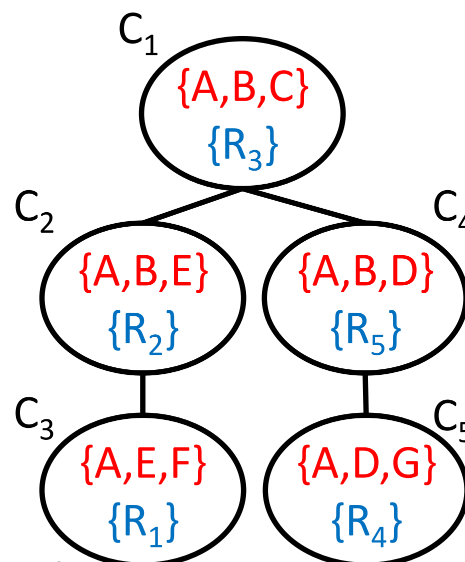
Dual graph

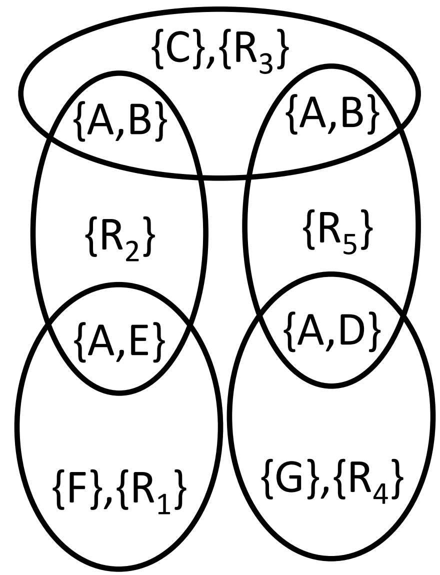


Tree Decomposition

- Tree: Vertices/clusters, edges
- Each cluster is labeled with
  - A set of variables$\subseteq \mathcal{V}$
  - A set of relations $\subseteq \mathcal{R}$
- Two conditions
  1. For each relation R, ∃cluster c_i
     - R appears c_i
     - Scope(R) is also in c_i
  2. Every variable
     - Induces a connected subtree
- Separators
  - Variables & relations common to 2 adjacent clusters
  - channel communications between clusters

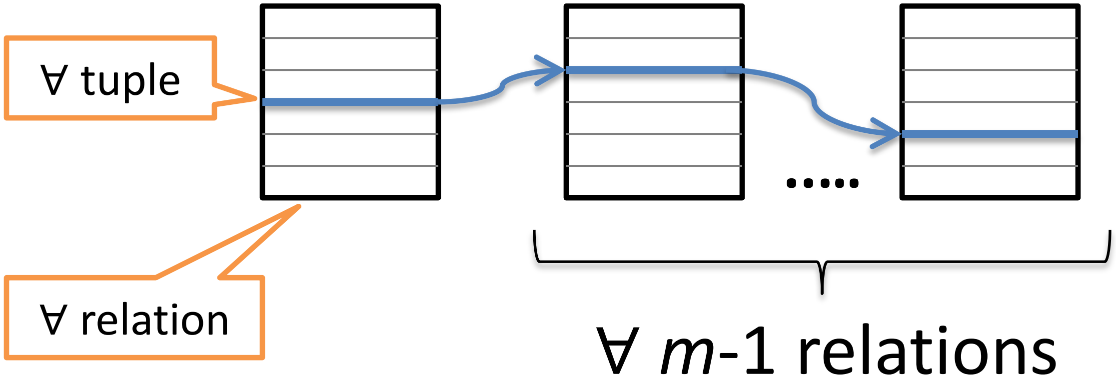






R(*,*m*)C Property [Karakashian+ 10]

- A CSP is R(*,*m*)C iff
  - Every ***tuple*** in a relation can be extended to the variables in the scope of any (*m*-1) other relations in an assignment satisfying all *m* relations simultaneously



ProcessQ: Algorithm for R(*,*m*)C

- Φ: combination of *m* connected relations in the dual graph

$$\Phi = \{ \omega_1=\{R_1,R_2,\ldots,R_m\}, \omega_2, \omega_3,\ldots, \omega_k\}$$

- Q propagation queue

$$Q=\{\langle R_1,\omega_1\rangle,\langle R_1,\omega_2\rangle,\langle R_1,\omega_3\rangle,\ldots ,\langle R_n,\omega_{k-1}\rangle,\langle R_n,\omega_k\rangle\}$$

- For each $\langle R_i,\omega_j\rangle$ in Q, ProcessQ
  - Deletes from R_i tuples that cannot extended to relations in ω_j
  - As some tuples of relations $R_x \in \omega_j$ may lose support, it requeues $\{\langle R_x,\omega_y\rangle\}$ for every threatened relation

ProcessQ: Animation

Q
$\langle R_1,\omega_1 \rangle$
$\langle R_2,\omega_1 \rangle$
$\langle R_5,\omega_1 \rangle$
$\langle R_2,\omega_2 \rangle$
$\langle R_5,\omega_2 \rangle$
$\langle R_4,\omega_2 \rangle$
$\langle R_3,\omega_3 \rangle$
$\langle R_4,\omega_3 \rangle$
$\langle R_5,\omega_3 \rangle$

ω_1 ω_2 ω_3

Extract $\langle R,\omega \rangle$ from Q

Define CSP P_ω

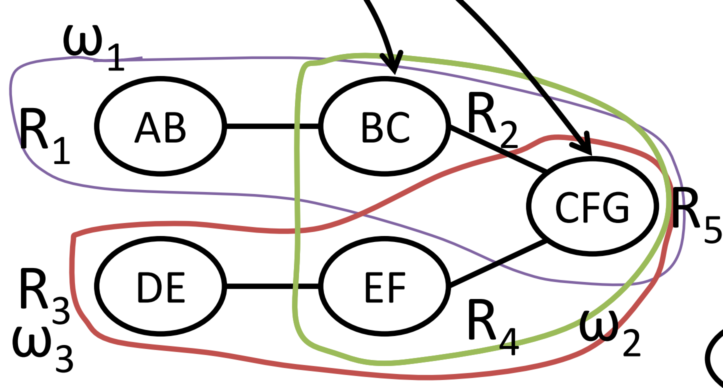
For each τ in R

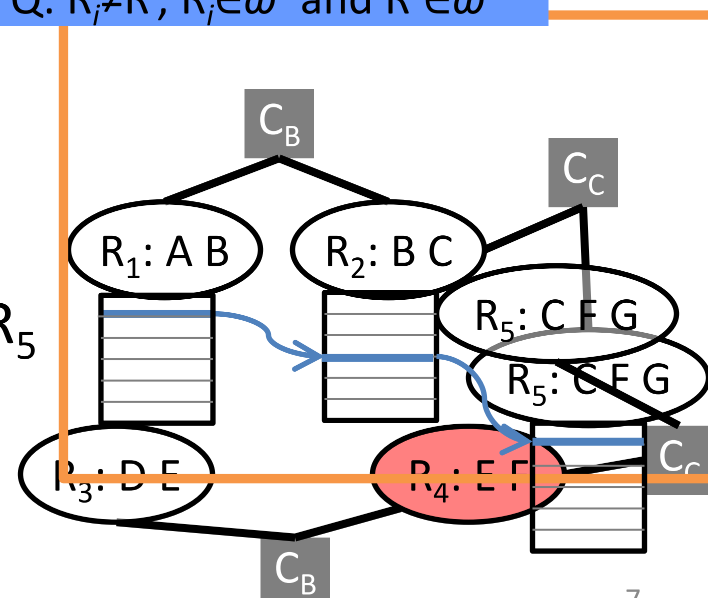
Assign τ as a value for R

Solve P_ω with forward checking

If no solution found: delete τ

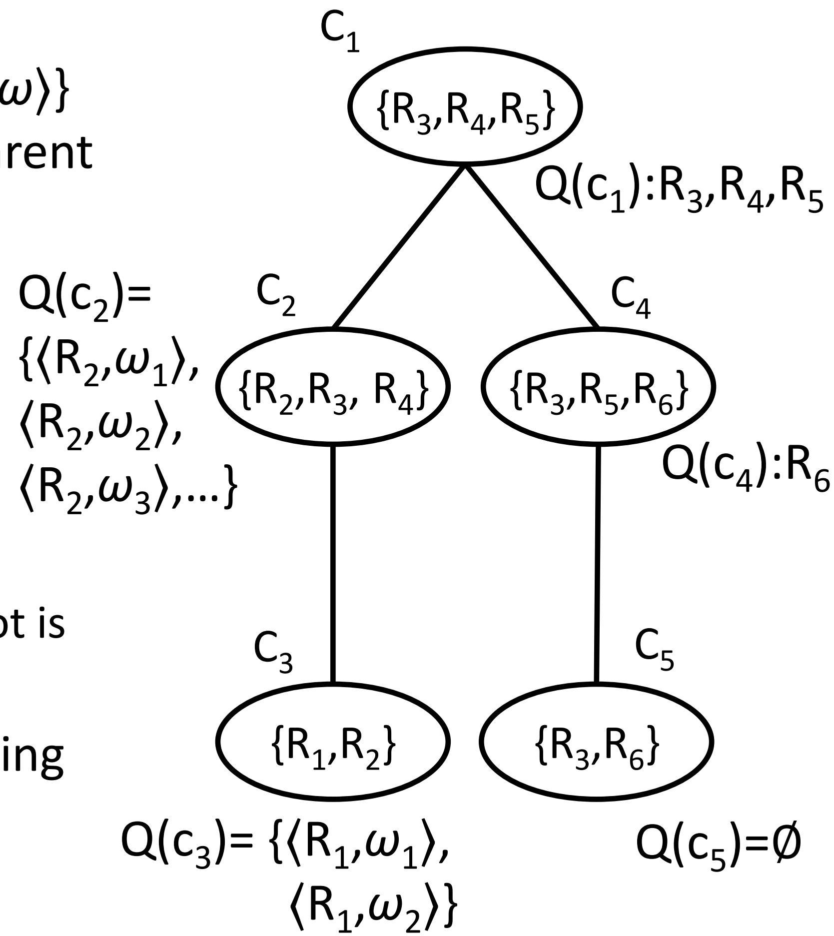
Add $\langle R',\omega' \rangle$ to Q: $R_i \neq R'$, $R_i \in \omega'$ and $R' \in \omega'$



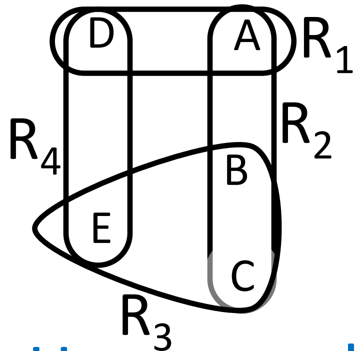


ProcessMQ: Intelligent update scheduling

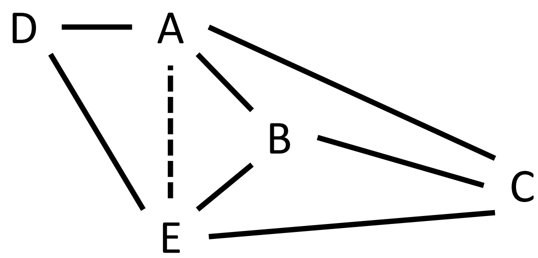
- Cluster c_i has a local queue $Q(c_i)=\{\langle R_i,\omega\rangle\}$ for relations R_i in cluster but not in parent
- Using the tree decomposition
  - As an ordering heuristic for checking consistency of $\langle R_i,\omega\rangle$
  - Repeat "leaves up to root, down to leaves," until quiescence
  - Update relations in only local queue
  - Example: R_3 is updated only when root is reached
- Advantage fewer updates, same filtering
  - In previous example, R_3 is updated once although it appears in 3 clusters



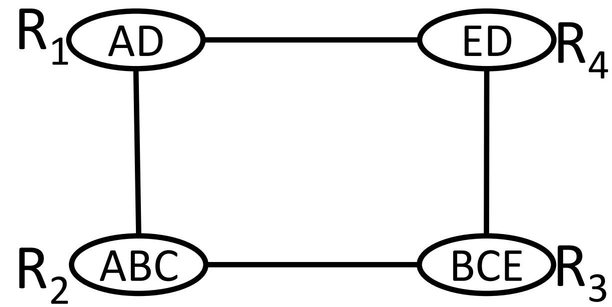
T-R(*,m,z)C [Rollon+ 10]



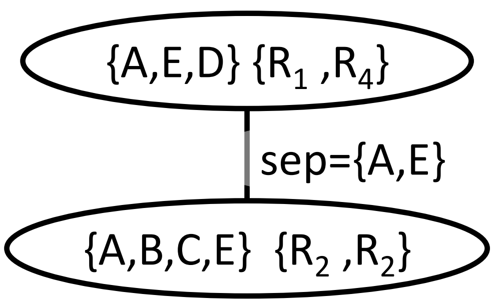
Hypergraph

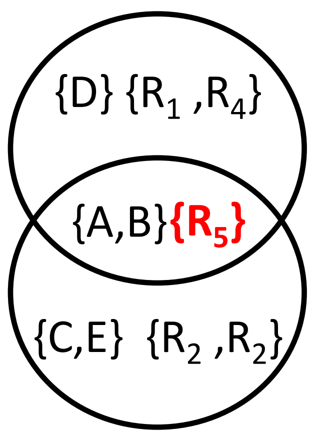


Primal graph

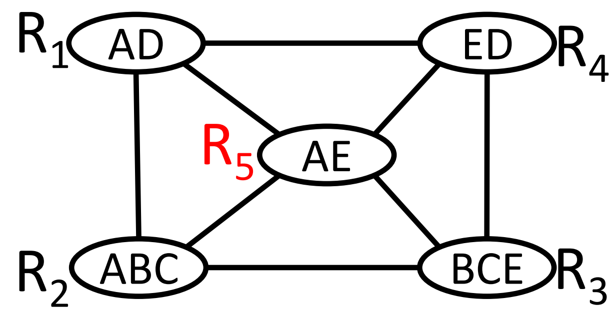


Dual graph





Tree decomposition



Adding R_5

T-R(*,*m*,*z*)C Strictly Stronger than R(*,*m*)C

Let A, B, C, D and E be Boolean variables

R_1

A	D
0	0
1	1

R_2

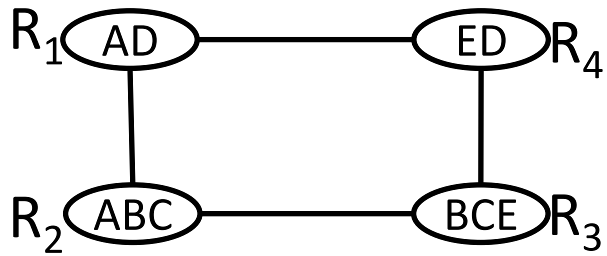
A	B	C
0	0	0
1	1	1

R_3

B	C	E
0	0	0
1	1	1

R_4

E	D
0	0
1	1

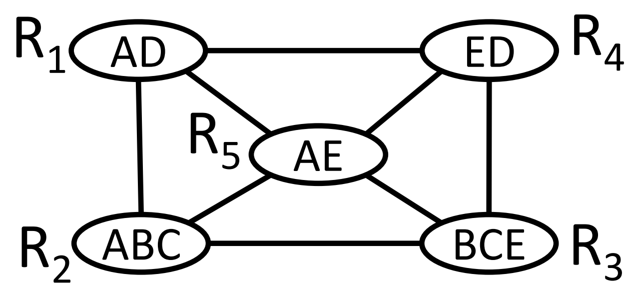


Assignment A=0 & E=1 is valid

Does not violate R(*,2)C

R_5

A	E
0	0
1	1



Assignment A=0 & E = 1 is **inconsistent**

Experimental Results

- Experiments for finding all solutions with BTD maintaining
  wR(*,best(2,3,4))C and T-wR(*,best(2,3,4), best(5,7,9))
- Results shown demonstrate the benefits of ProcessMQ & T-wR(*,m,z)C

Benchmark	#ins	#vars	tw		ProcessQ wR(*,best)C	ProcessMQ wR(*,best)C	ProcessQ T-wR(*,b,b)C
aim-200	24	200	104.92	#C	17	17	**22**
				t_{avg}	246.35	252.48	**238.99**
				t_{max}	3,352.54	3,452.98	**1,540.94**
ogdVg	59	134	85	#C	15	15	15
				t_{avg}	283.27	**242.06**	266.74
				t_{max}	1,834.11	**1,508.27**	1,720.97
rand-3-20-20	50	20	13	#C	13	**14**	-
				t_{avg}	2,191.56	**1,949.87**	-
				t_{max}	3,481.04	**3,145.77**	-

Conclusions

- Contributions
  - Reformulated R(*,*m*)C algorithm
  - New relational consistency property T-R(*,*m*,*z*)C
  - Experimental analysis
- Future work
  - Study impact of choice of parameters *z*, *m*
  - Develop strategies for dynamically choosing *z*, *m* as a function of the size of clusters & separators