Localizing & Bolstering Constraint Propagation in a Tree Decomposition

Shant Karakashian, Robert Woodward & Berthe Y. Choueiry

1. Paper & poster at AAAI 2013
2. PhD thesis of Shant Karakashian, ConSystLab, May 2013

Acknowledgments:

- Experiments conducted at UNL's Holland Computing Center
- NSF Award RI-111795

Outline

- Introduction & background information
- Key ideas
  - Localize consistency to clusters of a tree decomposition
  - Bolster propagation at separators
- Evaluation
  - Theoretical: Comparing resulting consistency properties
  - Empirical: Solving CSPs in a backtrack-free manner
- Conclusions & future work

✧ Robert's RNIC on binary CSPs

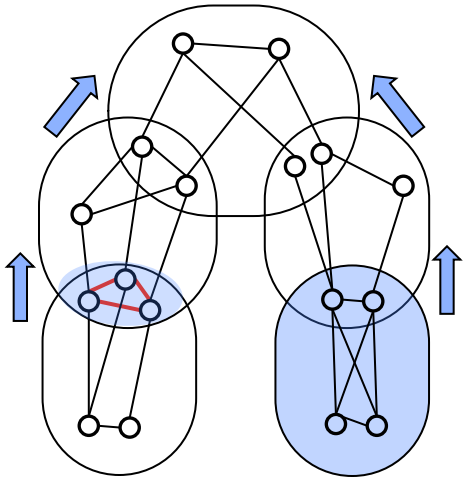
✧ Summary of contributions of Shant's thesis

Introduction

- One tractability condition links [Freuder 82]
  - Consistency level to
  - Width of the constraint network, a structural parameter
- Catch 22
  - Enforcing higher-levels of consistency may require adding constraints, which increases the width
  - Computing treewidth (≡ induced width) is in NP-hard
- Goal
  - Exploit above condition to achieve practical tractability

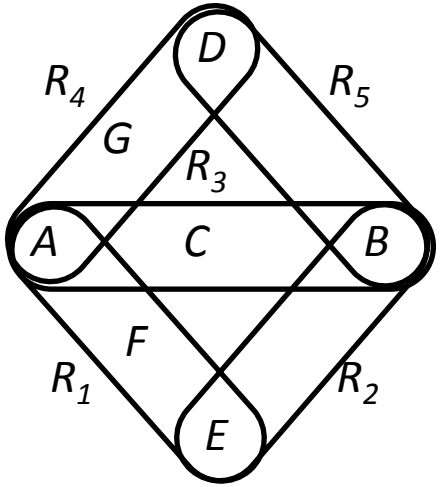
Our Approach

- Robert discussed consistency properties that
  - Preserve the structure of a problem instance
  - Enforce a (parameterized) consistency level
- Exploit a tree decomposition
  - Localize application of the consistency algorithm
  - Steer constraint propagation along the branches of the tree
  - Add redundant constraints at separators to enhance propagation



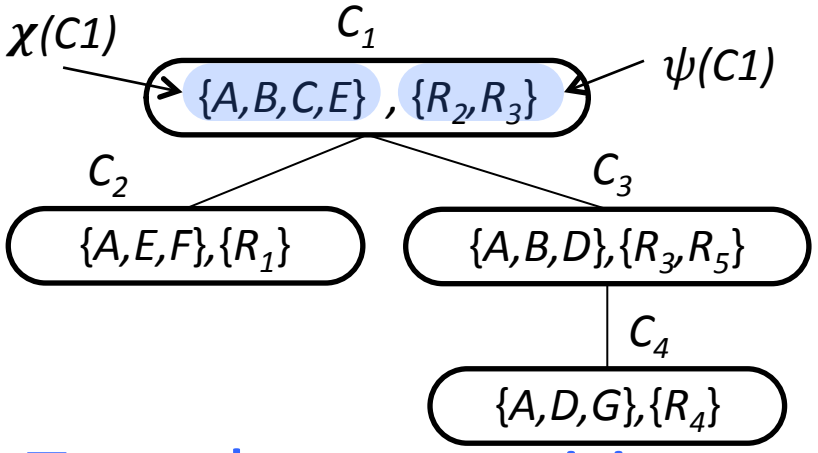
Tree Decomposition

- A tree decomposition: $\langle T, \chi, \psi \rangle$
  - T: a tree of clusters
  - χ: maps variables to clusters
  - ψ: maps constraints to clusters



Hypergraph

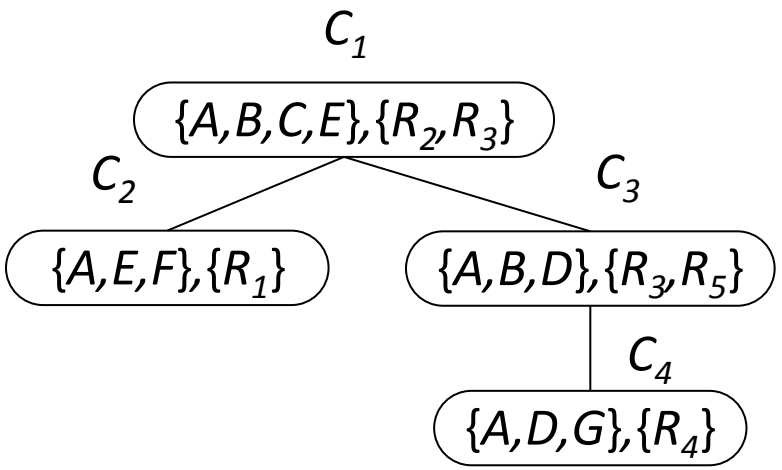
- Conditions
  - Each constraint appears in at least one cluster with all the variables in its scope
  - For every variable, the clusters where the variable appears induce a connected subtree

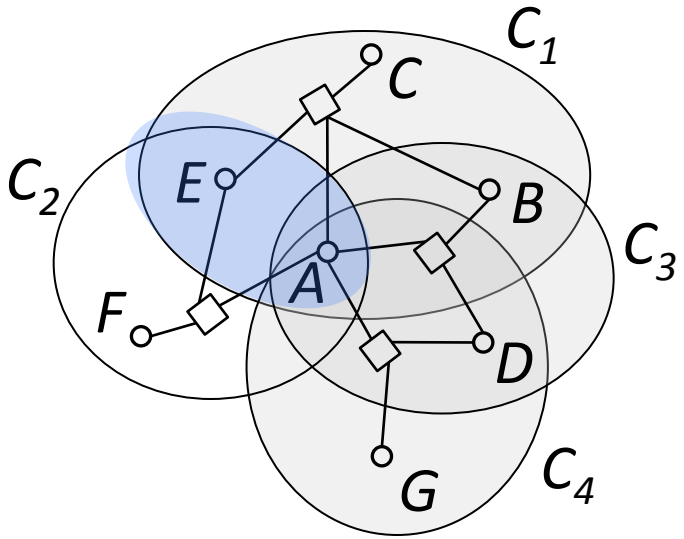


Tree decomposition

Tree Decomposition: Separators

- A separator of two adjacent clusters is the set of variables associated to both clusters



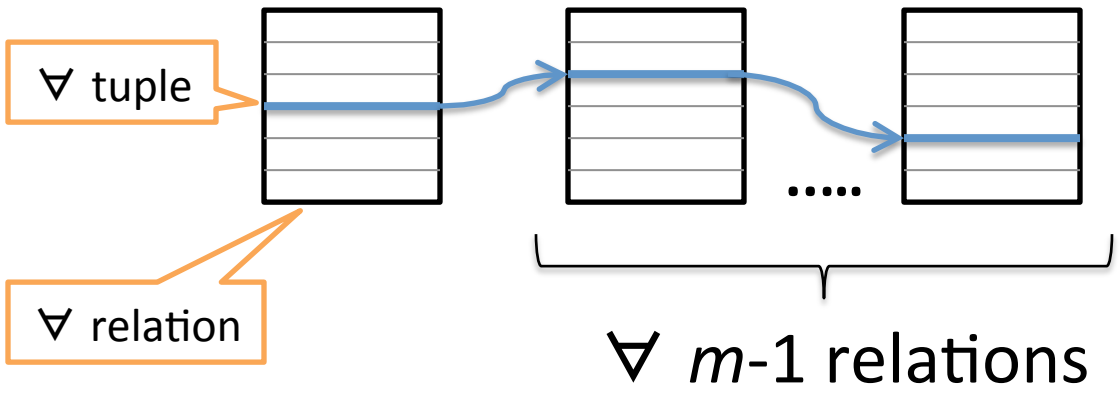


- Width of a decomposition/network
  - Treewidth = maximum number of variables in clusters - 1

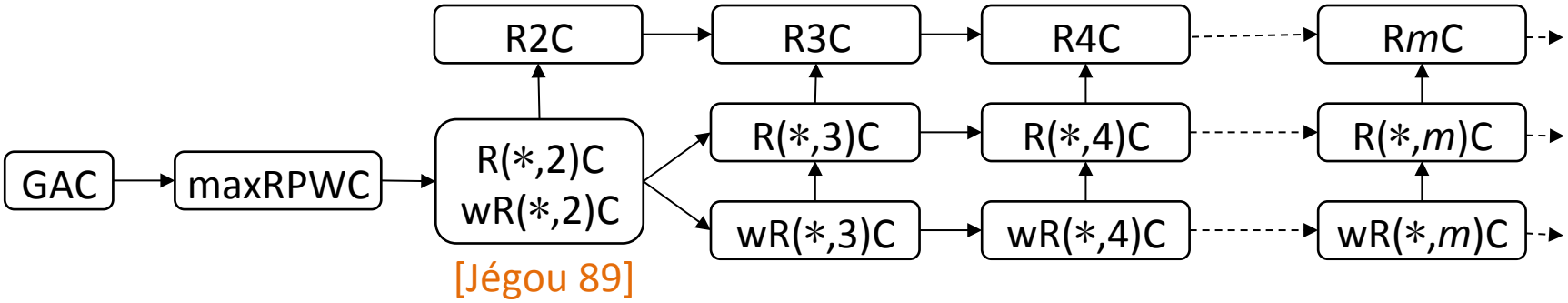
Relational Consistency R(*,*m*)C

[Karakashian+ AAAI 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
- R(*,*m*)C ≡ Every set of *m* relations is minimal



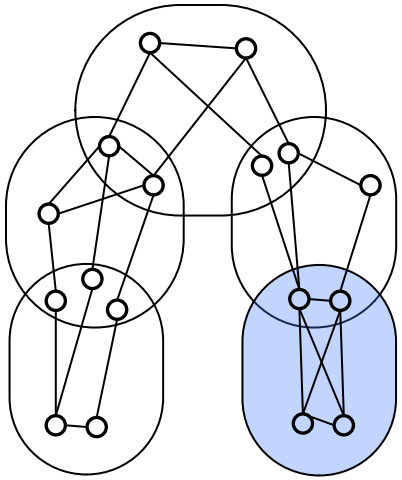
Characterizing R(*,m)C



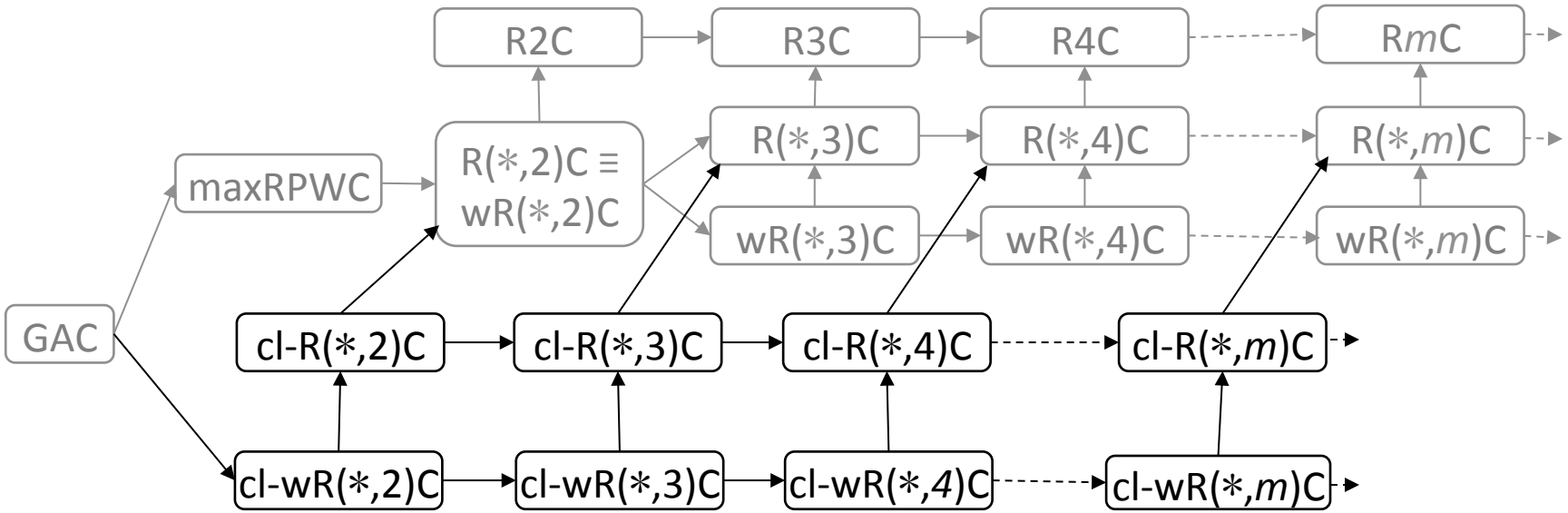
- GAC [Waltz 75]
- maxRPWC [Bessiere+ 08]
- RmC: Relational m Consistency [Dechter+ 97]

Localize Consistency

- Restricting R(*,*m*)C to clusters: cl-R(*,*m*)C
- Localization cl-R(*,*m*)C
  - Fewer combinations of *m* relations
  - Reduces the enforced consistency level



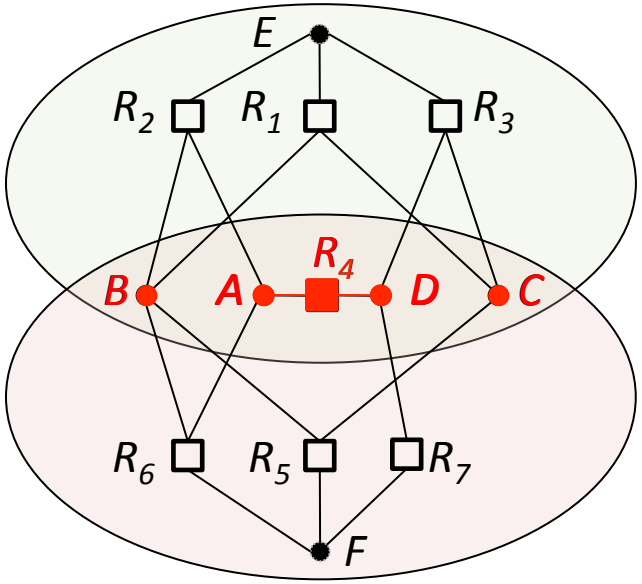
Characterizing cl-R(∗,*m*)C



- GAC [Waltz 75]
- maxRPWC [Bessiere+ 08]
- RmC: Relational m Consistency [Dechter+ 97]

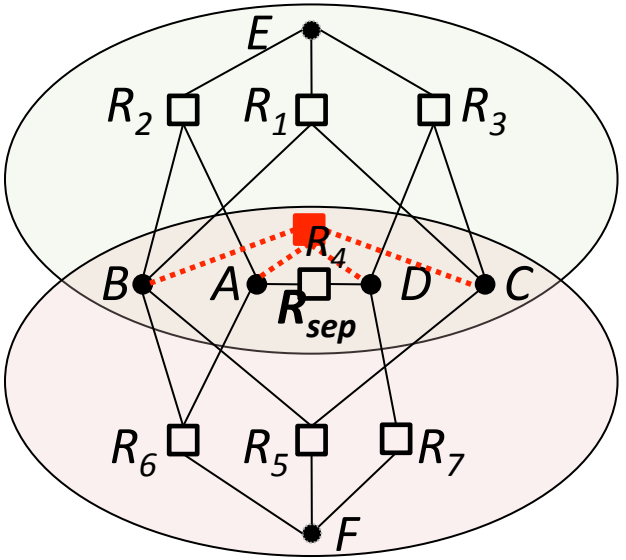
Communication Between Clusters

- Two clusters communicate via their separator
  - Constraints common to the two clusters
  - Domains of variables common to the two clusters

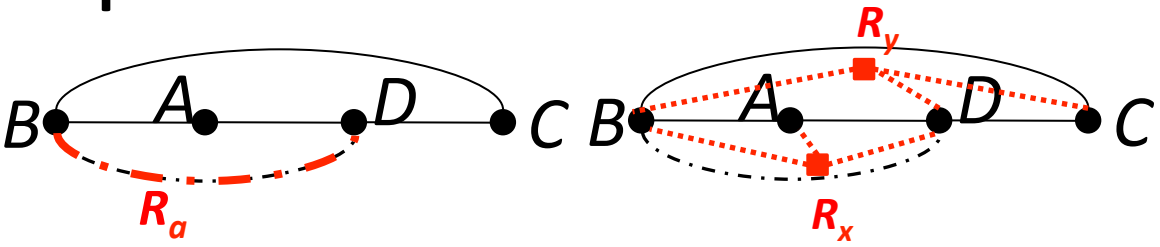


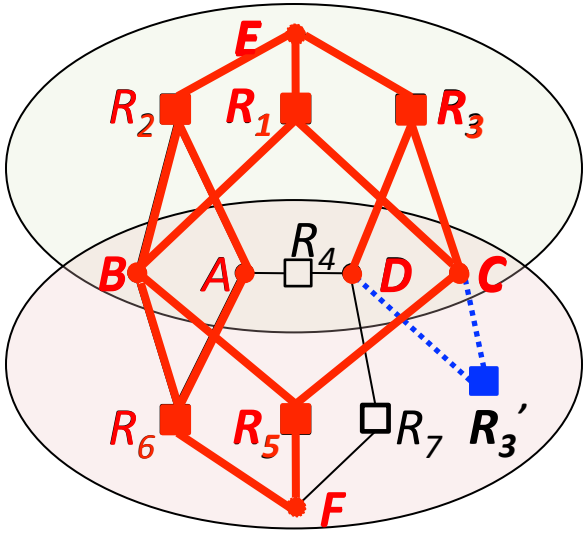
Bolstering Propagation at Separators

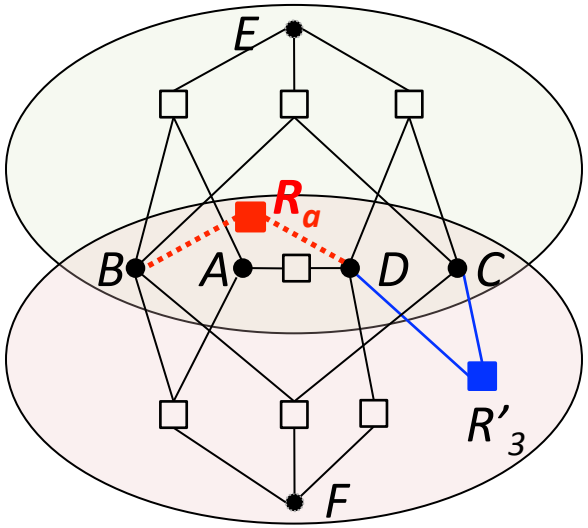
- For perfect communication between clusters
  - Ideally, add unique constraint
  - Space overhead, major bottleneck
- Enhance propagation by bolstering
  - Projection of existing constraints
  - Adding binary constraints
  - Adding clique constraints

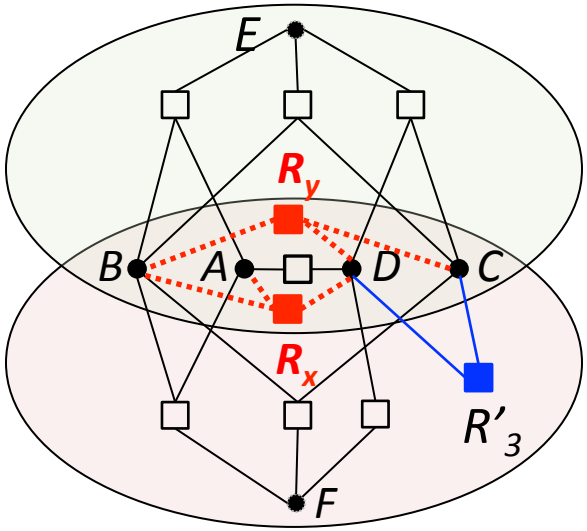


Bolstering Schemas: Approximate Unique Separator Constraint







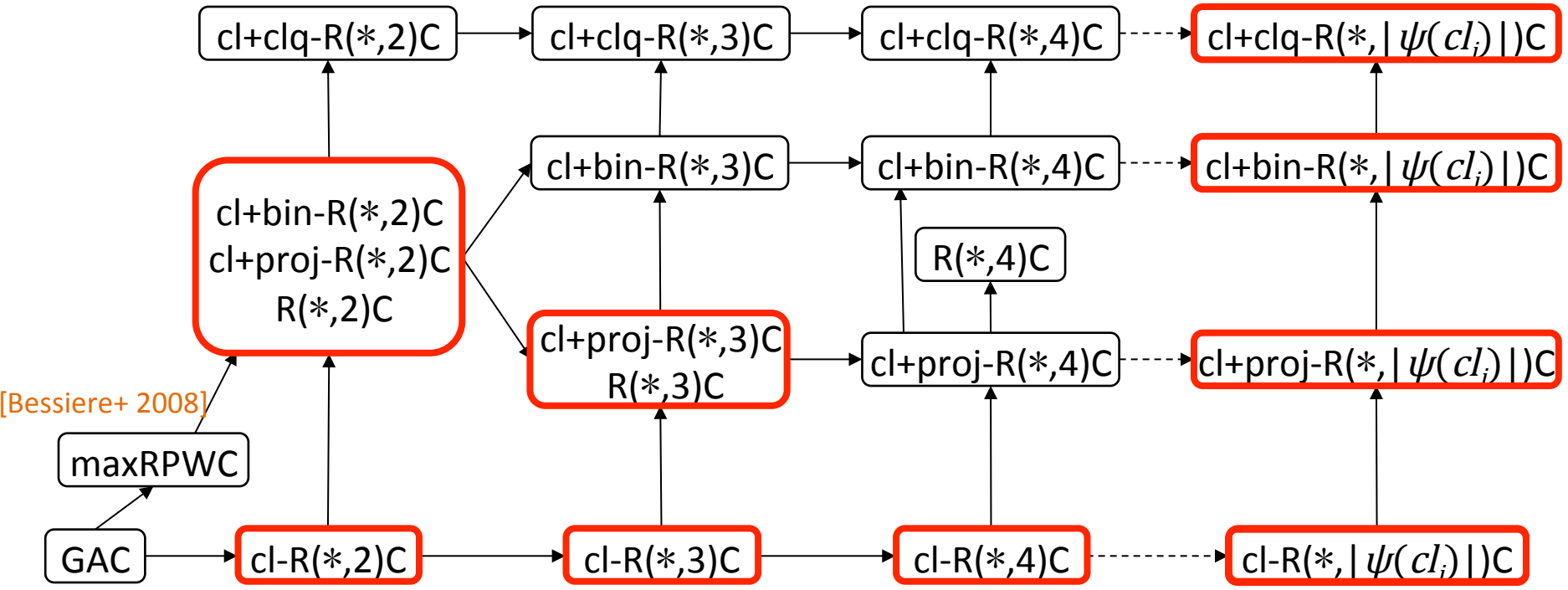


Projection
cl+proj-R(*,*m*)C

Binary constraints
cl+bin-R(*,*m*)C

Clique constraints
cl+clq-R(*,*m*)C

Resulting Consistency Properties



Experiments

- Domain-based filtering
  - GAC
  - maxRPWC
- Relation-based filtering
  - Global: wR(*,*m*)C for *m*=2,3,4
  - Local: cl-wR(*,*m*)C *m*=2,3,4 and cl-R(*,$|\psi(cl_i)|$)C
  - Bolstering
    - projection
    - binary
    - clique
- Backtrack search, full-lookahead, first solution
- Reported
  - Number of instances Completed, BT-free, Min(#NV), Fastest
  - Cumulative count of BT-Free as a function of treewidth

Characteristics of Benchmark Data

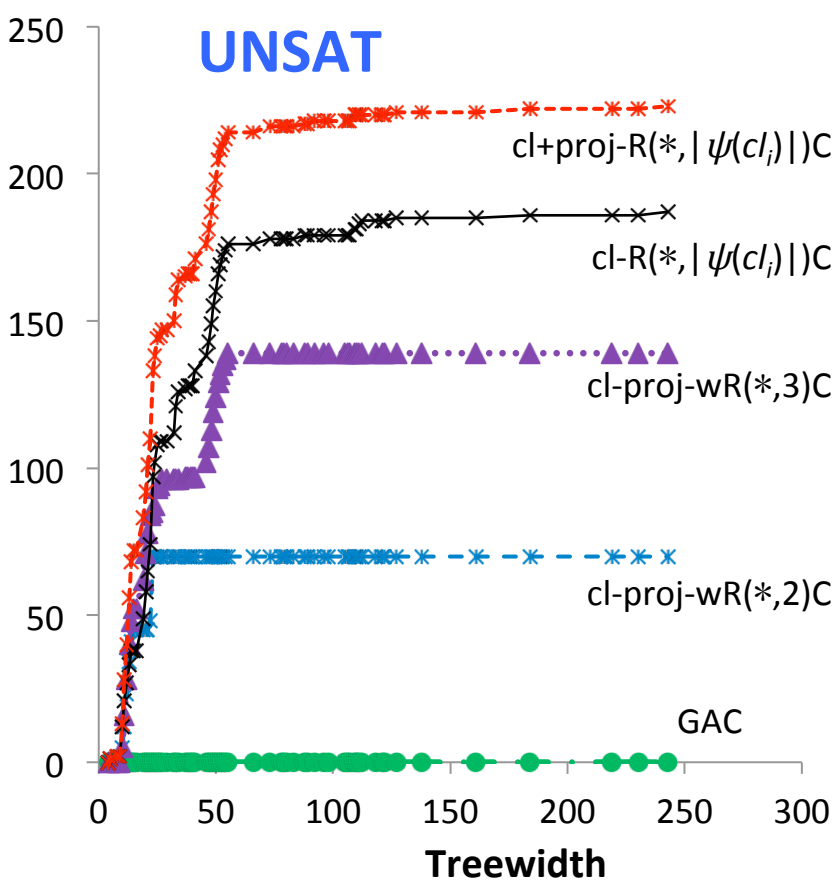
- UNSAT: 479 instances
- SAT: 200 instances

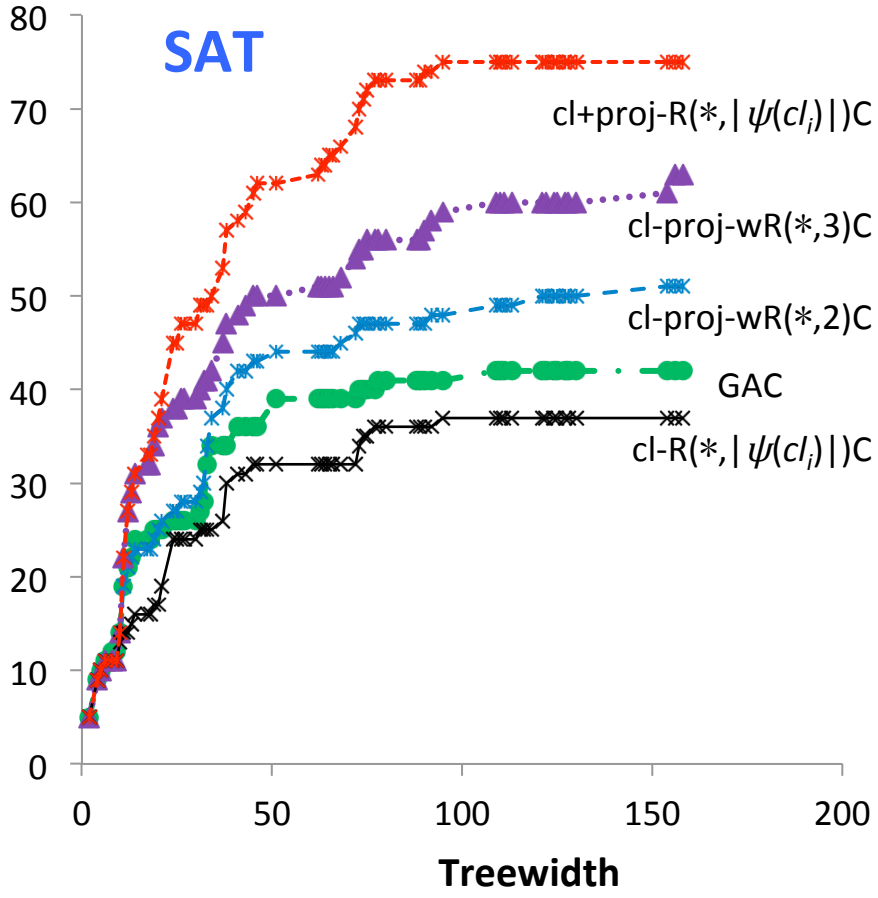
	max		median		mean	
	UNSAT	SAT	UNSAT	SAT	UNSAT	SAT
treewidth	243	158	33	18	43.45	34.44
largest sep.	214	157	28	16.5	39.02	31.33
$\max(\lvert\psi(cl_i)\rvert)$						
local	1,243	211	16	8	109.54	18.35
projection	1,243	211	18	11	114.70	37.35
binary	1,243	653	24	12	199.50	80.65
clique	1,243	148	18	10	113.35	25.87
clique arity	48	26	7	4	7.40	5.97

Empirical Evaluations

+ maxRPWC, *m*=3,4			wR(*,2)C					R(*,|ψ(cl$_i$)|)C			
	#inst.	GAC	global	local	Proj.	binary	clique	local	Proj.	binary	clique
Completed	UNSAT 479	167 34.9%	170 35.5%	167 34.9%	172 35.9%	169 35.3%	162 33.8%	285 59.5%	**286** **59.7%**	282 58.9%	271 56.6%
	SAT 200	174 87.0%	**179** **89.5%**	178 89.0%	176 88.0%	169 84.5%	104 52.0%	152 76.0%	138 69.0%	124 62.0%	113 56.5%
BT-Free	UNSAT 479	0 0.0%	70 14.6%	39 8.1%	70 14.6%	70 14.6%	74 15.4%	187 39.0%	**223** **46.6%**	223 46.6%	213 44.5%
	SAT 200	44 22.0%	55 27.5%	37 18.5%	53 26.5%	52 26.0%	38 19.0%	39 19.5%	**77** **38.5%**	71 35.5%	58 29.0%
Min(#NV)	UNSAT 479	17 3.5%	73 15.2%	43 9.0%	72 15.0%	72 15.0%	77 16.1%	220 45.9%	**249** **52.0%**	248 51.8%	239 49.9%
	SAT 200	47 23.5%	64 32.0%	37 18.5%	62 31.0%	61 30.5%	39 19.5%	83 41.5%	**111** **55.5%**	100 50.0%	79 39.5%
Fastest	UNSAT 479	72 15.0%	13 2.7%	35 7.3%	5 1.0%	1 0.2%	1 0.2%	**176** **36.7%**	108 22.5%	42 8.8%	37 7.7%
	SAT 200	**121** **60.5%**	45 22.5%	47 23.5%	23 11.5%	14 7.0%	12 6.0%	34 17.0%	18 9.0%	13 6.5%	12 6.0%

Cumulative Count of Instances Solved w/o Backtracking





Acknowledgment: Charts suggested by Rina Dechter

Conclusions & Future Work

- Adapted R(∗,*m*)C to a tree decomposition of the CSP
  - Localizing R(∗,*m*)C to the clusters
  - Bolstering separators to strengthen the enforced consistency
- Directions for future work
  - R(∗,*m*)C on non-table constraints via domain filtering
  - Automating the selection of a consistency property and of a bolstering scheme
    - On clusters/seperators
    - During search
  - Modify the structure of a tree decomposition to improve performance (e.g., merging clusters [Fattah & Dechter 1996])
  - Improve our algorithms for R(∗,*m*)C

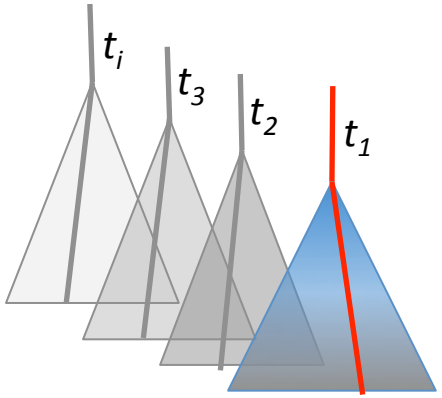
Thank You for Your Attention

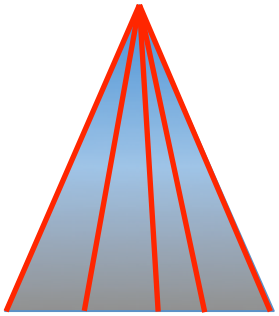
Shant's Thesis

- Question
  - Practical tractability of CSPs exploiting the condition linking
    - the level of consistency
    - to the width of the constraint graph
- Solution
  - Introduced a parameterized consistency property R(∗,m)C
  - Designed algorithms for implementing it
    - PERTUPLE and ALLSOL
    - Hybrid algorithms
  - Adapted R(∗,m)C to a tree decomposition of the CSP
    - Localizing R(∗,m)C to the clusters
    - Strategies for guiding propagation along the structure
    - Bolstering separators to strengthen the enforced consistency
  - Improved the BTD algorithm for solution counting, WITNESSBTD
- \+ Two incidental results in appendices

Algorithms for Enforcing R(*,m)C

- PERTUPLE
  - For each tuple find a solution for the variables in the *m-1* relations
  - Many satisfiability searches
    - Effective when there are many solutions
    - Each search is quick & easy
- ALLSOL
  - Find all solutions of problem induced by *m* relations, & keep their tuples
  - A single exhaustive search
    - Effective when there are few or no solutions
- Hybrid Solvers (portfolio based) [+Scott]





Hybrid Solver

- Choose between PERTUPLE & ALLSOL
- Parameters to characterize the problem
  - κ predicts if instance is at the phase transition
  - relLinkage approximates the likelihood of a tuple at the overlap two relations to appear in a solution
- Classifier built using Machine Learning
  - C4.5
  - Random Forests

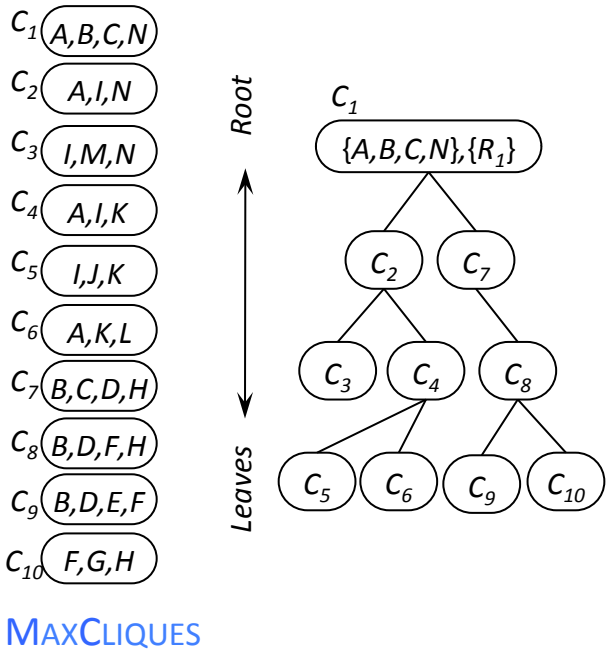
Empirical Evaluations: Hybrid

Task: compute the minimal CSP

	#Instances solved by…				**Average CPU** sec				
Partition	AllSol	PerTuple	Solver$_{C4.5}$	Solver$_{RF}$	#Instances	AllSol	PerTuple	Solver$_{C4.5}$	Solver$_{RF}$
$\mathcal{A}$	5,777	5,776	**5,777**	**5,777**	5,776	1.27	4.97	**2.14**	2.27
$\mathcal{P}$	10,095	15,457	**15,439**	14,012	10,095	109.61	5.21	**7.72**	31.53
$\mathcal{A} \cup \mathcal{P}$	15,872	**21,333**	21,216	19,789	15,871	70.18	**5.12**	5.69	20.88

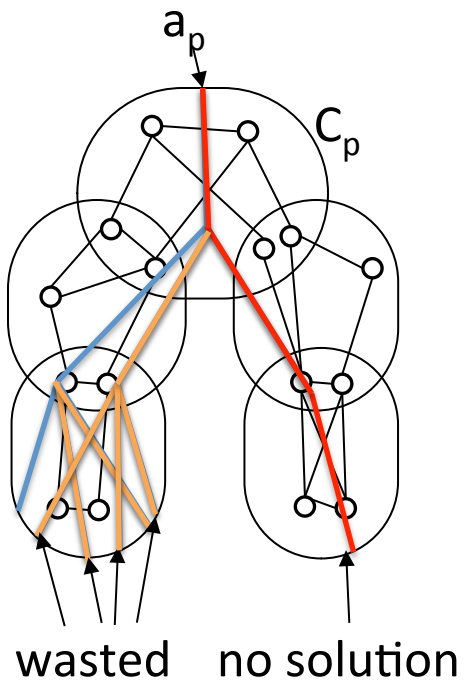
Structure-Guided Propagation

- Orderings
  - RANDOM: FIFO/arbitrary
  - STATIC, PRIORITY, DYNAMIC: Leaves⟷root
- Structure-based propagation
  - STATIC
    - Order of MAXCLIQUES
  - PRIORITY:
    - Process a cluster once in each direction
    - Select most significantly filtered cluster
  - DYNAMIC
    - Similar to PRIORITY
    - But may process 'active' clusters more than once



Solution Counting

- BTD is used to count the number of solutions [Favier+ 09]
- Using COUNT algorithm on trees [Dechter+ 87]
- Witness-based solution counting
- Find a witness solution before counting



Empirical Evaluations

- Task: Count solutions
  - BTD versus WITNESSBTD
  - GAC: Pre-processing & full look-ahead

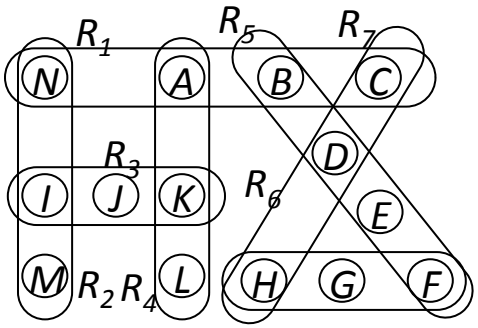
		BTD	WITNESSBTD
Avg. #NV	UNSAT (89)	1,437,909.79	**734,983.25**
	SAT (101)	4,785,737.57	**4,735,136.28**
Avg. Time (sec)	UNSAT (89)	145.55	**123.86**
	SAT (101)	1,151.35	**1,148.46**

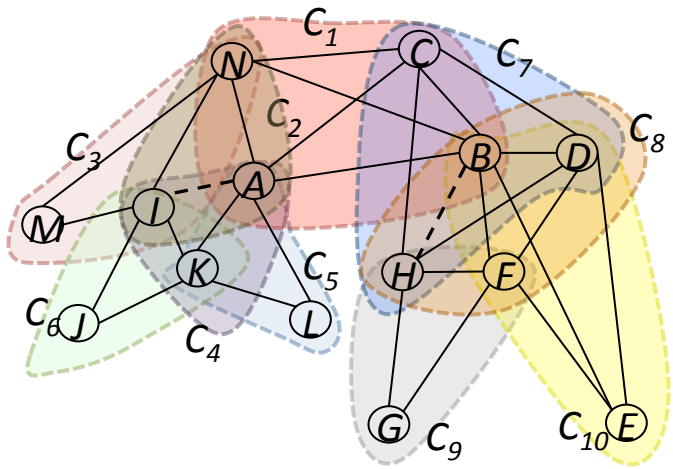
Empirical Evaluations: WITNESSBTD

+ maxRPWC, *m*=2,4			wR(*,3)C					R(*,|ψ(cl$_i$)|)C			
	#inst.	GAC	global	local	Proj.	binary	clique	local	Proj.	binary	clique
Completed	UNSAT 479	200 41.8%	192 40.1%	248 51.8%	237 49.5%	236 49.3%	220 45.9%	**302 63.0%**	290 60.5%	286 59.7%	277 57.8%
	SAT 200	111 55.5%	85 42.5%	**112 56.0%**	100 50.0%	96 48.0%	81 40.5%	110 55.0%	90 45.0%	88 44.0%	78 39.0%
BT-Free	UNSAT 479	0 0.0%	97 20.3%	104 21.7%	139 29.0%	139 29.0%	131 27.3%	186 38.8%	221 46.1%	**222 46.3%**	212 44.3%
	SAT 200	15 7.5%	42 21.0%	17 8.5%	47 23.5%	47 23.5%	45 22.5%	25 12.5%	**61 30.5%**	**61 30.5%**	52 26.0%
Min(#NV)	UNSAT 479	2 0.4%	101 21.1%	111 23.2%	145 30.3%	145 30.3%	136 28.4%	235 49.1%	263 54.9%	**264 55.1%**	244 50.9%
	SAT 200	19 9.5%	42 21.0%	23 11.5%	52 26.0%	49 24.5%	49 24.5%	57 28.5%	**74 37.0%**	73 36.5%	65 32.5%
Fastest	UNSAT 479	100 20.9%	26 5.4%	120 25.1%	71 14.8%	23 4.8%	23 4.8%	**189 39.5%**	126 26.3%	58 12.1%	52 10.9%
	SAT 200	**73 36.5%**	20 10.0%	20 10.0%	18 9.0%	9 4.5%	9 4.5%	27 13.5%	15 7.5%	9 4.5%	9 4.5%

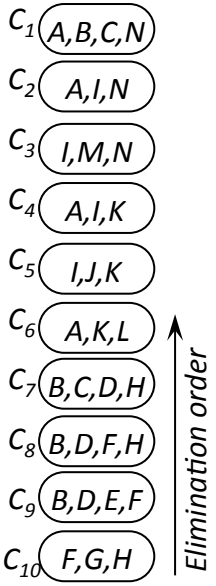
Generating a Tree Decomposition

- Triangulate the primal graph using min-fill
- Identify the maximal cliques using MAXCLIQUES
- Connect the clusters using JOINTREE
- Add constraints to clusters where their scopes appear

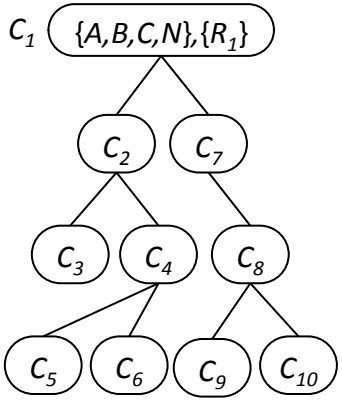




min-fill



MAXCLIQUES



JOINTREE