Recent Advances in High-Level Relational Consistency

Robert J. Woodward

Constraint Systems Laboratory

- **Joint work with**
  - Shant Karakashian, Daniel Geschwender, Christopher Reeson, and Berthe Y. Choueiry @ UNL
  - Christian Bessiere @ LIRMM-CNRS
- **Support**
  - Experiments conducted at UNL's Holland Computing Center
  - NSF Graduate Research Fellowship & NSF Grant No. RI-111795

Publications

- Relational *m*-wise consistency, R($*$,*m*)C
  - Relational Consistency by Constraint Filtering [SAC 10]
  - A First Practical Algorithm for High Levels of Relational Consistency [AAAI 10]
  - Improving the Performance of Consistency Algorithms by Localizing and Bolstering Propagation in a Tree Decomposition [AAAI 13]
- Relational Neighborhood Inverse Consistency, RNIC
  - Solving Difficult CSPs with Relational Neighborhood Inverse Consistency [AAAI 11]
  - Adaptive Neighborhood Inverse Consistency as Lookahead for Non-Binary CSPs [AAAI-SA 11]
  - Reformulating the Dual Graphs of CSPs to Improve the Performance of Relational Neighborhood Inverse Consistency [SARA 11]
  - Revisiting Neighborhood Inverse Consistency on Binary CSPs [CP 12]
  - Selecting the Appropriate Consistency Algorithm for CSPs Using Machine Learning Classifiers [AAAI-SA13]
- MS thesis, Woodward, Dec 2011
- PhD thesis, Karakashian, May 2013

- Papers and slides available on lab website, consystlab.unl.edu

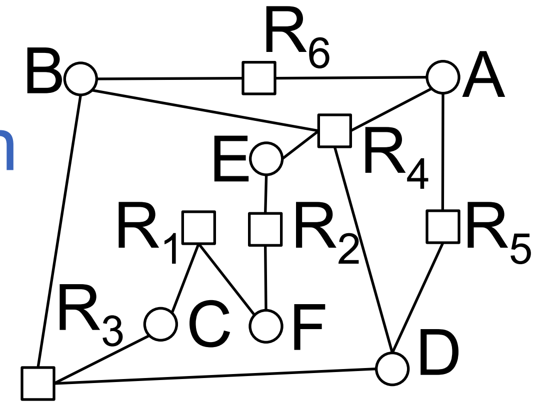
Overview

- Background
- Relational *m*-wise consistency, R($*$,*m*)C [SAC10,AAAI10]
  - Property, Algorithm, Weakening
  - Characterization, Evaluating
- Relational Neighborhood Inverse Consistency (RNIC) [AAAI11,SARA11]
  - Property, Algorithm
  - Dual-graph reformulation, Characterization, Selection strategy
  - Evaluating
- Dual Graphs of Binary CSPs [CP2012]
  - Complete constraint network, Non-complete constraint network
  - RNIC on binary CSPs
  - Characterization, Evaluating
- Conclusions

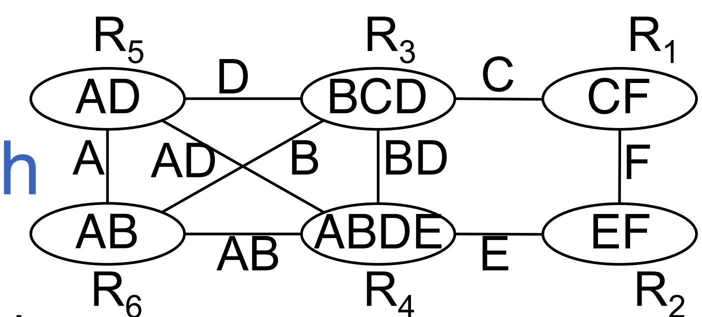
Constraint Satisfaction Problem

- CSP
  - Variables, Domains
  - Constraints: Relations & scopes
- Representation
  - Hypergraph
  - Dual graph
- Solved with
  - Search
  - Enforcing consistency
  - Lookahead = Search + enforcing consistency
- Key to our research
  - Operate on the dual graph

Hypergraph



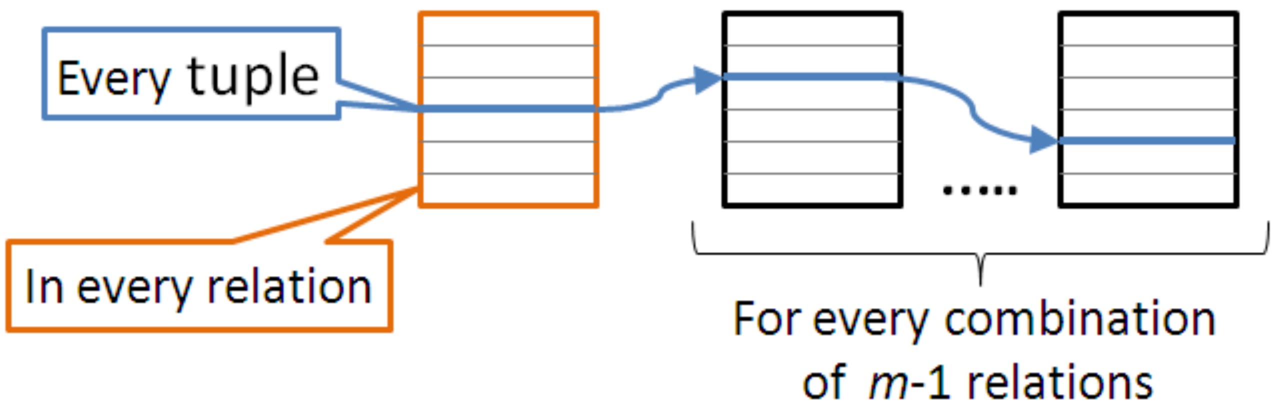
Dual graph



Relational *m*-wise consistency, R(∗,*m*)C

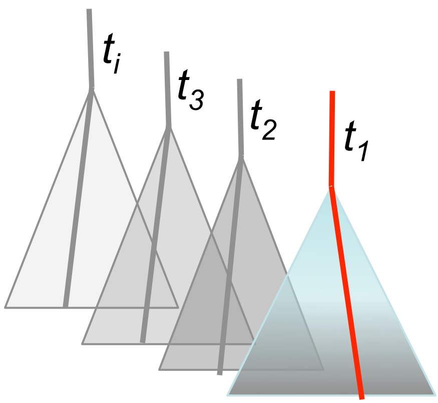
[SAC 2010, AAAI 2010]

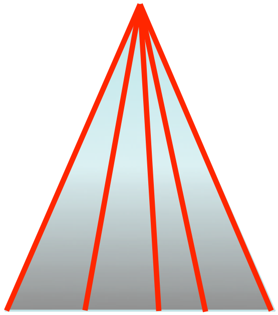
- A parameterized relational consistency property
- **Definition**
  - For every set of m constraints
  - every tuple in a relation can be extended to an assignment
  - of variables in the scopes of the other *m*-1 relations
- R(∗,*m*)C ≡ every *m* relations form a minimal CSP



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]



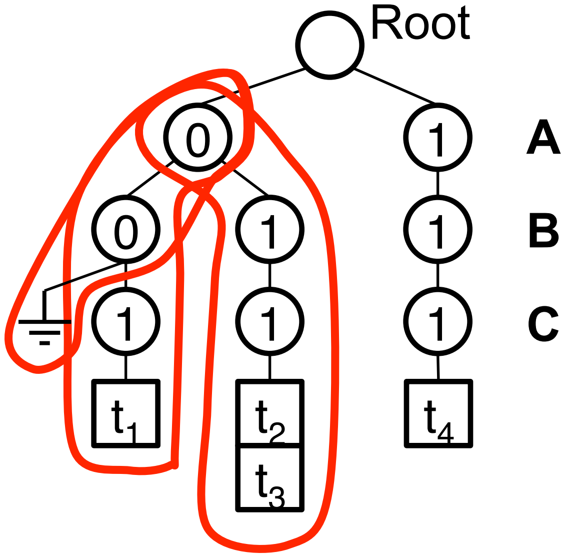


Index-Tree Data Structure

- Goal: quickly find matching tuples in other relations
- Given two relations, R_1 & R_2
- For a given tuple in R_1, find matching tuples in R_2

R_1

	X	A	B	C
τ_1	0	0	0	1
τ_2	1	0	0	1
τ_3	0	0	0	0
τ_4	0	0	1	1

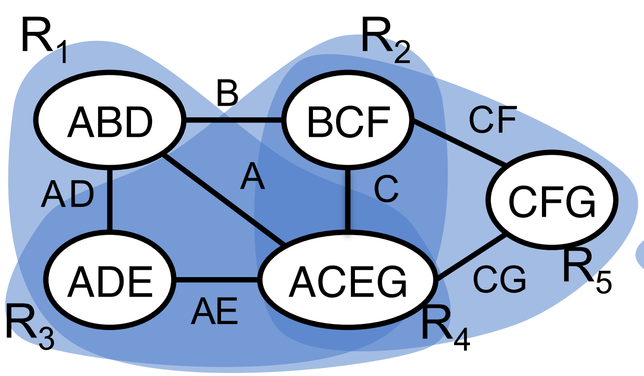


R_2

	A	B	C	D
t_1	0	0	1	0
t_2	0	1	1	0
t_3	0	1	1	1
t_4	1	1	1	1

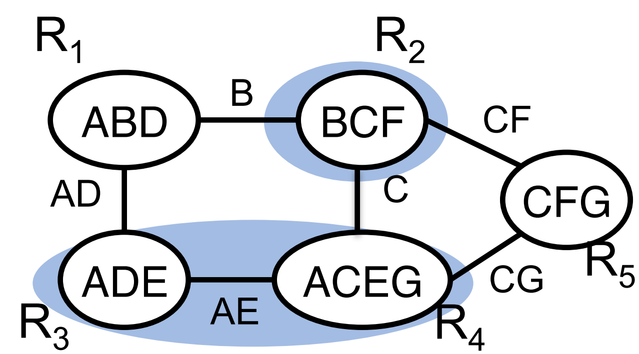
Weakening R(∗,*m*)C

- Weaken R(∗,*m*)C by removing redundant edges [Jégou 89]



R(∗,3)C

R_1 R_2 R_3
R_1 R_2 R_4
R_1 R_2 R_5
R_1 R_3 R_4
R_2 R_3 R_4
R_2 R_4 R_5
R_3 R_4 R_5



A minimal dual graph

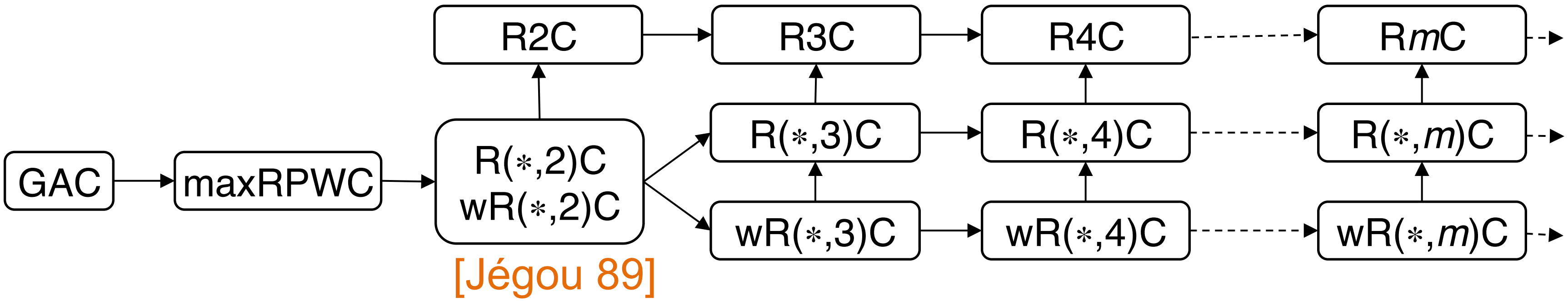
wR(∗,3)C

R_1 R_2 R_3

R_1 R_2 R_5
R_1 R_3 R_4

R_2 R_4 R_5
R_3 R_4 R_5

Characterizing R(∗,*m*)C



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

p → *p'* : *p* is strictly weaker than *p'*

Empirical Evaluations (1)

Algorithm	Avg. #Nodes	Avg. Time sec	#Completed	#Fastest	#BF
SAT **aim-100** (instances: 16, vars: 100, dom: 2, rels: 307, arity: 3)					
GAC	9,459,773.0	759.7	15	4	1
wR(∗,2)C	234,526.7	125.6	**16**	**7**	5
wR(∗,3)C	3,979.1	**19.4**	**16**	3	7
wR(∗,4)C	**559.1**	26.3	**16**	2	**9**
SAT **modifiedRenault** (instances: 19, vars: 110, dom: 42, rels: 128, arity: 10)					
GAC	1,171,458.4	108.5	17	**14**	5
wR(∗,2)C	211.5	**5.0**	**19**	5	7
wR(∗,3)C	110.4	13.3	**19**	0	14
wR(∗,4)C	**110.2**	81.3	**19**	0	**16**

Empirical Evaluations (2)

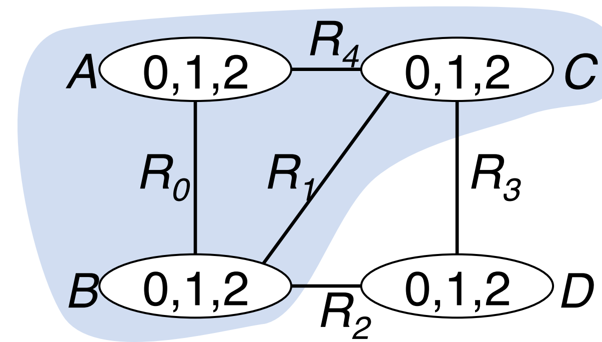
Algorithm	Avg. #Nodes	Avg. Time sec	#Completed	#Fastest	#BF
UNSAT **aim-100** (instances: 8, vars: 100, dom: 2, rels: 173, arity: 3)					
GAC	-	-	0	0	0
wR(∗,2)C	4,619,373.0	2,016.8	3	1	0
wR(∗,3)C	18,766.6	**97.4**	**4**	**3**	0
wR(∗,4)C	**18,685.3**	944.2	**4**	1	**1**
UNSAT **modifiedRenault** (instances: 31, vars: 111, dom: 42, rels: 130, arity: 10)					
GAC	1,171,458.4	782.3	9	2	0
wR(∗,2)C	487.0	**5.2**	28	**20**	25
wR(∗,3)C	**0.0**	9.6	30	2	28
wR(∗,4)C	**0.0**	44.2	**31**	2	**31**

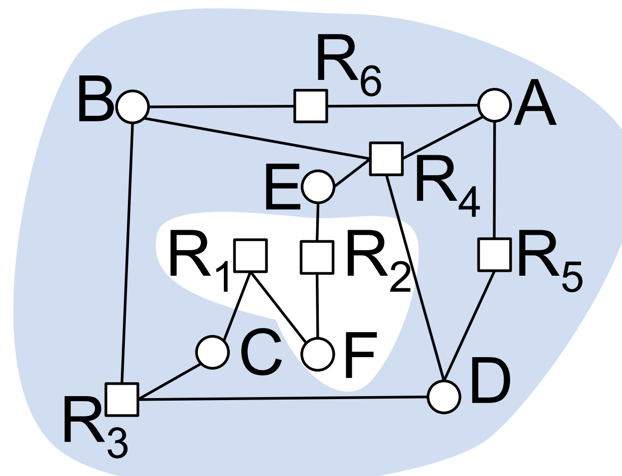
Overview

- Background
- Relational Consistency R($*$,m)C [SAC10,AAAI10]
  - Property, Algorithm, Weakening
  - Characterization, Evaluating
- Relational Neighborhood Inverse Consistency (RNIC) [AAAI11,SARA11]
  - Property, Algorithm
  - Dual-graph reformulation, Characterization, Selection strategy
  - Evaluating
- Dual Graphs of Binary CSPs [CP2012]
  - Complete constraint network, Non-complete constraint network
  - RNIC on binary CSPs
  - Characterization, Evaluating
- Conclusions

Neighborhood Inverse Consistency

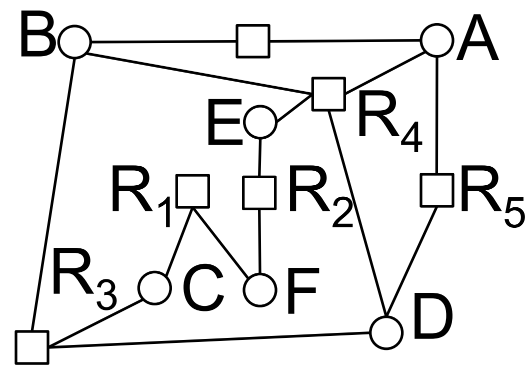
- Property [Freuder+ 96]
  - ↪ Every value can be extended to a solution in its variable's neighborhood
  - ↪ Domain-based property
- Algorithm
  - + No space overhead
  - + Adapts to graph connectivity
- Binary CSPs [Debruyene+ 01]
  - – Not effective on sparse problems
  - – Too costly on dense problems
- Non-binary CSPs?
  - – Neighborhoods likely too large



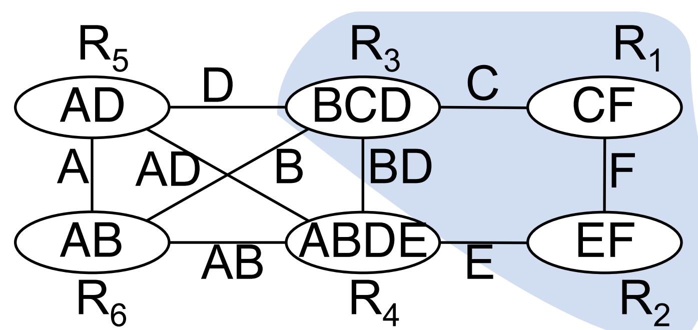


Relational NIC

- Property
  - ↪ Every tuple can be extended to a solution in its relation's neighborhood
  - ↪ Relation-based property
- Algorithm
  - Operates on dual graph
  - Filters relations
  - Does not alter topology of graphs
- Domain filtering
  - Property: RNIC+DF
  - Algorithm: Projection



Hypergraph



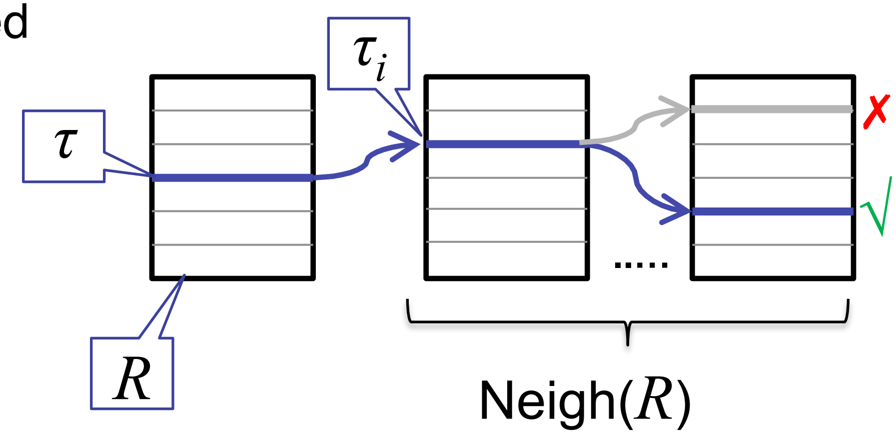
Dual graph

From NIC to RNIC

- Neighborhood Inverse Consistency (NIC) [Freuder+ 96]
  - Proposed for binary CSPs
  - Operates on constraint graph
  - Filters domain of variables
- Relational Neighborhood Inverse Consistency (RNIC)
  - Proposed for both binary & non-binary CSPs
  - Operates on dual graph
  - Filters relations; last step projects updated relations on domains
- Both
  - Adapt consistency level to local topology of constraint network
  - Add no new relations (no constraint synthesis)

Algorithm for Enforcing RNIC

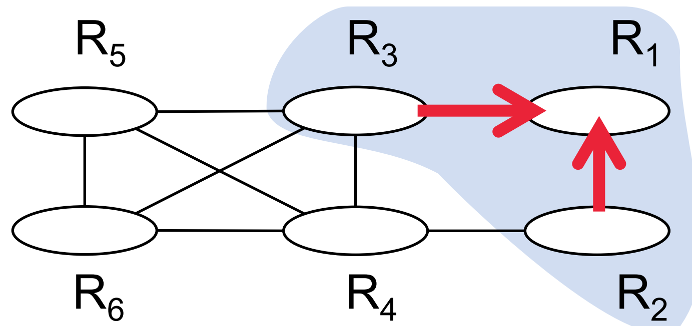
- Two queues
  1. Q: relations to be updated
  2. $Q_t(R)$: The tuples of relation R whose supports must be verified
- SearchSupport(τ,R)
  - Backtrack search on Neigh(R)
- Loop until all $Q_t(\cdot)$ are empty
- Complexity
  - Space: $O(ket\delta)$
  - Time: $O(t^{\delta+1}e\delta)$
  - Efficient for a fixed δ



Improving Algorithm's Performance

Dynamically detect dangles

- Tree structures may show in subproblem @ each instantiation
- Apply directional arc consistency

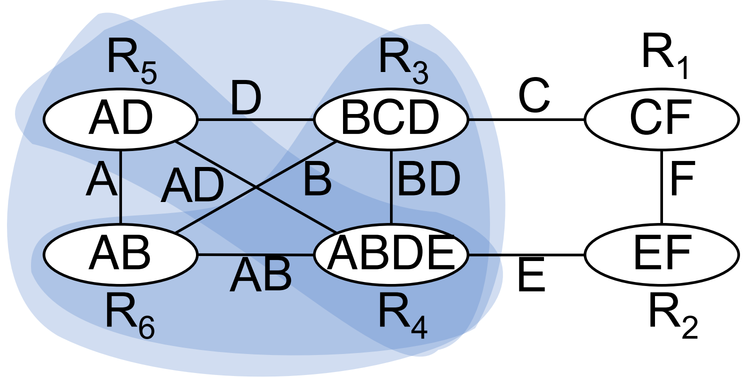


Note that exploiting dangles is

- Not useful in R($*$,m)C: small value of m, subproblem size
- Not applicable to GAC: does not operate on dual graph

Reformulation: Removing Redundant Edges

- High density
  - Large neighborhoods
  - Higher cost of RNIC
- Minimal dual graph
  - Equivalent CSP
  - Computed efficiently [Janssen+ 89]
- Run algorithm on a minimal dual graph
  + Smaller neighborhoods, solution set not affected
  - wRNIC: a strictly weaker property



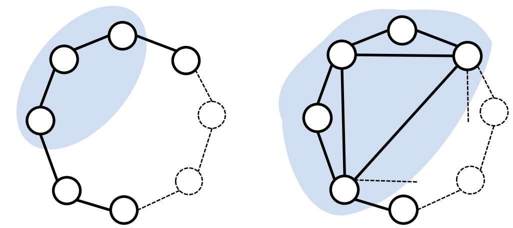
$d^{Go} = 60\%$

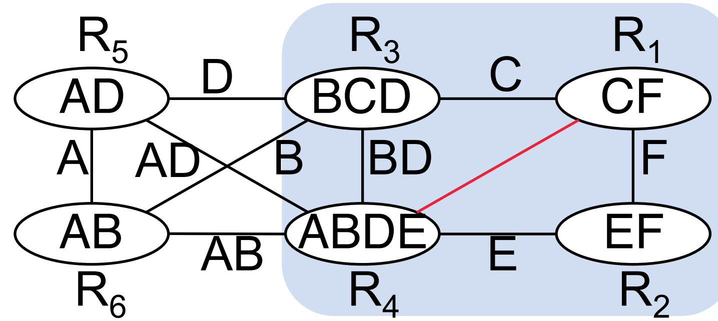
$d^{Gw} = 40\%$

wRNIC → RNIC

Reformulation: Triangulation

- Cycles of length ≥ 4
  - Hampers propagation
- Triangulating dual graph
  - Equivalent CSP
  - Min-fill heuristic
- Run algorithm on a triangulated dual graph
  - + Created loops enhance propagation
  - – triRNIC: a strictly stronger property



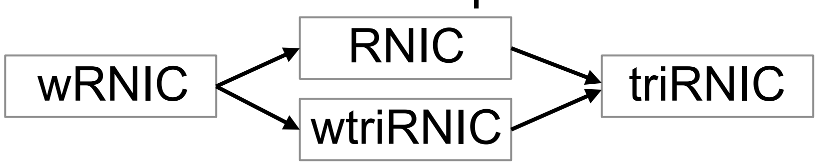


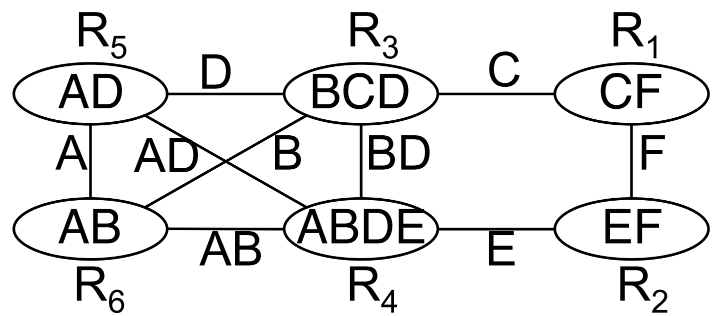
$d^{Go} = 60\%$

$d^{Gtri} = 67\%$

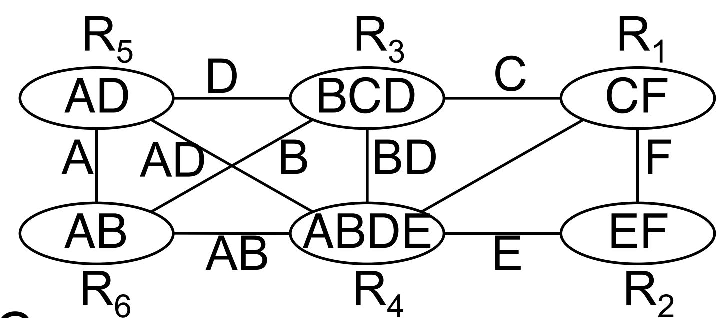
Reformulation: RR & Triangulation

- Fixing the dual graph
  - RR copes with high density
  - Triangulation boosts propagation
- RR+Tri
  - Both operate locally
  - Are complementary, do not ‘clash’
- Run algorithm on a RR+tri dual graph
  - CSP solution set is not affected
  - wtriRNIC is not comparable with RNIC





$d^{Go} = 60\%$



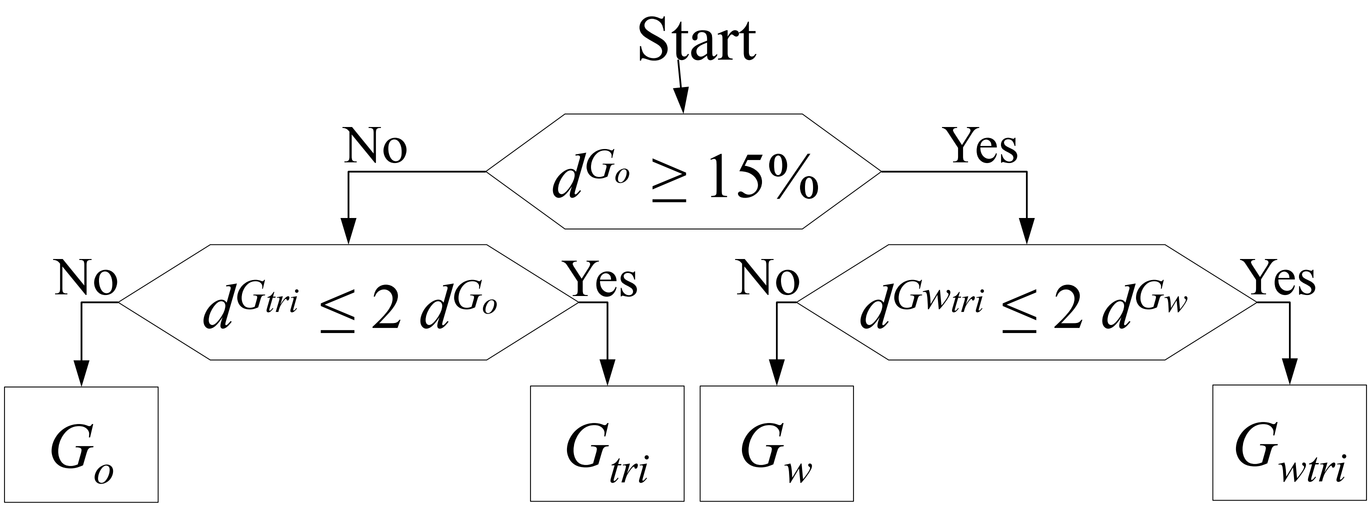
$d^{Gwtri} = 47\%$

Constraint Systems Laboratory

University of Nebraska Lincoln

Selection Strategy: Which? When?

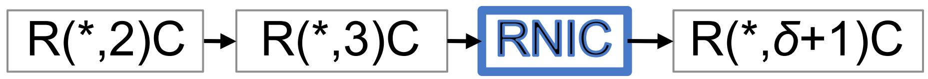
- Density of dual graph ≥ 15% is too dense
  - Remove redundant edges
- Triangulation increases density no more than two fold
  - Reformulate by triangulation
- Each reformulation executed at most once



Characterizing RNIC

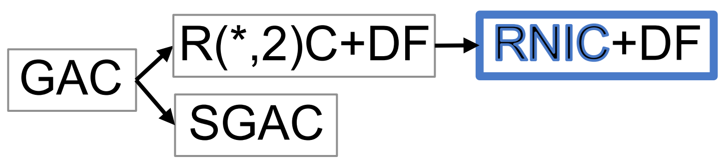
R(∗,*m*)C

- Relation-based property



GAC, SGAC

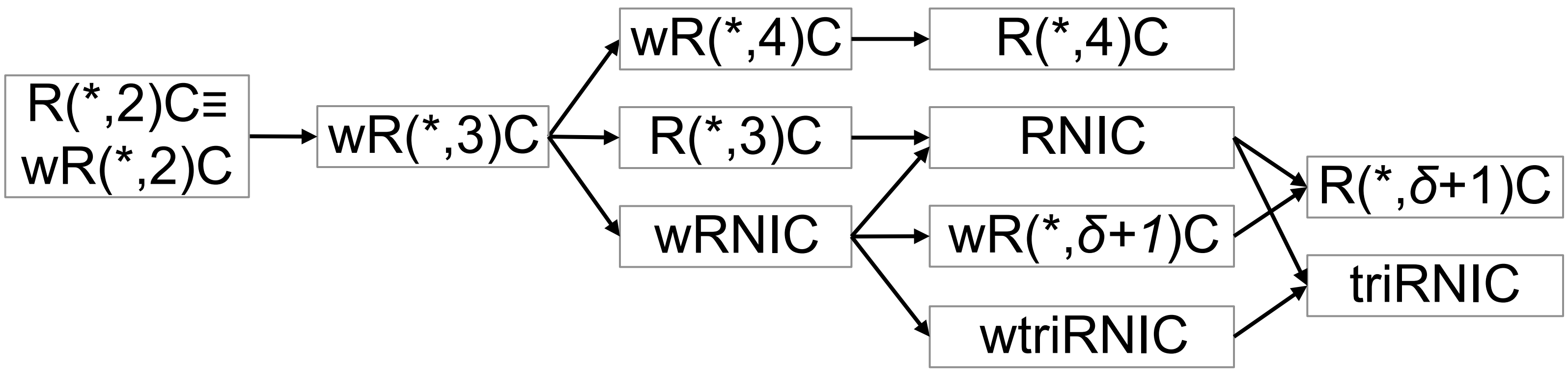
- Variable-based properties



$p \rightarrow p'$: p is strictly weaker than p'

Characterizing RNIC

- The fuller picture



  - w: Property weakened by redundancy removal
  - tri: Property strengthened by triangulation
  - δ: Degree of dual network

Experimental Setup

- Backtrack search with full lookahead
- We compare
  - wR($*$,m)C for $m = 2,3,4$
  - GAC
  - RNIC and its variations
- General conclusion
  - GAC best on random problems
  - RNIC-based best on structured/quasi-structued problems
- We focus on non-binary results (960 instances)
  - triRNIC theoretically has the least number of nodes visited
  - selRNIC solves most instances backtrack free (652 instances)

Category	#Binary	#Non-binary
Academic	31	0
Assignment	7	50
Boolean	0	160
Crossword	0	258
Latin square	50	0
Quasi-random	390	25
Random	980	290
TSP	0	30
Unsolvable		
Memory	10	60
All timed out	447	87

Experimental Results

- Statistical analysis on CP benchmarks
- **Time**: Censored data calculated mean
- **Rank**: Censored data rank based on probability of survival data analysis
- **#F**: Number of instances fastest
- **$[\cdot]_{CPU}$**: Equivalence classes based on CPU
- **$[\cdot]_{Completion}$**: Equivalence classes based on completion
- **#C**: Number of instances completed
- **#BT-free**: # instances solved backtrack free

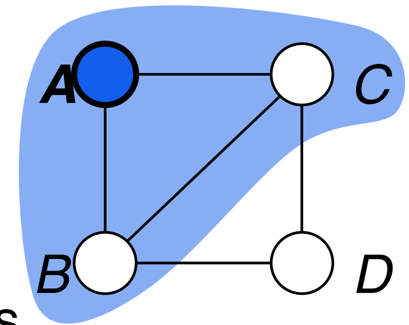
Algorithm	Time	Rank	#F	$[\cdot]_{CPU}$	#C	$[\cdot]_{Completion}$	#BT-free
169 instances: aim-100,aim-200,lexVg,modifiedRenault,ssa							
wR(∗,2)C	944,924	3	52	**A**	138	B	79
wR(∗,3)C	925,004	4	8	B	134	B	92
wR(∗,4)C	1,161,261	5	2	B	132	B	108
GAC	1,711,511	7	**83**	C	119	C	33
RNIC	6,161,391	8	19	C	100	C	66
triRNIC	3,017,169	9	9	C	84	C	80
wRNIC	1,184,844	6	8	B	131	B	84
wtriRNIC	937,904	2	3	B	144	B	129
selRNIC	**751,586**	**1**	17	**A**	**159**	**A**	**142**

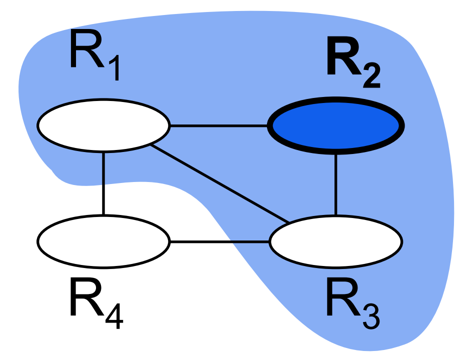
Overview

- Background
- Relational Consistency R($*$,m)C [SAC10,AAAI10]
  - Property, Algorithm, Weakening
  - Characterization, Evaluating
- Relational Neighborhood Inverse Consistency (RNIC) [AAAI11,SARA11]
  - Property, Algorithm
  - Dual-graph reformulation, Characterization, Selection strategy
  - Evaluating
- Dual Graphs of Binary CSPs [CP2012]
  - Complete constraint network, Non-complete constraint network
  - RNIC on binary CSPs
  - Characterization, Evaluating
- Conclusions

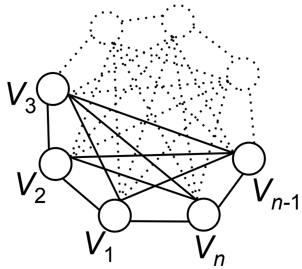
Neighborhood Inverse Consistency

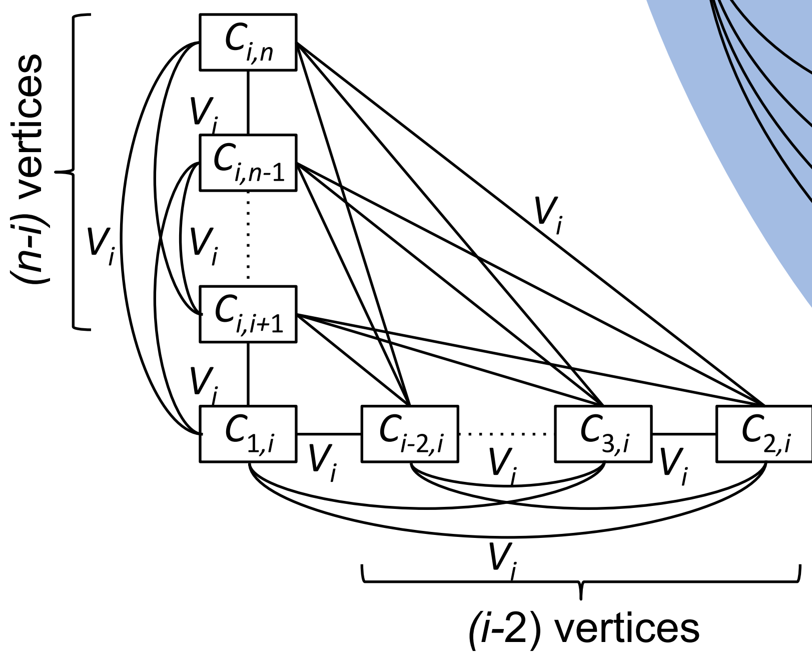
- Relational NIC [Woodward+ AAAI 11]
  - Reformulation of NIC [Freuder & Elfe, AAAI 96]
  - Defined for dual graph
  - Algorithm operates on dual graph & filter relations (not domains!)
  - Initially designed for non-binary CSPs
- How about RNIC on binary CSPs?
  - Impact of the structure of the dual graph



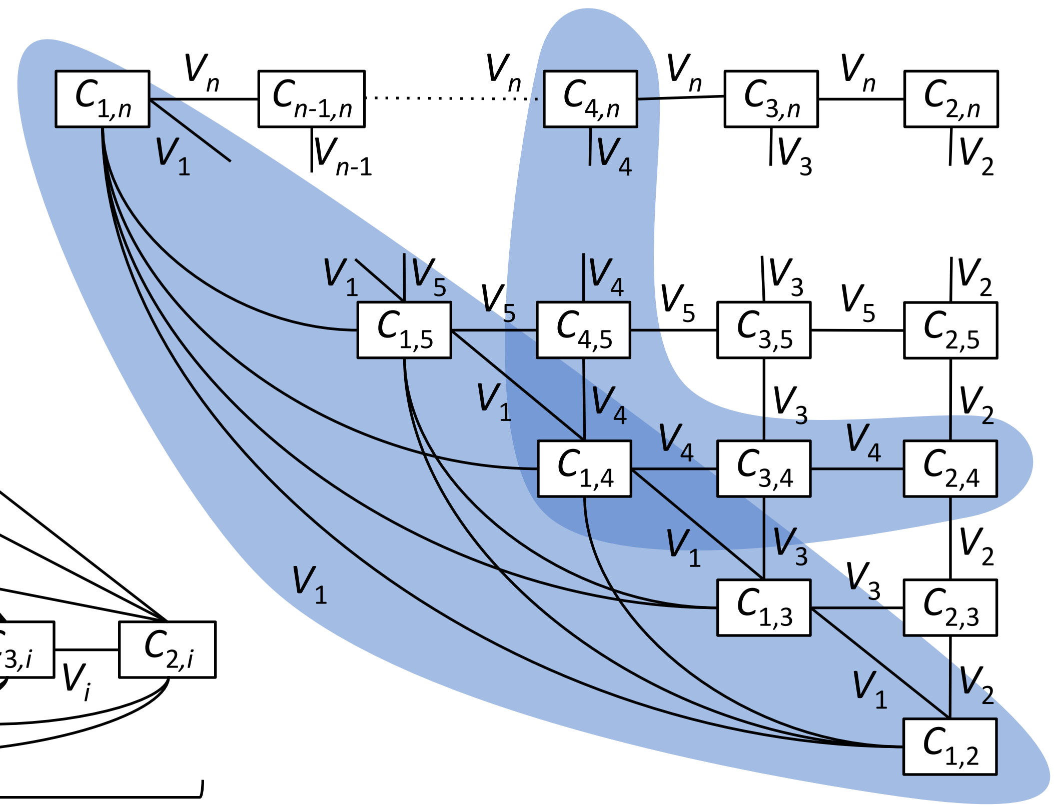


Complete Constraint Graph

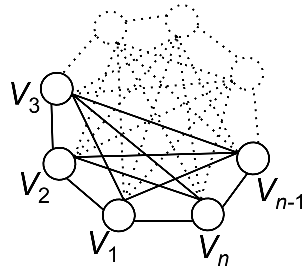




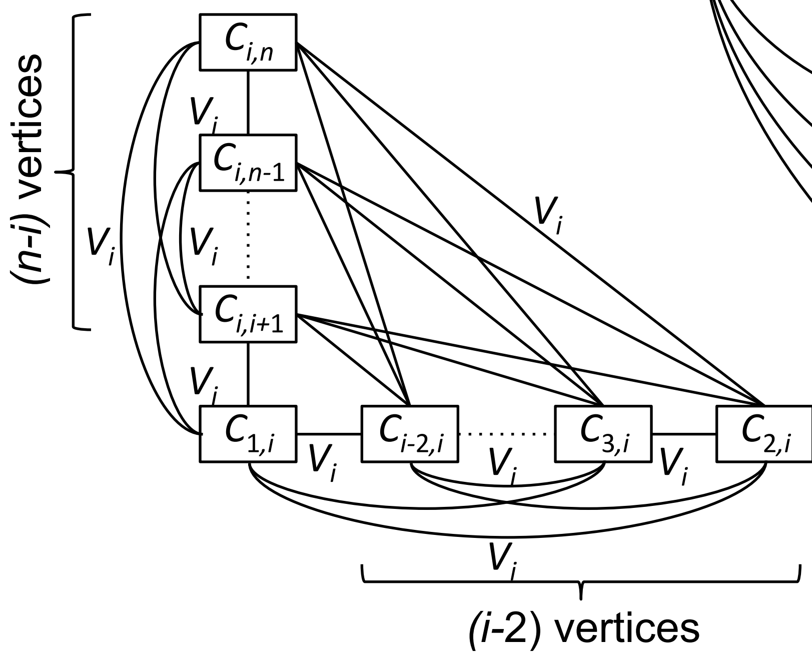
Dual Graph: Triangle shaped grids

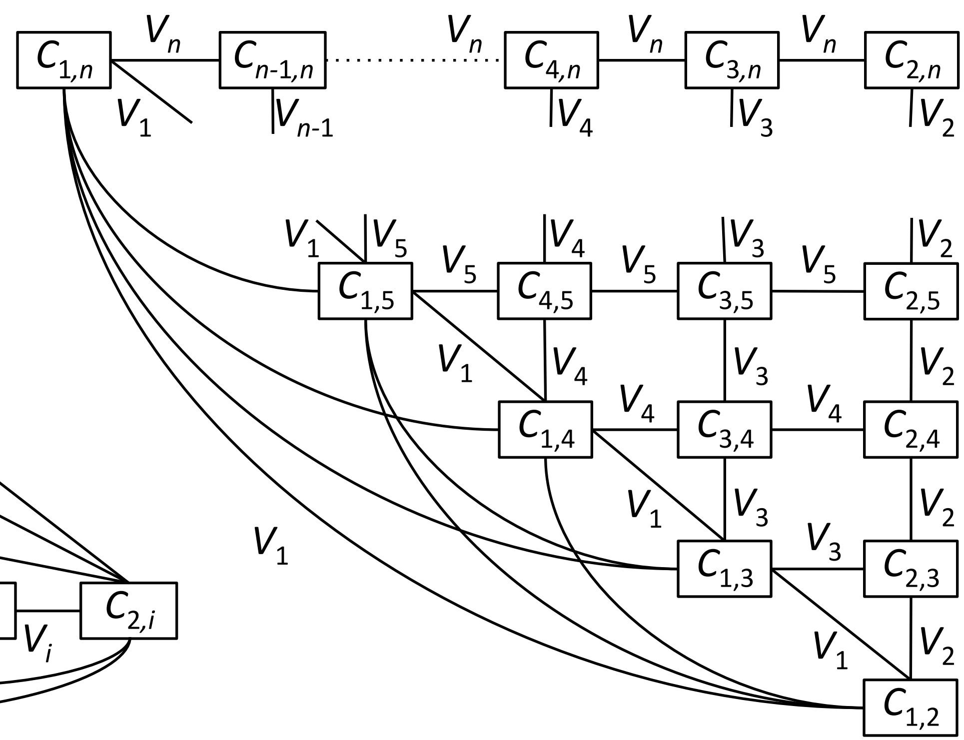


Minimal Dual Graph

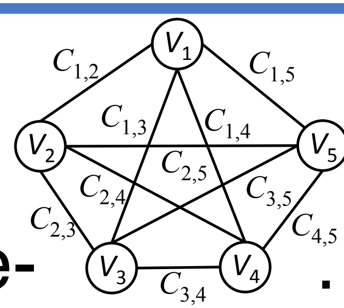


Dual Graph: Triangle shaped grids

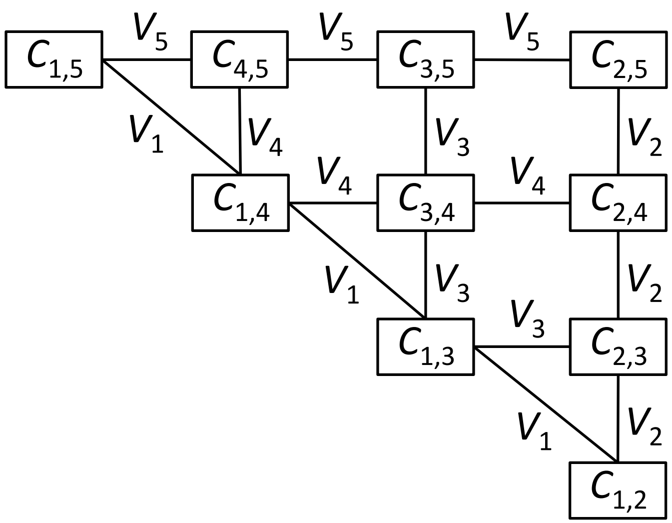




Minimal Dual Graph

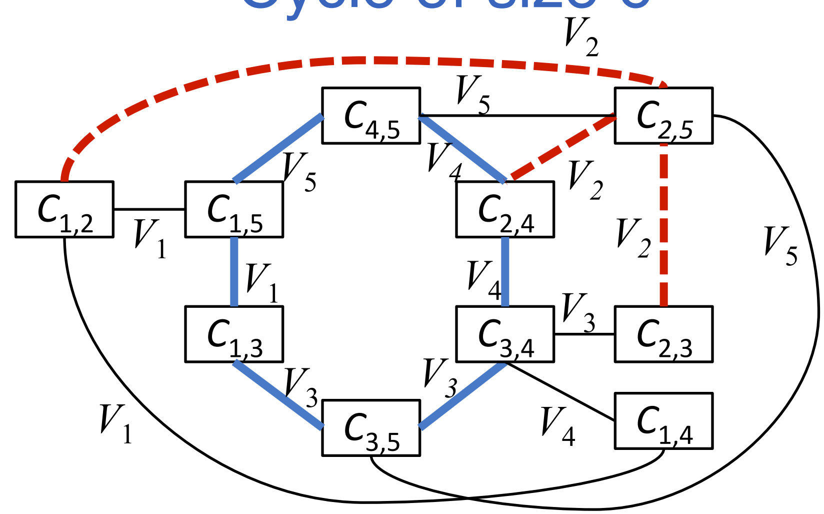


… can be a triangle-shaped grid (planar)



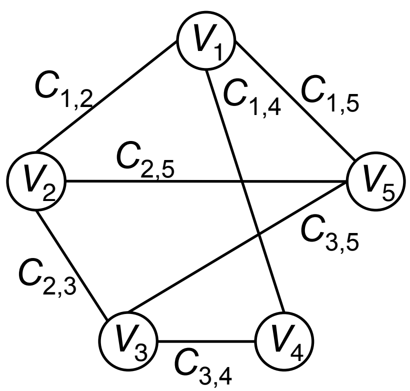
… but does not have to be

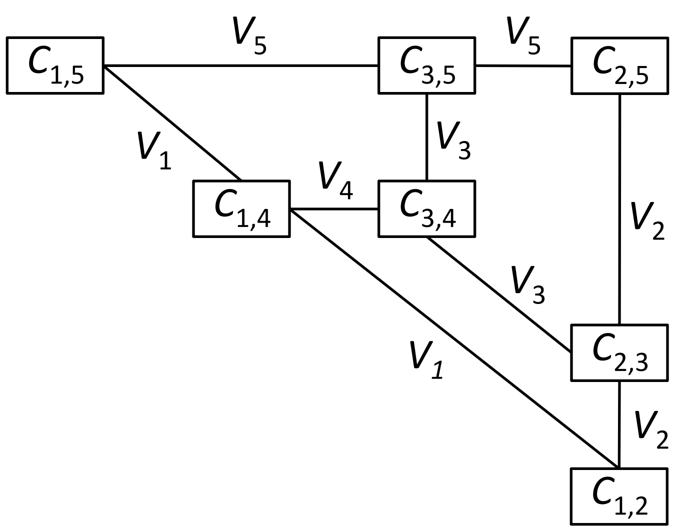
- Star on V_2
- Cycle of size 6



Non-Complete Constraint Graph

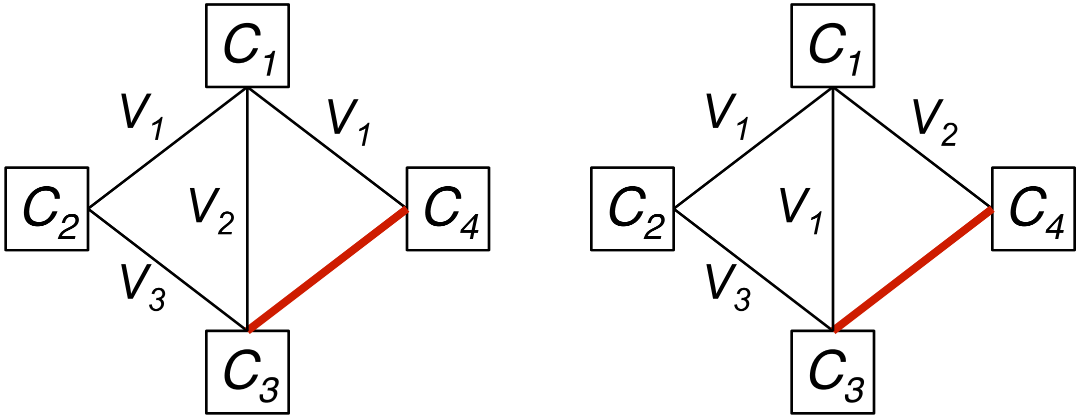
- Can still be a triangle-shaped grid
  - Have a chain of vertices
  - of length ≤ *n*-1





wRNIC on Binary CSPs

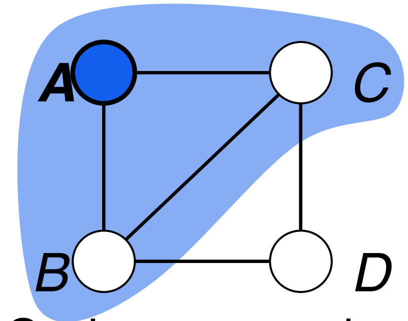
- *On a binary CSP, RNIC enforced on the minimal dual graph (wRNIC) is never strictly stronger than R(*,3)C.*
- wRNIC can never consider more than 3 relations



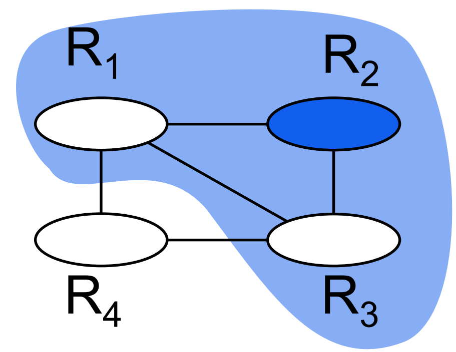
- In either case, it is not possible to have an edge between C_3 & C_4 (a common variable to C_3 & C_4) while keeping C_3 as a binary constraint

NIC, sCDC, and RNIC not comparable

- NIC Property [Freuder & Elfe, AAAI 96]
  - ↪ Every value can be extended to a solution in its variable's neighborhood



- sCDC Property [Lecoutre+, JAIR 11]
  - ↪ An instantiation {(*x*,*a*),(*y*,*b*)} is DC iff (*y*,*b*) holds in SAC when *x*=*a* and (*x*,*a*) holds in SAC when *y*=*b* and (*x*,*y*) in scope of some constraint. Further, the problem is also AC.
- RNIC Property [Woodward+, AAAI 11]
  - ↪ Every tuple can be extended to a solution in its relation's neighborhood
  - ↪ wRNIC, triRNIC, wtriRNIC enforce RNIC on a minimal, triangulated, and minimal triangulated dual graph, respectively
  - ↪ **selRNIC** automatically selects the RNIC variant based on the density of the dual graph



Experimental Results (CPU Time)

Benchmark	# inst.	AC3.1	sCDC1	NIC	selRNIC
		CPU Time (msec)			
		NIC Quickest			
bqwh-16-106	100/100	3,505	3,860	**1,470**	3,608
bqwh-18-141	100/100	68,629	82,772	**38,877**	77,981
coloring-sgb-queen	12/50	680,140	(+3) -	**(+9) 57,545**	634,029
coloring-sgb-games	3/4	41,317	33,307	**(+1) 860**	41,747
rand-2-23	10/10	1,467,246	1,460,089	**987,312**	1,171,444
rand-2-24	3/10	**567,620**	677,253	**(+7)** 3,456,437	677,883
		sCDC Quickest			
driver	2/7	(+5) 70,990	**(+5) 17,070**	358,790	(+4) 185,220
ehi-85	87/100	(+13) 27,304	**(+13) 573**	513,459	(+13) 75,847
ehi-90	89/100	(+11) 34,687	**(+11) 605**	713,045	(+11) 90,891
frb35-17	10/10	41,249	**38,927**	179,763	73,119
		RNIC Quickest			
composed-25-1-25	10/10	226	335	1,457	**114**
composed-25-1-2	10/10	223	283	1,450	**88**
composed-25-1-40	9/10	(+1) 288	(+1) 357	120,544	**(+1) 137**
composed-25-1-80	10/10	223	417	(+1) -	**190**
composed-75-1-25	10/10	2,701	1,444	363,785	**305**
composed-75-1-2	10/10	2,349	1,733	48,249	**292**
composed-75-1-40	7/10	(+1) 1,924	(+3) 1,647	631,040	**(+3) 286**
composed-75-1-80	10/10	1,484	1,473	(+1) -	**397**

Experimental Results (BT-free, #NV)

Benchmark	# inst.	AC3.1	sCDC1	NIC	selRNIC	AC3.1	sCDC1	NIC	selRNIC
		BT-Free				#NV			
		NIC Quickest							
bqwh-16-106	100/100	0	3	**8**	5	1,807	1,881	**739**	1,310
bqwh-18-141	100/100	0	0	**1**	0	25,283	25,998	**12,490**	22,518
coloring-sgb-queen	12/50	1	-	**16**	1	91,853	-	**15,798**	91,853
coloring-sgb-games	3/4	1	1	**4**	1	14,368	14,368	**40**	14,368
rand-2-23	10/10	0	0	**10**	0	471,111	471,111	**12**	471,111
rand-2-24	3/10	0	0	**10**	0	222,085	222,085	**24**	222,085
		sCDC Quickest							
driver	2/7	1	**2**	1	1	3,893	**409**	3,763	3,763
ehi-85	87/100	0	**100**	87	**100**	1,425	**0**	**0**	**0**
ehi-90	89/100	0	**100**	89	**100**	1,298	**0**	**0**	**0**
frb35-17	10/10	0	0	0	0	24,491	24,491	24,491	**24,346**
		RNIC Quickest							
composed-25-1-25	10/10	0	**10**	**10**	**10**	153	**0**	**0**	**0**
composed-25-1-2	10/10	0	**10**	**10**	**10**	162	**0**	**0**	**0**
composed-25-1-40	9/10	0	**10**	9	**10**	172	**0**	**0**	**0**
composed-25-1-80	10/10	0	**10**	-	**10**	112	**0**	-	**0**
composed-75-1-25	10/10	0	**10**	**10**	**10**	345	**0**	**0**	**0**
composed-75-1-2	10/10	0	**10**	**10**	**10**	346	**0**	**0**	**0**
composed-75-1-40	7/10	0	**10**	7	**10**	335	**0**	**0**	**0**
composed-75-1-80	10/10	0	**10**	-	**10**	199	**0**	-	**0**

Conclusions

- Introduced R($*$,m)C, RNIC
- Algorithm for enforcing R($*$,m)C and RNIC
  - BT-free search: hints to problem tractability
- Various reformulations of the dual graph
- Adaptive, unifying, self-regulatory, automatic strategy for RNIC
- Structure of binary dual graph
- Empirical evidence, supported by statistics

Thank You!

Questions?

Enforcing R(*,m)C on the Induced Dual CSP P_ω

Q	
$<\omega_1,R_1>$	ω_1
$<\omega_1,R_2>$	
$<\omega_1,R_5>$	
$<\omega_2,R_2>$	ω_2
$<\omega_2,R_5>$	
$<\omega_2,R_4>$	
$<\omega_3,R_3>$	ω_3
$<\omega_3,R_4>$	
$<\omega_3,R_5>$	

Extract $<\omega,R>$ from *Q*

- Define CSP P_ω
  - For each τ in *R*
    - Assign τ as a value for *R*
    - Solve P_ω (with τ fixed) with forward checking
    - If no solution found: delete τ
    - Add $<\omega', R'>$ to Q: $R_i \neq R'$, $R_i \in \omega'$ and $R' \in \omega'$