Improving Relational Consistency Algorithms Using Dynamic Relation Partitioning

Anthony Schneider[1] Robert J. Woodward[1,2]
Berthe Y. Choueiry[1] & Christian Bessiere[2]

[1]Constraint Systems Laboratory, University of Nebraska-Lincoln

[2]LIRMM, CNRS & University of Montpellier II

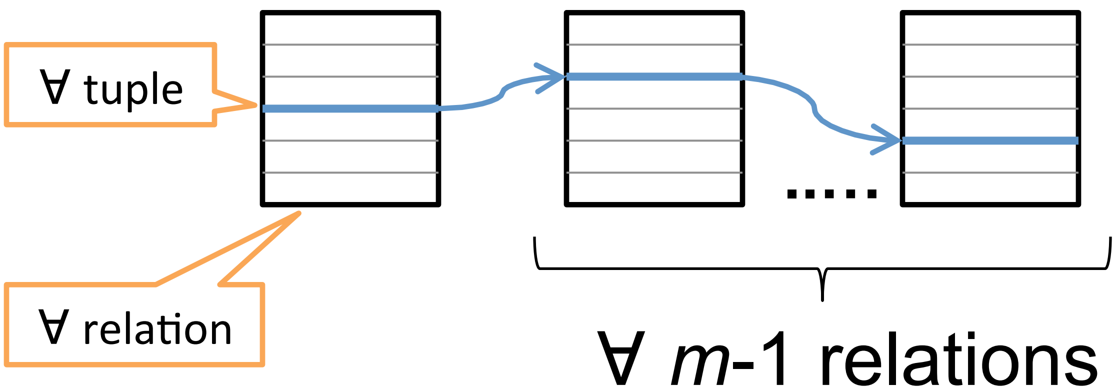
Acknowledgements
- Experiments conducted at UNL's Holland Computing Center
- Woodward supported by a NSF Graduate Research Fellowship grant number 1041000
- NSF Grant No. RI-111795 and EU project ICON (FP7-284715)

Outline

- Introduction
  - R($*$,m)C property and its algorithm PERTUPLE
- Partitioning a relation
  - Coarse, fine, intermediate blocks
- Improve PERTUPLE using partitions
  - PERTUPLE → PERFB
- Experimental results
- Conclusion

R($*$,*m*)C (a.k.a. *m*-wise consistency)

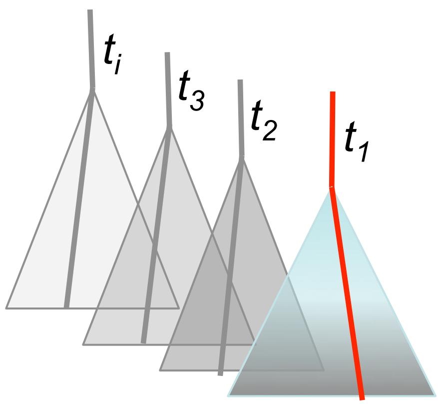
- A CSP is R($*$,*m*)C iff
  - Every tuple in a relation can be extended
  - to the variables in the scope of any (*m*-1) other relations
  - in an assignment satisfying all *m* relations simultaneously
- R($*$,*m*)C $\equiv$ Every set of *m* relations is minimal



PERTUPLE

[Karakashian+ AAAI 10]

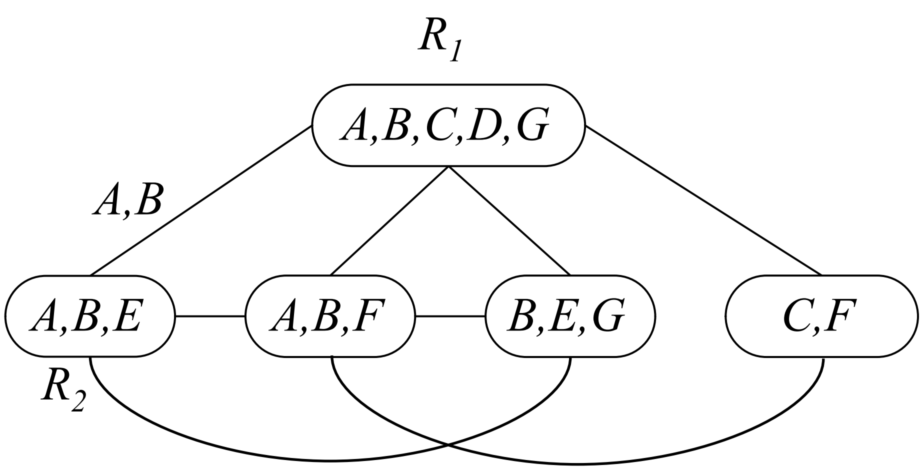
- PERTUPLE enforces R($*$,m)C
- Store all connected combinations of *m* relations
- For each relation in a combination
  - For each tuple in the relation
  - SEARCHSUPPORT: Conduct backtrack search with FC over the dual CSP induced by the *m* relations
  - Remove the tuple if no solution is found
- Update propagation queue



Piecewise Functional Constraints

- PW-AC enforces R(∗,2)C

[Samaras & Stergiou JAIR 05]



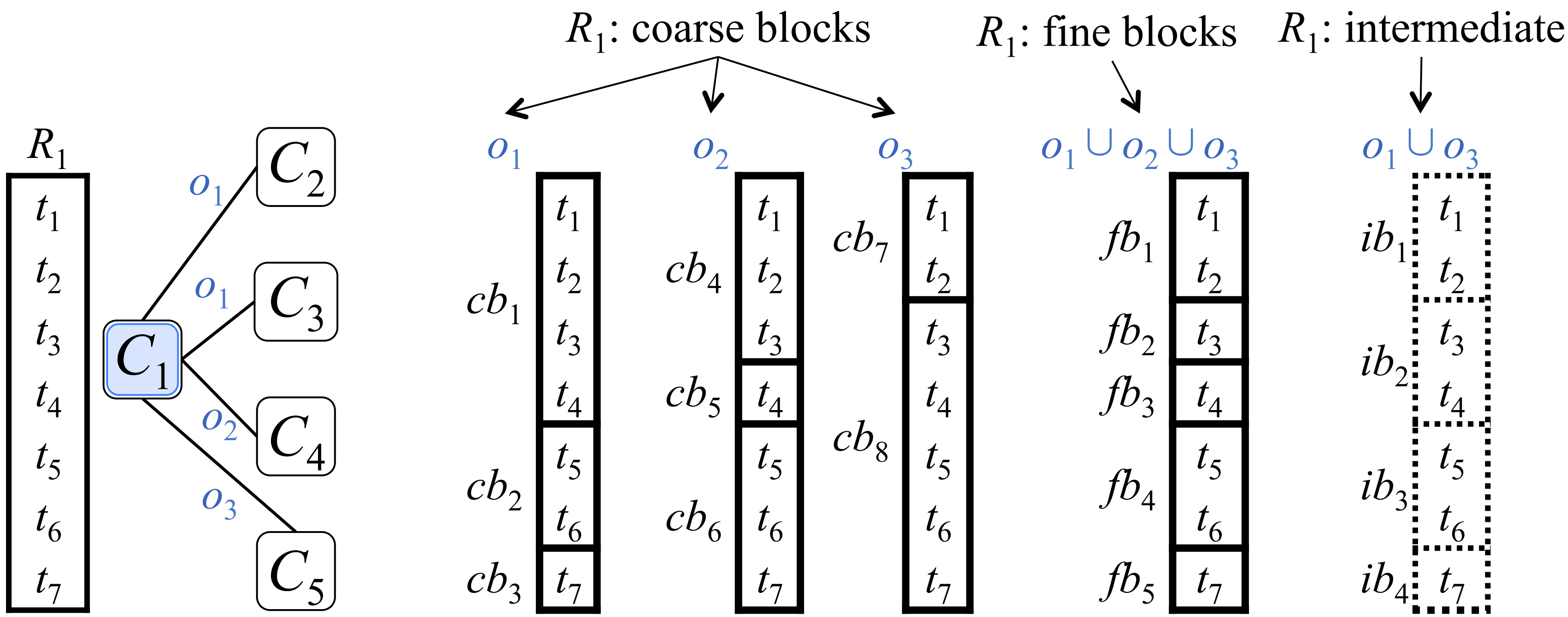
R_1

	A	B	C	D	G	
t_1	0	0	0	0	0	
t_2	0	0	0	1	0	
t_3	0	0	1	0	0	
t_4	0	0	1	1	1	
t_5	0	1	1	0	1	
t_6	0	1	1	1	1	
t_7	1	1	1	1	1	✗

R_2

A	B	E	
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	✗
1	0	1	✗

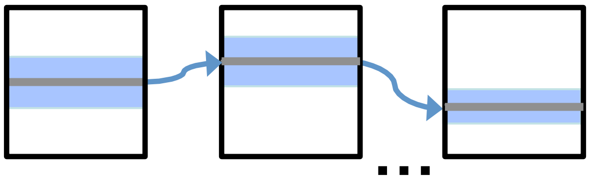
Types of Partitions



- For each relation, we store
  - A single partition of fine blocks
  - As many partitions of coarse blocks as shared subscopes

PerTuple → PerFB

- Tuple → Fine Block (FB)
- General mechanism is identical
  - Combinations, queuing, and propagation
- SearchSupport
  - Search enumerates fine blocks, not tuples
  - Forward checking operates on coarse blocks



- Calls to SearchSupport reduced
  - Skips tuples in the same fine blocks
  - Skips fine blocks in the same intermediate block
  - Using two local data structures
- Details in paper

Enforcing R($*$,m)C with PerFB

Vars in subscopes

Rel	***Variables***

Equiv FBs for R_1

A	***B***	***C***	***Consistent***
θ	θ	θ	True

R_1

		A	***B***	***C***	***D***	***G***
fb_1	t_1	0	0	0	0	0
	t_2	0	0	0	1	0
fb_2	t_3	0	0	1	0	0
fb_3	t_4	0	0	1	1	1
fb_4	t_5	0	1	1	0	1
	t_5	0	1	1	1	1
fb_5	t_7	1	1	1	1	1

R_2

	A	***B***	***E***
fb_6	0	0	0
fb_7	0	0	1
fb_8	0	1	0
fb_9	0	1	1
fb_{10}	1	0	0
fb_{11}	1	0	1

R_5

	C	***F***
fb_{20}	0	0
fb_{21}	0	1
fb_{22}	1	0

Enforcing R($*$,m)C with PerFB

Vars in subscopes

Rel	*Variables*
R_1	*ABC*

?

Equiv FBs for R_1

A	*B*	*C*	*Consistent*
0	0	0	True
0	0	1	True

R_1		*A*	*B*	*C*	*D*	*G*
fb_1	t_1	0	0	0	0	0
	t_2	0	0	0	1	0
fb_2	t_3	0	0	1	0	0
fb_3	t_4	0	0	1	1	1
fb_4	t_5	0	1	1	0	1
	t_5	0	1	1	1	1
fb_5	t_7	1	1	1	1	1

R_2	*A*	*B*	*E*
fb_6	0	0	0
fb_7	0	0	1
fb_8	0	1	0
fb_9	0	1	1
fb_{10}	1	0	0
fb_{11}	1	0	1

R_5	*C*	*F*
fb_{20}	0	0
fb_{21}	0	1
fb_{22}	1	0

Enforcing R(∗,*m*)C with PerFB

Vars in subscopes

Rel	**Variables**
R_1	*ABC*

Yes!

Equiv FBs for R_1

A	*B*	*C*	*Consistent*
0	0	0	True
0	0	1	True

➡

R_1

		A	*B*	*C*	*D*	*G*
fb_1	t_1	0	0	0	0	0
	t_2	0	0	0	1	0
fb_2	t_3	0	0	1	0	0
fb_3	t_4	0	0	1	1	1
fb_4	t_5	0	1	1	0	1
	t_5	0	1	1	1	1
fb_5	t_7	1	1	1	1	1

R_2

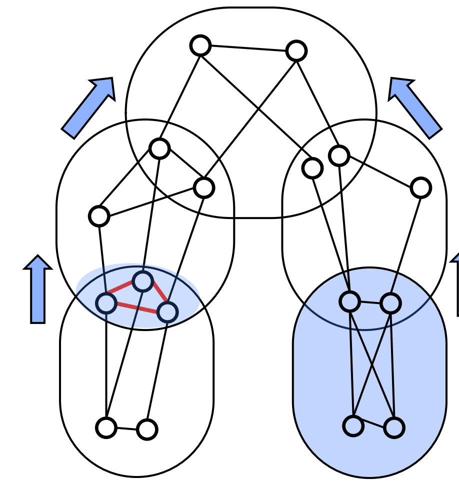
	A	*B*	*E*
fb_6	0	0	0
fb_7	0	0	1
fb_8	0	1	0
fb_9	0	1	1
fb_{10}	1	0	0
fb_{11}	1	0	1

R_5

	C	*F*
fb_{20}	0	0
fb_{21}	0	1
fb_{22}	1	0

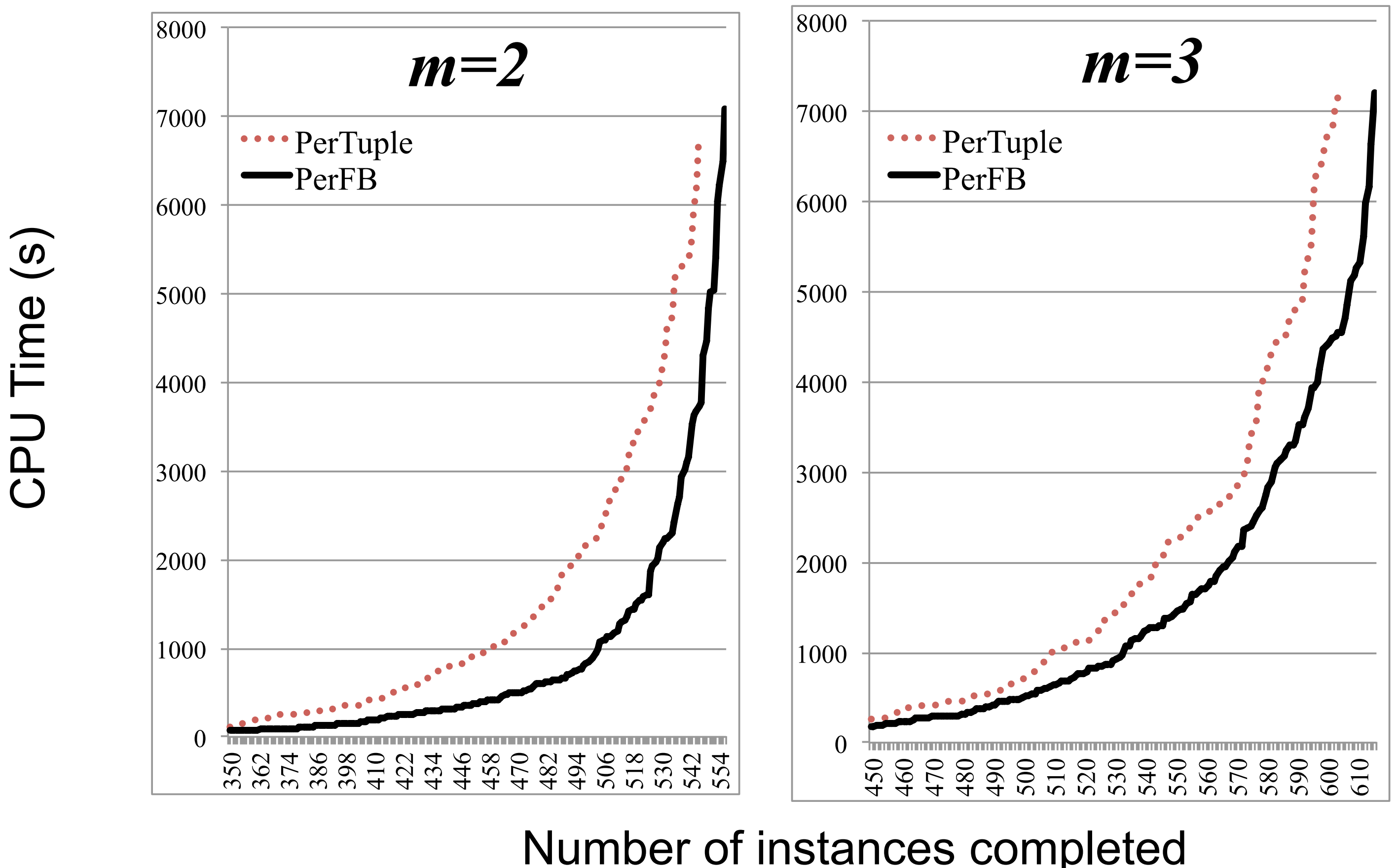
Experimental Setup

- Backtrack search, find first solution, dom/deg
- Maintaining R($*$,m)C [Karakashian+ AAAI 13]
  - Use a tree-decomposition of the CSP
  - Enforce consistency on individual clusters
  - Add projection of constraints to separators to bolster propagation between adjacent clusters
  - Use a minimal dual graph to reduce number of combinations of m constraints
  - Better performance than GAC, maxRPWC



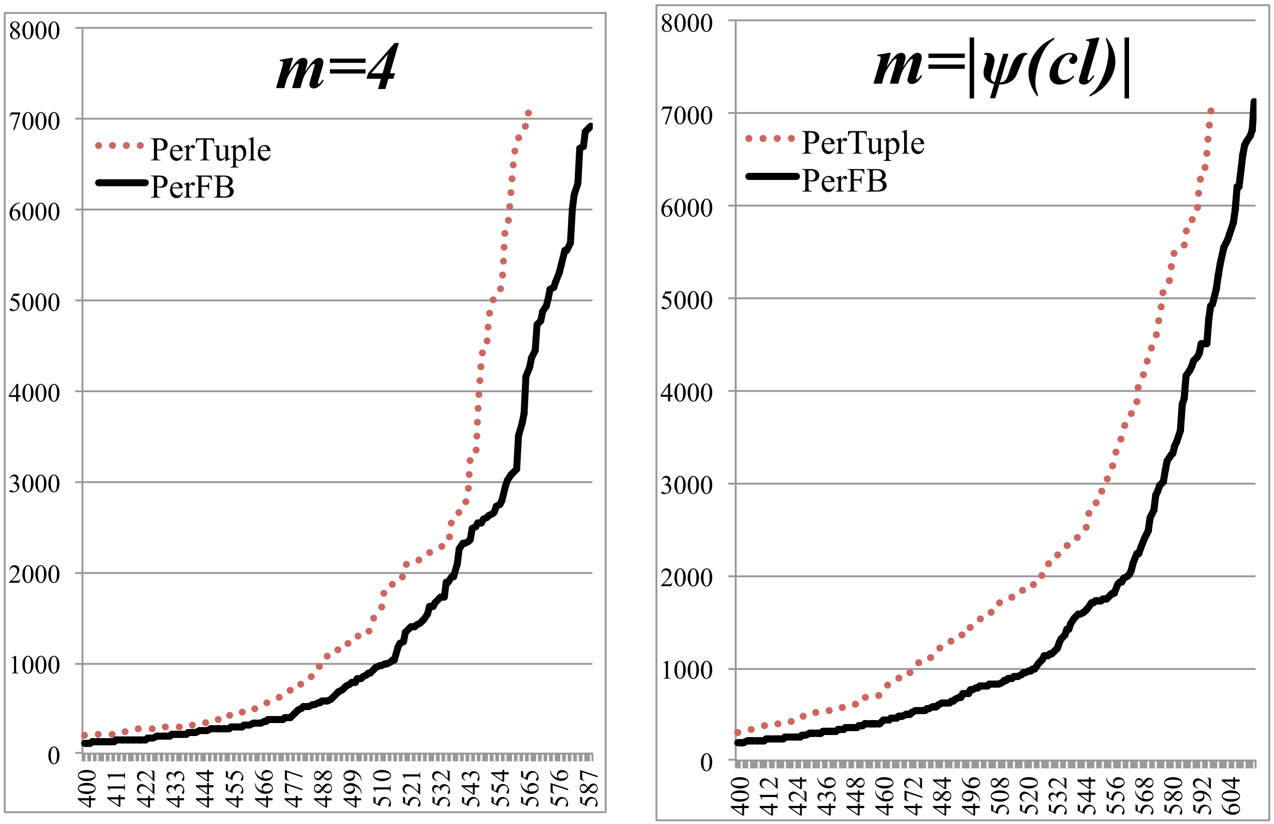
- $m = 2, 3, 4, |\psi(\text{cl})|$
- XCSP benchmark of CP Solver Competition [Lecoutre+]

PERFB vs. PERTUPLE: completed instances



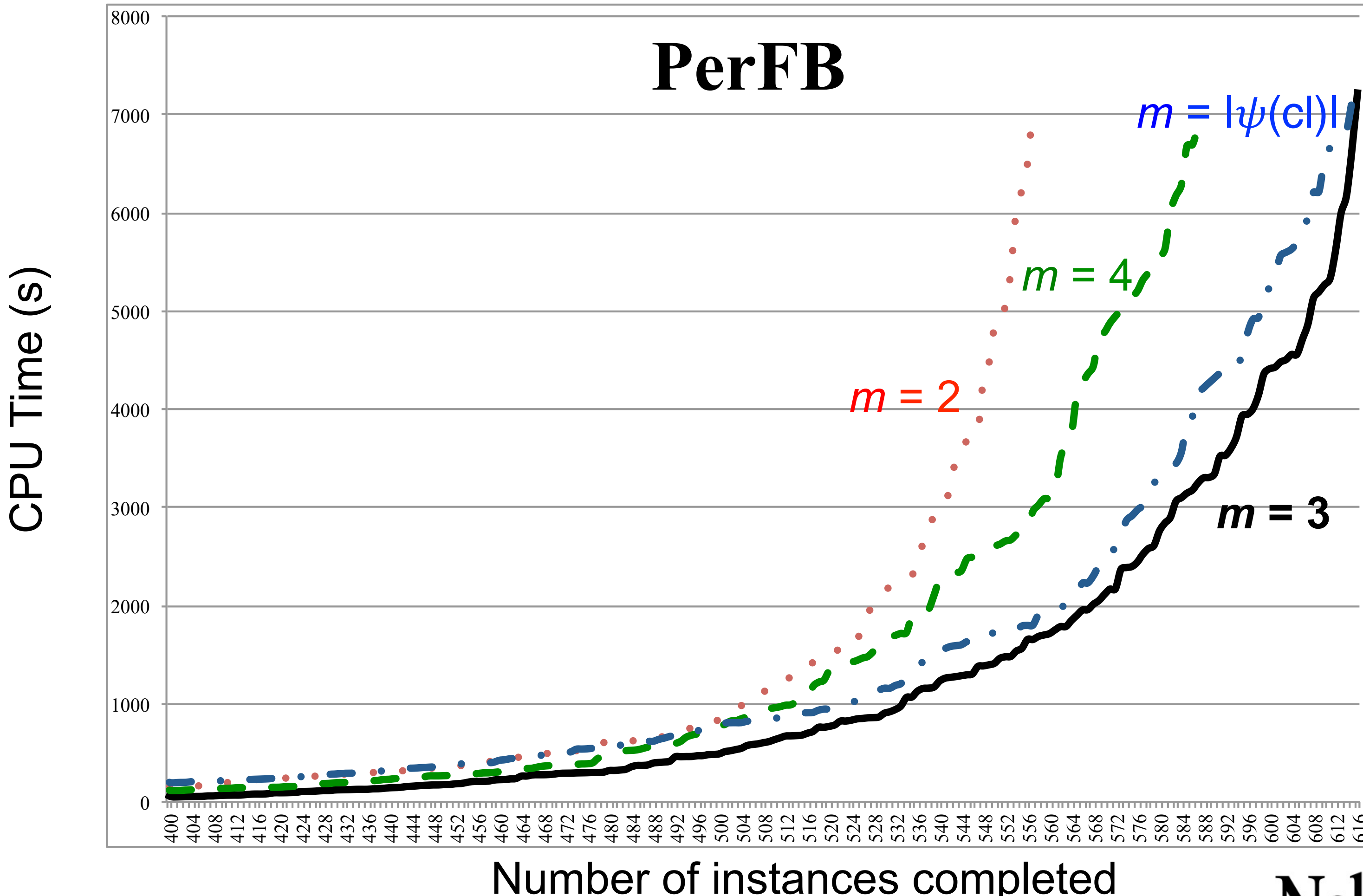
PerFB vs. PerTuple: completed instances

CPU Time (s)



Number of instances completed

PerFB $m = 2,3,4, |\psi(cl)|$



Detailed Results

Number of instances tested: 853

	$m = 2$		$m = 3$		$m = 4$		$m = \lvert\psi(cl)\rvert$	
	PerTuple	PerFB	PerTuple	PerFB	PerTuple	PerFB	PerTuple	PerFB
#Completed	546	**557**	604	**616**	566	**589**	597	**615**
… only by	5	**16**	1	**13**	2	**25**	8	**26**
… by both	541		603		564		589	
Avg. CPU (sec)	538	**227**	521	**362**	472	**314**	669	**458**
SearchSupport calls (10^9)	86.4	0.0	88.1	26.1	52.7	19.6	24.7	8.1
Call ratio	--		3.37		2.69		3.06	

Conclusions

- Contributions
  - Designed PERFB
    - to replace PERTUPLE of [Karakashian+ AAAI 10]
    - by extending the work of [Samaras & Stergiou JAIR 05]
  - Empirically showed benefits of our approach

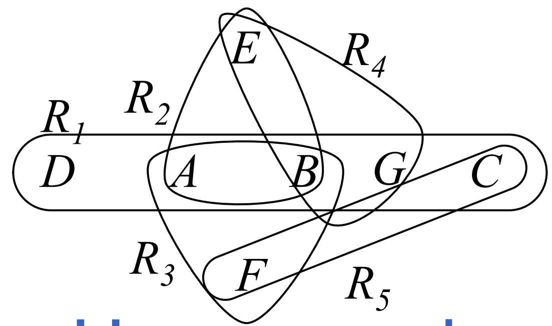
- Future work
  - Extend our approach to our other algorithm for enforcing R($*$,m)C [Karakashian PhD 13]

Thank you for listening

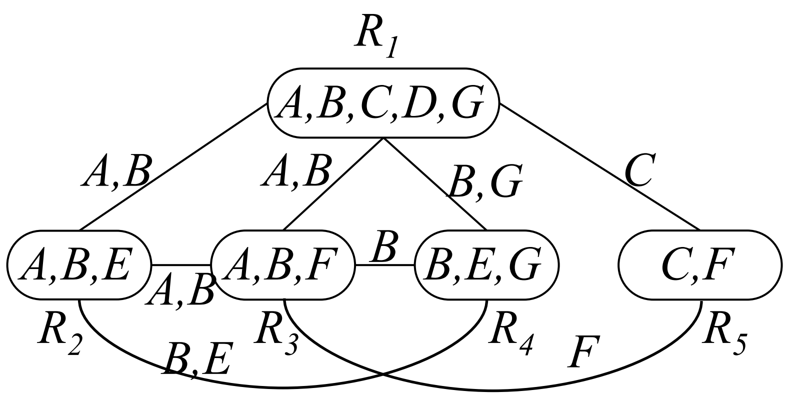
Wake up the Chair!

CSP: Graphical Representations

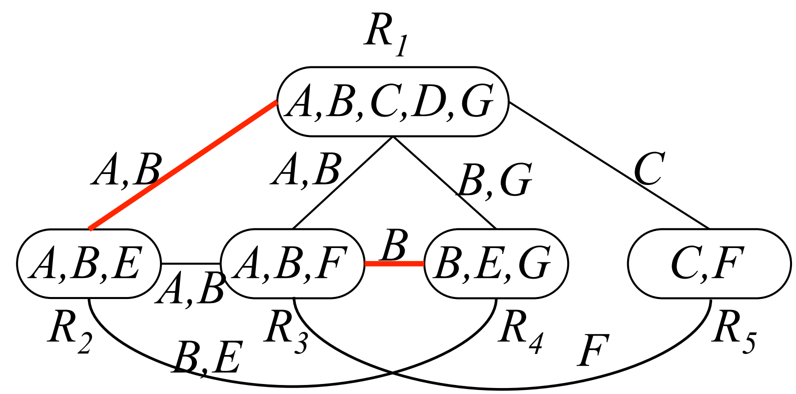
- Hyper graph
- Dual graph
- Minimal dual graph



Hyper graph



Dual graph



Minimal dual graph

[Janssen+,1989]

Block Statistics

Benchmark	Absolute Max	Averages Min	Averages Max	Averages Mean
geom	17	1.0	1.2	1.0
graphColoring-hos	3	1.0	2.0	1.0
graphColoring-sgb-book	12	1.0	7.7	1.1
hanoi	2	1.0	2.0	1.0
modifiedRenault	260	1.0	25.6	1.0
rand-10-20-10	2	1.0	1.3	1.0
renault	4	1.0	4.0	1.0
ssa	8	1.0	3.1	1.1
tightness0.9	38	1.0	28.1	1.0
varDimacs	16	1.0	3.4	1.1

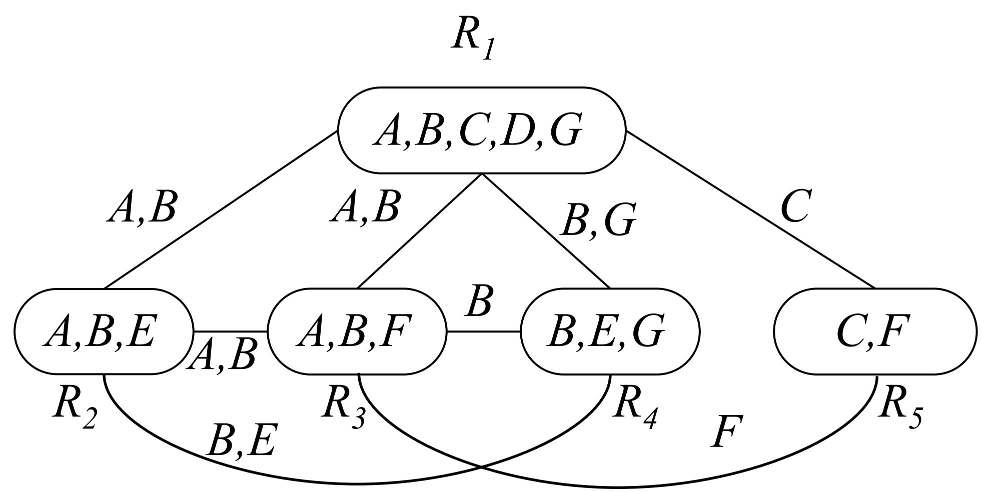
Block Statistics

Benchmark	Abs Max	Averages Min	Max	Mean
aim-50,100,200,pseudo	4	1.0	4.0	2.1
cmpsed-25-1-2,25,40,80	10	1.0	10.0	8.0-8.4
cmpsed-25-10-20	10	1.0	10.0	7.6
cmpsed-75-1-2,25,40,80	10	1.0	10.0	8.3-8.5
dag-rand	108	1.0	91.6	2.9
dubois,pret	2	1.0	2.0	1.5
geom	20	6.4	20.0	15.0
grCol-hos	6	1.0	3.3	3.3
grCol-mug	3	1.0	2.5	2.4
grCol-register-mulsol	48	23.2	23.2	23.2
grCol-sgb-book	12	1.0	7.7	7.5
grCol-sgb-games	8	1.0	6.3	6.1

Benchmark	Abs Max	Averages Min	Max	Mean
grCol-sgb-queen	17	10.3	10.3	10.3
hanoi	3	1.0	3.0	2.9
lexVg	875	1.0	484.7	3.6
modifiedRenault	48,720	1.0	48,720.0	7.9
rand-10-20-10	1,046	1.0	119.2	1.3
rand-3-20-20-fcd	190	1.0	181.5	12.8
renault	48,720	1.0	48,720.0	7.7
rlfapGr/ScensMod	44,43	1.0	30.0,35.6	18.5,19.4
ssa	31	1.0	14.7	2.1
super-queens	49	15.6	17.6	16.4
tightness0.9	40	1.0	36.3	16.9
varDimacs	512	1.0	115.0	5.6

Partitioning Relations -- Definitions

- Scope
- Subscope
- Combination



Coarse Partitions

R_1

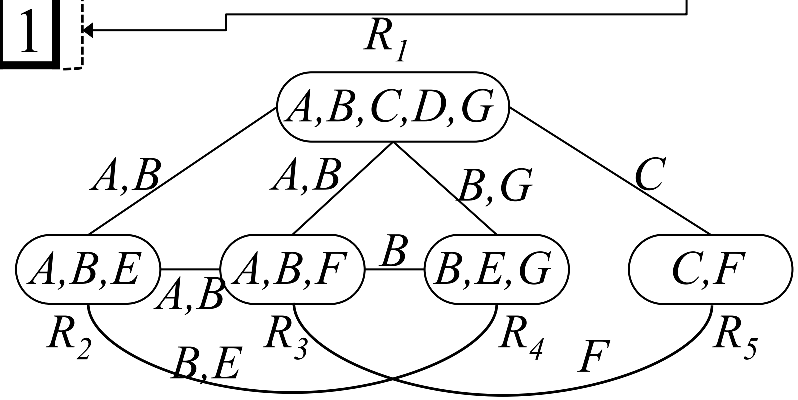
		A	**B**	**C**	**D**	**G**
fb_1	t_1	0	0	0	0	0
	t_2	0	0	0	1	0
fb_2	t_3	0	0	1	0	0
fb_3	t_4	0	0	1	1	1
fb_4	t_5	0	1	1	0	1
	t_5	0	1	1	1	1
fb_5	t_7	1	1	1	1	1

R_2

	A	**B**	**E**
fb_6	0	0	0
fb_7	0	0	1
fb_8	0	1	0
fb_9	0	1	1
fb_{10}	1	0	0
fb_{11}	1	0	1

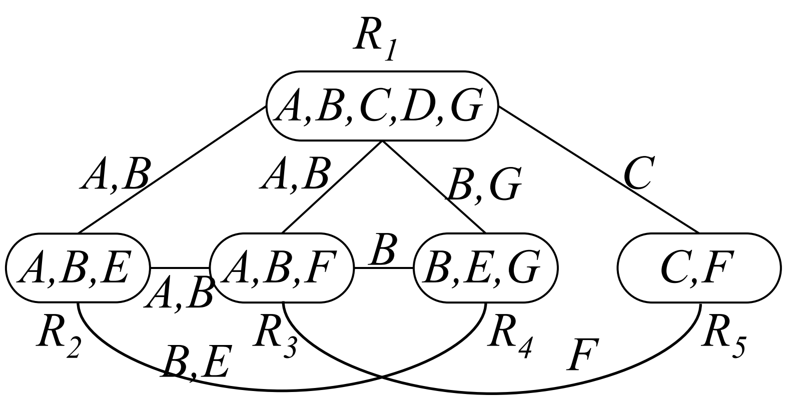
R_3

	A	**B**	**F**
fb_{12}	0	0	0
fb_{13}	0	0	1
fb_{14}	0	1	1
fb_{15}	1	1	0
fb_{16}	1	1	1



Fine Partitions

- Equivalence class that induces smallest set of tuples

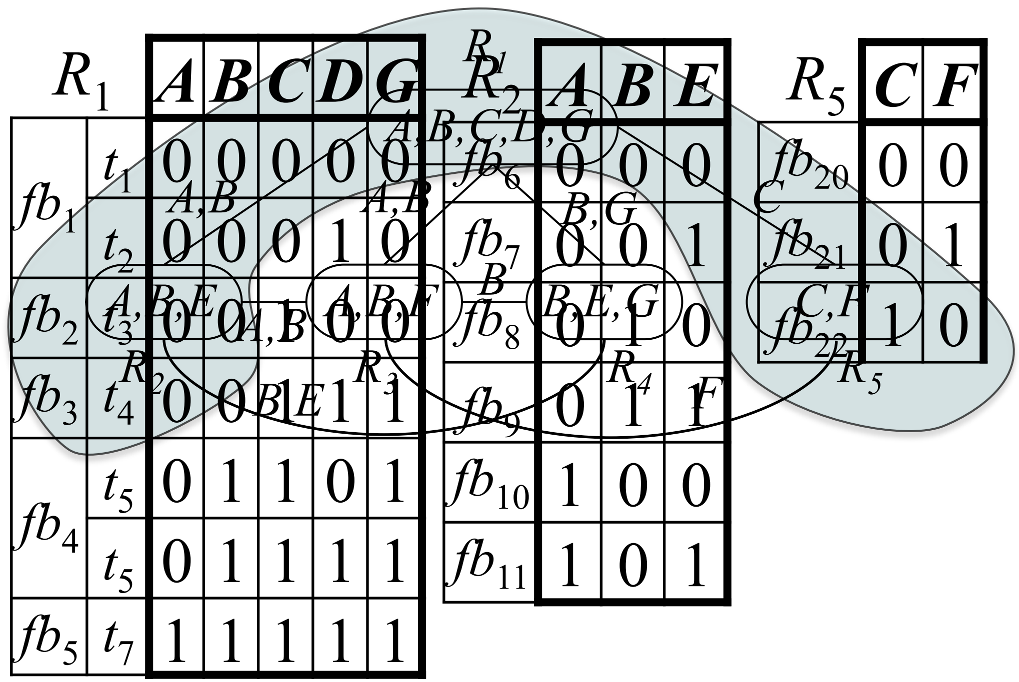


R_1		A	B	C	D	G
fb_1	t_1	0	0	0	0	0
	t_2	0	0	0	1	0
fb_2	t_3	0	0	1	0	0
fb_3	t_4	0	0	1	1	1
fb_4	t_5	0	1	1	0	1
	t_5	0	1	1	1	1
fb_5	t_7	1	1	1	1	1

R_2	A	B	E
fb_6	0	0	0
fb_7	0	0	1
fb_8	0	1	0
fb_9	0	1	1
fb_{10}	1	0	0
fb_{11}	1	0	1

Intermediate Partitions

- Consider a combination of size *m=3* …



R_1		A	B	C	D	G
fb_1	t_1	0	0	0	0	0
	t_2	0	0	0	1	0
fb_2	t_3	0	0	[illegible]	0	0
fb_3	t_4	0	0	1	1	1
fb_4	t_5	0	1	1	0	1
	t_5	0	1	1	1	1
fb_5	t_7	1	1	1	1	1

R_2	A	B	E
fb_6	0	0	0
fb_7	0	0	1
fb_8	0	1	0
fb_9	0	1	1
fb_{10}	1	0	0
fb_{11}	1	0	1

R_5	C	F
fb_{20}	0	0
fb_{21}	0	1
fb_{22}	1	0

Intermediate Partitions

- First, identify the subscopes that affect R_1

R_1		A	B	C	D	G
fb_1	t_1	0	0	0	0	0
	t_2	0	0	0	1	0
fb_2	t_3	0	0	1	0	0
fb_3	t_4	0	0	1	1	1
fb_4	t_5	0	1	1	0	1
	t_5	0	1	1	1	1
fb_5	t_7	1	1	1	1	1

R_2	A	B	E
fb_6	0	0	0
fb_7	0	0	1
fb_8	0	1	0
fb_9	0	1	1
fb_{10}	1	0	0
fb_{11}	1	0	1

R_5	C	F
fb_{20}	0	0
fb_{21}	0	1
fb_{22}	1	0

Intermediate Partitions

- Project the union of those subscopes over the relation

R_1			A	B	C	D	G
ib_1	fb_1	t_1	0	0	0	0	0
		t_2	0	0	0	1	0
ib_2	fb_2	t_3	0	0	1	0	0
	fb_3	t_4	0	0	1	1	1
ib_3	fb_4	t_5	0	1	1	0	1
		t_5	0	1	1	1	1
ib_4	fb_5	t_7	1	1	1	1	1

R_2	A	B	E
fb_6	0	0	0
fb_7	0	0	1
fb_8	0	1	0
fb_9	0	1	1
fb_{10}	1	0	0
fb_{11}	1	0	1

R_5	C	F
fb_{20}	0	0
fb_{21}	0	1
fb_{22}	1	0