Improving Relational Consistency Algorithms Using Dynamic Relation Partitioning

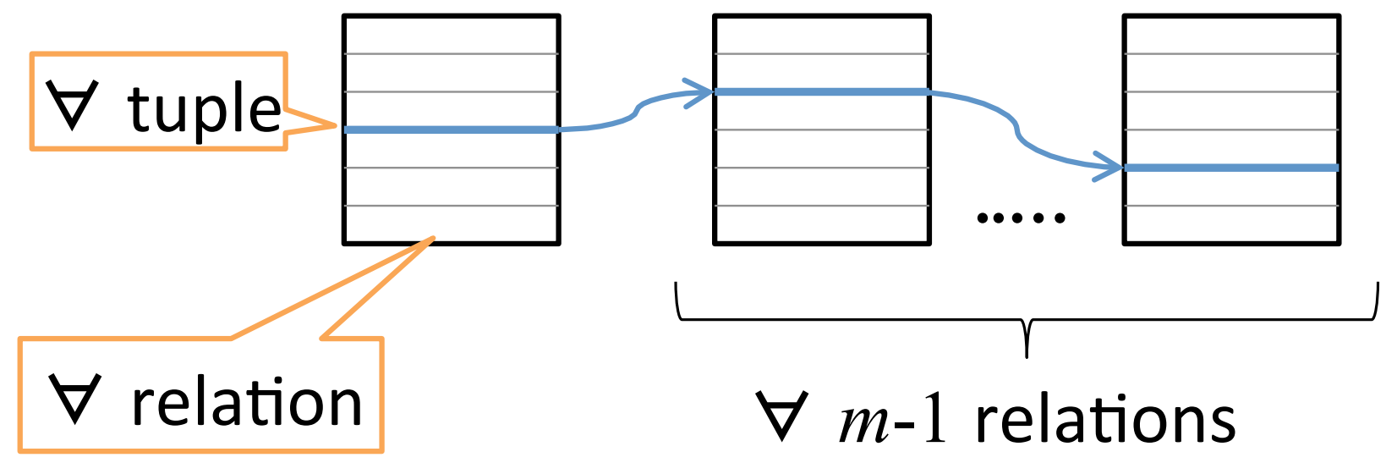
A.Schneider[1], R.J.Woodward[1,2], B.Y.Choueiry[1], and C.Bessiere[2]
[1]Constraint Systems Laboratory • University of Nebraska-Lincoln • USA
[2]LIRMM-CNRS • University of Montpellier • France

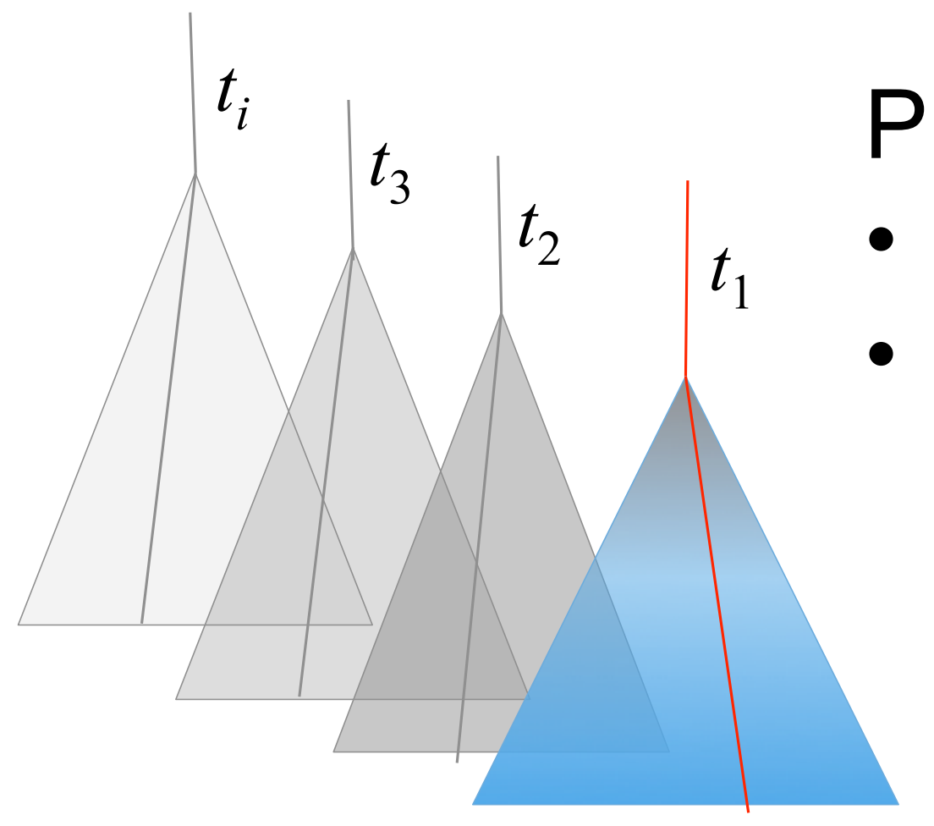
Contributions

1. Designed PERFB, an algorithm for enforcing R(*,*m*)C, exploiting the fact that constraints in dual CSP are piecewise functional.
2. Compared performance of PERFB and PERTUPLE (previous algorithm) to empirically establish improvements.

Relational Consistency

R(∗,*m*)C, *m*-wise consistency, ensures that every combination of *m* relations is minimal.





PERTUPLE enforces R(*,*m*)C [Karakshian+ AAAI10]

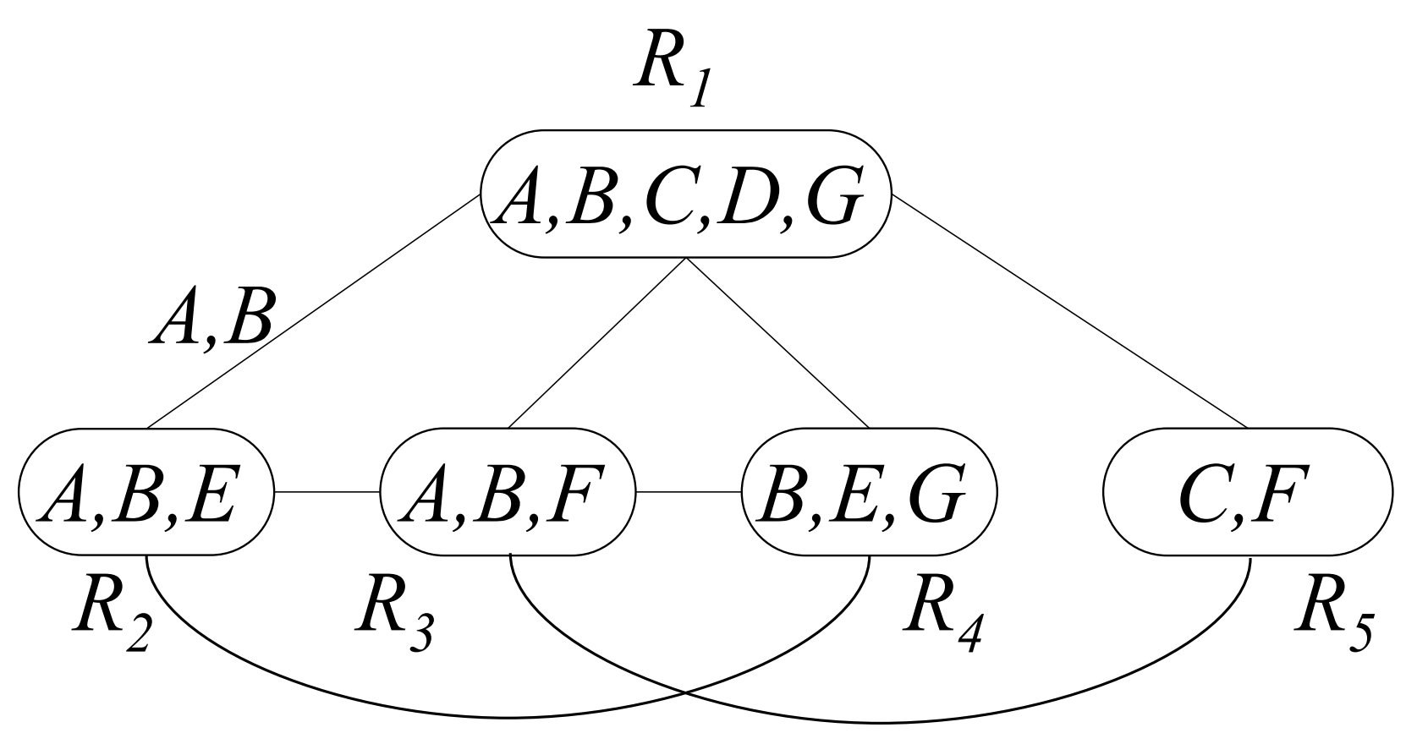
- Given all combinations of *m* relations
- For each relation in each combination
  - SEARCHSUPPORT (a backtrack search with FC) ensures each tuple can be extended to the other *m-1* relations
  - If no solution is found, tuple is removed

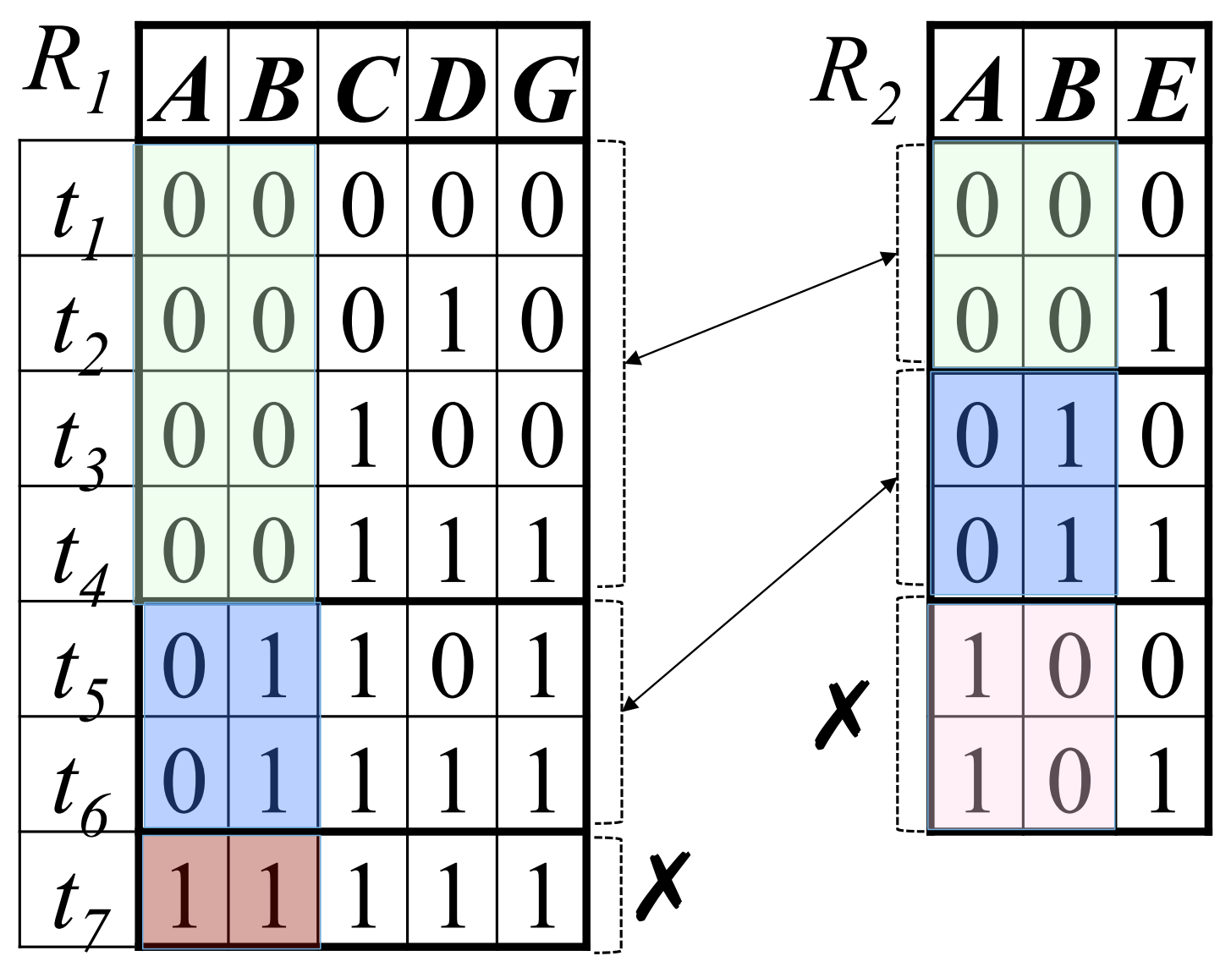
Piecewise Functional Constraints

Samaras & Stergiou [JAIR 05] noted that the constraints in the dual CSP are piecewise functional

1. Each relation can be partitioned into blocks of equivalent tuples
2. Each block is supported by at most one other block

They used above property to design PW-AC algorithm (*m*=2)





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

✗

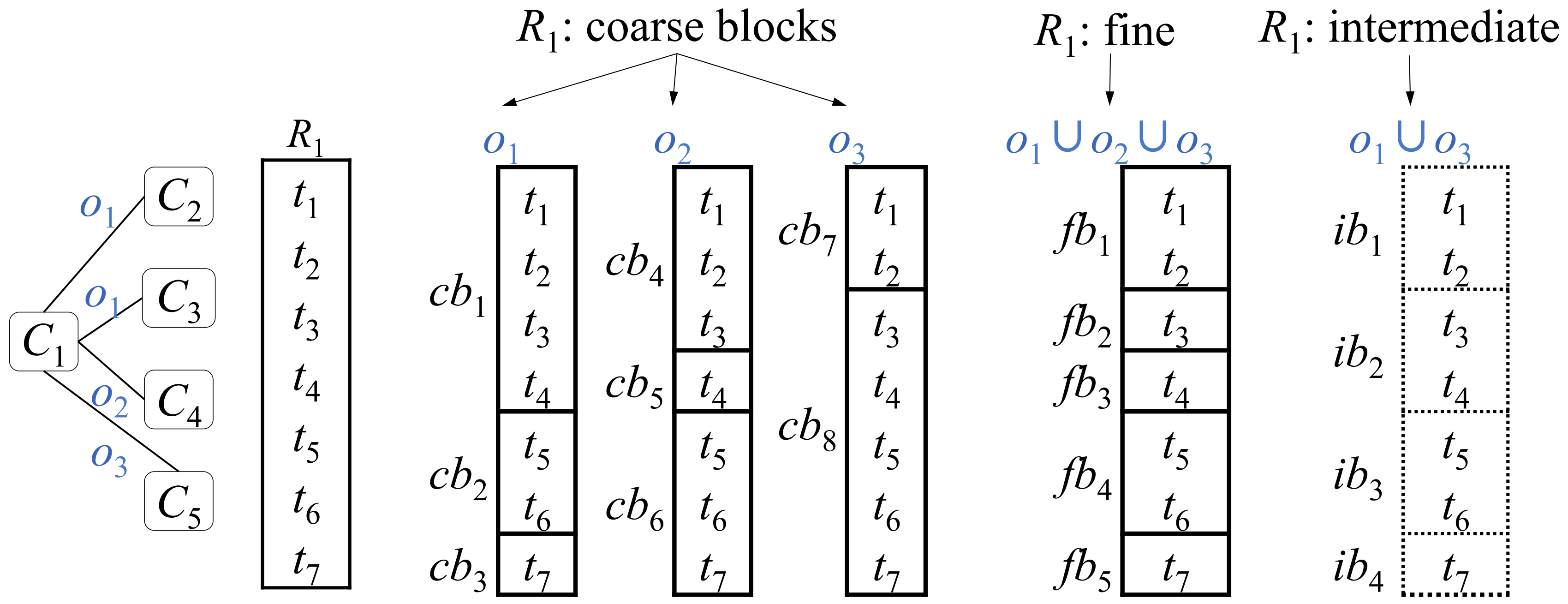
R_2	*A*	*B*	*E*
	0	0	0
	0	0	1
	0	1	0
	0	1	1
✗	1	0	0
	1	0	1

The "subscope equality constraint" {*A*,*B*} between R_1 and R_2 determines the partition of R_1.

- What partition do two subscopes (e.g., {*A*,*B*}, {*C*}) induce on R_1?
- What partition do all the subscopes with R_1's neighbors (i.e., R_2, R_3, R_4, and R_5) induce on R_1?
- How do those various partitions relate?
- How to exploit them in PERTUPLE?

Partitions: Coarse, Fine, Intermediate

The considered set of subscopes determines the partition of R_1.



We compute and store fine and coarse blocks at preprocessing.

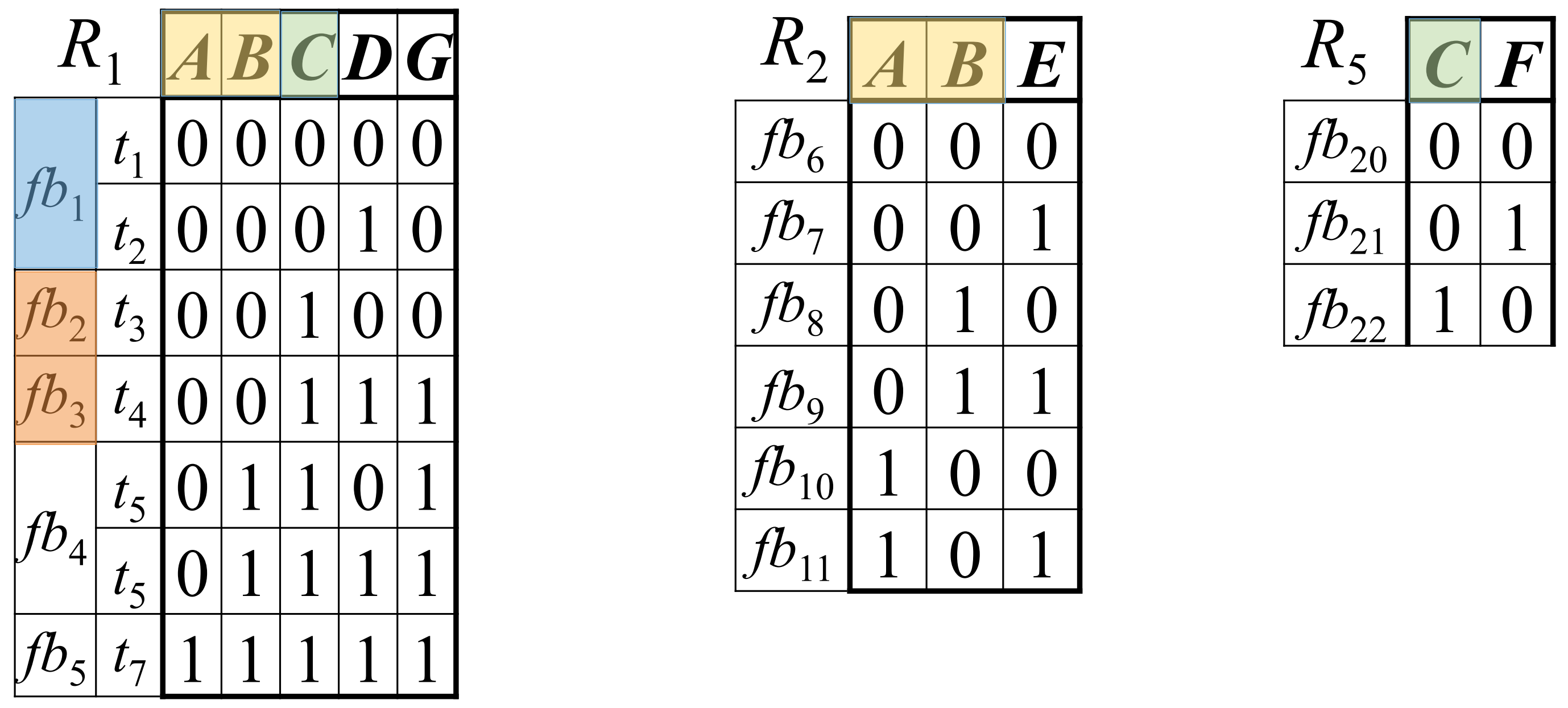
From PERTUPLE To PERFB

PERFB makes fewer calls to SEARCHSUPPORT than PERTUPLE

1. PERFB iterates over fine blocks rather than tuples
2. At each call, it dynamically determines the intermediate blocks induced on a relation by the considered other relations.

Considering relations R_1,$R_2$$R_5$

- The union of the subscopes of R_1,R_2 and R_1,R_5 determines the intermediate partition induced by $R_2$$R_5$ on R_1.
- Projecting a fine block over this union forms a *signature* of a fine block.
- Once SEARCHSUPPORT finds (or not) a support for a fine block, it reuses this result for future fine blocks with the same signature.



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

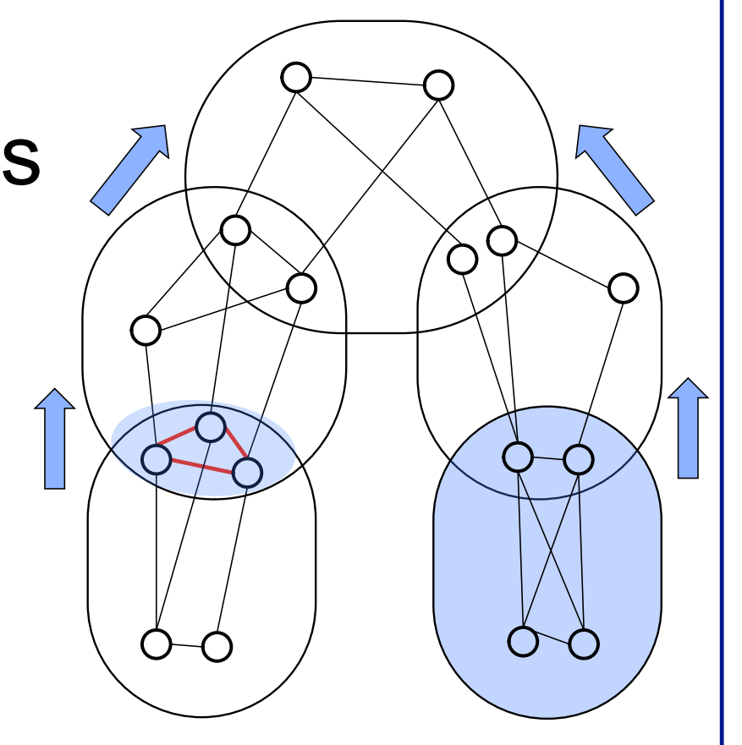
R_5	*C*	*F*
fb_{20}	0	0
fb_{21}	0	1
fb_{22}	1	0

- $\langle R_1, fb_1 \rangle$ has support $\langle R_2, fb_6 \rangle$, $\langle R_5, fb_{20} \rangle$.
- $\langle R_1, fb_2 \rangle$ has support $\langle R_2, fb_6 \rangle$, $\langle R_5, fb_{22} \rangle$.
- fb_2, fb_3 have the same signature (intermediate block $\{fb_2, fb_3\}$). SEARCHSUPPORT is not called on fb_3.

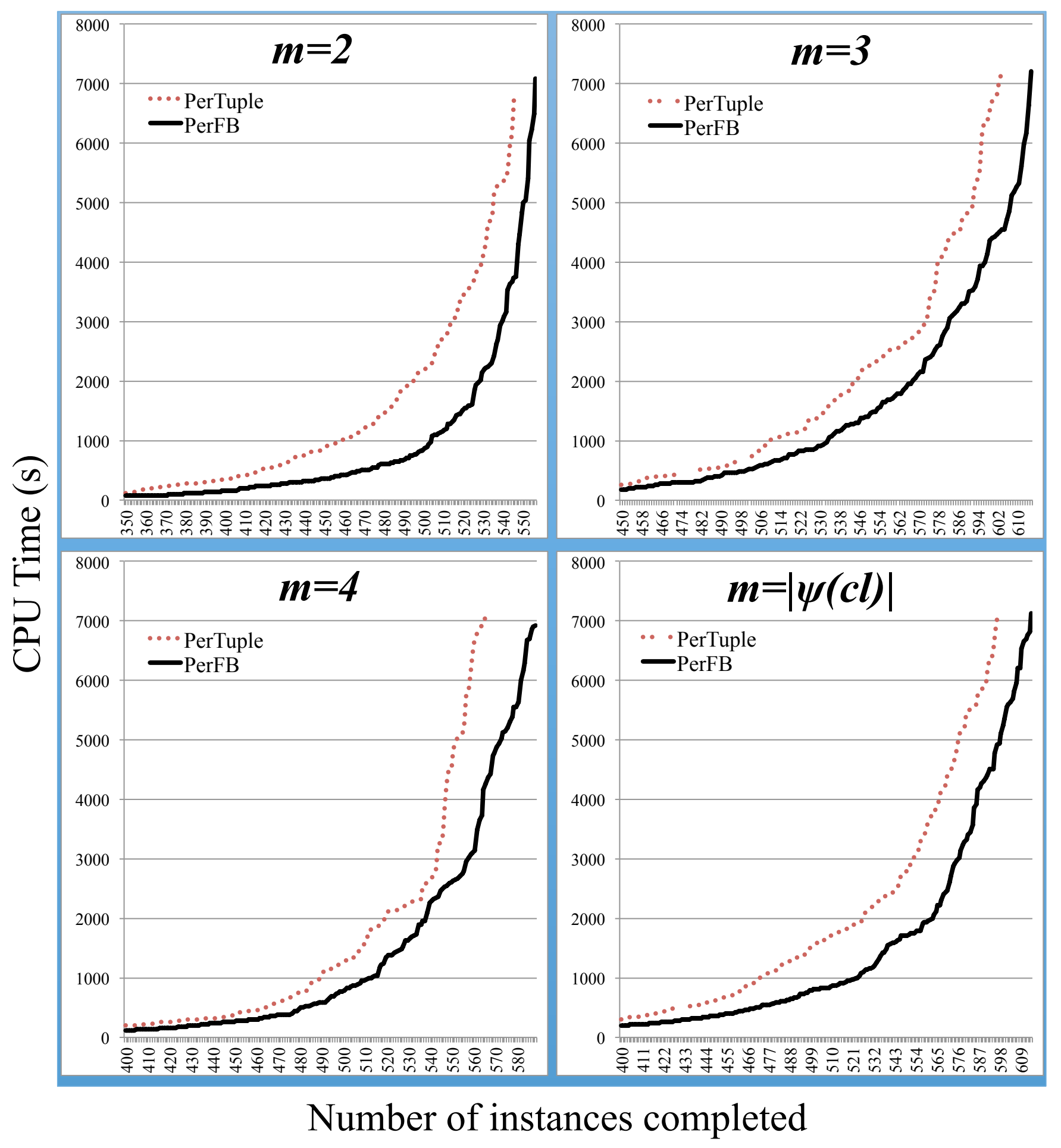
Empirical Evaluation

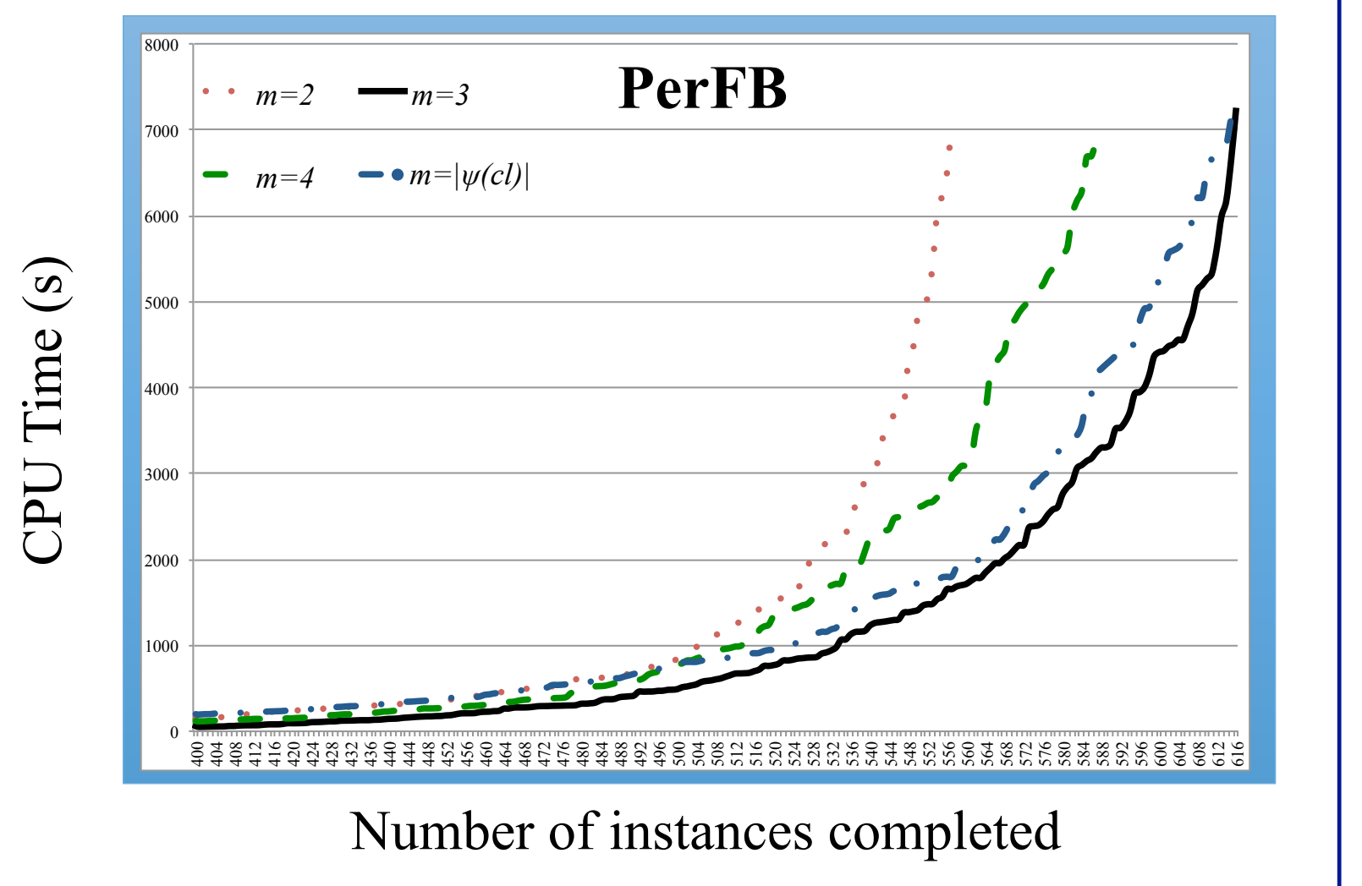
Experimental Setup

- 853 instances from the 2008 CP Solver Competition. [Lecoutre+]
- Real full lookahead cl+proj-wR(*,*m*)C, which enforces R(*, *m*)C on each cluster, adds projection of constraints to cluster separators to bolster propagation, and uses the minimal dual graph to reduce the number of combinations. [Karakashian+ AAAI 13]
- *m* = 2,3,4,|ψ(cl)| (i.e., minimal clusters)
- 2 hours and 8 GB per instance, 853 total instances.



Cumulative Charts





- For *m*=2,3,4, |ψ(cl)|, PERFB dominates PERTUPLE
- *m=3* performs the best, *m*=|ψ(cl)| trails closely behind

Summary Results

For all tested combination sizes,

- PERFB solves more instances than PERTUPLE, and,
- On instances solved by both algorithms, PERFB has a smaller average CPU time.
- Dynamic partitions reduce redundant calls to SEARCHSUPPORT.

	m = 2		*m* = 3		*m* = 4		*m* = |ψ(cl)|	
Total: 853 instances	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	86.4	0.0	88.1	26.1	52.7	19.6	24.7	8.1
ratio	--		3.37		2.69		3.06	

Future Research

Extend our approach to ALLSOL, our other algorithm for enforcing minimality of *m* relations [Karakashian PhD 13]

Experiments were conducted on the equipment of the Holland Computing Center at the University of Nebraska-Lincoln.
This research was supported by NSF Grant No. RI-111795 and EU project ICON (FP7-284715).
Woodward was supported by an NSF GRF Grant No. 1041000 and a Chateaubriand Fellowship.

Paper: CP 2014. September 4th, 2014