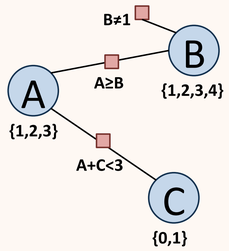
Characterizing Performance of Consistency Algorithms by Algorithm Configuration of Random CSP Generators

Daniel J. Geschwender, Robert J. Woodward, Berthe Y. Choueiry
Constraint Systems Laboratory • Department of Computer Science & Engineering • University of Nebraska-Lincoln

Constraint Satisfaction Problem:



- Used to model constrained combinatorial problems
- Important real-world applications: hardware & software verification, scheduling, resource allocation, etc.

A CSP is defined as follows:

Given

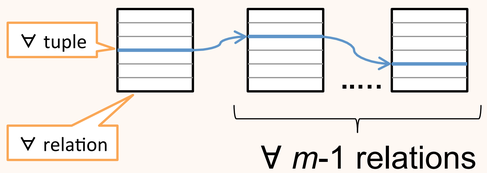
- A set of variables {A,B,C}
- Their domains D_A={1,2,3}, D_B={1,2,3,4}, D_C={0,1}
- A set of constraints: {A≥B,B≠2,A+C<3}

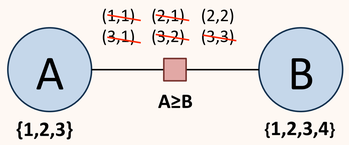
Question

- Determine consistency NP-complete
- Count number of solutions #P
- Find minimal network NP-complete
- Minimize number of broken constraints NP-hard

Minimal Network:

- Is a consistency property
- Guarantees that every tuple allowed by a constraint must participate in some solution to the CSP (i.e., the constraints are as minimal as possible)

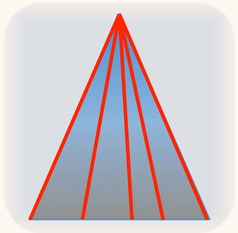




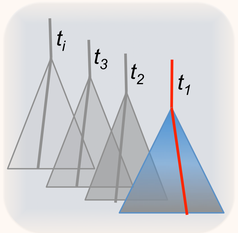
Two Algorithms for Enforcing Minimality:

[Karakashian, PhD 2013]

- AllSol: better when there are many 'almost' solutions
  - One search explores the entire search space
  - Finds all solutions without storing them, keeps tuples that appear in at least one solution
- PerTuple: better when many solutions are available
  - For each tuple, finds one solution where it appears
  - Many searches that stop after the first solution



AllSol



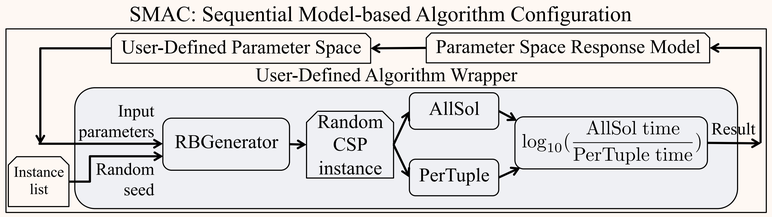
PerTuple

RBGenerator:

[Xu+ AIJ 2007]

- Generates hard satisfiable CSP instances at the phase transition

- k : arity of the constraints
- n : number of variables
- α : domain size $d=n^{\alpha}$
- r : # constraints $m=rn\ln(n)$
- δ : distance from phase transition, $p_{cr}+\delta/1000$
- ***forced*** : forced satisfiable?
- ***merged*** : merge similar scopes?



Sequential Model-based Algorithm Configuration:

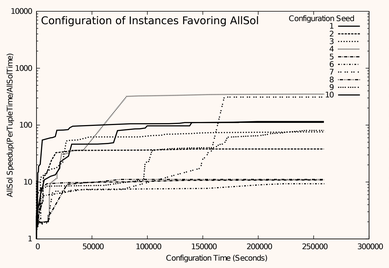
[Hutter+ LION 2011]

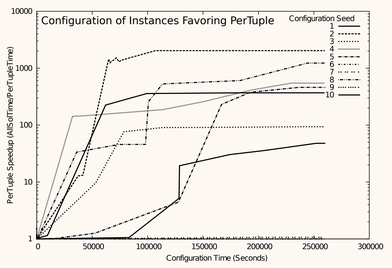
- SMAC tunes the parameter configuration of RBGenerator
- RBGenerator creates CSP to run on PerTuple and AllSol
- Compare runtimes and update SMAC response model
- Move toward parameters which favor one algorithm over the other

Experiments:

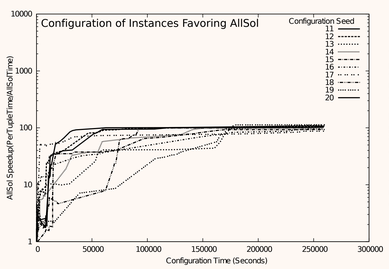
- 4 tests run, testing two factors:
  - Configuring to favor PerTuple and AllSol
  - With adjustable and fixed problem size parameters
- Each test run over 10 configuration seeds
- Configuration run for 4 days
- Algorithm time limit of 20 minutes

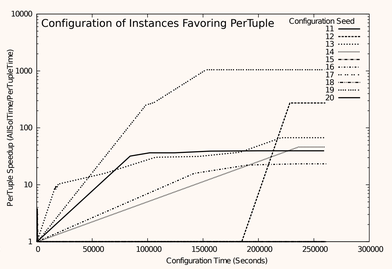
Adjustable Problem Size:





Fixed Problem Size:

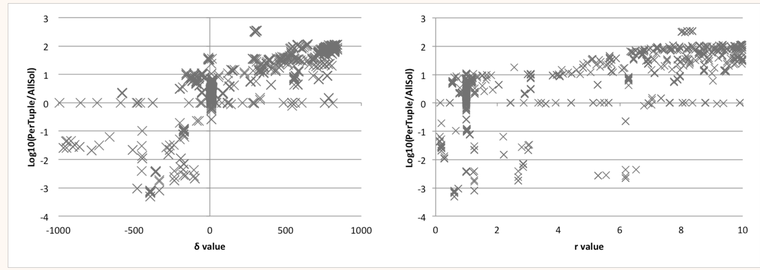


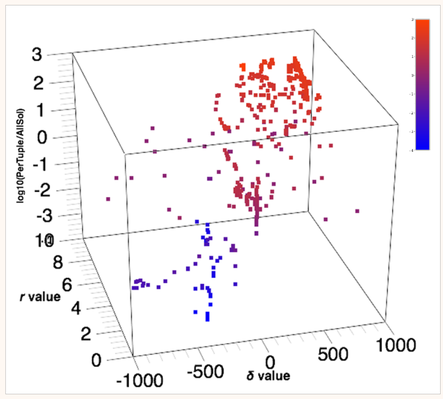


Final Parameter Configurations:

		seed	k	n	α	r	δ	*forced*	*merged*	speedup	CoV
Adjustable Size	AllSol	4	2	20	1.74	8.28	309	n	y	348	5%
	PerTuple	2	2	17	0.79	0.61	-394	y	n	4627	79%
Fixed Size	AllSol	19	4	16	1.00	9.94	840	y	y	110	4%
	PerTuple	19	2	16	1.00	0.74	-481	n	n	1301	27%

Effect of Parameters r and δ:





Conclusions:

- Configured PerTuple 1000x faster, AllSol 100x faster
- PerTuple configuration: fewer constraints, lower constraint tightness
- AllSol configuration: more constraints, higher constraint tightness
- Adjustable problem size only offers marginally better configuration

Future Work:

- Compare other consistency algorithms
- Use a CSP generator with more parameters
- Apply results found to algorithm selection

Supported by NSF Grant No. RI-111795. Geschwender was also supported by NSF GRF Grant No. 1041000 and a Barry M. Goldwater Scholarship, and Woodward by NSF GRF Grant No. 1041000 and a Chateaubriand Fellowship. Experiments conducted at UNL's Holland Computing Center.

Date January 29, 2015