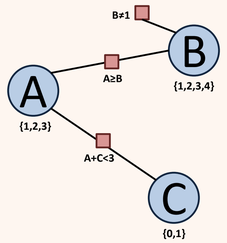
Configuring Random CSP Generators to Favor a Particular Consistency Algorithm

Daniel J. Geschwender, Robert J. Woodward, Berthe Y. Choueiry
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

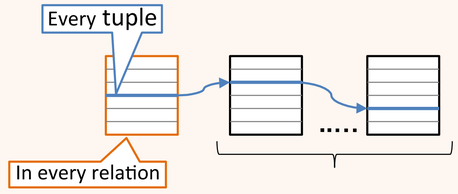
- A set of variables — {A,B,C}
- Their domains — D_A={1,2,3}, D_B={1,2,3,4}, D_C={0,1}
- A set of constraints: — {A≥B,B≠2,A+C<3}

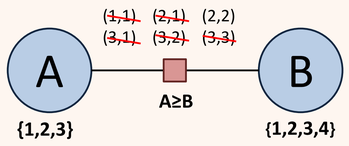
Question

- Find a solution — NP-complete
- Count number of solutions — #P
- Find minimal network — NP-complete
- Minimize number of broken constraints — NP-hard

Minimal Network:

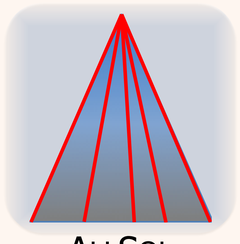
- Is a consistency property
- Guarantees that every tuple allowed by a constraint must participate in some solution to the CSP (i.e., the constraints are as minimal as possible)

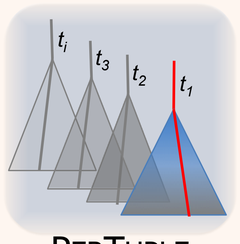




Two Algorithms for Enforcing Minimality:

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





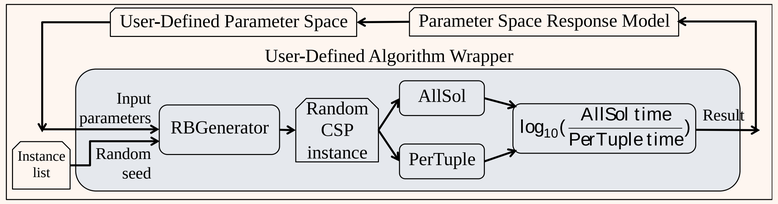
RBGenerator:

[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^{α}
- ***r*** : # constraints m=rn/*r*(n)
- ***δ*** : distance from phase transition, p_{cr}+ δ/1000
- ***forced*** : forced satisfiable?
- ***merged*** : merge similar scopes?

SMAC: Sequential Model-based Algorithm Configuration



Sequential Model-based Algorithm Configuration:

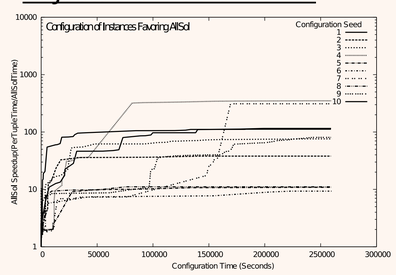
[Hutter+ LION-5]

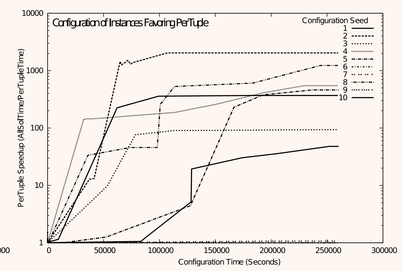
- SMAC tunes the parameter configuration of RBGenerator
- RBGenerator creates CSP to run on PerTuple and AllSol
- Compare runtimes and update SMAC response model
- Move toward parameters which favor one algorithm over the other

Experiments:

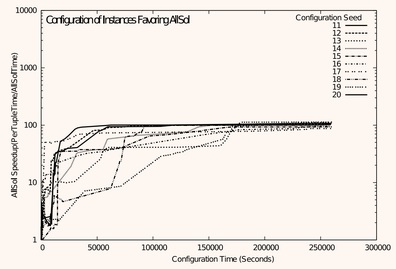
- 4 tests run, testing two factors:
  - Configuring to favor PerTuple and AllSol
  - With adjustable and fixed problem size parameters
- Each test run over 10 configuration seeds
- Configuration run for 4 days
- Algorithm time limit of 20 minutes

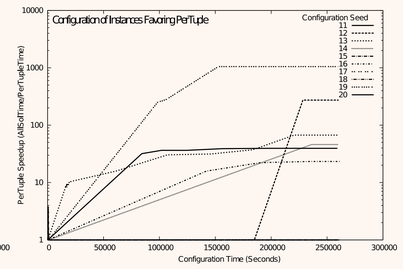
Adjustable Problem Size:





Fixed Problem Size:





Experiment Results:

		seed	k	n	α	r	δ	forced	merged	speedup	CoV
Adjustable Size	AllSol	1	4	19	0.75	9.91	578	n	y	101.36	14.55%
		2	7	18	0.20	5.88	-9	y	y	36.30	6.49%
		3	5	17	0.68	9.18	759	n	y	75.73	8.10%
		4	2	20	1.74	8.28	309	n	y	348.43	4.83%
		5	2	12	1.45	4.43	-54	n	n	11.21	32.69%
		6	3	14	0.78	1.52	-100	y	n	9.63	27.00%
		† 7	2	20	1.75	8.12	302	y	n	289.63	27.61%
		8	3	8	1.70	1.00	-155	n	n	13.72	77.29%
		9	5	19	0.72	7.70	825	n	n	80.70	3.56%
		10	4	16	0.86	9.95	646	n	y	115.01	17.31%
	PerTuple	1	2	9	1.28	5.29	-162	n	n	378.29	15.12%
		† 2	2	17	0.79	0.61	-394	y	n	4627.35	78.67%
		3	3	16	1.01	0.29	-190	n	n	296.94	102.49%
		4	2	14	0.77	2.69	-235	y	n	531.85	15.82%
		5	2	12	0.85	6.19	-98	y	y	442.20	12.32%
		‡ 6	4	13	9.80	8.05	542	n	n	1.00	0.00%
		7	2	19	0.36	3.59	291	n	y	0.97	17.70%
		8	2	14	0.87	1.25	-334	n	n	1510.80	40.12%
		∗ 9	4	11	1.11	7.61	-52	y	n	1.00	0.00%
		10	3	18	0.99	0.18	-780	y	y	63.75	73.30%
Fixed Size	AllSol	11	4	16	1.00	8.35	799	n	n	103.05	7.73%
		12	4	16	1.00	9.74	832	n	n	98.89	5.45%
		13	4	16	1.00	9.33	826	n	n	107.16	6.87%
		14	4	16	1.00	9.00	811	n	n	89.80	13.29%
		15	4	16	1.00	8.36	794	n	y	87.45	11.21%
		16	4	16	1.00	7.20	764	n	y	92.09	8.51%
		17	4	16	1.00	7.18	757	y	n	87.79	10.54%
		18	4	16	1.00	8.59	808	n	n	102.17	6.20%
		19	4	16	1.00	9.94	840	y	y	109.75	4.05%
		20	4	16	1.00	7.73	786	n	y	100.21	7.66%
	PerTuple	∗ 11	4	16	1.00	5.13	-62	y	n	1.00	0.00%
		12	2	16	1.00	1.22	-361	y	y	311.41	35.37%
		13	2	16	1.00	2.21	-276	y	y	69.74	28.92%
		14	2	16	1.00	3.00	-265	n	n	47.06	15.05%
		‡ 15	5	16	1.00	0.46	757	y	n	1.00	0.00%
		† 16	3	16	1.00	0.14	-950	y	n	40.36	102.40%
		‡ 17	7	16	1.00	9.91	-377	n	n	1.00	0.00%
		‡ 18	8	16	1.00	4.41	292	n	n	1.00	0.00%
		19	2	16	1.00	0.74	-481	n	n	1300.63	27.01%
		† 20	3	16	1.00	0.18	-967	n	n	69.71	116.22%

*: all instances timeout, †: one or two instances crash, ‡: all instances crash

Conclusion:

- Configured PerTuple 1000x faster, AllSol 100x faster
- PerTuple configuration: less 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 more parameterized CSP generator
- Apply results found to algorithm selection

Supported by NSF Grant No. RI-111795, Barry M. Goldwater scholarship, and Undergraduate Creative Activities and Research Experiences Program (UCARE) of the University of Nebraska-Lincoln. Experiments conducted at UNL's Holland Computing Center.

Date April 13, 2014