Adaptive Parameterized Consistency for Non-Binary CSPs by Counting Supports

R.J. Woodward[1,2], A. Schneider[1], B.Y. Choueiry[1], C. Bessiere[2]

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

[2]LIRMM-CNRS, University of Montpellier

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)

Motivation

- Solvers maintain local consistency
- Standard level of consistency is AC
- Sometimes, AC is not enough
- Others, higher level may be too expensive
- Issue
  - Adapt the consistency level dynamically
  - On every variable-value pair + constraint

Outline

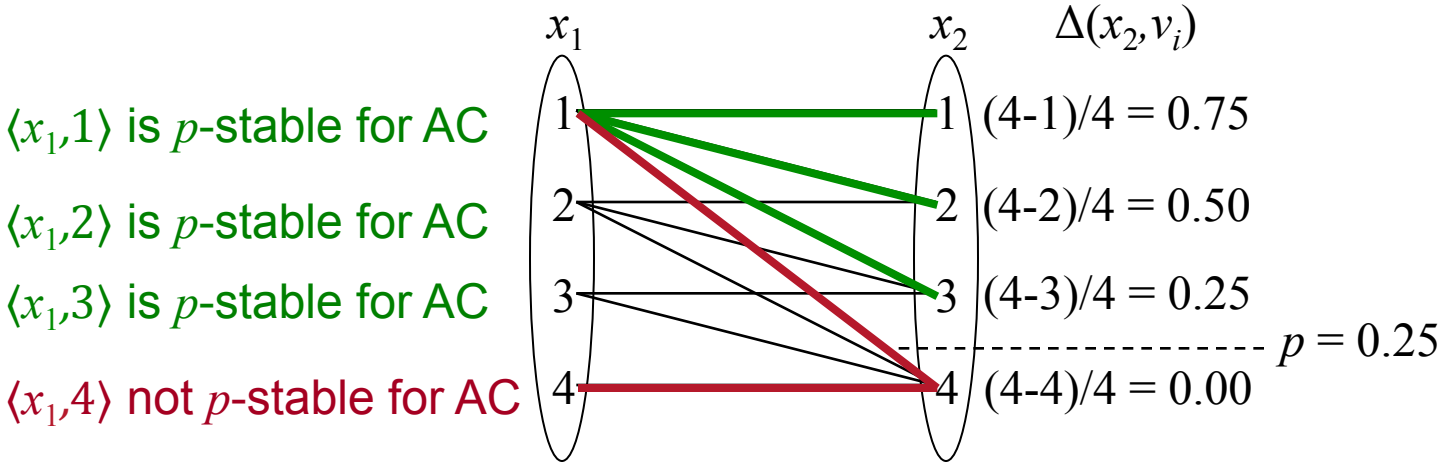
- Background
  - p-stability for AC [Balafrej+ CP2013]
  - Parameterized consistency [Balafrej+ CP2013]
  - Local consistency properties
- p-stability for GAC
- Empirical evaluations
- Conclusions

p-stable for AC

[Balafrej+ CP2013]

- For x_2, relative position of v_i in dom(x_2)

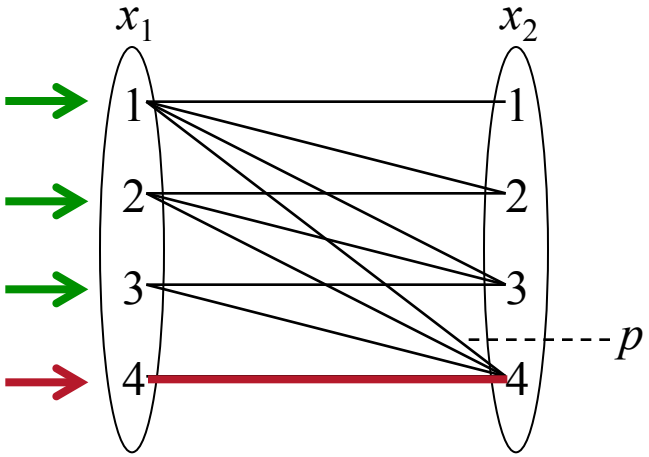
$$\Delta(x_2, v_i) = \frac{|dom^o(x_2)| - rank(v_i, dom^o(x_2))}{|dom^o(x_2)|}$$



- $\langle x_1, v_i\rangle$ p-stable for AC on x_2 iff it has an AC-support with $\Delta(x_2, v_j) \geq p$

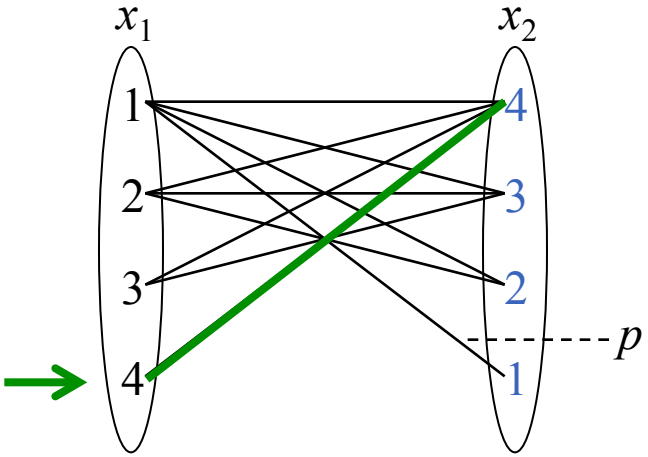
Parameterized Consistency [Balafrej+ CP2013]

- LC: Some local consistency stronger than AC
- pc-LC: $\forall c_{ij}$ on x_i, x_j
  - $\langle x_j, v_j \rangle$ is p-stable for AC on c_{ij} or
  - $\langle x_j, v_j \rangle$ is LC on c_{ij}
- apc-LC: adaptive
  - Use weight of constraint, like dom/wdeg
  - Low weight $\Rightarrow$ Low $p(c_{ij}) \Rightarrow$ Low consistency
  - High weight $\Rightarrow$ High $p(c_{ij}) \Rightarrow$ High consistency
- Issue: position is not a precise measure of support



Parameterized Consistency [Balafrej+ CP2013]

- LC: Some local consistency stronger than AC
- pc-LC: $\forall c_{ij}$ on x_i, x_j
  - $\langle x_j, v_j \rangle$ is p-stable for AC on c_{ij} or
  - $\langle x_j, v_j \rangle$ is LC on c_{ij}
- apc-LC: adaptive
  - Use weight of constraint, like dom/wdeg
  - Low weight $\Rightarrow$ Low $p(c_{ij}) \Rightarrow$ Low consistency
  - High weight $\Rightarrow$ High $p(c_{ij}) \Rightarrow$ High consistency
- Issue: position is not a precise measure of support



Non-Binary CSPs: Local Consistency

- Generalized Arc Consistent: STR
  - GAC (domain filtering)
  - Removes unsupported tuples (relation filtering)
- m-wise consistency
  - R(*,m)C
  - $m = 2 \equiv$ pair-wise consistent
- Algorithms maintain count of supports alive

Outline

- Background
  - p-stability for AC [Balafrej+ CP2013]
  - Parameterized Consistency [Balafrej+ CP2013]
  - Local consistency properties
- p-stability for GAC
- Empirical evaluations
- Conclusions

p-stability for GAC

- Recall p-stable for AC
  - $\Delta(x_i, v_i) = \dfrac{|dom^o(x_i)| - rank(v_i, dom^o(x_i))}{|dom^o(x_i)|} \geq p$

- p-stable for GAC:
  - $\forall c_j$ on x_i $\dfrac{|\sigma_{x_i=v_i}(c_j)|}{|c_j^o|} \geq p$
  - $\sigma_{xi=vi}(c_j)$ selection operator
    - Stored in $gacSupports[c_j](\langle x_i,v_i\rangle)$

$gacSupports[\sigma_{x_1=1}(c_j)]=\{0,1,2,3\}$
$gacSupports[c_j](\langle x_1,2\rangle)=\{4,5,6\}$
$gacSupports[c_j](\langle x_1,3\rangle)=\{7,8\}$
$gacSupports[c_j](\langle x_1,4\rangle)=\{9\}$

	x_1	x_2
0	1	1
1	1	2
2	1	3
3	1	4
4	2	2
5	2	3
6	2	4
7	3	3
8	3	4
9	4	4

$\langle x_1,1\rangle$ is 0.25-stable for GAC (4/10 ≥ 0.25)

$\langle x_1,2\rangle$ is 0.25-stable for GAC (3/10 ≥ 0.25)

$\langle x_1,3\rangle$ not 0.25-stable for GAC (2/10 < 0.25)

$\langle x_1,4\rangle$ not 0.25-stable for GAC (1/10 < 0.25)

Algorithm for *apc*-LC

Input: A constraint R_j

Output: Set of variables whose domain have been modified

$X_{modified} \leftarrow \emptyset$

$\forall \langle x_i, v_i \rangle \mid x_i \in scope(R_j), v_i \in dom(x_i)$

  if $|gacSupports[R_j][\langle x_i, v_i \rangle]| \neq 0$ **and** $\frac{|gacSupports[R_j][\langle x_i, v_i \rangle]|}{|R_j^o|} \geq p(R_j)$

    APPLY-LC(R_j, $gacSupports[R_j][\langle x_i, v_i \rangle]$)

  if $|gacSupports[R_j][\langle x_i, v_i \rangle]| = 0$

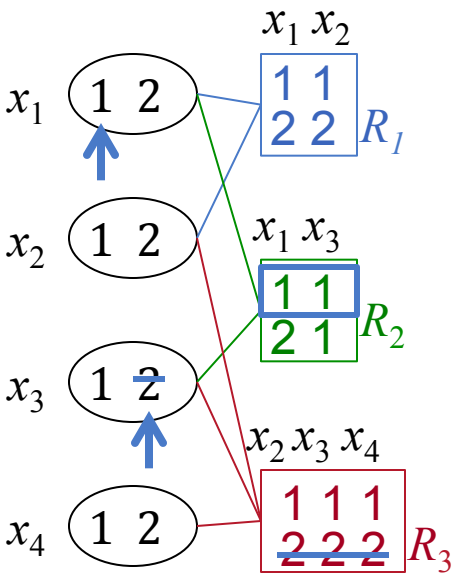
    $dom(x_i) \leftarrow dom(x_i) \setminus \{v_i\}$

    PROPAGATE-STR

    if $dom(x_i) = \emptyset$ **then** throw INCONSISTENCY

    $X_{modified} \leftarrow X_{modified} \cup \{x_i\}$

return $X_{modified}$



$gacSupports[R_2](\langle x_3, 2 \rangle) = \{\}$

$gacSupports[R_2](\langle x_1, 2 \rangle) = \{1\}$

Empirical Evaluations (I)

- BT, first solution, dom/wdeg
- Maintain: STR, R(*,2)C, *apc*-R(*,2)C
- Number of instances: 623

	✔ STR	✗ R(*,2)C	✗ *apc*-R(*,2)C
Number of instances solved by	504	550	**552**
# instances solved only by	10	5	0
# instances solved by STR, but missed by	0	18	11
# instances solved by R(*,2)C, but missed by	64	0	6
# instances solved by *apc*-R(*,2)C, but missed by	59	8	0
Average CPU time (sec.) over 458 instances	328.41	378.12	**313.31**
Median CPU time (sec.) over 458 instances	7.23	17.35	**7.21**

Empirical Evaluations (I)

- BT, first solution, dom/wdeg
- Maintain: STR, R(*,2)C, *apc*-R(*,2)C
- Number of instances: 623

	✗ STR	✔ R(*,2)C	✗ *apc*-R(*,2)C
Number of instances solved by	504	550	**552**
# instances solved only by	10	5	0
# instances solved by STR, but missed by	0	18	11
# instances solved by R(*,2)C, but missed by	64	0	6
# instances solved by *apc*-R(*,2)C, but missed by	59	8	0
Average CPU time (sec.) over 458 instances	328.41	378.12	**313.31**
Median CPU time (sec.) over 458 instances	7.23	17.35	**7.21**

Empirical Evaluations (I)

- BT, first solution, dom/wdeg
- Maintain: STR, R(*,2)C, *apc*-R(*,2)C
- Number of instances: 623

	✗ STR	✗ R(*,2)C	✔ *apc*-R(*,2)C
Number of instances solved by	504	550	**552**
# instances solved only by	10	5	0
# instances solved by STR, but missed by	0	18	11
# instances solved by R(*,2)C, but missed by	64	0	6
# instances solved by *apc*-R(*,2)C, but missed by	59	8	0
Average CPU time (sec.) over 458 instances	328.41	378.12	**313.31**
Median CPU time (sec.) over 458 instances	7.23	17.35	**7.21**

Empirical Evaluations (II)

Average CPU Time (sec.)					
benchmark	# Instances	All Comp.	STR	R(*,2)C	*apc*-R(*,2)C
a) *apc*-R(*,2)C is the best					
aim-50	24	24	**0.04**	0.07	**0.04**
allIntervalSeries	25	22	7.09	141.85	**6.00**
jnhSat	16	16	13.07	357.66	**11.74**
modifiedRenault	50	50	6.39	11.17	**6.29**
rand-3-20-20	50	31	1,666.10	939.88	**932.77**
c) *apc*-R(*,2)C is the worst					
dubois	13	6	1,000.54	**451.91**	1,456.01
TSP-20	15	15	**101.20**	318.37	335.13
d) Not solved by STR					
dag-rand	25	0	-	**123.70**	149.64

benchmark	# Instances	All Comp.	STR	R(*,2)C	*apc*-R(*,2)C
b) *apc*-R(*,2)C is competitive					
aim-100	24	24	0.38	**0.26**	0.41
aim-200	24	22	414.48	**6.52**	286.27
jnhUnsat	34	34	**13.61**	294.77	13.95
lexVg	63	63	**69.81**	341.87	338.74
pret	8	4	**117.89**	347.03	136.04
rand-3-20-20-fcd	50	39	928.06	**546.84**	615.23
rand-8-20-5	20	9	2,564.94	**355.57**	372.76
rand-10-20-10	20	12	6.72	**1.67**	2.76
ssa	8	5	**64.60**	100.64	69.59
TSP-25	15	10	**232.38**	1,072.72	743.33
ukVg	65	31	**166.82**	796.90	421.35
varDimacs	9	6	**89.23**	587.55	319.20
wordsVg	65	58	**119.76**	532.05	400.22

Empirical Evaluations (III)

Number of calls to STR and R(*,2)C by *apc*-R(*,2)C

benchmark	STR Calls	R(*,2)C Calls
allIntervalSeries	38,281,694	0
modifiedRenault	4,618,778	601,641
rand-3-20-20	489,441,126	3,480,216,943

Conclusions

- Contributions
  - Extended p-stability for AC to GAC
  - A more precise mechanism for computing it
  - Algorithm for enforcing apc-LC
- Future work
  - Investigate an adaptive criterion that operates also during pre-processing
  - Extend to more than 2 consistency levels

Thank You!

Enjoy Lunch ☺

Empirical Evaluations (III)

benchmark	# Instances	# Completions STR	R(*,2)C	*apc*-R(*,2)C
a) *apc*-R(*,2)C is the best				
aim-50	24	24	24	24
allIntervalSeries	25	22	22	22
jnhSat	16	16	16	16
modifiedRenault	50	50	50	50
rand-3-20-20	50	31	43	41
c) *apc*-R(*,2)C is the worst				
dubois	13	7	8	6
TSP-20	15	15	15	15
d) Not solved by STR				
dag-rand	25	0	25	25

benchmark	# Instances	STR	R(*,2)C	*apc*-R(*,2)C
b) *apc*-R(*,2)C is competitive				
aim-100	24	24	24	24
aim-200	24	22	24	24
jnhUnsat	34	34	34	34
lexVg	63	63	63	63
pret	8	4	4	4
rand-3-20-20-fcd	50	39	48	47
rand-8-20-5	20	9	20	20
rand-10-20-10	20	12	12	12
ssa	8	6	5	6
TSP-25	15	13	10	13
ukVg	65	37	31	34
varDimacs	9	6	6	6
wordsVg	65	65	58	58

Emp

benchmark	# Instances	Completed				Average CPU Time (sec)		
		STR	R(*,2)C	*apc*-R(*,2)C	All	STR	R(*,2)C	*apc*-R(*,2)C
a) *apc*-R(*,2)C is the best								
aim-50	24	24	24	24	24	**0.04**	0.07	**0.04**
allIntervalSeries	25	22	22	22	22	7.09	141.85	**6.00**
jnhSat	16	16	16	16	16	13.07	357.66	**11.74**
modifiedRenault	50	50	50	50	50	6.39	11.17	**6.29**
rand-3-20-20	50	31	43	41	31	1,666.10	939.88	**932.77**
b) *apc*-R(*,2)C is competitive								
aim-100	24	24	24	24	24	0.38	**0.26**	0.41
aim-200	24	22	24	24	22	414.48	**6.52**	286.27
jnhUnsat	34	34	34	34	34	**13.61**	294.77	13.95
lexVg	63	63	63	63	63	**69.81**	341.87	338.74
pret	8	4	4	4	4	**117.89**	347.03	136.04
rand-3-20-20-fcd	50	39	48	47	39	928.06	**546.84**	615.23
rand-8-20-5	20	9	20	20	9	2,564.94	**355.57**	372.76
rand-10-20-10	20	12	12	12	12	6.72	**1.67**	2.76
ssa	8	6	5	6	5	**64.60**	100.64	69.59
TSP-25	15	13	10	13	10	**232.38**	1,072.72	743.33
ukVg	65	37	31	34	31	**166.82**	796.90	421.35
varDimacs	9	6	6	6	6	**89.23**	587.55	319.20
wordsVg	65	65	58	58	58	**119.76**	532.05	400.22
c) *apc*-R(*,2)C is the worst								
dubois	13	7	8	6	6	1,000.54	**451.91**	1,456.01
TSP-20	15	15	15	15	15	**101.20**	318.37	335.13
d) Not solved by STR								
dag-rand	25	0	25	25	0	-	**123.70**	149.64

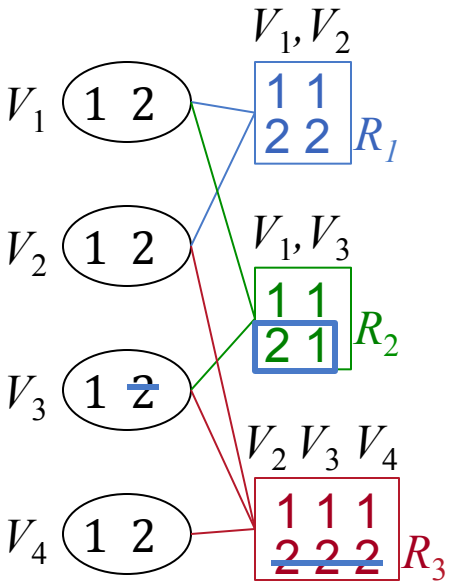
benchmark	# Instances	Completed				Average CPU Time (sec)			Median CPU Time (sec)		
		STR	R(*,2)C	*apc*-R(*,2)C	All	STR	R(*,2)C	*apc*-R(*,2)C	STR	R(*,2)C	*apc*-R(*,2)C
a) *apc*-R(*,2)C is the best											
aim-50	24	24	24	24	24	**0.04**	0.07	**0.04**	0.02	0.04	**0.03**
allIntervalSeries	25	22	22	22	22	7.09	141.85	**6.00**	0.13	0.31	0.12
jnhSat	16	16	16	16	16	13.07	357.66	**11.74**	8.15	142.24	7.21
modifiedRenault	50	50	50	50	50	6.39	11.17	**6.29**	7.24	8.79	6.98
rand-3-20-20	50	31	43	41	31	1,666.10	939.88	**932.77**	1,211.50	822.54	811.74
b) *apc*-R(*,2)C is competitive											
aim-100	24	24	24	24	24	0.38	**0.26**	0.41	0.18	0.25	**0.16**
aim-200	24	22	24	24	22	414.48	**6.52**	286.27	2.39	1.37	**2.60**
jnhUnsat	34	34	34	34	34	**13.61**	294.77	13.95	10.74	153.50	**9.78**
lexVg	63	63	63	63	63	**69.81**	341.87	338.74	0.50	1.38	0.89
pret	8	4	4	4	4	**117.89**	347.03	136.04	115.81	354.82	145.70
rand-3-20-20-fcd	50	39	48	47	39	928.06	**546.84**	615.23	501.30	422.24	464.00
rand-8-20-5	20	9	20	20	9	2,564.94	**355.57**	372.76	1,987.35	314.26	**261.68**
rand-10-20-10	20	12	12	12	12	6.72	**1.67**	2.76	6.40	1.66	2.75
ssa	8	6	5	6	5	**64.60**	100.64	69.59	1.51	1.60	1.58
TSP-25	15	13	10	13	10	**232.38**	1,072.72	743.33	69.00	211.41	131.69
ukVg	65	37	31	34	31	**166.82**	796.90	421.35	36.29	54.65	**30.39**
varDimacs	9	6	6	6	6	**89.23**	587.55	319.20	1.56	6.43	2.94
wordsVg	65	65	58	58	58	**119.76**	532.05	400.22	0.39	0.95	0.59
c) *apc*-R(*,2)C is the worst											
dubois	13	7	8	6	6	1,000.54	**451.91**	1,456.01	552.13	255.25	779.57
TSP-20	15	15	15	15	15	**101.20**	318.37	335.13	23.32	61.55	**46.34**
d) Not solved by STR											
dag-rand	25	0	25	25	0	-	**123.70**	149.64	-	124.47	151.33

Algorithm for *apc*-LC

Algorithm 1: LIVING-STR(c_i): set of variables

Input: c_j: a constraint of $\mathcal{P}$
Output: Set of variables in $scope(c_j)$ whose domains have been modified

```
1  X_modified ← ∅
2  foreach x_i ∈ scope(c_j) | x_i ∉ past(P) do
3      foreach v_i ∈ dom(x_i) do
4          if |gacSupports[R_j](⟨x_i,v_i⟩)| ≠ 0 and |gacSupports[R_j](⟨x_i,v_i⟩)| / |R_j| ≱ p(c_j)
           then
5              APPLY-LC(R_j, gacSupports[R_j](⟨x_i,v_i⟩))
6          if |gacSupports[R_j](⟨x_i,v_i⟩)| = 0 then
7              foreach c_k ∈ cons(x_i) do
8                  delTuples(c_k, gacSupports[R_k](⟨x_i,v_i⟩), |past(P)|)
9              dom(x_i) ← dom(x_i) \ {v_i}
10             if dom(x_i) = ∅ then throw INCONSISTENCY
11             X_modified ← X_modified ∪ {x_i}
12 return X_modified
```



$gacSupports[R_2](\langle V_3,2\rangle)=\{\}$

$gacSupports[R_2](\langle V_1,2\rangle)=\{2\}$