PROPAGATING SOFT TABLE CONSTRAINTS

CHRISTOPHE LECOUTRE AND NICOLAS PARIS AND OLIVIER ROUSSEL AND SÉBASTIEN TABARY

{*lecoutre,paris,roussel,tabary*}@cril.univ-artois.fr

Introduction

Context :

- WCSP framework
- Soft table constraints (extensional) of large arity with a default cost equal to 0 or k
- Filtering approach based on the concept of Equivalence-Preserving Transformations

Goal :

- Exploit the efficiency of cost transfer operations to solve constraints of large arity

Principle :

- Algorithm to enforce a weak version of GAC on soft table constraints
- Combine both the techniques of Simple Tabular Reduction and cost transfer

Background

A soft constraint c_S involves an ordered set S of variables and is defined as a cost function from $l(S)$ to $\{0, \ldots, k\}$ where $l(S)$ is the set of possible instantiations of S.

The existence of a nullary constraint $c_\emptyset$ (a constant) as well as the presence of a unary constraint c_x for every variable x is assumed.

DEFINITION 1: The *extended* cost of an instantiation $I \in l(S)$ on a soft constraint c_S is defined by $ecost(c_S, I) = c_\emptyset + \sum_{x \in S} c_x(I[x]) + c_S(I)$.

DEFINITION 2: A value (x, a) of a soft constraint c_S is GAC^w-consistent on c_S iff $\exists I \in l(S) \mid I[x] = a \wedge c_S(I) = 0 \wedge ecost(c_S, I) < k$. A soft constraint c_S is GAC^w-consistent iff every value of c_S is GAC^w-consistent.

Algorithm

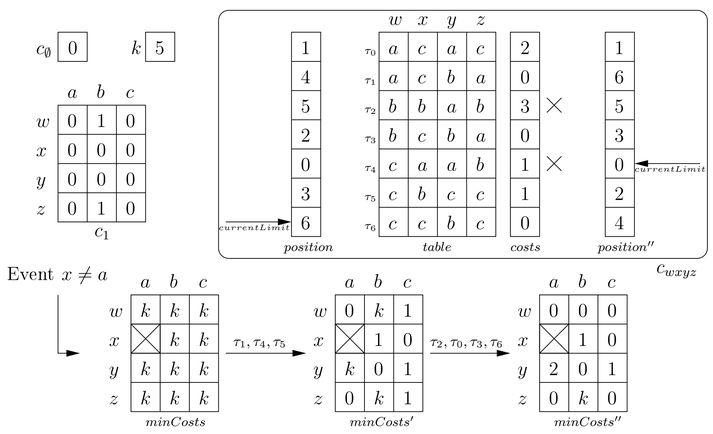
GAC^w-WSTR(c_S: soft constraint)

```
1:  cnt ← 1
2:  if default[c_S] = k then
3:      traverse-k(c_S, cnt)
4:  else
5:      traverse-0(c_S, cnt)        \\default[c_S] = 0
6:  pruneVars(c_S)
7:  while S^sup ≠ ∅ do
8:      pick and delete x from S^sup
9:      α ← +∞
10:     for each a ∈ dom(x)
11:         if minCosts[c_S][x][a] > 0 then
12:             project(c_S, x, a, minCosts[c_S][x][a])
13:         α ←min(α, c_x(a))
14:     if α > 0 then
15:         unaryProject(x, α)
16:     if S^sup ≠ ∅ then
17:         cnt + +
18:         if default[c_S] = k then
19:             traverse-k(c_S, cnt)
20:         else
21:             traverse-0(c_S, cnt)        \\default[c_S] = 0
22: end while
```

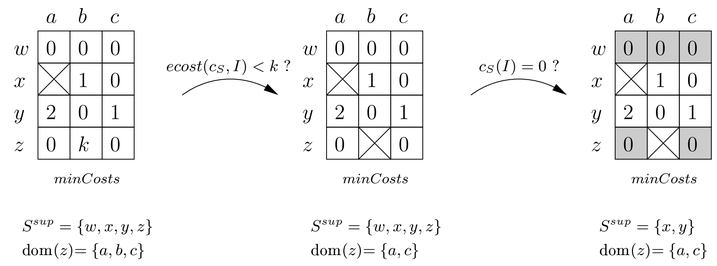
Space complexity: $O(tr + rd)$. Time complexity: $O(r^2(d + t))$ when default cost is k, $O(r^3 dt \log(t))$ when default cost is 0.

Evolution of data structures

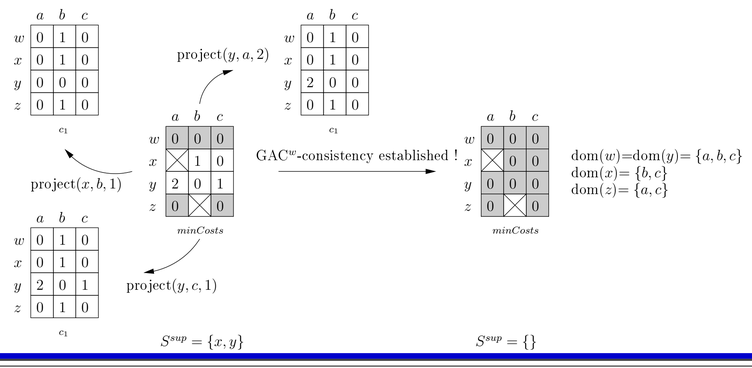
Step 1 : Compute minimum costs for values



Step 2 : Prune GAC^w-inconsistent values



Step 3 : Search again minimum costs for values after making successive projections



Experiments

Series	*#Inst*	PFC-MRDAC- WSTR	PFC-MRDAC- GEN	Maintaining- GAC^w-WSTR	Maintaining- AC*	Maintaining- FDAC
crossoft-herald	50	33	10	**47**	11	11
crossoft-puzzle	22	**22**	9	**22**	18	18
crossoft-vg	64	**14**	6	**14**	7	7
poker	18	**10**	2	**10**	5	5
rand-3 (rb)	48	20	29	20	**32**	30
rand-10	20	**20**	0	**20**	0	0
ergo	19	13	10	15	15	**17**
linkage	30	0	0	0	1	**9**
renault-mod	50	**50**	32	**50**	**50**	47

Number of solved instances per series (time-out of 1,200 seconds set per instance)

Instance (arity max)	PFC-MRDAC- WSTR	PFC-MRDAC- GEN	Maintaining- GAC^w-WSTR	Maintaining- AC*	Maintaining- FDAC
crossoft-ogd-15-09 (7)	26.5	$\perp$	**25.2**	273	269
crossoft-ogd-23-01 (23)	$\perp$	$\perp$	**565**	$\perp$	$\perp$
crossoft-ogd-puzzle-18 (10)	**6.29**	$\perp$	6.66	$\perp$	$\perp$
crossoft-ogd-vg-5-6 (6)	**0.4**	155	0.77	31.5	32.3
poker-5 (5)	0.26	92.4	**0.24**	1.39	1.5
poker-6 (5)	**0.38**	463	0.39	6.58	6.99
poker-9 (5)	0.79	$\perp$	**0.63**	782	1022
poker-12 (5)	1.51	$\perp$	**0.89**	$\perp$	$\perp$
rb-3-12-12-30-0.630 (3)	4.13	1.2	3.61	**0.79**	0.86
rb-3-16-16-44-0.635 (3)	94.3	7.41	51.6	**2.31**	3.11
rb-3-20-20-60-0.632 (3)	614	34.4	830	24.3	**23.3**
pedigree1 (5)	$\perp$	$\perp$	890	819	**35.0**
barley (5)	$\perp$	$\perp$	40.7	23	**20.3**
cpcs422b (18)	**7.4**	113	8.61	58.3	111
link (4)	68.6	$\perp$	5.41	**4.55**	6.5
rand-10-20-10-5-9 (10)	3.94	$\perp$	**2.39**	$\perp$	$\perp$
rand-10-20-10-5-10 (10)	5.27	$\perp$	**2.67**	$\perp$	$\perp$
renault-mod-12 (10)	1.74	680	**1.39**	6.01	14.4
renault-mod-14 (10)	2.49	$\perp$	**1.49**	6.83	14.9

CPU time (in seconds) to prove optimality on various selected instances (time-out of 1,200 seconds set per instance, $\perp$: time-out reached)

Conclusion and future work

- Conclusion
  - New filtering algorithm enforcing a weak version of Generalized Arc Consistency called GAC^w
  - Can be applied on soft table constraints with a default cost of either 0 or k
  - A filtering method combining simple tabular reduction and cost transfer operations
  - Efficient approach compared to generic algorithms when soft table constraints have large arity
  - Generic algorithms not efficient to their full extent, particularly with cost transfer postponed
- Future work
  - Generalize our approach to soft table constraints with any default cost