5.357 soft_all_equal_min_ctr

DESCRIPTION LINKS GRAPH

Origin [205]

Constraint soft_all_equal_min_ctr(N, VARIABLES)

Synonyms soft_alldiff_max_ctr, soft_alldifferent_max_ctr, soft_alldistinct_max_ctr.

Arguments

N	:	int
VARIABLES	:	collection(var−dvar)

Restrictions

$N \geq 0$
$N \leq |VARIABLES| * |VARIABLES| - |VARIABLES|$
required(VARIABLES, var)

Purpose Consider the *equality* constraints involving two distinct variables of the collection VARIABLES. Among the previous set of constraints, N is less than or equal to the number of *equality* constraints that hold.

Example $(6, \langle 5, 1, 5, 5 \rangle)$

Within the collection $\langle 5, 1, 5, 5 \rangle$ six equality constraints holds. Consequently, the soft_all_equal_ctr constraint holds since the argument $N = 6$ is less than or equal to the number of equality constraints that hold.

Typical

$N > 0$
$N < |VARIABLES| * |VARIABLES| - |VARIABLES|$
$|VARIABLES| > 1$

Symmetries

- N can be decreased to any value ≥ 0.
- Items of VARIABLES are permutable.
- All occurrences of two distinct values of VARIABLES.var can be swapped; all occurrences of a value of VARIABLES.var can be renamed to any unused value.

Remark It was shown in [205] that, finding out whether the soft_all_equal_ctr constraint has a solution or not is NP-hard. This was achieved by reduction from 3-dimensional-matching. Hebrard *et al.* also identify a tractable class when no value occurs in more than two variables of the collection VARIABLES that is equivalent to the vertex matching problem. One year later, [149] shows how to achieve bound-consistency in polynomial time.

See also **common keyword:** soft_all_equal_max_var, soft_all_equal_min_var, soft_alldifferent_ctr, soft_alldifferent_var *(soft constraint)*.

hard version: all_equal.

implied by: `and`, `balance`, `equivalent`, `nor`.

related: `atmost_nvalue`.

Keywords

complexity: 3-dimensional-matching.

constraint type: soft constraint, value constraint, relaxation, decomposition-based violation measure.

filtering: bound-consistency.

Arc input(s)	VARIABLES
Arc generator	$CLIQUE(\neq) \mapsto$ collection(variables1, variables2)
Arc arity	2
Arc constraint(s)	variables1.var = variables2.var
Graph property(ies)	**NARC**$\geq$ N

Graph model We generate an initial graph with binary *equalities* constraints between each vertex and its successors. We use the arc generator $CLIQUE(\neq)$ in order to avoid considering *equality* constraints between the same variable. The graph property states that N is less than or equal to the number of *equalities* that hold in the final graph.

Parts (A) and (B) of Figure 5.698 respectively show the initial and final graph associated with the **Example** slot. Since we use the **NARC** graph property, the arcs of the final graph are stressed in bold. Six equality constraints remain in the final graph.

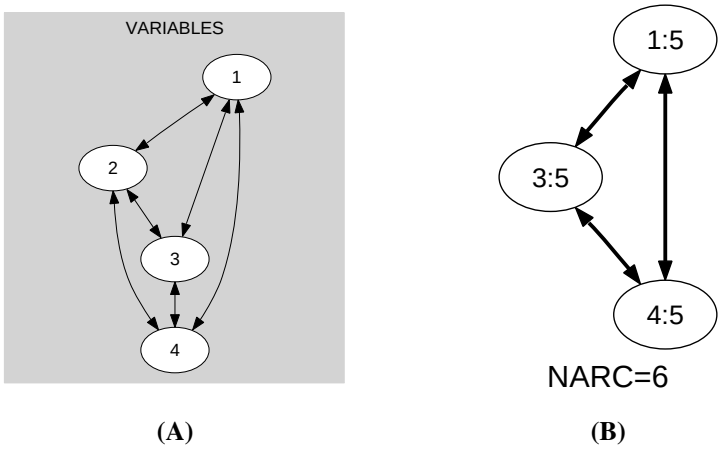

Figure 5.698: Initial and final graph of the soft_all_equal_min_ctr constraint