5.367 soft_used_by_modulo_var

DESCRIPTION LINKS GRAPH

Origin Derived from used_by_modulo

Constraint soft_used_by_modulo_var(C, VARIABLES1, VARIABLES2, M)

Synonym soft_used_by_modulo.

Arguments

C	:	dvar
VARIABLES1	:	collection(var−dvar)
VARIABLES2	:	collection(var−dvar)
M	:	int

Restrictions

$C \geq 0$
$C \leq |\text{VARIABLES2}|$
$|\text{VARIABLES1}| \geq |\text{VARIABLES2}|$
required(VARIABLES1, var)
required(VARIABLES2, var)
$M > 0$

Purpose For each integer R in $[0, M-1]$, let $N1_R$ (respectively $N2_R$) denote the number of variables of VARIABLES1 (respectively VARIABLES2) that have R as a rest when divided by M. C is the minimum number of values to change in the VARIABLES1 and VARIABLES2 collections so that for all R in $[0, M-1]$ we have $N2_R > 0 \Rightarrow N1_R \geq N2_R$.

Example $(2, \langle 9,1,1,8,8\rangle, \langle 9,9,9,1\rangle, 3)$

In the example, the values of the collections $\langle 9,1,1,8,8\rangle$ and $\langle 9,9,9,1\rangle$ are respectively associated with the equivalence classes $9 \bmod 3 = 0$, $1 \bmod 3 = 1$, $1 \bmod 3 = 1$, $8 \bmod 3 = 2$, $8 \bmod 3 = 2$ and $9 \bmod 3 = 0$, $9 \bmod 3 = 0$, $9 \bmod 3 = 0$, $1 \bmod 3 = 1$. Since there is a correspondence between two pairs of equivalence classes we must unset at least $4 - 2$ items (4 is the number of items of the VARIABLES2 collection). Consequently, the soft_used_by_modulo_var constraint holds since its first argument C is set to $4 - 2$.

Typical

$C > 0$
$|\text{VARIABLES1}| > 1$
$|\text{VARIABLES2}| > 1$
range(VARIABLES1.var) > 1
range(VARIABLES2.var) > 1
$M > 1$
M <maxval(VARIABLES1.var)
M <maxval(VARIABLES2.var)

Symmetries

- Items of `VARIABLES1` are permutable.
- Items of `VARIABLES2` are permutable.
- An occurrence of a value u of `VARIABLES1.var` can be replaced by any other value v such that v is congruent to u modulo `M`.
- An occurrence of a value u of `VARIABLES2.var` can be replaced by any other value v such that v is congruent to u modulo `M`.

Usage

A soft `used_by_modulo` constraint.

See also

hard version: `used_by_modulo`.

implied by: `soft_same_modulo_var`.

Keywords

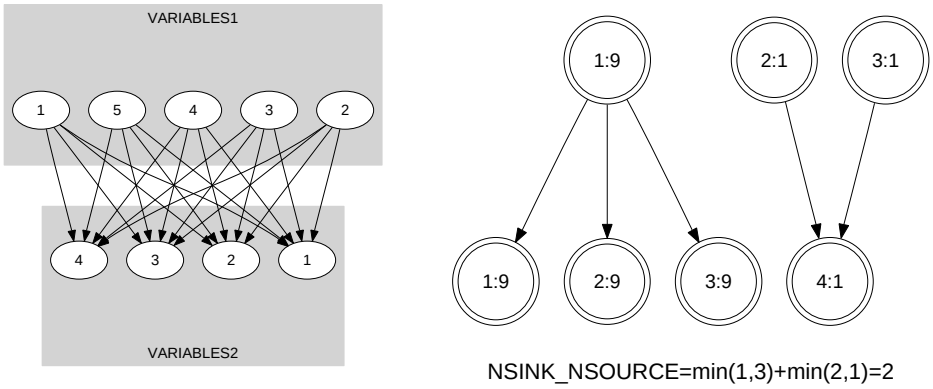
characteristic of a constraint: modulo.

constraint arguments: constraint between two collections of variables.

constraint type: soft constraint, relaxation, variable-based violation measure.

Arc input(s)	VARIABLES1 VARIABLES2
Arc generator	$PRODUCT \mapsto$ collection(variables1, variables2)
Arc arity	2
Arc constraint(s)	variables1.var mod M = variables2.var mod M
Graph property(ies)	NSINK_NSOURCE= |VARIABLES2| − C

Graph model Parts (A) and (B) of Figure 5.710 respectively show the initial and final graph associated with the **Example** slot. Since we use the **NSINK_NSOURCE** graph property, the source and sink vertices of the final graph are stressed with a double circle. The soft_used_by_modulo_var constraint holds since the cost 2 corresponds to the difference between the number of variables of VARIABLES2 and the sum over the different connected components of the minimum number of sources and sinks.



(A) **(B)**

Figure 5.710: Initial and final graph of the soft_used_by_modulo_var constraint