5.365 soft_same_var

DESCRIPTION LINKS GRAPH

Origin [423]

Constraint soft_same_var(C, VARIABLES1, VARIABLES2)

Synonym soft_same.

Arguments

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

Restrictions

$\texttt{C} \geq 0$
$\texttt{C} \leq |\texttt{VARIABLES1}|$
$|\texttt{VARIABLES1}| = |\texttt{VARIABLES2}|$
required(VARIABLES1, var)
required(VARIABLES2, var)

Purpose C is the minimum number of values to change in the VARIABLES1 and VARIABLES2 collections so that the variables of the VARIABLES2 collection correspond to the variables of the VARIABLES1 collection according to a permutation.

Example $(4, \langle 9, 9, 9, 9, 9, 1\rangle, \langle 9, 1, 1, 1, 1, 8\rangle)$

As illustrated by Figure 5.707, there is a correspondence between two pairs of values of the collections $\langle 9, 9, 9, 9, 9, 1\rangle$ and $\langle 9, 1, 1, 1, 1, 8\rangle$. Consequently, we must unset at least $6 - 2$ items (6 is the number of items of the VARIABLES1 and VARIABLES2 collections). The soft_same_var constraint holds since its first argument C is set to $6 - 2$.

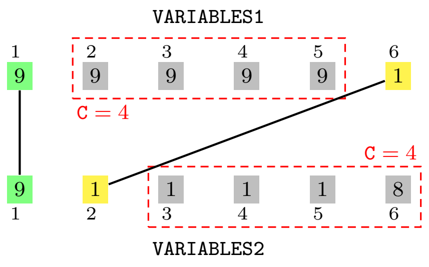


Figure 5.707: Illustration of the partial correspondence between the items of the VARIABLES1 and of the VARIABLES2 collections of the **Example** slot, i.e., C = 4 items of the VARIABLES1 or of the VARIABLES2 collections need to be changed in order to have a full correspondence

Typical

$\mathtt{C} > 0$
$|\mathtt{VARIABLES1}| > 1$
$\mathtt{range}(\mathtt{VARIABLES1.var}) > 1$
$\mathtt{range}(\mathtt{VARIABLES2.var}) > 1$

Symmetries

- Arguments are permutable w.r.t. permutation $(\mathtt{C})$ $(\mathtt{VARIABLES1}, \mathtt{VARIABLES2})$.
- Items of VARIABLES1 are permutable.
- Items of VARIABLES2 are permutable.
- All occurrences of two distinct values in VARIABLES1.var or VARIABLES2.var can be swapped; all occurrences of a value in VARIABLES1.var or VARIABLES2.var can be renamed to any unused value.

Usage

A soft same constraint.

Algorithm

A first filtering algorithm is described in [423, page 80]. A second filtering algorithm is presented in [129, 130].

See also

hard version: same.

implies: soft_used_by_var.

Keywords

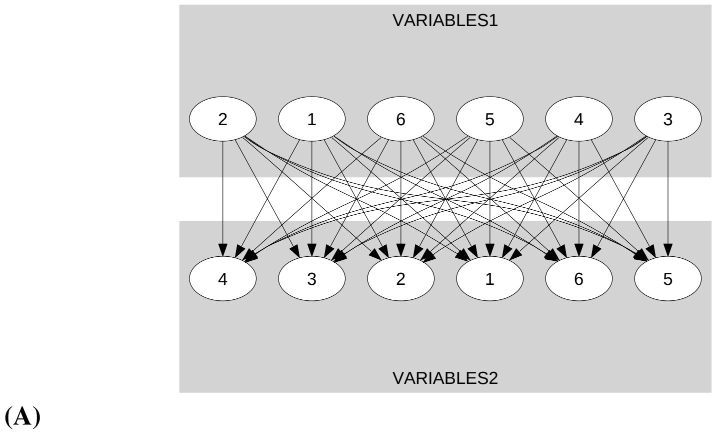
constraint arguments: constraint between two collections of variables.

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

filtering: minimum cost flow, bipartite matching.

Arc input(s)	`VARIABLES1` `VARIABLES2`
Arc generator	*PRODUCT* $\mapsto$ `collection(variables1, variables2)`
Arc arity	2
Arc constraint(s)	`variables1.var` $=$ `variables2.var`
Graph property(ies)	**NSINK_NSOURCE** $= \vert$`VARIABLES1`$\vert -$ `C`

Graph model Parts (A) and (B) of Figure 5.708 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_same_var` constraint holds since the cost 4 corresponds to the difference between the number of variables of `VARIABLES1` and the sum over the different connected components of the minimum number of sources and sinks.



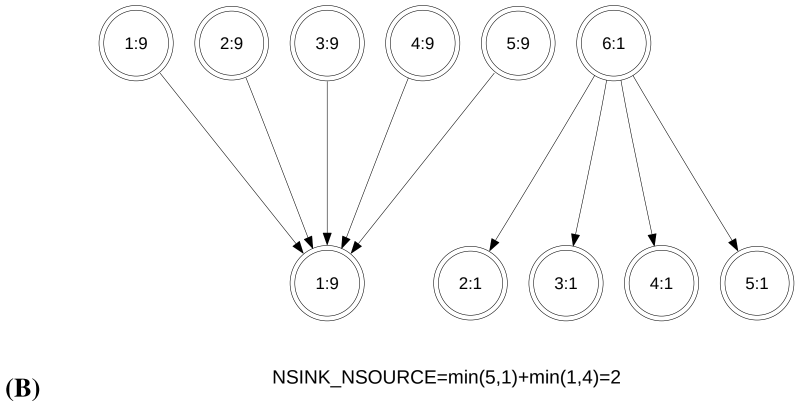


Figure 5.708: Initial and final graph of the `soft_same_var` constraint