5.302 soft_alldifferent_ctr

DESCRIPTION LINKS GRAPH

Origin [281]

Constraint `soft_alldifferent_ctr(C, VARIABLES)`

Synonyms `soft_alldiff_ctr`, `soft_alldistinct_ctr`, `soft_alldiff_min_ctr`, `soft_alldifferent_min_ctr`, `soft_alldistinct_min_ctr`, `soft_all_equal_max_ctr`.

Arguments

```
C         : dvar
VARIABLES : collection(var-dvar)
```

Restrictions

$\mathtt{C} \geq 0$
`required(VARIABLES, var)`

Purpose Consider the *disequality* constraints involving two distinct variables `VARIABLES`$[i]$`.var` and `VARIABLES`$[j]$`.var` $(i < j)$ of the collection `VARIABLES`. Among the previous set of constraints, `C` is greater than or equal to the number of *disequality* constraints that do not hold.

Example

$$\left(4, \left\langle \begin{array}{l} \mathtt{var}-5, \\ \mathtt{var}-1, \\ \mathtt{var}-9, \\ \mathtt{var}-1, \\ \mathtt{var}-5, \\ \mathtt{var}-5 \end{array} \right\rangle \right)$$

Within the collection $\langle 5, 1, 9, 1, 5, 5\rangle$ the first and fifth values, the first and sixth values, the second and fourth values, and the fifth and sixth values are identical. Consequently, the argument $\mathtt{C} = 4$ is greater than or equal to the number of *disequality* constraints that do not hold (i.e, 4) and the `soft_alldifferent_ctr` constraint holds.

Symmetries

- `C` can be increased.
- 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.

Usage A soft `alldifferent` constraint.

Remark The `soft_alldifferent_ctr` constraint is called `soft_alldiff_min_ctr` or `soft_all_equal_max_ctr` in [132].

Algorithm Since it focus on the soft aspect of the `alldifferent` constraint, the original article [281] that introduces this constraint describes how to evaluate the minimum value of `C` and how to prune according to the maximum value of `C`. The corresponding filtering algorithm does not achieve arc-consistency. W.-J. van Hoeve [383] presents a new filtering algorithm that achieves arc-consistency. This algorithm is based on a reformulation into a minimum-cost flow problem.

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

hard version: `alldifferent`.

related: `atmost_nvalue`.

Keywords **characteristic of a constraint:** all different, disequality.

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

filtering: minimum cost flow.

modelling: degree of diversity of a set of solutions.

modelling exercises: degree of diversity of a set of solutions.

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

Graph model We generate an initial graph with binary *equalities* constraints between each vertex and its successors. We use the arc generator $CLIQUE(<)$ in order to avoid counting twice the same *equality* constraint. The graph property states that C is greater than or equal to the number of *equalities* that hold in the final graph.

Parts (A) and (B) of Figure 5.544 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. Since four equality constraints remain in the final graph the *cost* variable C is greater than or equal to 4.

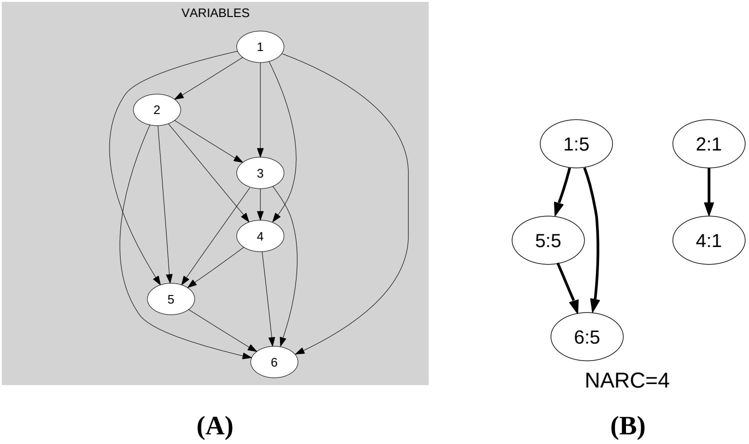

Figure 5.544: Initial and final graph of the soft_alldifferent_ctr constraint