5.331 symmetric_alldifferent

DESCRIPTION LINKS GRAPH

Origin [312]

Constraint symmetric_alldifferent(NODES)

Synonyms symmetric_alldiff, symmetric_alldistinct, symm_alldifferent, symm_alldiff, symm_alldistinct, one_factor, two_cycle.

Argument NODES : collection(index−int, succ−dvar)

Restrictions
required(NODES, [index, succ])
NODES.index ≥ 1
NODES.index $\leq |$NODES$|$
distinct(NODES, index)
NODES.succ ≥ 1
NODES.succ $\leq |$NODES$|$

Purpose All variables associated with the succ attribute of the NODES collection should be pairwise distinct. In addition enforce the following condition: if variable NODES$[i]$.succ takes value j then variable NODES$[j]$.succ takes value i. This can be interpreted as a graph-covering problem where one has to cover a digraph G with circuits of length two in such a way that each vertex of G belongs to one single circuit.

Example

$$\left\{ \left\langle \begin{array}{ll} \texttt{index} - 1 & \texttt{succ} - 3, \\ \texttt{index} - 2 & \texttt{succ} - 4, \\ \texttt{index} - 3 & \texttt{succ} - 1, \\ \texttt{index} - 4 & \texttt{succ} - 2 \end{array} \right\rangle \right\}$$

The symmetric_alldifferent constraint holds since:

- NODES$[1]$.succ $= 3 \Leftrightarrow$ NODES$[3]$.succ $= 1$,
- NODES$[2]$.succ $= 4 \Leftrightarrow$ NODES$[4]$.succ $= 2$.

Symmetry Items of NODES are permutable.

Usage As it was reported in [312, page 420], this constraint is useful to express matches between persons. The symmetric_alldifferentconstraint also appears implicitly in the *cycle cover problem* and corresponds to the four conditions given in section 1 *Modeling the Cycle Cover Problem* of [277].

Remark This constraint is referenced under the name one_factor in [187] as well as in [371]. From a modelling point of view this constraint can be expressed with the cycle constraint [37] where one imposes the additional condition that each cycle has only two nodes.

Algorithm A filtering algorithm for the `symmetric_alldifferent` constraint was proposed by J.-C. Régin in [312]. It achieves arc-consistency and its running time is dominated by the complexity of finding all edges that do not belong to any maximum cardinality matching in an undirected n-vertex, m-edge graph, i.e., $O(m \cdot n)$.

Reformulation The `symmetric_alldifferent`(`NODES`) constraint can be expressed in term of a conjunction of $|\texttt{NODES}|^2$ reified constraints of the form $\texttt{NODES}[i].\texttt{succ} = j \Leftrightarrow \texttt{NODES}[j].\texttt{succ} = i$ $(1 \leq i, j \leq |\texttt{NODES}|)$. The `symmetric_alldifferent` constraint can also be reformulated as an `inverse` constraint as shown below:

$$\texttt{symmetric\_alldifferent}\left(\left\langle \begin{array}{ll} \texttt{index} - 1 & \texttt{succ} - s_1, \\ \texttt{index} - 2 & \texttt{succ} - s_2, \\ \vdots & \vdots \\ \texttt{index} - n & \texttt{succ} - s_n \end{array} \right\rangle\right)$$

$$\texttt{inverse}\left(\left\langle \begin{array}{lll} \texttt{index} - 1 & \texttt{succ} - s_1 & \texttt{pred} - s_1, \\ \texttt{index} - 2 & \texttt{succ} - s_2 & \texttt{pred} - s_2, \\ \vdots & \vdots & \vdots \\ \texttt{index} - n & \texttt{succ} - s_n & \texttt{pred} - s_n \end{array} \right\rangle\right)$$

See also **common keyword:** `alldifferent`, `cycle`, `inverse` (*permutation*).

related: `roots`.

Keywords **application area:** sport timetabling.

characteristic of a constraint: all different, disequality.

combinatorial object: permutation, matching.

constraint type: graph constraint, timetabling constraint, graph partitioning constraint.

filtering: arc-consistency.

final graph structure: circuit.

modelling: cycle.

Arc input(s)	NODES
Arc generator	$CLIQUE(\neq) \mapsto$ collection(nodes1, nodes2)
Arc arity	2
Arc constraint(s)	• nodes1.succ = nodes2.index • nodes2.succ = nodes1.index
Graph property(ies)	**NARC**= |NODES|

Graph model In order to express the binary constraint that links two vertices one has to make explicit the identifier of the vertices.

Parts (A) and (B) of Figure 5.595 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.

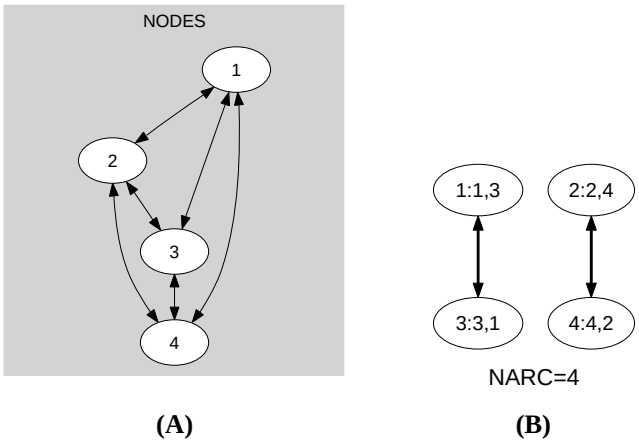

Figure 5.595: Initial and final graph of the symmetric_alldifferent constraint

Signature Since all the index attributes of the NODES collection are distinct, and because of the first condition nodes1.succ = nodes2.index of the arc constraint, each vertex of the final graph has at most one successor. Therefore the maximum number of arcs of the final graph is equal to the maximum number of vertices |NODES| of the final graph. So we can rewrite **NARC** = |NODES| to **NARC** $\geq$ |NODES| and simplify $\overline{\mathbf{NARC}}$ to $\overline{\mathbf{NARC}}$.