5.42 atmost_nvector

DESCRIPTION LINKS GRAPH

Origin Derived from `nvector`

Constraint `atmost_nvector(NVEC, VECTORS)`

Type `VECTOR : collection(var-dvar)`

Arguments
`NVEC : dvar`
`VECTORS : collection(vec - VECTOR)`

Restrictions
$|\texttt{VECTOR}| \geq 1$
$\texttt{NVEC} \geq \texttt{min}(1, |\texttt{VECTORS}|)$
`required(VECTORS, vec)`
`same_size(VECTORS, vec)`

Purpose The number of distinct tuples of values taken by the vectors of the collection `VECTORS` is less than or equal to `NVEC`. Two tuples of values $\langle A_1, A_2, \ldots, A_m\rangle$ and $\langle B_1, B_2, \ldots, B_m\rangle$ are *distinct* if and only if there exist an integer $i \in [1, m]$ such that $A_i \neq B_i$.

Example

$$\left(3, \left\langle \begin{array}{l} \texttt{vec} - \langle 5,6\rangle, \\ \texttt{vec} - \langle 5,6\rangle, \\ \texttt{vec} - \langle 9,3\rangle, \\ \texttt{vec} - \langle 5,6\rangle, \\ \texttt{vec} - \langle 9,3\rangle \end{array} \right\rangle \right)$$

The `atmost_nvector` constraint holds since the collection `VECTORS` involves at most 3 distinct tuples of values (i.e., in fact the 2 distinct tuples $\langle 5,6\rangle$ and $\langle 9,3\rangle$).

Typical
$|\texttt{VECTOR}| > 1$
$\texttt{NVEC} > 1$
$\texttt{NVEC} < |\texttt{VECTORS}|$
$|\texttt{VECTORS}| > 1$

Symmetries

- `NVEC` can be increased.
- Items of `VECTORS` are permutable.
- Items of `VECTORS.vec` are permutable (*same permutation used*).
- All occurrences of two distinct tuples of values of `VECTORS.vec` can be swapped; all occurrences of a tuple of values of `VECTORS.vec` can be renamed to any unused tuple of values.

Arg. properties Contractible wrt. `VECTORS`.

Reformulation By introducing an extra variable $\mathtt{NV} \in [0, |\mathtt{VECTORS}|]$, the atmost_nvector(NV, VECTORS) constraint can be expressed in term of an nvector(NV, VECTORS) constraint and of an inequality constraint $\mathtt{NV} \leq \mathtt{NVEC}$.

See also

comparison swapped: atleast_nvector.

implied by: nvector *($\leq$ NVEC replaced by $=$ NVEC)*, ordered_atmost_nvector.

used in graph description: lex_equal.

Keywords

characteristic of a constraint: vector.

constraint type: counting constraint, value partitioning constraint.

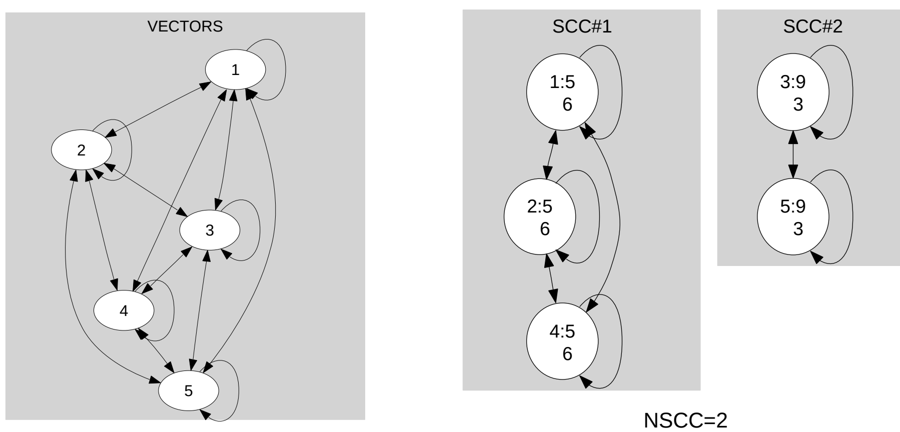
final graph structure: strongly connected component, equivalence.

modelling: number of distinct equivalence classes.

problems: domination.

Arc input(s)	`VECTORS`
Arc generator	*CLIQUE*$\mapsto$`collection`(`vectors1`, `vectors2`)
Arc arity	2
Arc constraint(s)	`lex_equal`(`vectors1.vec`, `vectors2.vec`)
Graph property(ies)	**NSCC**$\leq$ `NVEC`
Graph class	`EQUIVALENCE`

Graph model Parts (A) and (B) of Figure 5.96 respectively show the initial and final graph associated with the **Example** slot. Since we use the **NSCC** graph property we show the different strongly connected components of the final graph. Each strongly connected component corresponds to a tuple of values that is assigned to some vectors of the `VECTORS` collection. The 2 following tuple of values $\langle 5, 6\rangle$ and $\langle 9, 3\rangle$ are used by the vectors of the `VECTORS` collection.



(A) **(B)**

Figure 5.96: Initial and final graph of the `atmost_nvector` constraint