5.217 minimum

DESCRIPTION LINKS GRAPH AUTOMATON

Origin CHIP

Constraint minimum(MIN, VARIABLES)

Synonym min.

Arguments

MIN	:	dvar
VARIABLES	:	collection(var−dvar)

Restrictions

$|\texttt{VARIABLES}| > 0$
required(VARIABLES, var)

Purpose MIN is the minimum value of the collection of domain variables VARIABLES.

Example $(2, \langle 3, 2, 7, 2, 6\rangle)$

The minimum constraint holds since its first argument MIN $=$ 2 is set to the minimum value of the collection $\langle 3, 2, 7, 2, 6\rangle$.

Symmetries

- Items of VARIABLES are permutable.
- All occurrences of two distinct values of VARIABLES.var can be swapped.
- One and the same constant can be added to MIN as well as to the var attribute of all items of VARIABLES.

Usage In some project scheduling problems one has to introduce dummy activities that correspond for instance to the starting time of a given set of activities. In this context one can use the minimum constraint to get the minimum starting time of a set of tasks.

Remark Note that minimum is a constraint and not just a function that computes the minimum value of a collection of variables: potential values of MIN influence the variables of VARIABLES, and reciprocally potential values that can be assigned to variables of VARIABLES influence MIN.

The minimum constraint is called min in **JaCoP** (http://www.jacop.eu/).

Algorithm [26].

Systems min in **Choco**, min in **Gecode**, min in **JaCoP**, minimum in **SICStus**.

Used in minimum_greater_than, next_element, next_greater_element.

See also

common keyword: `maximum` *(order constraint)*.

comparison swapped: `maximum`.

generalisation: `minimum_modulo` (`variable` *replaced by* `variable` mod `constant`).

implied by: `and`.

implies: `between_min_max`, `in`.

soft variant: `minimum_except_0` *(value* 0 *is ignored)*, `open_minimum` *(open constraint)*.

specialisation: `min_n` *(minimum or order* `n` *replaced by absolute minimum)*.

Keywords

characteristic of a constraint: minimum, maxint, automaton, automaton without counters, reified automaton constraint.

constraint network structure: centered cyclic(1) constraint network(1).

constraint type: order constraint.

filtering: arc-consistency.

Arc input(s)	VARIABLES
Arc generator	$CLIQUE \mapsto$ collection(variables1, variables2)
Arc arity	2
Arc constraint(s)	$\bigvee \left(\begin{array}{l} \texttt{variables1.key} = \texttt{variables2.key}, \\ \texttt{variables1.var} < \texttt{variables2.var} \end{array} \right)$
Graph property(ies)	$\mathbf{ORDER}(0, \texttt{MAXINT}, \texttt{var}) = \texttt{MIN}$

Graph model The condition variables1.key = variables2.key holds if and only if variables1 and variables2 corresponds to the same vertex. It is used in order to enforce to keep all the vertices of the initial graph. $\mathbf{ORDER}(0, \texttt{MAXINT}, \texttt{var})$ refers to the source vertices of the graph, i.e., those vertices that do not have any predecessor.

Parts (A) and (B) of Figure 5.416 respectively show the initial and final graph associated with the **Example** slot. Since we use the ORDER graph property, the vertices of rank 0 (without considering the loops) of the final graph are outlined with a thick circle.

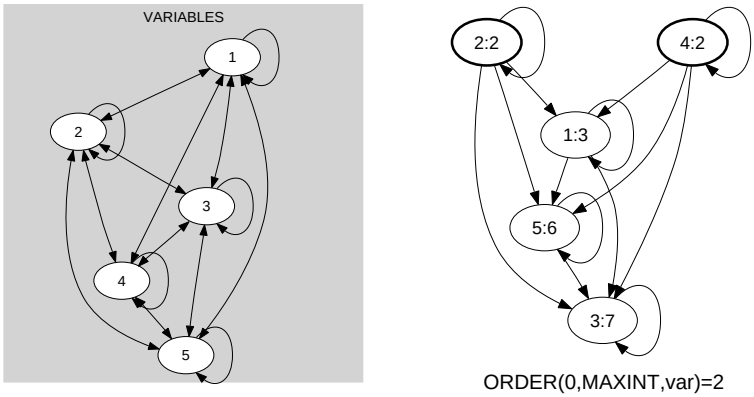

Figure 5.416: Initial and final graph of the minimum constraint

Automaton

Figure 5.417 depicts the automaton associated with the `minimum` constraint. Let $\texttt{VAR}_i$ be the i^{th} variable of the `VARIABLES` collection. To each pair $(\texttt{MIN}, \texttt{VAR}_i)$ corresponds a signature variable $\texttt{S}_i$ as well as the following signature constraint: $(\texttt{MIN} < \texttt{VAR}_i \Leftrightarrow \texttt{S}_i = 0) \wedge (\texttt{MIN} = \texttt{VAR}_i \Leftrightarrow \texttt{S}_i = 1) \wedge (\texttt{MIN} > \texttt{VAR}_i \Leftrightarrow \texttt{S}_i = 2)$.

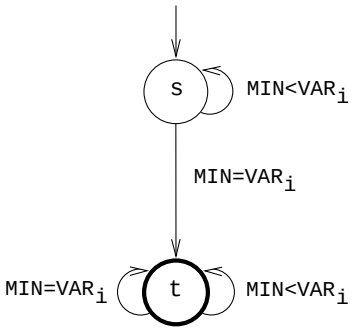

Figure 5.417: Automaton of the `minimum` constraint

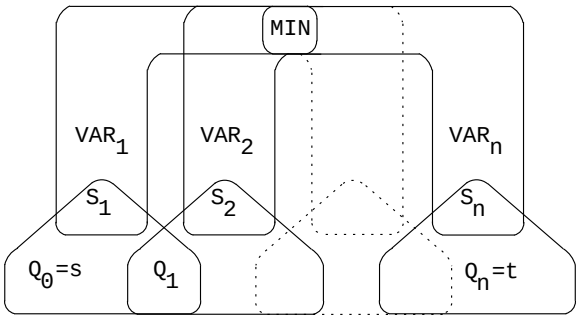

Figure 5.418: Hypergraph of the reformulation corresponding to the automaton of the `minimum` constraint