5.317 stretch_circuit

DESCRIPTION LINKS GRAPH

Origin [274]

Constraint stretch_circuit(VARIABLES, VALUES)

Usual name stretch

Arguments

VARIABLES	:	collection(var−dvar)
VALUES	:	collection(val−int, lmin−int, lmax−int)

Restrictions

|VARIABLES| > 0
required(VARIABLES, var)
|VALUES| > 0
required(VALUES, [val, lmin, lmax])
distinct(VALUES, val)
VALUES.lmin ≤ VALUES.lmax

Purpose

In order to define the meaning of the stretch_path constraint, we first introduce the notions of *stretch* and *span*. Let n be the number of variables of the collection VARIABLES and let i, j $(0 \leq i < n, 0 \leq j < n)$ be two positions within the collection of variables VARIABLES such that the following conditions apply:

- If $i \leq j$ then all variables $X_i, \ldots, X_j$ take a same value from the set of values of the val attribute.
  If $i > j$ then all variables $X_i, \ldots, X_{n-1}, X_0, \ldots, X_j$ take a same value from the set of values of the val attribute.
- $X_{(i-1) \bmod n}$ is different from X_i.
- $X_{(j+1) \bmod n}$ is different from X_j.

We call such a set of variables a *stretch*. The *span* of the stretch is equal to $1 + (j - i) \bmod n$, while the *value* of the stretch is X_i. We now define the condition enforced by the stretch_circuit constraint.

Each item $(\texttt{val} - v, \texttt{lmin} - s, \texttt{lmax} - t)$ of the VALUES collection enforces the minimum value s as well as the maximum value t for the span of a stretch of value v.

Note that:

1. Having an item $(\texttt{val} - v, \texttt{lmin} - s, \texttt{lmax} - t)$ with s strictly greater than 0 does not mean that value v should be assigned to one of the variables of collection VARIABLES. It rather means that, when value v is used, all stretches of value v must have a span that belong to interval $[s, t]$.
2. A variable of the collection VARIABLES may be assigned a value that is not defined in the VALUES collection.

Example

$$\left(\begin{array}{l} \texttt{var}-6,\\ \texttt{var}-6,\\ \texttt{var}-3,\\ \left\langle \begin{array}{l}\texttt{var}-1,\\ \texttt{var}-1,\end{array}\right\rangle,\\ \texttt{var}-1,\\ \texttt{var}-6,\\ \texttt{var}-6 \end{array}\right.\quad \left.\begin{array}{llll} & & & \\ & & & \\ & & & \\ & & & \\ & & & \\ & & & \\ & & & \\ \texttt{val}-1 & \texttt{lmin}-2 & \texttt{lmax}-4,\\ \texttt{val}-2 & \texttt{lmin}-2 & \texttt{lmax}-3,\\ \texttt{val}-3 & \texttt{lmin}-1 & \texttt{lmax}-6,\\ \texttt{val}-6 & \texttt{lmin}-2 & \texttt{lmax}-4 \end{array}\right\rangle\right)$$

The `stretch_circuit` constraint holds since the sequence $6\ 6\ 3\ 1\ 1\ 1\ 6\ 6$ contains three stretches $6\ 6\ 6\ 6$, 3, and $1\ 1\ 1$ respectively verifying the following conditions:

- The span of the first stretch $6\ 6\ 6\ 6$ is located within interval $[2, 4]$ (i.e., the limit associated with value 6).
- The span of the second stretch 3 is located within interval $[1, 6]$ (i.e., the limit associated with value 3).
- The span of the third stretch $1\ 1\ 1$ is located within interval $[2, 4]$ (i.e., the limit associated with value 1).

Symmetries

- Items of `VARIABLES` can be shifted.
- Items of `VALUES` are permutable.
- All occurrences of two distinct values in `VARIABLES.var` or `VALUES.val` can be swapped; all occurrences of a value in `VARIABLES.var` or `VALUES.val` can be renamed to any unused value.

Usage

The article [274], which originally introduced the `stretch` constraint, quotes rostering problems as typical examples of use of this constraint.

Remark

We split the origin `stretch` constraint into the `stretch_circuit` and the `stretch_path` constraints that respectively use the $PATH\ LOOP$ and $CIRCUIT\ LOOP$ arc generators. We also reorganise the parameters: the `VALUES` collection describes the attributes of each value that can be assigned to the variables of the `stretch_circuit` constraint. Finally we skipped the pattern constraint that tells what values can follow a given value.

Algorithm

A first filtering algorithm was described in the original article of G. Pesant [274]. An algorithm that also generates explanations is given in [327]. The first filtering algorithm achieving arc-consistency is depicted in [184, 185]. This algorithm is based on dynamic programming and handles the fact that some values can be followed by only a given subset of values.

Reformulation

The `stretch_circuit` constraint can be reformulated in term of a `stretch_path` constraint. Let $LMAX$ denote the maximum value taken by the `lmax` attribute within the items of the collection `VALUES`, let n be the number of variables of the collection `VARIABLES`, and let $\delta = \min(LMAX, n)$. The first and second arguments of the `stretch_path` constraint are created in the following way:

- We pass to the stretch_path the variables of the collection VARIABLES to which we add the δ first variables of the collection VARIABLES.
- We pass to the stretch_path the values of the collection VALUES with the following modification: to each value v for which the corresponding lmax attribute is greater than or equal to n we reset its value to $n + \delta$.

Even if stretch_path can achieve arc-consistency this reformulation may not achieve arc-consistency since it duplicates variables.

Using this reformulation, the example

stretch_circuit$(\langle 6,6,3,1,1,1,6,6\rangle,$
$\langle \mathtt{val}-1\ \mathtt{lmin}-2\ \mathtt{lmax}-4,\ \mathtt{val}-2\ \mathtt{lmin}-2\ \mathtt{lmax}-3,$
$\mathtt{val}-3\ \mathtt{lmin}-1\ \mathtt{lmax}-6,\ \mathtt{val}-6\ \mathtt{lmin}-2\ \mathtt{lmax}-4\rangle)$

of the **Example** slot is reformulated as:

stretch_path$(\langle 6,6,3,1,1,1,6,6,6,3,1,1,1\rangle,$
$\langle \mathtt{val}-1\ \mathtt{lmin}-2\ \mathtt{lmax}-4,\ \mathtt{val}-2\ \mathtt{lmin}-2\ \mathtt{lmax}-3,$
$\mathtt{val}-3\ \mathtt{lmin}-1\ \mathtt{lmax}-6,\ \mathtt{val}-6\ \mathtt{lmin}-2\ \mathtt{lmax}-4\rangle)$

In the reformulation δ was equal to 6, and the VALUES collection was left unchanged since no lmax attribute was equal to the number of variables of the VARIABLES collection (i.e., 8).

See also **common keyword:** group *(timetabling constraint)*,
pattern *(sliding sequence constraint,timetabling constraint)*,
sliding_distribution *(sliding sequence constraint)*,
stretch_path *(sliding sequence constraint,timetabling constraint)*.

used in reformulation: stretch_path.

Keywords **characteristic of a constraint:** cyclic.

constraint type: timetabling constraint, sliding sequence constraint.

filtering: dynamic programming, arc-consistency, duplicated variables.

For all items of VALUES:

Arc input(s)	VARIABLES
Arc generator	$CIRCUIT \mapsto$ collection(variables1, variables2) $LOOP \mapsto$ collection(variables1, variables2)
Arc arity	2
Arc constraint(s)	• variables1.var = VALUES.val • variables2.var = VALUES.val
Graph property(ies)	• not_in(MIN_NCC, 1, VALUES.lmin − 1) • MAX_NCC≤ VALUES.lmax

Graph model Part (A) of Figure 5.578 shows the initial graphs associated with values 1, 2, 3 and 6 of the **Example** slot. Part (B) of Figure 5.578 shows the corresponding final graphs associated with values 1, 3 and 6. Since value 2 is not assigned to any variable of the VARIABLES collection the final graph associated with value 2 is empty. The stretch_circuit constraint holds since:

- For value 1 we have one connected component for which the number of vertices is greater than or equal to 2 and less than or equal to 4,
- For value 2 we do not have any connected component,
- For value 3 we have one connected component for which the number of vertices is greater than or equal to 1 and less than or equal to 6,
- For value 6 we have one connected component for which the number of vertices is greater than or equal to 2 and less than or equal to 4.

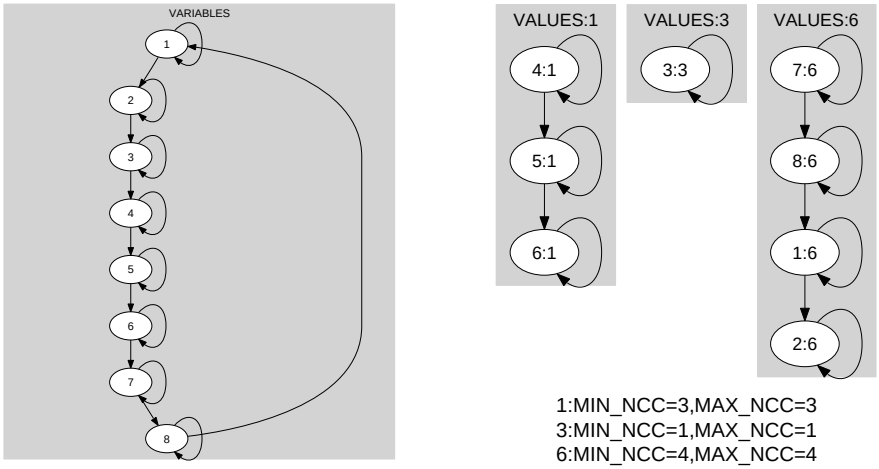

Figure 5.578: Initial and final graph of the stretch_circuit constraint