5.318 stretch_path

DESCRIPTION LINKS GRAPH AUTOMATON

Origin [274]

Constraint stretch_path(VARIABLES, VALUES)

Usual name stretch

Arguments

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

Restrictions

$|\texttt{VARIABLES}| > 0$
required(VARIABLES, var)
$|\texttt{VALUES}| > 0$
required(VALUES, [val, lmin, lmax])
distinct(VALUES, val)
VALUES.lmin $\leq$ 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. Let $X_i, \ldots, X_j$ $(1 \leq i \leq j \leq n)$ be consecutive variables of the collection of variables VARIABLES such that the following conditions apply:

- All variables $X_i, \ldots, X_j$ take a same value from the set of values of the val attribute,
- $i = 1$ or X_{i-1} is different from X_i,
- $j = n$ or X_{j+1} is different from X_j.

We call such a set of variables a *stretch*. The *span* of the stretch is equal to $j - i + 1$, while the *value* of the stretch is X_i. We now define the condition enforced by the stretch_path 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 over consecutive variables of the VARIABLES collection.

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

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

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

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

Symmetries

- Items of `VARIABLES` can be reversed.
- 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 original `stretch` constraint into the `stretch_path` and the `stretch_circuit` constraints that respectively use the *PATH LOOP* and the *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_path` constraint. Finally we skipped the pattern constraint that tells what values can follow a given value. A extension of this constraint (i.e., stretch plus pattern), called `forced_shift_stretch`, where one can specify for each value v with a 0-1 variable, whether it should occur at least once or not at all, was proposed in [185]. By reduction to Hamiltonian path it was shown that enforcing arc-consistency for `forced_shift_stretch` is NP-hard [185].

Algorithm

A first filtering algorithm was described in the original article of G. Pesant [274]. A second filtering algorithm, based on dynamic programming, achieving arc-consistency is depicted in [184, 185]. It also handles the fact that some values can be followed by only a given subset of values. An other alternative achieving arc-consistency is to use the automaton described in the **Automaton** slot.

Systems stretchPath in **Choco**, stretch in **JaCoP**.

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

generalisation: stretch_path_partition *(*variable *replaced by* variable ∈ partition*)*.

uses in its reformulation: stretch_circuit.

Keywords **characteristic of a constraint:** automaton, automaton without counters, reified automaton constraint.

combinatorial object: sequence.

constraint network structure: Berge-acyclic constraint network.

constraint type: timetabling constraint, sliding sequence constraint.

filtering: dynamic programming, arc-consistency.

final graph structure: consecutive loops are connected.

	For all items of VALUES:
Arc input(s)	VARIABLES
Arc generator	$PATH \mapsto$ 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 5.579 shows the initial graphs associated with values 1, 2, 3 and 6 of the **Example** slot. Part (B) of Figure 5.579 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_path constraint holds since:

- For value 1 we have one connected component for which the number of vertices 3 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 1 is greater than or equal to 1 and less than or equal to 6,
- For value 6 we have two connected components that both contain two vertices: this is greater than or equal to 2 and less than or equal to 2.

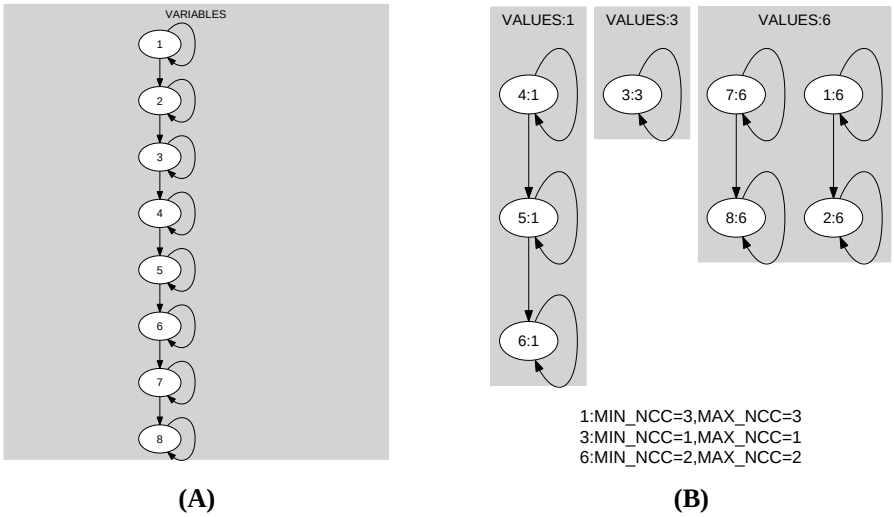

Figure 5.579: Initial and final graph of the stretch_path constraint

During the presentation of this constraint at CP'2001 the following point was mentioned: it could be useful to allow domain variables for the minimum and the maximum values of a stretch. This could be achieved in the following way: the `lmin` (respectively `lmax`) attribute would now be a domain variable that gives the size of the shortest (respectively longest) stretch. Finally within the **Graph property(ies)** slot we would replace $\geq$ (and $\leq$) by $=$.

Automaton

Let n and m respectively denote the quantities |VARIABLES| and |VALUES|. Furthermore, let $\texttt{val}_i$, $\texttt{lmin}_i$ and $\texttt{lmax}_i$, $(i \in [1,m])$, respectively be shortcuts for the expressions $\texttt{VALUES}[i].\texttt{val}$, $\texttt{VALUES}[i].\texttt{lmin}$ and $\texttt{VALUES}[i].\texttt{lmax}$. Without loss of generality, we assume that all the lmin attributes of the items of the VALUES collection are at least equal to 1. The following automaton $\mathcal{A}$ involving $1 + \texttt{lmax}_1 + \texttt{lmax}_2 + \ldots + \texttt{lmax}_m$ states only accepts solutions of the stretch_path constraint. Automaton $\mathcal{A}$ has the following states:

- an initial state s that is also a terminal state,
- $\forall i \in [1,m], \forall j \in [1, \texttt{lmin}_i - 1]$, a non-terminal state $s_{i,j}$,
- $\forall i \in [1,m], \forall j \in [\texttt{lmin}_i, \texttt{lmax}_i]$, a terminal state $s_{i,j}$.

Transitions of $\mathcal{A}$ are defined in the following way:

- $\forall i \in [1,m]$, a transition from s to $s_{i,1}$ labelled by condition $X_l = \texttt{val}_i$,
- a transition from s to s labelled by condition $X_l \neq \texttt{val}_1 \wedge X_l \neq \texttt{val}_2 \wedge \ldots \wedge X_l \neq \texttt{val}_m$,
- $\forall i \in [1,m], \forall j \in [\texttt{lmin}_i, \texttt{lmax}_i]$, a transition from $s_{i,j}$ to s labelled by condition $X_l \neq \texttt{val}_1 \wedge X_l \neq \texttt{val}_2 \wedge \ldots \wedge X_l \neq \texttt{val}_m$,
- $\forall i \in [1,m], \forall j \in [1, \texttt{lmax}_i - 1]$, a transition from $s_{i,j}$ to $s_{i,j+1}$ labelled by condition $X_l = \texttt{val}_i$,
- $\forall i \in [1,m], \forall j \in [\texttt{lmin}_i, \texttt{lmax}_i], \forall k \neq i \in [1,m]$, a transition from $s_{i,j}$ to $s_{k,1}$ labelled by condition $X_l = \texttt{val}_k$.

Figure 5.580 depicts the automaton associated with the stretch_path constraint of the **Example** slot. Transitions labels 0, 1, 2, 3 and 4 respectively correspond to the conditions $X_l \neq 1 \wedge X_l \neq 2 \wedge X_l \neq 3 \wedge X_l \neq 6$, $X_l = 1$, $X_l = 2$, $X_l = 3$, $X_l = 6$ (since values 1, 2, 3 and 6 respectively correspond to the values of the first, second, third and fourth item of the VALUES collection). The stretch_path constraint holds since the corresponding sequence of visited states, $s\ s_{41}\ s_{42}\ s_{31}\ s_{11}\ s_{12}\ s_{13}\ s_{41}\ s_{42}$, ends up in a terminal state (i.e., terminal states are depicted by thick circles in the figure).

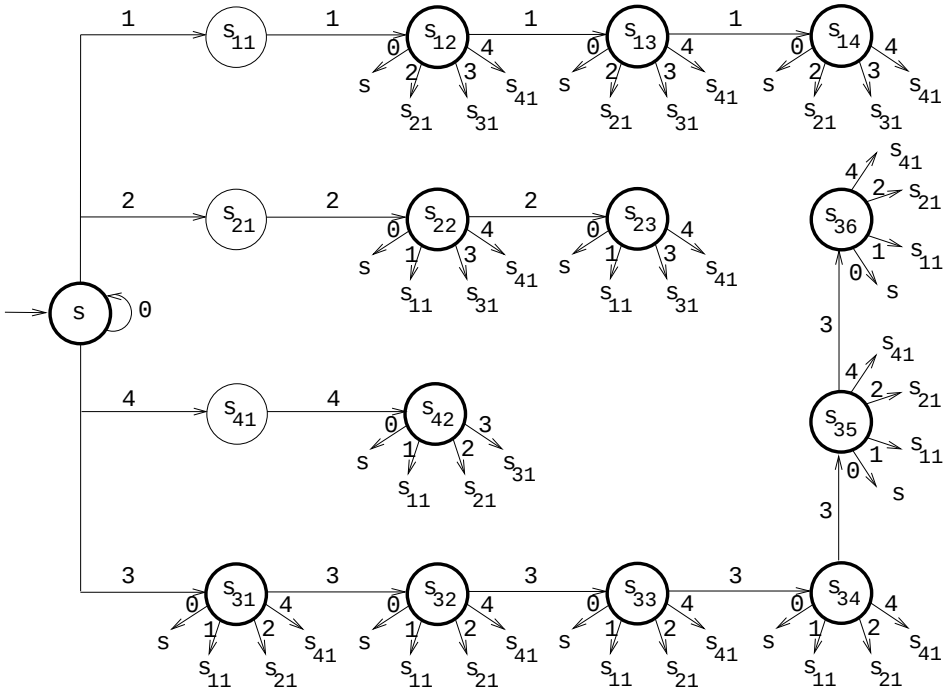

Figure 5.580: Automaton of the `stretch_path` constraint of the **Example** slot