

5.90 cycle_card_on_path

	DESCRIPTION	LINKS	GRAPH
Origin	CHIP		
Constraint	<code>cycle_card_on_path(NCYCLE, NODES, ATLEAST, ATMOST, PATH_LEN, VALUES)</code>		
Arguments	<pre> NCYCLE : dvar NODES : collection(index-int, succ-dvar, colour-dvar) ATLEAST : int ATMOST : int PATH_LEN : int VALUES : collection(val-int) </pre>		
Restrictions	<pre> NCYCLE ≥ 1 NCYCLE ≤ NODES required(NODES, [index, succ, colour]) NODES.index ≥ 1 NODES.index ≤ NODES distinct(NODES, index) NODES.succ ≥ 1 NODES.succ ≤ NODES ATLEAST ≥ 0 ATLEAST ≤ PATH_LEN ATMOST ≥ ATLEAST PATH_LEN ≥ 0 required(VALUES, val) distinct(VALUES, val) </pre>		
Purpose	<p>Consider a digraph G described by the <code>NODES</code> collection. <code>NCYCLE</code> is the number of circuits for covering G in such a way that each vertex belongs to one single circuit. In addition the following constraint must also hold: on each set of <code>PATH_LEN</code> consecutive distinct vertices of each final circuit, the number of vertices for which the attribute <code>colour</code> takes his value in the collection of values <code>VALUES</code> should be located within the range <code>[ATLEAST, ATMOST]</code>.</p>		
Example	$2, \left(\begin{array}{l} \text{index} - 1 \quad \text{succ} - 7 \quad \text{colour} - 2, \\ \text{index} - 2 \quad \text{succ} - 4 \quad \text{colour} - 3, \\ \text{index} - 3 \quad \text{succ} - 8 \quad \text{colour} - 2, \\ \text{index} - 4 \quad \text{succ} - 9 \quad \text{colour} - 1, \\ \text{index} - 5 \quad \text{succ} - 1 \quad \text{colour} - 2, \\ \text{index} - 6 \quad \text{succ} - 2 \quad \text{colour} - 1, \\ \text{index} - 7 \quad \text{succ} - 5 \quad \text{colour} - 1, \\ \text{index} - 8 \quad \text{succ} - 6 \quad \text{colour} - 1, \\ \text{index} - 9 \quad \text{succ} - 3 \quad \text{colour} - 1 \end{array} \right), 1, 2, 3, \langle 1 \rangle$		

The constraint `cycle_card_on_path` holds since the vertices of the `NODES` collection correspond to a set of disjoint circuits and since, for each set of 3 (i.e., `PATH_LEN = 3`) consecutive vertices, colour 1 (i.e., the value provided by the `VALUES` collection) occurs at least once (i.e., `ATLEAST = 1`) and at most twice (i.e., `ATMOST = 2`).

Typical

```
|NODES| > 2
NCYCLE < |NODES|
ATLEAST < PATH_LEN
ATMOST > 0
PATH_LEN > 1
|NODES| > |VALUES|
```

Symmetries

- Items of `NODES` are [permutable](#).
- An occurrence of a value of `NODES.colour` that belongs to `VALUES.val` (resp. does not belong to `VALUES.val`) can be [replaced](#) by any other value in `VALUES.val` (resp. not in `VALUES.val`).
- `ATLEAST` can be [decreased](#) to any value ≥ 0 .
- `ATMOST` can be [increased](#).
- Items of `VALUES` are [permutable](#).

Usage

Assume that the vertices of G are partitioned into the following two categories:

- Clients to visit.
- Depots where one can reload a vehicle.

Using the `cycle_card_on_path` constraint we can express a constraint like: after visiting three consecutive clients we should visit a depot. This is typically not possible with the `atmost` constraint since we do not know in advance the set of variables involved in the `atmost` constraint.

Remark

This constraint is a special case of the `sequence` parameter of the `cycle` constraint of **CHIP** [76, pages 121–128].

See also

common keyword: `cycle` (*graph partitioning constraint*).

used in graph description: `among_low_up`.

Keywords

characteristic of a constraint: `coloured`.

combinatorial object: `sequence`.

constraint type: `graph constraint`, `graph partitioning constraint`, `sliding sequence constraint`.

final graph structure: `connected component`, `one_succ`.

Arc input(s)	NODES
Arc generator	<code>CLIQUE</code> \mapsto <code>collection(nodes1, nodes2)</code>
Arc arity	2
Arc constraint(s)	<code>nodes1.succ = nodes2.index</code>
Graph property(ies)	<ul style="list-style-type: none"> • NTREE = 0 • NCC = NCYCLE
Graph class	<code>ONE_SUCC</code>
Sets	<code>PATH_LENGTH(PATH_LEN) \mapsto</code> $\left[\text{variables} - \text{col} \left(\begin{array}{c} \text{VARIABLES} - \text{collection}(\text{var} - \text{dvar}), \\ \text{[item}(\text{var} - \text{NODES.colour})] \end{array} \right) \right]$
Constraint(s) on sets	<code>among_low_up(ATLEAST, ATMOST, variables, VALUES)</code>

Graph model

Parts (A) and (B) of Figure 5.181 respectively show the initial and final graph associated with the **Example** slot. Since we use the **NCC** graph property, we show the two **connected components** of the final graph. The constraint `cycle_card_on_path` holds since all the vertices belong to a circuit (i.e., **NTREE** = 0) and since for each set of three consecutive vertices, colour 1 occurs at least once and at most twice (i.e., the `among_low_up` constraint holds).

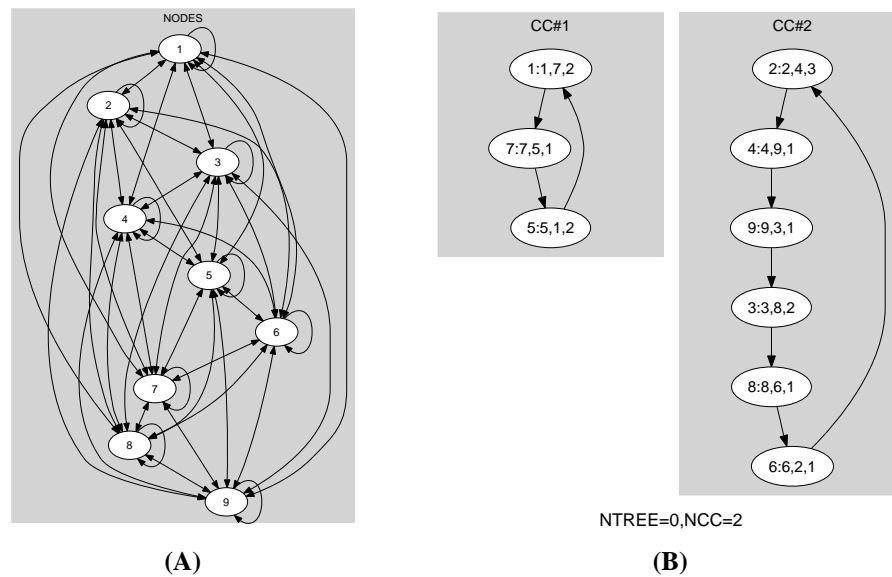


Figure 5.181: Initial and final graph of the `cycle_card_on_path` constraint

20000128

761