5.104 cycle_card_on_path

DESCRIPTION LINKS GRAPH

Origin CHIP

Constraint cycle_card_on_path(NCYCLE, NODES, ATLEAST, ATMOST, PATH_LEN, VALUES)

Arguments

NCYCLE	:	dvar
NODES	:	collection(index−int, succ−dvar, colour−dvar)
ATLEAST	:	int
ATMOST	:	int
PATH_LEN	:	int
VALUES	:	collection(val−int)

Restrictions

NCYCLE ≥ 1
NCYCLE $\leq |$NODES$|$
required(NODES, [index, succ, colour])
NODES.index ≥ 1
NODES.index $\leq |$NODES$|$
distinct(NODES, index)
NODES.succ ≥ 1
NODES.succ $\leq |$NODES$|$
ATLEAST ≥ 0
ATLEAST $\leq$ PATH_LEN
ATMOST $\geq$ ATLEAST
PATH_LEN ≥ 0
$|$VALUES$| \geq 1$
required(VALUES, val)
distinct(VALUES, val)

Purpose Consider a digraph G described by the NODES collection. NCYCLE is the number of circuits for covering G in such a way that each vertex belongs to a single circuit. In addition the following constraint must also hold: on each set of PATH_LEN consecutive distinct vertices of each final circuit, the number of vertices for which the attribute colour takes his value in the collection of values VALUES should be located within the range [ATLEAST, ATMOST].

Example

$$\left\langle \begin{array}{l} \{\mathtt{index}-1, \mathtt{succ}-7, \mathtt{colour}-2\}, \\ \{\mathtt{index}-2, \mathtt{succ}-4, \mathtt{colour}-3\}, \\ \{\mathtt{index}-3, \mathtt{succ}-8, \mathtt{colour}-2\}, \\ 2, \left\langle \{\mathtt{index}-4, \mathtt{succ}-9, \mathtt{colour}-1\}, \right. \\ \{\mathtt{index}-5, \mathtt{succ}-1, \mathtt{colour}-2\}, \\ \left. \{\mathtt{index}-6, \mathtt{succ}-2, \mathtt{colour}-1\} \right\rangle, 1, 2, 3, \\ \{\mathtt{index}-7, \mathtt{succ}-5, \mathtt{colour}-1\}, \\ \{\mathtt{index}-8, \mathtt{succ}-6, \mathtt{colour}-1\}, \\ \{\mathtt{index}-9, \mathtt{succ}-3, \mathtt{colour}-1\} \\ \langle 1 \rangle \end{array} \right\rangle$$

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

$|\text{NODES}| > 2$
$\text{NCYCLE} < |\text{NODES}|$
$\text{ATLEAST} < \text{PATH\_LEN}$
$\text{ATMOST} > 0$
$\text{PATH\_LEN} > 1$
$|\text{NODES}| > |\text{VALUES}|$
$\text{ATLEAST} > 0 \vee \text{ATMOST} < \text{PATH\_LEN}$

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** [84, 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	$CLIQUE \mapsto$ collection(nodes1, nodes2)
Arc arity	2
Arc constraint(s)	nodes1.succ = nodes2.index
Graph property(ies)	• NTREE = 0 • NCC = NCYCLE
Graph class	ONE_SUCC
Sets	PATH_LENGTH(PATH_LEN) $\mapsto$ [variables − col (VARIABLES−collection(var−dvar), [item(var − NODES.colour)])]
Constraint(s) on sets	among_low_up(ATLEAST, ATMOST, variables, VALUES)

Graph model Parts (A) and (B) of Figure 5.248 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).

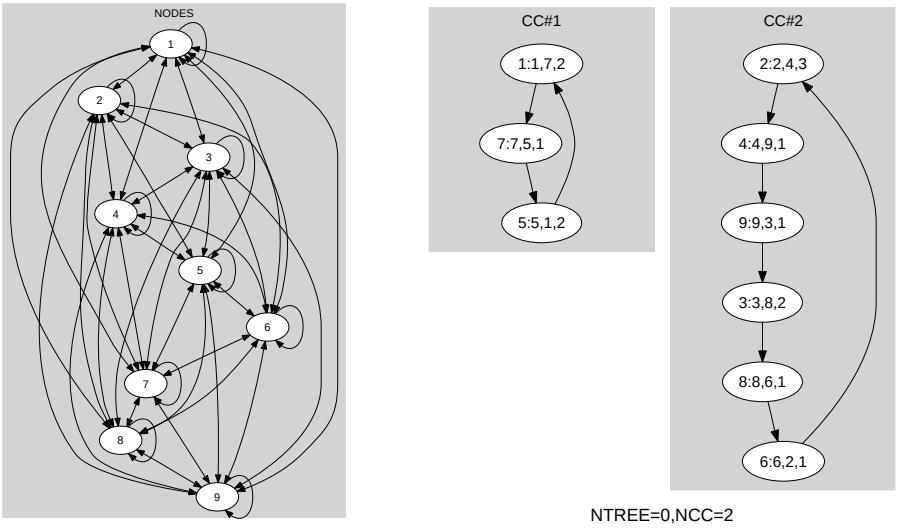

Figure 5.248: Initial and final graph of the cycle_card_on_path constraint