5.67 circuit_cluster

DESCRIPTION LINKS GRAPH

Origin Inspired by [252].

Constraint `circuit_cluster(NCIRCUIT, NODES)`

Arguments

```
NCIRCUIT : dvar
NODES    : collection(index-int, cluster-int, succ-dvar)
```

Restrictions

```
NCIRCUIT ≥ 1
NCIRCUIT ≤ |NODES|
required(NODES, [index, cluster, succ])
NODES.index ≥ 1
NODES.index ≤ |NODES|
distinct(NODES, index)
NODES.succ ≥ 1
NODES.succ ≤ |NODES|
```

Purpose Consider a digraph G, described by the `NODES` collection, such that its vertices are partitioned among several clusters. `NCIRCUIT` is the number of circuits containing more than one vertex used for covering G in such a way that each cluster is visited by exactly one circuit of length greater than 1.

Example

```
(1, ⟨ index-1 cluster-1 succ-1,
      index-2 cluster-1 succ-4,
      index-3 cluster-2 succ-3,
      index-4 cluster-2 succ-5,
      index-5 cluster-3 succ-8,
      index-6 cluster-3 succ-6,
      index-7 cluster-3 succ-7,
      index-8 cluster-4 succ-2,
      index-9 cluster-4 succ-9 ⟩)
(2, ⟨ index-1 cluster-1 succ-1,
      index-2 cluster-1 succ-4,
      index-3 cluster-2 succ-3,
      index-4 cluster-2 succ-2,
      index-5 cluster-3 succ-5,
      index-6 cluster-3 succ-9,
      index-7 cluster-3 succ-7,
      index-8 cluster-4 succ-8,
      index-9 cluster-4 succ-6 ⟩)
```

Both examples involve 9 vertices $1, 2, \ldots, 9$ such that vertices 1 and 2 belong to cluster number 1, vertices 3 and 4 belong to cluster number 2, vertices 5, 6 and 7 belong to cluster number 3, and vertices 8 and 9 belong to cluster number 4.

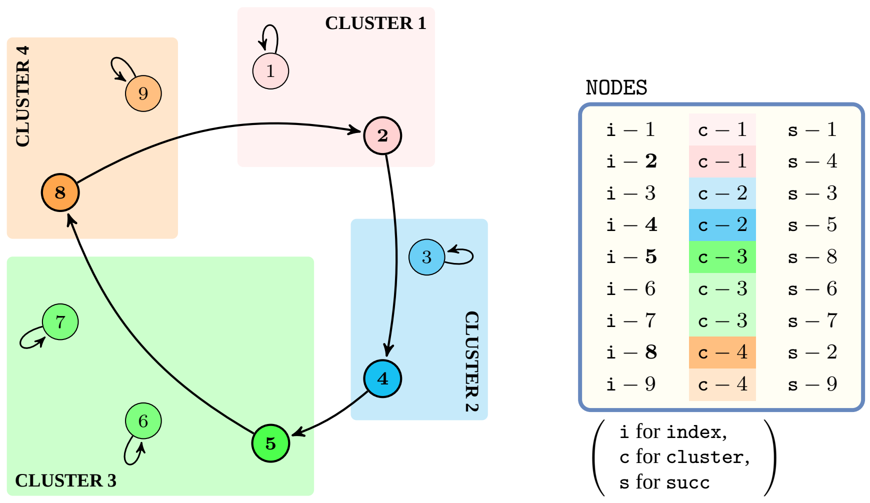


Figure 5.182: Four clusters and a covering with one circuit corresponding to the first example of the **Example** slot

The first example involves only a single circuit containing more than one vertex (i.e., see in Figure 5.182 the circuit $2 \to 4 \to 5 \to 8 \to 2$). The corresponding `circuit_cluster` constraint holds since exactly one vertex of each cluster (i.e., vertex 2 for cluster 1, vertex 4 for cluster 2, vertex 5 for cluster 3, vertex 8 for cluster 4) belongs to this circuit.

The second example contains the two circuits $2 \to 4 \to 2$ and $6 \to 9 \to 6$ that both involve more than one vertex. The corresponding `circuit_cluster` constraint holds since exactly one vertex of each cluster (i.e., see in Figure 5.183 vertex 2 in $2 \to 4 \to 2$ for cluster 1, vertex 4 in $2 \to 4 \to 2$ for cluster 2, vertex 6 in $6 \to 9 \to 6$ for cluster 3, vertex 9 in $6 \to 9 \to 6$ for cluster 4) belongs to these two circuits.

Typical

```
NCIRCUIT < |NODES|
|NODES| > 2
range(NODES.cluster) > 1
```

Symmetry

Items of `NODES` are permutable.

Usage

A related abstraction in Operations Research was introduced in [252]. It was reported as the Generalised Travelling Salesman Problem (GTSP). The `circuit_cluster` constraint differs from the GTSP because of the two following points:

- Each node of our graph belongs to a single cluster,
- We do not constrain the number of circuits to be equal to 1: The number of circuits should be equal to one of the values of the domain of the variable `NCIRCUIT`.

See also

common keyword: `alldifferent` (*permutation*), `circuit`, `cycle` (*graph constraint*, *one_succ*).

used in graph description: `alldifferent`, `nvalues`.

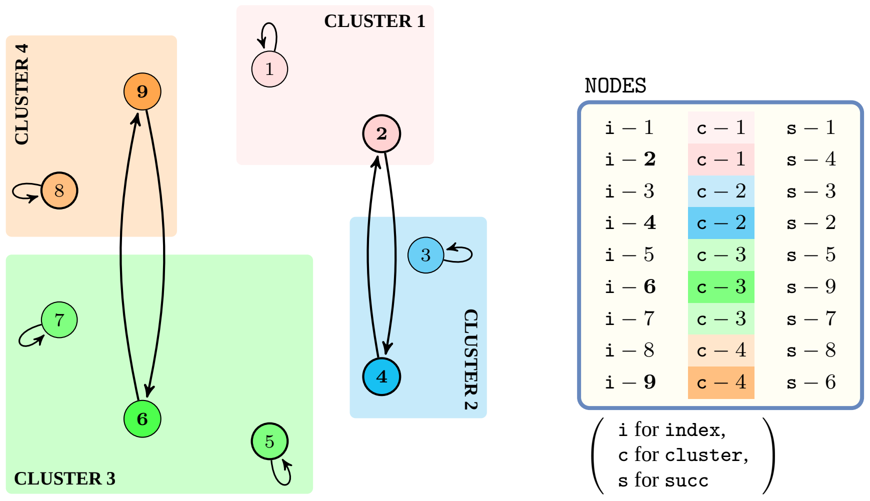


Figure 5.183: The same clusters as in the first example of the **Example** slot and a covering with two circuits corresponding to the second example of the **Example** slot

Keywords

combinatorial object: permutation.

constraint type: graph constraint.

final graph structure: strongly connected component, one_succ.

modelling: cluster.

Arc input(s)	NODES
Arc generator	$CLIQUE \mapsto$ `collection`(`nodes1`, `nodes2`)
Arc arity	2
Arc constraint(s)	• `nodes1.succ` $\neq$ `nodes1.index` • `nodes1.succ` $=$ `nodes2.index`
Graph property(ies)	• **NTREE** $= 0$ • **NSCC** $=$ NCIRCUIT
Graph class	ONE_SUCC
Sets	ALL_VERTICES $\mapsto$ $\left[\texttt{variables} - \texttt{col}\left(\begin{array}{l}\texttt{VARIABLES}-\texttt{collection}(\texttt{var}-\texttt{dvar}),\\ [\texttt{item}(\texttt{var}-\texttt{NODES.cluster})]\end{array}\right)\right]$
Constraint(s) on sets	• `alldifferent`(`variables`) • `nvalues`(`variables`, $=$, `size`(NODES, `cluster`))

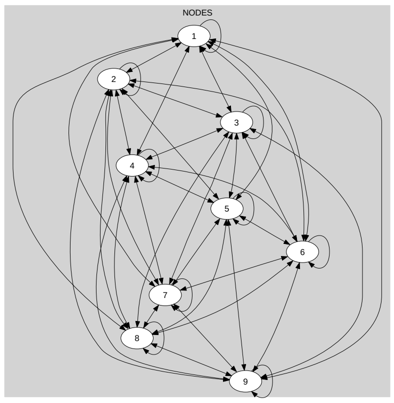
Graph model In order to express the binary constraint linking two vertices one has to make explicit the identifier of each vertex as well as the cluster to which belongs each vertex. This is why the `circuit_cluster` constraint considers objects that have the following three attributes:

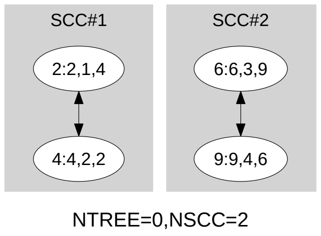
- The attribute `index` that is the identifier of a vertex.
- The attribute `cluster` that is the cluster to which belongs a vertex.
- The attribute `succ` that is the unique successor of a vertex.

The partitioning of the clusters by different circuits is expressed in the following way:

- First note the condition `nodes1.succ` $\neq$ `nodes1.index` prevents the final graph of containing any loop. Moreover the condition `nodes1.succ` $=$ `nodes2.index` imposes no more than one successor for each vertex of the final graph.
- The graph property **NTREE** $= 0$ enforces that all vertices of the final graph belong to one circuit.
- The graph property **NSCC** $=$ NCIRCUIT express the fact that the number of strongly connected components of the final graph is equal to NCIRCUIT.
- The constraint `alldifferent`(`variables`) on the set ALL_VERTICES (i.e., all the vertices of the final graph) states that the cluster attributes of the vertices of the final graph should be pairwise distinct. This concretely means that no cluster should be visited more than once.
- The constraint `nvalues`(`variables`, $=$, `size`(NODES, `cluster`)) on the set ALL_VERTICES conveys the fact that the number of distinct values of the cluster attribute of the vertices of the final graph should be equal to the total number of clusters. This implies that each cluster is visited at least one time.

Parts (A) and (B) of Figure 5.184 respectively show the initial and final graph associated with the second example of the **Example** slot. Since we use the **NSCC** graph property, we show the two strongly connected components of the final graph. They respectively correspond to the two circuits $2 \rightarrow 4 \rightarrow 2$ and $6 \rightarrow 9 \rightarrow 6$. Since all the vertices belongs to a circuit we have that **NTREE** $= 0$.





(A) **(B)**

Figure 5.184: Initial and final graph of the `circuit_cluster` constraint