5.105 cycle_or_accessibility

DESCRIPTION LINKS GRAPH

Origin

Inspired by [244].

Constraint

`cycle_or_accessibility(MAXDIST, NCYCLE, NODES)`

Arguments

```
MAXDIST : int
NCYCLE  : dvar
NODES   : collection(index-int, succ-dvar, x-int, y-int)
```

Restrictions

```
MAXDIST ≥ 0
NCYCLE ≥ 1
NCYCLE ≤ |NODES|
required(NODES, [index, succ, x, y])
NODES.index ≥ 1
NODES.index ≤ |NODES|
distinct(NODES, index)
NODES.succ ≥ 0
NODES.succ ≤ |NODES|
NODES.x ≥ 0
NODES.y ≥ 0
```

Purpose

Consider a digraph G described by the NODES collection. Cover a subset of the vertices of G by a set of vertex-disjoint circuits in such a way that the following property holds: for each uncovered vertex v_1 of G there exists at least one covered vertex v_2 of G such that the Manhattan distance between v_1 and v_2 is less than or equal to MAXDIST.

Example

$$\left(3, 2, \left\langle \begin{array}{llll} \texttt{index}-1 & \texttt{succ}-6 & \texttt{x}-4 & \texttt{y}-5, \\ \texttt{index}-2 & \texttt{succ}-0 & \texttt{x}-9 & \texttt{y}-1, \\ \texttt{index}-3 & \texttt{succ}-0 & \texttt{x}-2 & \texttt{y}-4, \\ \texttt{index}-4 & \texttt{succ}-1 & \texttt{x}-2 & \texttt{y}-6, \\ \texttt{index}-5 & \texttt{succ}-5 & \texttt{x}-7 & \texttt{y}-2, \\ \texttt{index}-6 & \texttt{succ}-4 & \texttt{x}-4 & \texttt{y}-7, \\ \texttt{index}-7 & \texttt{succ}-0 & \texttt{x}-6 & \texttt{y}-4 \end{array} \right\rangle \right)$$

Figure 5.232 represents the solution associated with the example. The covered vertices are coloured in blue, while the links starting from the uncovered vertices are dashed. The `cycle_or_accessibility` constraint holds since:

- In the solution we have NCYCLE $= 2$ disjoint circuits.
- All the 3 uncovered nodes are located at a distance that does not exceed MAXDIST $= 3$ from at least one covered node.

Typical

```
MAXDIST > 0
NCYCLE < |NODES|
|NODES| > 2
```

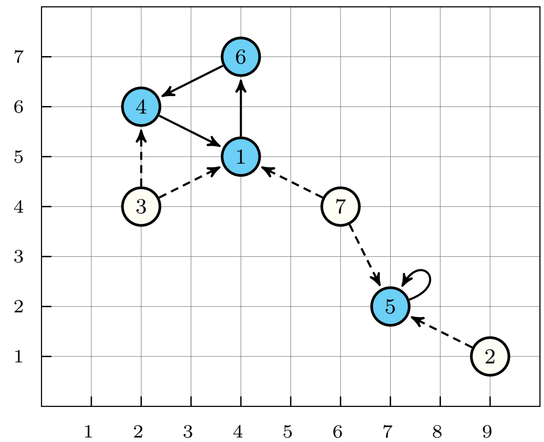


Figure 5.232: Final graph associated with the facilities location problem

Symmetries

- Items of NODES are permutable.
- Attributes of NODES are permutable w.r.t. permutation (index) (succ) (x, y) (*permutation applied to all items*).
- One and the same constant can be added to the x attribute of all items of NODES.
- One and the same constant can be added to the y attribute of all items of NODES.

Arg. properties

Functional dependency: NCYCLE determined by NODES.

Remark

This kind of facilities location problem is described in [244, pages 187–189] pages. In addition to our example they also mention the cost problem that is usually a trade-off between the vertices that are directly covered by circuits and the others.

See also

common keyword: cycle (*graph constraint*).

used in graph description: nvalues_except_0.

Keywords

constraint type: graph constraint.

final graph structure: strongly connected component.

geometry: geometrical constraint.

modelling: functional dependency.

problems: facilities location problem.

Arc input(s)	`NODES`
Arc generator	$CLIQUE$ ↦ `collection(nodes1, nodes2)`
Arc arity	2
Arc constraint(s)	`nodes1.succ` = `nodes2.index`
Graph property(ies)	• **NTREE** = 0 • **NCC** = `NCYCLE`

Arc input(s)	`NODES`
Arc generator	$CLIQUE$ ↦ `collection(nodes1, nodes2)`
Arc arity	2
Arc constraint(s)	$\bigvee\left(\begin{array}{l}\texttt{nodes1.succ} = \texttt{nodes2.index},\\ \bigwedge\left(\begin{array}{l}\texttt{nodes1.succ} = 0,\\ \texttt{nodes2.succ} \neq 0,\\ \texttt{abs}(\texttt{nodes1.x} - \texttt{nodes2.x}) + \texttt{abs}(\texttt{nodes1.y} - \texttt{nodes2.y}) \leq \texttt{MAXDIST}\end{array}\right)\end{array}\right)$
Graph property(ies)	**NVERTEX** = |`NODES`|
Sets	$\mathsf{PRED} \mapsto \left[\begin{array}{l}\texttt{variables} - \texttt{col}\left(\begin{array}{l}\texttt{VARIABLES}-\texttt{collection}(\texttt{var}-\texttt{dvar}),\\ [\texttt{item}(\texttt{var} - \texttt{NODES.succ})]\end{array}\right),\\ \texttt{destination}\end{array}\right]$
Constraint(s) on sets	`nvalues_except_0(variables, =, 1)`

Graph model For each vertex v we have introduced the following attributes:

- `index`: the label associated with v,
- `succ`: if v is not covered by a circuit then 0; If v is covered by a circuit then index of the successor of v.
- `x`: the `x`-coordinate of v,
- `y`: the `y`-coordinate of v.

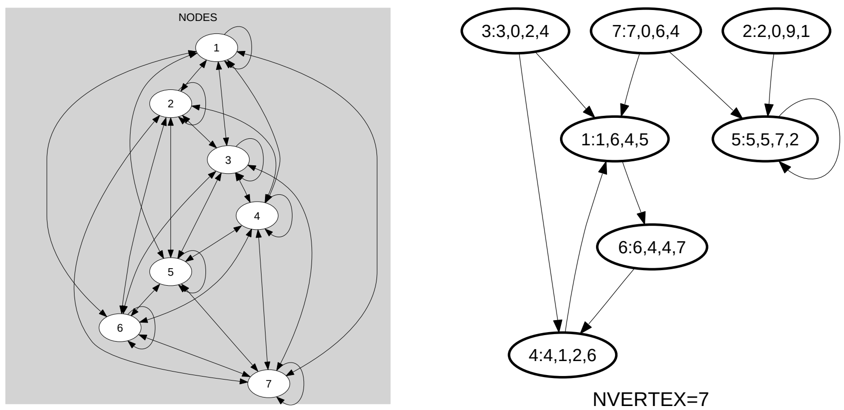
The first graph constraint forces all vertices, which have a non-zero successor, to form a set of `NCYCLE` vertex-disjoint circuits.

The final graph associated with the second graph constraint contains two types of arcs:

- The arcs belonging to one circuit (i.e., `nodes1.succ` = `nodes2.index`),
- The arcs between one vertex v_1 that does not belong to any circuit (i.e., `nodes1.succ` = 0) and one vertex v_2 located on a circuit (i.e., `nodes2.succ` ≠ 0) such that the Manhattan distance between v_1 and v_2 is less than or equal to `MAXDIST`.

In order to specify the fact that each vertex is involved in at least one arc we use the graph property **NVERTEX** = |`NODES`|. Finally the dynamic constraint `nvalues_except_0(variables, =, 1)` expresses the fact that, for each vertex v, there is exactly one predecessor of v that belongs to a circuit.

Parts (A) and (B) of Figure 5.233 respectively show the initial and final graph associated with the second graph constraint of the **Example** slot.



(A) **(B)**

Figure 5.233: Initial and final graph of the `cycle_or_accessibility` constraint

Signature

Since $|\texttt{NODES}|$ is the maximum number of vertices of the final graph associated with the second graph constraint we can rewrite $\mathbf{NVERTEX} = |\texttt{NODES}|$ to $\mathbf{NVERTEX} \geq |\texttt{NODES}|$. This leads to simplify $\underline{\overline{\mathbf{NVERTEX}}}$ to $\overline{\mathbf{NVERTEX}}$.