

5.338 tree_range

	DESCRIPTION	LINKS	GRAPH
Origin	Derived from tree .		
Constraint	tree_range(NTREES, R, NODES)		
Arguments	NTREES : dvar R : dvar NODES : collection (index- int , succ- dvar)		
Restrictions	$NTREES \geq 0$ $R \geq 0$ $R < NODES $ $ NODES > 0$ required (NODES, [index, succ]) $NODES.index \geq 1$ $NODES.index \leq NODES $ distinct (NODES, index) $NODES.succ \geq 1$ $NODES.succ \leq NODES $		
Purpose	Cover the digraph G described by the NODES collection with NTREES trees in such a way that each vertex of G belongs to one distinct tree. R is the difference between the longest and the shortest paths (from a leaf to a root) of the final graph.		
Example	$2, 1, \left\langle \begin{array}{l} index - 1 \quad succ - 1, \\ index - 2 \quad succ - 5, \\ index - 3 \quad succ - 5, \\ index - 4 \quad succ - 7, \\ index - 5 \quad succ - 1, \\ index - 6 \quad succ - 1, \\ index - 7 \quad succ - 7, \\ index - 8 \quad succ - 5 \end{array} \right\rangle$		
	The <code>tree_range</code> constraint holds since the graph associated with the items of the NODES collection corresponds to two trees (i.e., NTREES = 2): each tree respectively involves the vertices {1, 2, 3, 5, 6, 8} and {4, 7}. Furthermore R = 1 is set to the difference between the longest path (for instance 2 → 5 → 1) and the shortest path (for instance 4 → 7) from a leaf to a root. Figure 5.605 provides the two trees associated with the example.		
Symmetry	Items of NODES are permutable .		
Reformulation	By introducing a <i>distance variable</i> D_i , an <i>occurrence variable</i> O_i and a <i>leave variable</i> L_i ($1 \leq i \leq NODES $) for each item i of the NODES collection, where:		

- D_i represents the number of vertices from i to the root of the corresponding tree,
- O_i gives the number of occurrences of value i within variables $\text{NODES}[1].\text{succ}, \text{NODES}[2].\text{succ}, \dots, \text{NODES}[n].\text{succ}$,
- L_i is set to 1 if item i corresponds to a leaf (i.e., $O_i > 0$) and 0 otherwise,

the `tree_range(NTREES, R, NODES)` constraint can be expressed in term of a conjunction of one `tree` constraint, $|\text{NODES}|$ `element` constraints, $|\text{NODES}|$ linear constraints, one `global_cardinality` constraint, $|\text{NODES}|$ reified constraints, one `open_minimum`, one `maximum` and one linear constraint, where:

- The `tree` constraint models the fact that we have a forest of `NTREES` trees.
- Each `element` constraint provides the link between the attribute `succ` of the i -th item and the distance variable $D_{\text{NODES}[i].\text{succ}}$ associated with item $\text{NODES}[i].\text{succ}$.
- Each linear constraint associated with the i -th item states that the difference between the distance variable D_i and the distance variable $D_{\text{NODES}[i].\text{succ}}$ is equal to 1.
- The `global_cardinality` constraint provides the number of occurrences O_i of value i ($1 \leq i \leq |\text{NODES}|$) within variables $\text{NODES}[1].\text{succ}, \text{NODES}[2].\text{succ}, \dots, \text{NODES}[|\text{NODES}|].\text{succ}$. Note that, when O_i is equal to 0, the corresponding i -th item is a leaf of one of the `NTREES` trees.
- Each reified constraint of the form $L_i \Leftrightarrow O_i > 0$ makes the link between the i -th occurrence variable O_i and the i -th leaf variable L_i .
- The `open_minimum` constraint computes the minimum distance `MIN` from the leaves to the corresponding roots. The leaf variable L_i is used in order to select only the distance variables corresponding to leaves.
- The `maximum` constraint computes the maximum distance `MAX` from the vertices to the roots. Since the maximum is achieved by a leaf we do not need to focus just on the leaves as it was the case for the minimum distance `MIN`.
- The linear constraint $\text{MAX} - \text{MIN} = R$ links together argument `R` to the minimum and maximum distances.

With respect to the **Example** slot we get the following conjunction of constraints:

```
tree(2, (index - 1 succ - 1, index - 2 succ - 5,
        index - 3 succ - 5, index - 4 succ - 7,
        index - 5 succ - 1, index - 6 succ - 1,
        index - 7 succ - 7, index - 8 succ - 5)),
domain((D1, D2, D3, D4, D5, D6, D7, D8), 0, 8),
DS1 ∈ [0, 8], element(1, (0, D2, D3, D4, D5, D6, D7, D8), DS1), D1 - 0 = 1,
DS2 ∈ [0, 8], element(5, (1, 0, D3, D4, D5, D6, D7, D8), DS2), D2 - D5 = 1,
DS3 ∈ [0, 8], element(5, (1, D2, 0, D4, D5, D6, D7, D8), DS3), D3 - D5 = 1,
DS4 ∈ [0, 8], element(7, (1, D2, D3, 0, D5, D6, D7, D8), DS4), D4 - D7 = 1,
```

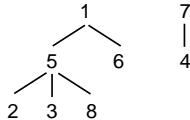


Figure 5.605: The two trees associated with the example

```

DS5 ∈ [0, 8], element(1, ⟨1, D2, D3, D4, 0, D6, D7, D8⟩, DS5), D5 - 1 = 1,
DS6 ∈ [0, 8], element(1, ⟨1, 3, 3, D4, 2, 0, D7, D8⟩, DS6), D6 - 1 = 1,
DS7 ∈ [0, 8], element(7, ⟨1, 3, 3, D4, 2, 2, 0, D8⟩, DS7), D7 - 0 = 1,
DS8 ∈ [0, 8], element(5, ⟨1, 3, 3, 2, 2, 2, 1, 0⟩, DS8), D8 - 2 = 1,
global_cardinality(⟨1, 5, 5, 7, 1, 1, 7, 5⟩, ⟨val - 1 noccurrence - 3,
val - 2 noccurrence - 0,
val - 3 noccurrence - 0,
val - 4 noccurrence - 0,
val - 5 noccurrence - 3,
val - 6 noccurrence - 0,
val - 7 noccurrence - 2,
val - 8 noccurrence - 0⟩),
1 ⇔ 3 > 0, 0 ⇔ 0 > 0, 0 ⇔ 0 > 0, 0 ⇔ 0 > 0,
1 ⇔ 3 > 0, 0 ⇔ 0 > 0, 1 ⇔ 2 > 0, 0 ⇔ 0 > 0,
open_minimum(MIN, ⟨var - 3 bool - 1, var - 0 bool - 0,
var - 0 bool - 0, var - 0 bool - 0,
var - 3 bool - 1, var - 0 bool - 0,
var - 2 bool - 1, var - 0 bool - 0⟩),
maximum(MAX, ⟨1, 3, 3, 2, 2, 2, 1, 3⟩),
MAX - MIN = R = 1.

```

See also**related:** [balance](#) (*balanced tree versus balanced assignment*).**root concept:** [tree](#).**used in reformulation:** [domain](#), [element](#), [global_cardinality](#), [maximum](#), [open_minimum](#), [tree](#).**Keywords****constraint type:** [graph constraint](#), [graph partitioning constraint](#).**final graph structure:** [connected component](#), [tree](#).**modelling:** [balanced tree](#).

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"> • <code>MAX_NSCC</code> \leq 1 • <code>NCC</code> = NTREES • <code>RANGE_DRG</code> = R

Graph model

Parts (A) and (B) of Figure 5.606 respectively show the initial and final graph associated with the **Example** slot. Since we use the `RANGE_DRG` graph property, we respectively display the longest and shortest paths of the final graph with a bold and a dash line.

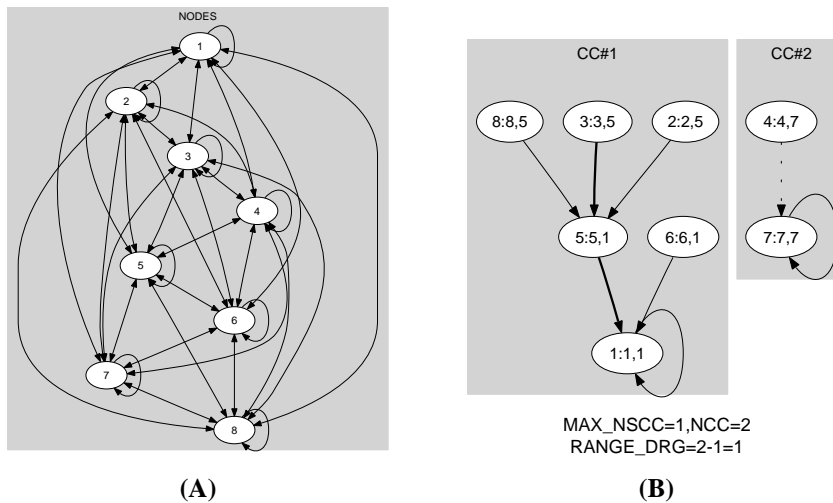


Figure 5.606: Initial and final graph of the `tree_range` constraint