

5.49 balance_tree

	DESCRIPTION	LINKS	GRAPH
Origin	derived from balance and tree		
Constraint	<code>balance_tree(BALANCE, NODES)</code>		
Arguments	BALANCE : <code>dvar</code> NODES : <code>collection(index-int, succ-dvar)</code>		
Restrictions	$BALANCE \geq 0$ $BALANCE \leq \max(0, NODES - 2)$ <code>required(NODES, [index, succ])</code> $NODES.index \geq 1$ $NODES.index \leq NODES $ <code>distinct(NODES, index)</code> $NODES.succ \geq 1$ $NODES.succ \leq NODES $		
Purpose	Consider a digraph G described by the <code>NODES</code> collection. Partition G into a set of vertex disjoint trees in such a way that each vertex of G belongs to a single tree . $BALANCE$ is equal to the difference between the number of vertices of the largest tree and the number of vertices of the smallest tree.		
Example	$ \left(\begin{array}{l} \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} \\ 4, \left\langle \begin{array}{l} \\ \\ \\ \\ \\ \\ \\ \end{array} \right\rangle \\ \begin{array}{l} index - 1 \quad succ - 1, \\ index - 2 \quad succ - 1, \\ index - 3 \quad succ - 1, \\ index - 4 \quad succ - 2, \\ index - 5 \quad succ - 6, \\ index - 6 \quad succ - 6 \end{array} \\ 2, \left\langle \begin{array}{l} \\ \\ \\ \\ \\ \end{array} \right\rangle \end{array} \right) $ <p>In the first example we have two trees involving respectively the set of vertices $\{1, 2, 3, 5, 6, 8\}$ and the set $\{4, 7\}$. They are depicted by Figure 5.127. Since $BALANCE = 6 - 2 = 4$ is the difference between the number of vertices of the largest tree (i.e., 6) and the number of vertices of the smallest tree (i.e., 2) the corresponding <code>balance_tree</code> constraint holds.</p>		
All solutions	Figure 5.128 gives all solutions to the following non ground instance of the <code>balance_tree</code> constraint: $BALANCE = 0$, $S_1 \in [1, 2]$, $S_2 \in [1, 2]$, $S_3 \in [4, 5]$, $S_4 \in [2, 4]$, $S_5 \in [4, 5]$, $S_6 \in [5, 6]$, <code>balance_tree(BALANCE, (1 S₁, 2 S₂, 3 S₃, 4 S₄, 5 S₅, 6 S₆))</code> .		

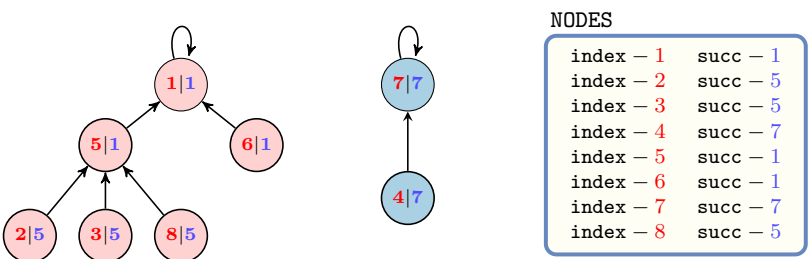


Figure 5.127: The two trees associated with the first example of the **Example** slot, respectively containing 6 and 2 vertices, therefore $BALANCE = 6 - 2 = 4$; each vertex contains the information `index|succ` where `succ` is the index of its father in the tree (by convention the father of the root is the root itself).

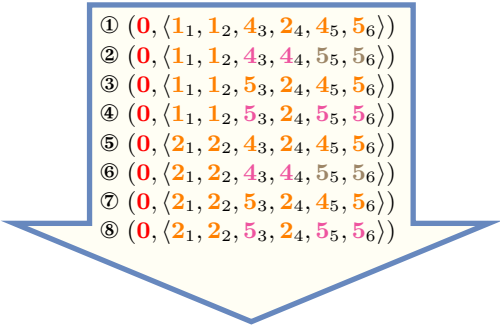
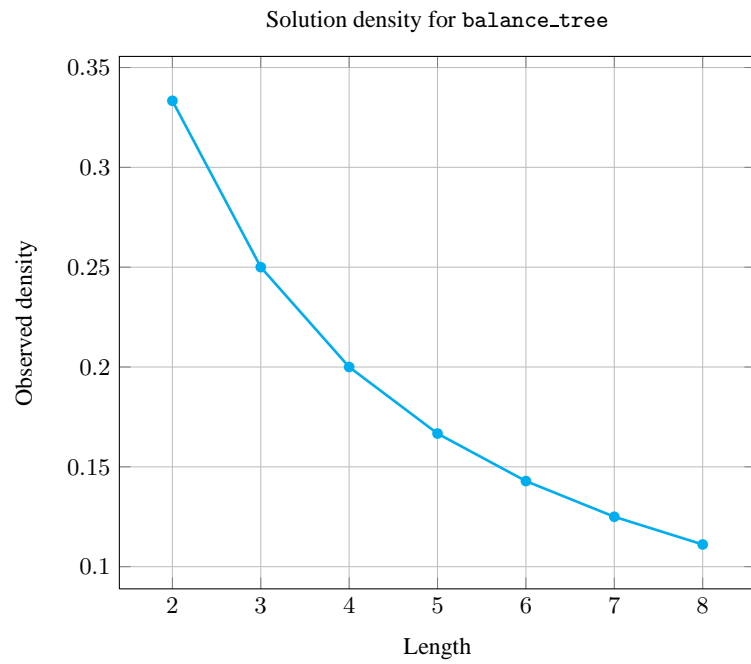
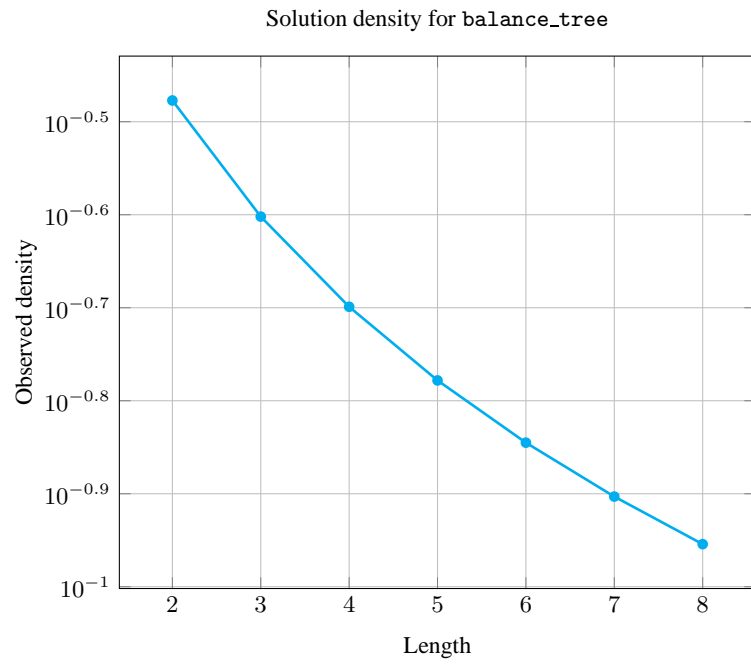


Figure 5.128: All solutions corresponding to the non ground example of the `balance_tree` constraint of the **All solutions** slot; the `index` attribute is displayed as indices of the `succ` attribute and all vertices of a same tree are coloured by the same colour.

Typical	<code> NODES > 2</code>
Symmetry	Items of <code>NODES</code> are permutable .
Arg. properties	Functional dependency : <code>BALANCE</code> determined by <code>NODES</code> .
Counting	

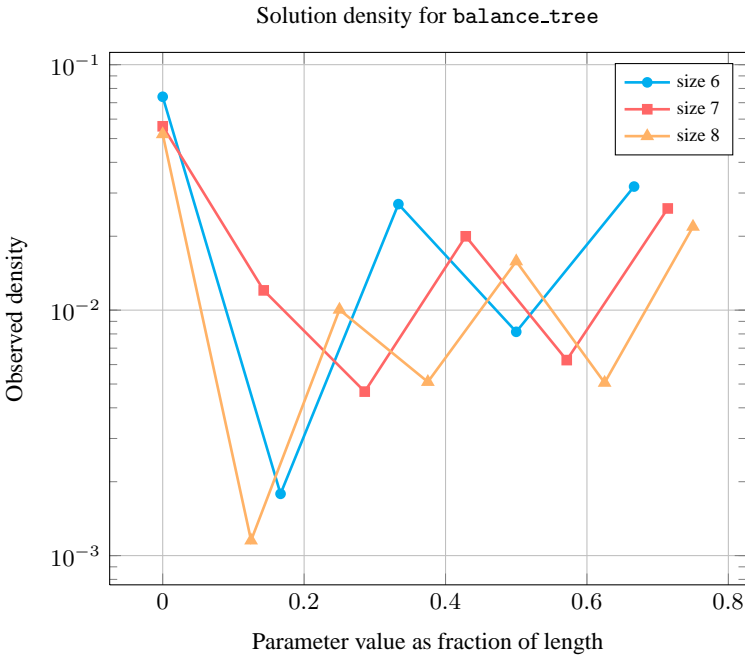
Length (<i>n</i>)	2	3	4	5	6	7	8
Solutions	3	16	125	1296	16807	262144	4782969

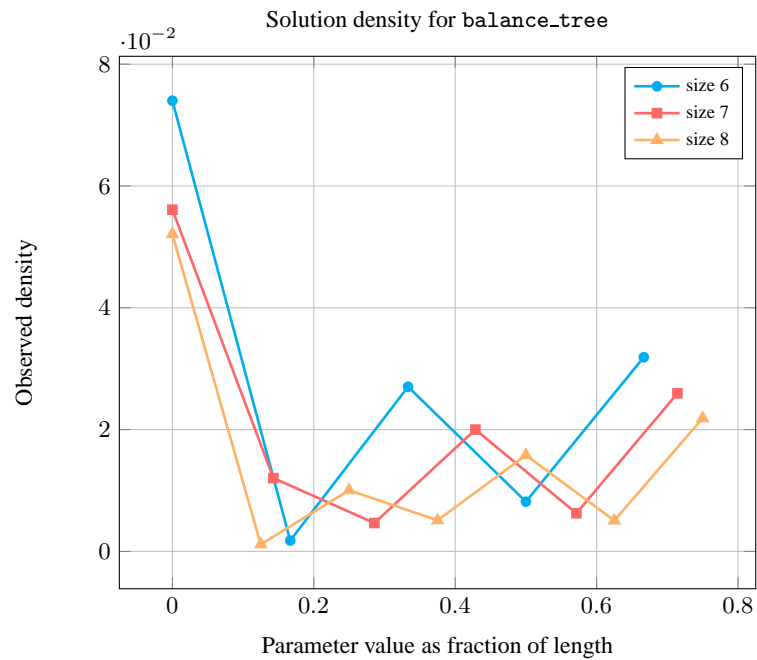
Number of solutions for `balance_tree`: domains 0..*n*



Length (n)		2	3	4	5	6	7	8
Total		3	16	125	1296	16807	262144	4782969
Parameter value	0	3	10	77	626	8707	117650	2242193
	1	-	6	12	260	210	25242	49616
	2	-	-	36	90	3180	9765	432264
	3	-	-	-	320	960	41930	219520
	4	-	-	-	-	3750	13125	680456
	5	-	-	-	-	-	54432	217728
	6	-	-	-	-	-	-	941192

Solution count for `balance_tree`: domains $0..n$



**See also**

implied by: `balance_path`.

related: `balance` (equivalence classes correspond to vertices in same tree rather than variables assigned to the same value), `tree` (do not care how many trees but how balanced the trees are).

Keywords

constraint type: graph constraint, graph partitioning constraint.

filtering: DFS-bottleneck.

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

modelling: functional dependency.

Cond. implications

```
balance_tree(BALANCE, NODES)
  with BALANCE > 0
  and BALANCE ≤ |NODES|
implies ordered_atleast_nvector(NVEC : BALANCE, VECTORS : NODES).
```

Arc input(s)	NODES
Arc generator	<code>CLIQUE</code> \mapsto <code>collection</code> (nodes1,nodes2)
Arc arity	2
Arc constraint(s)	nodes1.succ = nodes2.index
Graph property(ies)	<ul style="list-style-type: none"> • <code>MAX_NSCC</code> \leq 1 • <code>RANGE_NCC</code> = BALANCE

Graph model

In order to express the binary constraint that links two vertices one has to make explicit the identifier of the vertices. This is why the `balance_tree` constraint considers objects that have two attributes:

- One fixed attribute `index` that is the identifier of the vertex,
- One variable attribute `succ` that is the successor of the vertex.

We use the graph property `MAX_NSCC` \leq 1 in order to specify the fact that the size of the largest strongly connected component should not exceed one. In fact each root of a tree is a strongly connected component with a single vertex.

Parts (A) and (B) of Figure 5.129 respectively show the initial and final graph associated with the first example of the **Example** slot. Since we use the `RANGE_NCC` graph property, we show the connected components of the final graph. The constraint holds since all the vertices belong to a `tree` and since $\text{BALANCE} = \text{RANGE_NCC}6 - 2 = 4$.

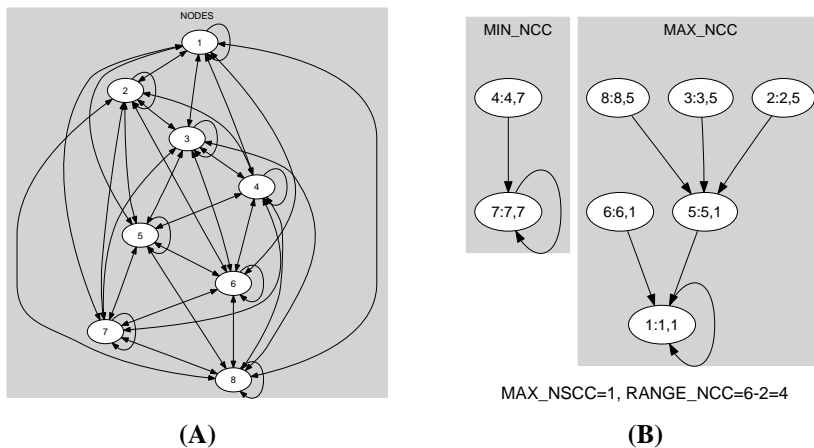


Figure 5.129: Initial and final graph of the `balance_tree` constraint