

## 5.27 assign\_and\_counts

	DESCRIPTION	LINKS	GRAPH	AUTOMATON
<b>Origin</b>	N. Beldiceanu			
<b>Constraint</b>	<code>assign_and_counts(COLOURS, ITEMS, RELOP, LIMIT)</code>			
<b>Arguments</b>	COLOURS : <code>collection(val-int)</code> ITEMS : <code>collection(bin-dvar, colour-dvar)</code> RELOP : <code>atom</code> LIMIT : <code>dvar</code>			
<b>Restrictions</b>	<code>required(COLOURS, val)</code> <code>distinct(COLOURS, val)</code> <code>required(ITEMS, [bin, colour])</code> $RELOP \in [=, \neq, <, \geq, >, \leq]$			
<b>Purpose</b>	Given several items (each of them having a specific colour that may not be initially fixed), and different bins, assign each item to a bin, so that the total number $n$ of items of colour COLOURS in each bin satisfies the condition $n$ RELOP LIMIT.			

### Example

$$\left( \begin{array}{l} \langle 4 \rangle, \\ \left\langle \begin{array}{l} \text{bin} - 1 \quad \text{colour} - 4, \\ \text{bin} - 3 \quad \text{colour} - 4, \\ \text{bin} - 1 \quad \text{colour} - 4, \\ \text{bin} - 1 \quad \text{colour} - 5 \end{array} \right\rangle, \leq, 2 \end{array} \right)$$

Figure 5.50 shows the solution associated with the example. The items and the bins are respectively represented by little squares and by the different columns. Each little square contains the value of the key attribute of the item to which it corresponds. The items for which the colour attribute is equal to 4 are located under the thick line. The `assign_and_counts` constraint holds since for each used bin (i.e., namely bins 1 and 3) the number of assigned items for which the colour attribute is equal to 4 is less than or equal to the limit 2.

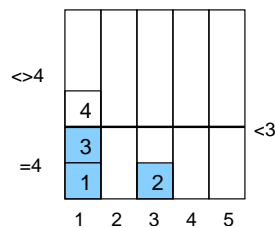


Figure 5.50: Assignment of the items to the bins

**Typical**

```
|COLOURS| > 0  
|ITEMS| > 1  
range(ITEMS.bin) > 1  
LIMIT > 0  
LIMIT < |ITEMS|
```

**Symmetries**

- Items of COLOURS are [permutable](#).
- Items of ITEMS are [permutable](#).
- All occurrences of two distinct values of ITEMS.bin can be [swapped](#); all occurrences of a value of ITEMS.bin can be [renamed](#) to any unused value.

**Usage**

Some persons have pointed out that it is impossible to use constraints such as [among](#), [atleast](#), [atmost](#), [count](#), or [global\\_cardinality](#) if the set of variables is not initially known. However, this is for instance required in practice for some timetabling problems.

**See also**

[assignment dimension removed](#): [count](#), [counts](#).  
[used in graph description](#): [counts](#).

**Keywords**

[application area](#): [assignment](#).  
[characteristic of a constraint](#): [coloured](#), [automaton](#), [automaton with array of counters](#), [derived collection](#).  
[final graph structure](#): [acyclic](#), [bipartite](#), [no loop](#).  
[modelling](#): [assignment dimension](#).

**Derived Collection**


---


$$\text{col}(\text{VALUES} - \text{collection}(\text{val} - \text{int}), [\text{item}(\text{val} - \text{COLOURS.val})])$$


---

**Arc input(s)**

ITEMS ITEMS

**Arc generator** $\text{PRODUCT} \mapsto \text{collection}(\text{items1}, \text{items2})$ **Arc arity**

2

**Arc constraint(s)** $\text{items1.bin} = \text{items2.bin}$ **Graph class**

- ACYCLIC
- BIPARTITE
- NO\_LOOP

**Sets**

SUCC  $\mapsto$

$$\left[ \begin{array}{l} \text{source}, \\ \text{variables} - \text{col} \left( \begin{array}{l} \text{VARIABLES} - \text{collection}(\text{var} - \text{dvar}), \\ [\text{item}(\text{var} - \text{ITEMS.colour})] \end{array} \right) \end{array} \right]$$
**Constraint(s) on sets**


---

 $\text{counts}(\text{VALUES}, \text{variables}, \text{RELOP}, \text{LIMIT})$ 


---

**Graph model**

We enforce the `counts` constraint on the colour of the items that are assigned to the same bin.

Parts (A) and (B) of Figure 5.51 respectively show the initial and final graph associated with the **Example** slot. The final graph consists of the following two connected components:

- The connected component containing six vertices corresponds to the items that are assigned to bin 1.
- The connected component containing two vertices corresponds to the items that are assigned to bin 3.

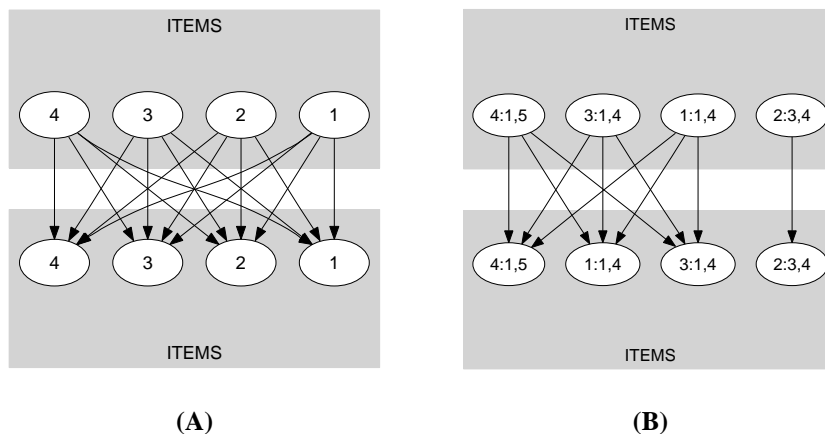


Figure 5.51: Initial and final graph of the `assign_and_counts` constraint

The `assign_and_counts` constraint holds since for each set of successors of the vertices of the final graph no more than two items take colour 4.

**Automaton**

Figure 5.52 depicts the automaton associated with the `assign_and_counts` constraint. To each `colour` attribute  $\text{COLOUR}_i$  of the collection `ITEMS` corresponds a 0-1 signature variable  $S_i$ . The following signature constraint links  $\text{COLOUR}_i$  and  $S_i$ :  $\text{COLOUR}_i \in \text{COLOURS} \Leftrightarrow S_i$ . For all items of the collection `ITEMS` for which the `colour` attribute takes its value in `COLOURS`, counts for each value assigned to the `bin` attribute its number of occurrences  $n$ , and finally imposes the condition  $n \text{ RELOP LIMIT}$ .

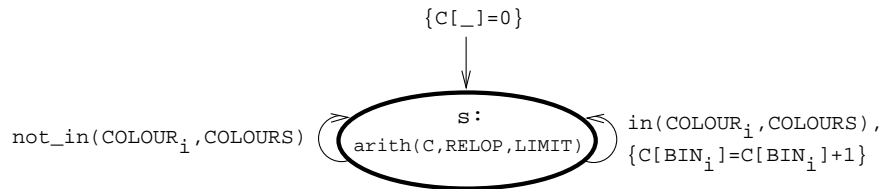


Figure 5.52: Automaton of the `assign_and_counts` constraint