5.34 assign_and_counts

DESCRIPTION **LINKS** **GRAPH** **AUTOMATON**

Origin N. Beldiceanu

Constraint `assign_and_counts(COLOURS, ITEMS, RELOP, LIMIT)`

Arguments

COLOURS	:	`collection(val−int)`
ITEMS	:	`collection(bin−dvar, colour−dvar)`
RELOP	:	`atom`
LIMIT	:	`dvar`

Restrictions

`required(COLOURS, val)`
`distinct(COLOURS, val)`
`required(ITEMS, [bin, colour])`
RELOP $\in [=, \neq, <, \geq, >, \leq]$

Purpose 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

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

Figure 5.95 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.

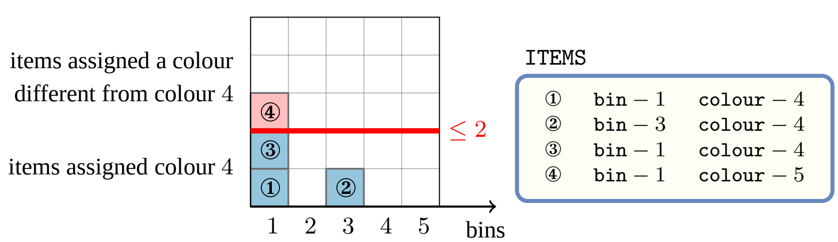


Figure 5.95: Assignment of the items to the bins

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.

All solutions

Figure 5.96 gives all solutions to the following non ground instance of the assign_and_counts constraint: $B_1 \in [1,2]$, $B_2 \in [2,3]$, $B_3 \in [2,3]$, $B_4 \in [3,4]$, $C_1 \in [0,1]$, $C_2 \in [0,1]$, $C_3 \in [0,0]$, $C_4 \in [0,1]$, assign_and_counts$(\langle 0\rangle, \langle B_1\ C_1,\ B_2\ C_2,\ B_3\ C_3,\ B_4\ C_4\rangle, \geq, 3)$.

① $(\langle 1\ 1,\ \mathbf{3\ 0},\ \mathbf{3\ 0},\ \mathbf{3\ 0}\rangle, \geq, 3)$
② $(\langle \mathbf{2\ 0},\ \mathbf{2\ 0},\ \mathbf{2\ 0},\ 3\ 1\rangle, \geq, 3)$
③ $(\langle \mathbf{2\ 0},\ \mathbf{2\ 0},\ \mathbf{2\ 0},\ 4\ 1\rangle, \geq, 3)$
④ $(\langle 2\ 1,\ \mathbf{3\ 0},\ \mathbf{3\ 0},\ \mathbf{3\ 0}\rangle, \geq, 3)$

Figure 5.96: All solutions corresponding to the non ground example of the assign_and_counts constraint of the **All solutions** slot, where items that are assigned colour 0 are shown in orange

Typical

$|\text{COLOURS}| > 0$
$|\text{ITEMS}| > 1$
$\text{range}(\text{ITEMS.bin}) > 1$
$\text{RELOP} \in [<, \leq]$
$\text{LIMIT} > 0$
$\text{LIMIT} < |\text{ITEMS}|$

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.

Arg. properties

- Contractible wrt. ITEMS when $\text{RELOP} \in [<, \leq]$.
- Extensible wrt. ITEMS when $\text{RELOP} \in [\geq, >]$.

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 col(VALUES−collection(val−int), [item(val − COLOURS.val)])

Arc input(s) ITEMS ITEMS

Arc generator $PRODUCT \mapsto$ collection(items1, items2)

Arc arity 2

Arc constraint(s) items1.bin = items2.bin

Graph class
- ACYCLIC
- BIPARTITE
- NO_LOOP

Sets

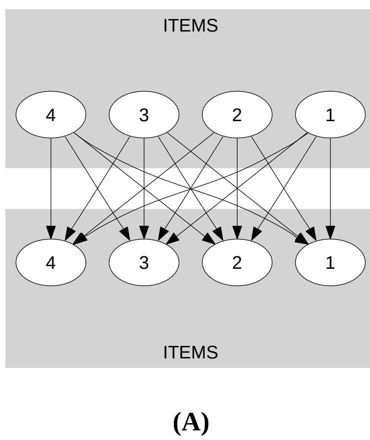
$$\text{SUCC} \mapsto \left[\begin{array}{l}\texttt{source},\\ \texttt{variables} - \texttt{col}\left(\begin{array}{l}\texttt{VARIABLES}-\texttt{collection}(\texttt{var}-\texttt{dvar}),\\ [\texttt{item}(\texttt{var} - \texttt{ITEMS.colour})]\end{array}\right)\end{array}\right]$$

Constraint(s) on sets counts(VALUES, variables, RELOP, 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 5.97 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.



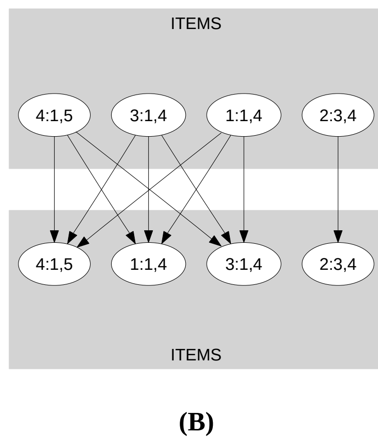


Figure 5.97: Initial and final graph of the assign_and_counts constraint

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 5.98 depicts the automaton associated with the `assign_and_counts` constraint. To each `colour` attribute $\texttt{COLOUR}_i$ of the collection `ITEMS` corresponds a 0-1 signature variable S_i. The following signature constraint links $\texttt{COLOUR}_i$ and S_i: $\texttt{COLOUR}_i \in \texttt{COLOURS} \Leftrightarrow S_i$. 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 `RELOP` `LIMIT`.

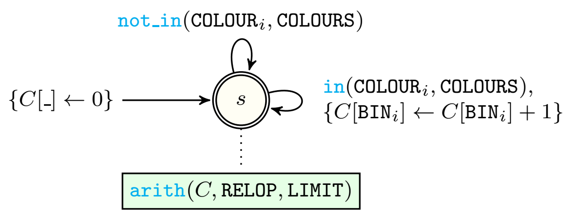


Figure 5.98: Automaton of the `assign_and_counts` constraint