## 5.36 balance

DESCRIPTION LINKS GRAPH AUTOMATON

**Origin** N. Beldiceanu

**Constraint** balance(BALANCE, VARIABLES)

**Arguments**

| | | |
|---|---|---|
| BALANCE | : | dvar |
| VARIABLES | : | collection(var−dvar) |

**Restrictions**

BALANCE $\geq 0$
BALANCE $\leq$ max$(0, |$VARIABLES$| - 2)$
required(VARIABLES, var)

**Purpose** BALANCE is equal to the difference between the number of occurrence of the value that occurs the most and the value that occurs the least within the collection of variables VARIABLES.

**Example** $(2, \langle 3, 1, 7, 1, 1 \rangle)$

In this example, values $1, 3$ and $7$ are respectively used $3, 1$ and $1$ times. The balance constraint holds since its first argument BALANCE is assigned to the difference between the maximum and minimum number of the previous occurrences (i.e., $3 - 1$). Figure 5.65 shows the solution associated with the example.

**Typical** $|$VARIABLES$| > 2$

**Symmetries**

- Items of VARIABLES are permutable.
- All occurrences of two distinct values of VARIABLES.var can be swapped; all occurrences of a value of VARIABLES.var can be renamed to any unused value.

**Usage** An application of the balance constraint is to enforce a *balanced assignment* of values, no matter how many distinct values will be used. In this case one will *push down* the maximum value of the first argument of the balance constraint.

**Remark** If we do not want to use an automaton with an array of counters a possible reformulation of the balance constraint can be achieved in the following way. We use a sort constraint in order to reorder the variables of the collection VARIABLES and compute the difference between the longest and the smallest sequences of consecutive values.

**See also** **generalisation:** balance_interval *(*variable *replaced by* variable/constant*)*, balance_modulo *(*variable *replaced by* variable mod constant*)*, balance_partition *(*variable *replaced by* variable $\in$ partition*)*.

**related:** tree_range *(balanced assignment versus balanced tree)*.

**Keywords**

**application area:** assignment.

**characteristic of a constraint:** automaton, automaton with array of counters.

**constraint type:** value constraint.

**final graph structure:** equivalence.

**modelling:** balanced assignment.

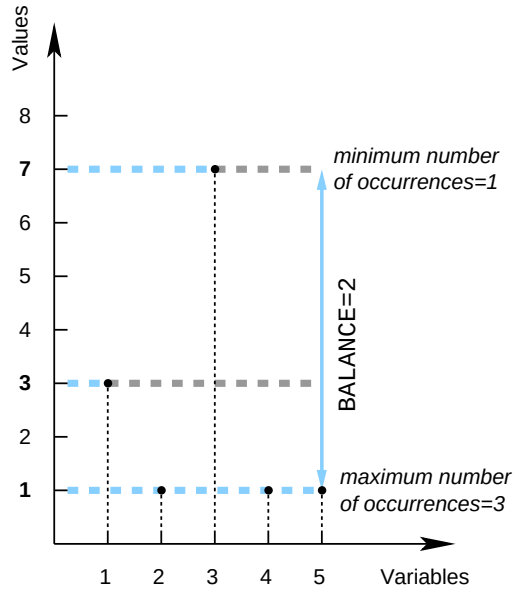


Figure 5.65: Illustration of the example: five variables respectively fixed to values 3, 1, 7, 1 and 1, and the corresponding value of BALANCE $= 2$

| | |
|---|---|
| **Arc input(s)** | VARIABLES |
| **Arc generator** | $CLIQUE \mapsto$ collection(variables1, variables2) |
| **Arc arity** | 2 |
| **Arc constraint(s)** | variables1.var = variables2.var |
| **Graph property(ies)** | RANGE_NSCC = BALANCE |
| **Graph class** | EQUIVALENCE |

**Graph model** The graph property RANGE_NSCC constraints the difference between the sizes of the largest and smallest strongly connected components.

Parts (A) and (B) of Figure 5.66 respectively show the initial and final graph associated with the **Example** slot. Since we use the RANGE_NSCC graph property, we show the largest and smallest strongly connected components of the final graph.

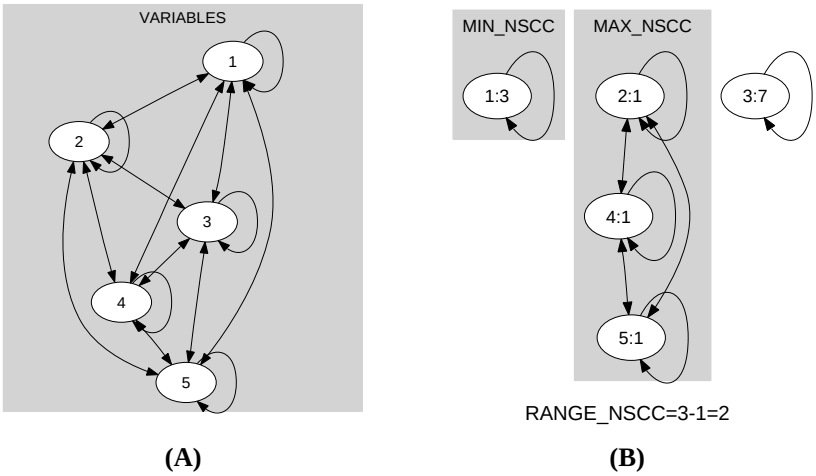


Figure 5.66: Initial and final graph of the balance constraint

**Automaton**

Figure 5.67 depicts the automaton associated with the `balance` constraint. To each item of the collection `VARIABLES` corresponds a signature variable $S_i$ that is equal to 1.

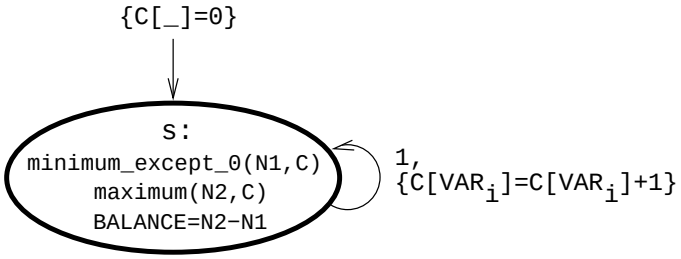


Figure 5.67: Automaton of the `balance` constraint