5.43 balance

DESCRIPTION LINKS GRAPH AUTOMATON

Origin N. Beldiceanu

Constraint balance(BALANCE, VARIABLES)

Arguments

BALANCE	:	dvar
VARIABLES	:	collection(var−dvar)

Restrictions

$\texttt{BALANCE} \geq 0$
$\texttt{BALANCE} \leq \texttt{max}(0, |\texttt{VARIABLES}| - 2)$
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

$(2, \langle 3, 1, 7, 1, 1\rangle)$
$(0, \langle 3, 3, 1, 1, 1, 3\rangle)$
$(4, \langle 3, 1, 1, 1, 1, 1\rangle)$

In the first example, values $1, 3$ and 7 are respectively used $3, 1$ and 1 times. The corresponding 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 5.97 shows the solution associated with this first example.

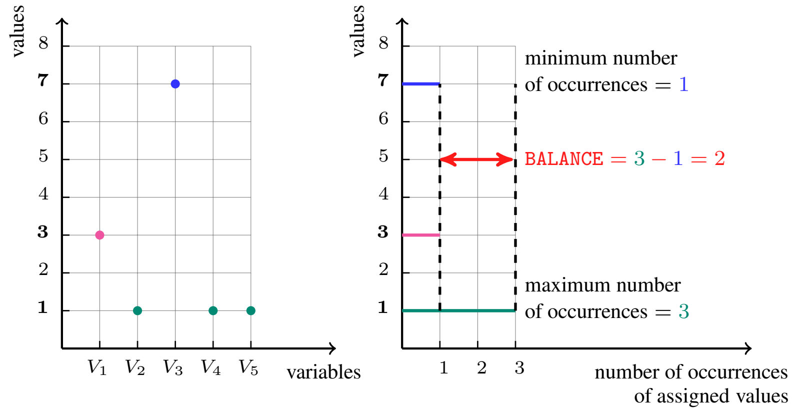
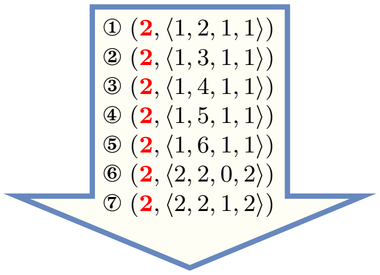


Figure 5.97: Illustration of the first example of the **Example** slot: five variables V_1, V_2, V_3, V_4, V_5 respectively fixed to values 3, 1, 7, 1 and 1, and the corresponding value of $\texttt{BALANCE} = 2$

All solutions

Figure 5.98 gives all solutions to the following non ground instance of the `balance` constraint: `BALANCE` $\in [2,3]$, $V_1 \in [0,5]$, $V_2 \in [2,6]$, $V_3 \in [0,1]$, $V_4 \in [1,2]$, `balance`(`BALANCE`, $\langle V_1, V_2, V_3, V_4\rangle$).



① $(\mathbf{2}, \langle 1,2,1,1\rangle)$
② $(\mathbf{2}, \langle 1,3,1,1\rangle)$
③ $(\mathbf{2}, \langle 1,4,1,1\rangle)$
④ $(\mathbf{2}, \langle 1,5,1,1\rangle)$
⑤ $(\mathbf{2}, \langle 1,6,1,1\rangle)$
⑥ $(\mathbf{2}, \langle 2,2,0,2\rangle)$
⑦ $(\mathbf{2}, \langle 2,2,1,2\rangle)$

Figure 5.98: All solutions corresponding to the non ground example of the `balance` constraint of the **All solutions** slot

Typical

`BALANCE` $\leq 2 + |$`VARIABLES`$|/10$
$|$`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.

Arg. properties

Functional dependency: `BALANCE` determined by `VARIABLES`.

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.

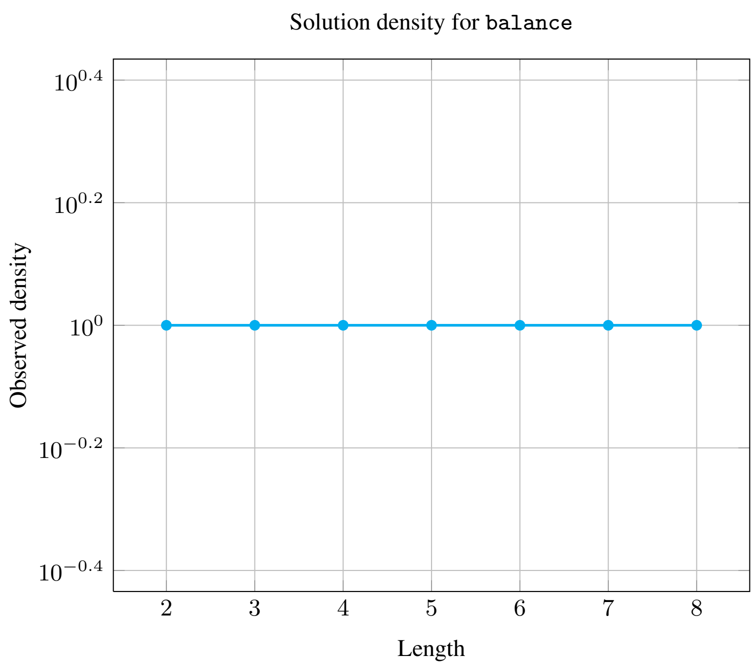
Remark

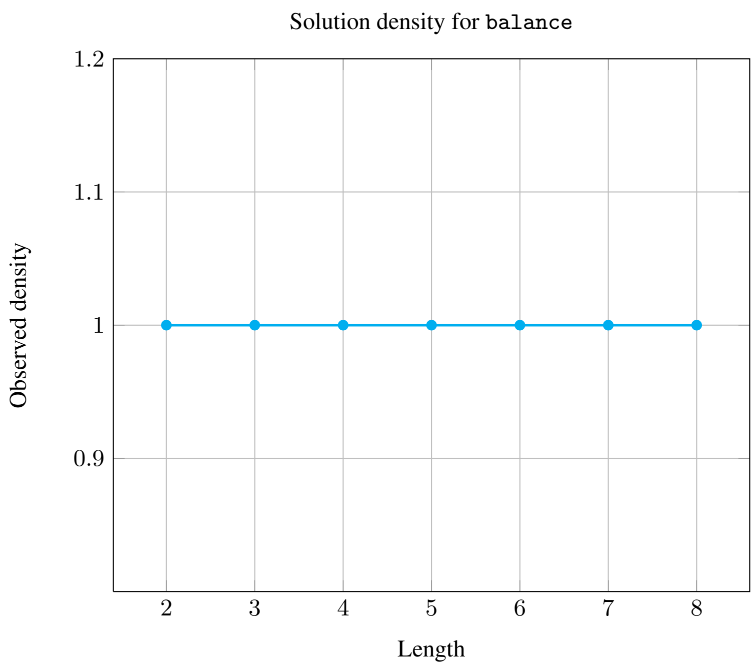
If we do not want to use an automaton with an array of counters a possible reformulation of the `balance` constraint can be achieved in the following way. We use a `sort` constraint in order to reorder the variables of the collection `VARIABLES` and compute the difference between the longest and the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

Counting

Length (n)	2	3	4	5	6	7	8
Solutions	9	64	625	7776	117649	2097152	43046721

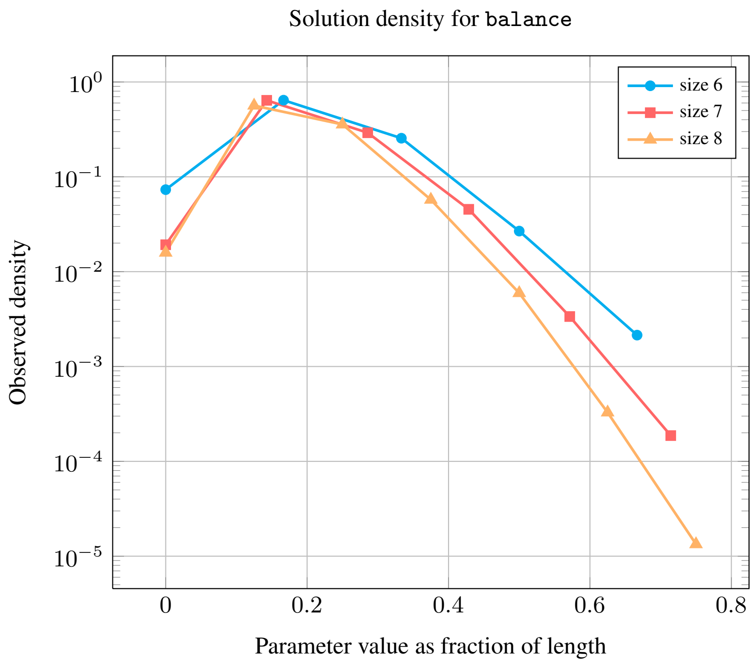
Number of solutions for `balance`: domains $0..n$

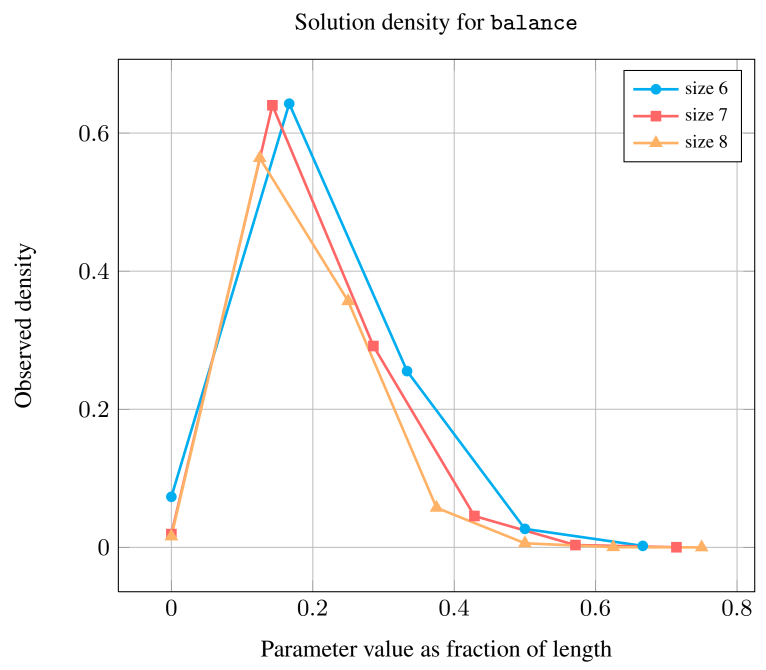

Solution density for balance
Observed density
$10^{0.4}$
$10^{0.2}$
10^{0}
$10^{-0.2}$
$10^{-0.4}$
2
3
4
5
6
7
8
Length


Solution density for balance
Observed density
1.2
1.1
1
0.9
2
3
4
5
6
7
8
Length

Length (n)		2	3	4	5	6	7	8
Total		9	64	625	7776	117649	2097152	43046721
Parameter value	0	9	28	185	726	8617	40328	682929
	1	-	36	360	5700	75600	1342600	24272640
	2	-	-	80	1200	30030	611520	15350832
	3	-	-	-	150	3150	95256	2469600
	4	-	-	-	-	252	7056	256032
	5	-	-	-	-	-	392	14112
	6	-	-	-	-	-	-	576

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





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`*)*.

implies: `soft_all_equal_min_ctr`.

related: `balance_cycle` *(balanced assignment versus graph partitionning with balanced cycles)*, `balance_path` *(balanced assignment versus graph partitionning with balanced paths)*, `balance_tree` *(balanced assignment versus graph partitionning with balanced trees)*, `nvalue` *(no restriction on how balanced an assignment is)*, `tree_range` *(balanced assignment versus balanced tree)*.

shift of concept: `equilibrium`.

Keywords

application area: assignment.

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

constraint arguments: pure functional dependency.

constraint type: value constraint.

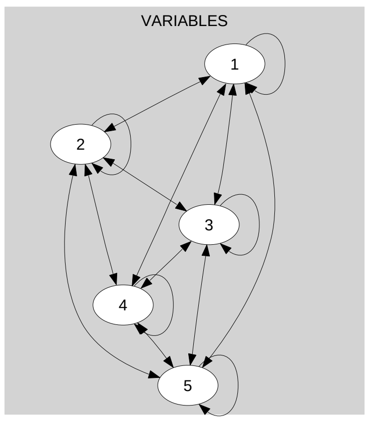
final graph structure: equivalence.

modelling: balanced assignment, functional dependency.

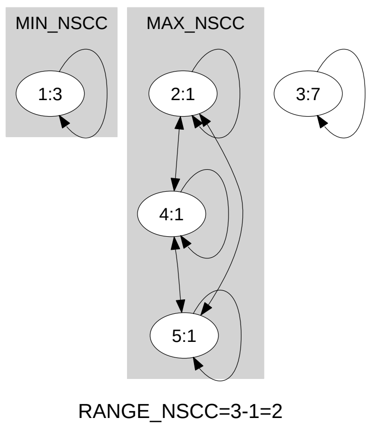
Arc input(s)	`VARIABLES`
Arc generator	$CLIQUE \mapsto$ `collection`(`variables1`, `variables2`)
Arc arity	2
Arc constraint(s)	`variables1.var` = `variables2.var`
Graph property(ies)	**RANGE_NSCC**= `BALANCE`
Graph class	`EQUIVALENCE`

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 5.99 respectively show the initial and final graph associated with the first example of the **Example** slot. Since we use the **RANGE_NSCC** graph property, we show the largest and smallest strongly connected components of the final graph.



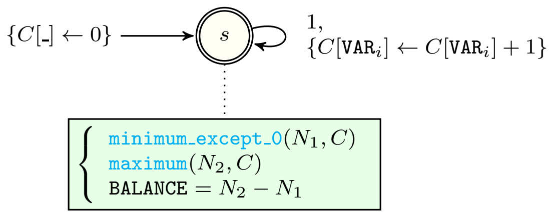
(A)



(B)

Figure 5.99: Initial and final graph of the `balance` constraint

Automaton Figure 5.100 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.



$\{C[\_] \leftarrow 0\}$
s
1,
$\{C[\texttt{VAR}_i] \leftarrow C[\texttt{VAR}_i] + 1\}$
$\texttt{minimum\_except\_0}(N_1, C)$
$\texttt{maximum}(N_2, C)$
$\texttt{BALANCE} = N_2 - N_1$

Figure 5.100: Automaton of the `balance` constraint