5.417 valley

DESCRIPTION LINKS AUTOMATON

Origin Derived from inflexion.

Constraint valley(N, VARIABLES)

Arguments

N	:	dvar
VARIABLES	:	collection(var−dvar)

Restrictions

$\mathtt{N} \geq 0$
$2 * \mathtt{N} \leq \mathtt{max}(|\mathtt{VARIABLES}| - 1, 0)$
required(VARIABLES, var)

Purpose

A variable V_v $(1 < v < m)$ of the sequence of variables $\mathtt{VARIABLES} = V_1, \ldots, V_m$ is a *valley* if and only if there exists an i (with $1 < i \leq v$) such that $V_{i-1} > V_i$ and $V_i = V_{i+1} = \cdots = V_v$ and $V_v < V_{v+1}$. N is the total number of valleys of the sequence of variables VARIABLES.

Example

$(1, \langle 1, 1, 4, 8, 8, 2, 7, 1\rangle)$
$(0, \langle 1, 1, 4, 5, 8, 8, 4, 1\rangle)$
$(4, \langle 1, 0, 4, 0, 8, 2, 4, 1, 2\rangle)$

The first valley constraint holds since the sequence 1 1 4 8 8 2 7 1 contains one valley that corresponds to the variable that is assigned to value 2.

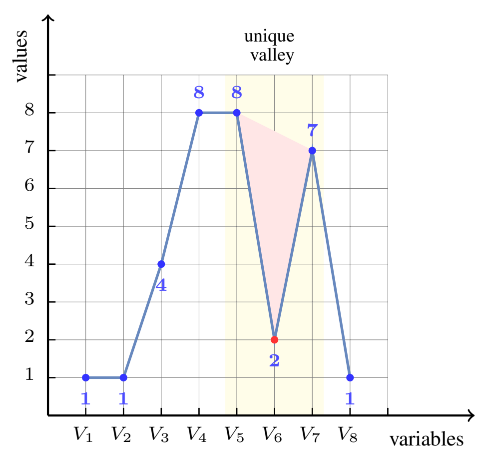


Figure 5.790: Illustration of the first example of the **Example** slot: a sequence of eight variables V_1, V_2, V_3, V_4, V_5, V_6, V_7, V_8 respectively fixed to values 1, 1, 4, 8, 8, 2, 7, 1 and its corresponding unique valley ($\mathtt{N} = 1$)

All solutions Figure 5.791 gives all solutions to the following non ground instance of the `valley` constraint: $\texttt{N} \in [1,2]$, $V_1 \in [0,1]$, $V_2 \in [0,2]$, $V_3 \in [0,2]$, $V_4 \in [0,1]$, $\texttt{valley}(\texttt{N}, \langle V_1, V_2, V_3, V_4 \rangle)$.

① $(\mathbf{1}, \langle 0, 1, 0, 1 \rangle)$
② $(\mathbf{1}, \langle 0, 2, 0, 1 \rangle)$
③ $(\mathbf{1}, \langle 1, 0, 0, 1 \rangle)$
④ $(\mathbf{1}, \langle 1, 0, 1, 0 \rangle)$
⑤ $(\mathbf{1}, \langle 1, 0, 1, 1 \rangle)$
⑥ $(\mathbf{1}, \langle 1, 0, 2, 0 \rangle)$
⑦ $(\mathbf{1}, \langle 1, 0, 2, 1 \rangle)$
⑧ $(\mathbf{1}, \langle 1, 1, 0, 1 \rangle)$
⑨ $(\mathbf{1}, \langle 1, 2, 0, 1 \rangle)$

Figure 5.791: All solutions corresponding to the non ground example of the `valley` constraint of the **All solutions** slot where each valley is coloured in orange

Typical

$|\texttt{VARIABLES}| > 2$
$\texttt{range}(\texttt{VARIABLES.var}) > 1$

Symmetries

- Items of `VARIABLES` can be reversed.
- One and the same constant can be added to the `var` attribute of all items of `VARIABLES`.

Arg. properties

- Functional dependency: `N` determined by `VARIABLES`.
- Contractible wrt. `VARIABLES` when $\texttt{N} = 0$.

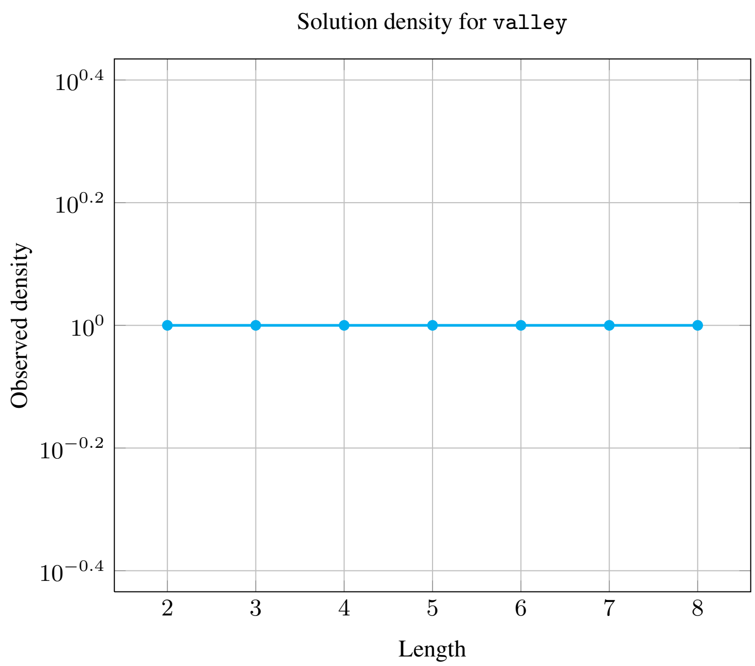
Usage Useful for constraining the number of *valleys* of a sequence of domain variables.

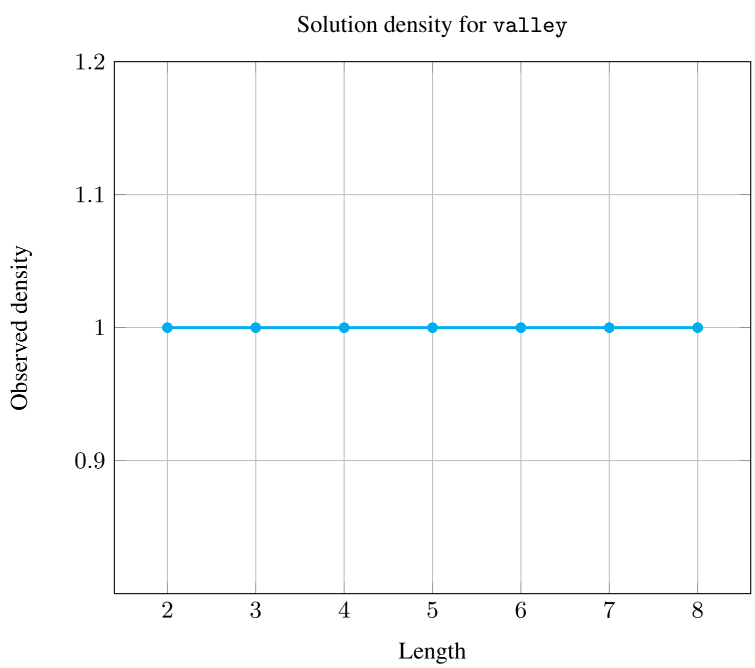
Remark Since the arity of the arc constraint is not fixed, the `valley` constraint cannot be currently described with the graph-based representation. However, this would not hold anymore if we were introducing a slot that specifies how to merge adjacent vertices of the final graph.

Counting

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

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

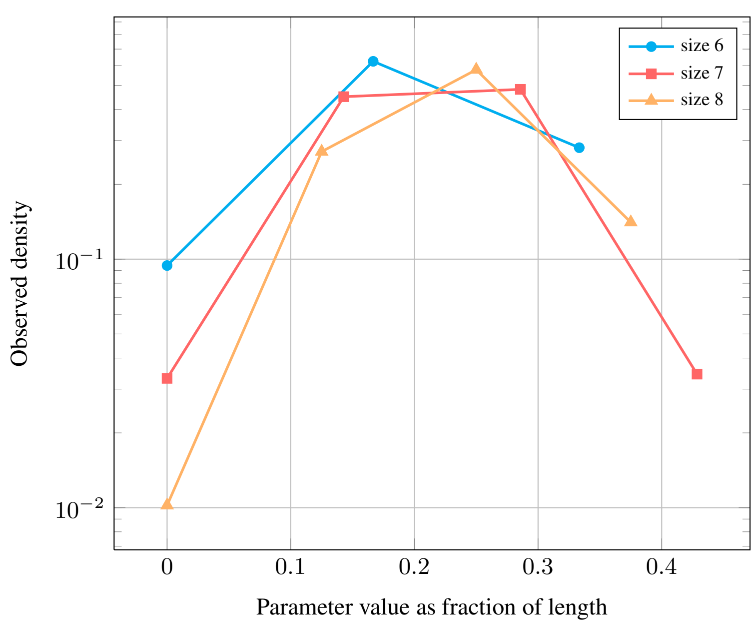

Solution density for valley
Observed density
$10^{0.4}$
$10^{0.2}$
10^{0}
$10^{-0.2}$
$10^{-0.4}$
2 3 4 5 6 7 8
Length

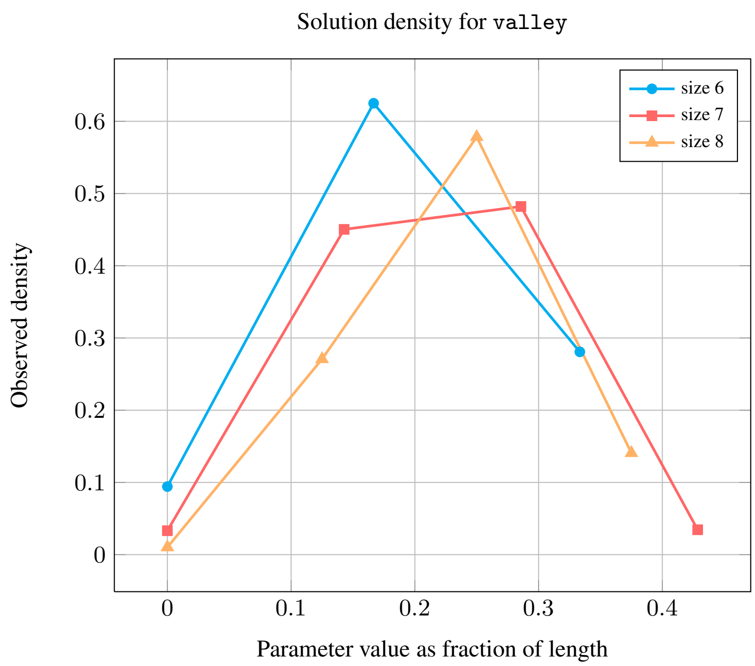

Solution density for valley
Observed density
1.2
1.1
1
0.9
2 3 4 5 6 7 8
Length

Length (n)		2	3	4	5	6	7	8
Total		9	64	625	7776	117649	2097152	43046721
Parameter value	0	9	50	295	1792	11088	69498	439791
	1	-	14	330	5313	73528	944430	11654622
	2	-	-	-	671	33033	1010922	24895038
	3	-	-	-	-	-	72302	6057270

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

Solution density for `valley`





See also

common keyword: deepest_valley, inflexion, min_dist_between_inflexion, min_width_valley *(sequence)*.

comparison swapped: peak.

generalisation: big_valley *(a tolerance parameter is added for counting only big valleys)*.

related: all_equal_valley, all_equal_valley_min, decreasing_valley, increasing_valley, no_peak.

specialisation: no_valley *(the variable counting the number of valleys is set to* 0 *and removed)*.

Keywords

characteristic of a constraint: automaton, automaton with counters, automaton with same input symbol.

combinatorial object: sequence.

constraint arguments: reverse of a constraint, pure functional dependency.

constraint network structure: sliding cyclic(1) constraint network(2).

filtering: glue matrix.

modelling: functional dependency.

Cond. implications

• valley(N, VARIABLES)
with $N > 0$
implies atleast_nvalue(NVAL, VARIABLES)
when $NVAL = 2$.

- `valley`(`N`, `VARIABLES`)
  implies `inflexion`(`N`, `VARIABLES`)
  when `N` =`peak`(`VARIABLES`.`var`) + `valley`(`VARIABLES`.`var`).

Automaton Figure 5.792 depicts the automaton associated with the `valley` constraint. To each pair of consecutive variables $(\texttt{VAR}_i, \texttt{VAR}_{i+1})$ of the collection `VARIABLES` corresponds a signature variable S_i. The following signature constraint links $\texttt{VAR}_i$, $\texttt{VAR}_{i+1}$ and S_i: $(\texttt{VAR}_i < \texttt{VAR}_{i+1} \Leftrightarrow S_i = 0) \wedge (\texttt{VAR}_i = \texttt{VAR}_{i+1} \Leftrightarrow S_i = 1) \wedge (\texttt{VAR}_i > \texttt{VAR}_{i+1} \Leftrightarrow S_i = 2)$.

STATES SEMANTICS

s	: stationary/increasing mode	$(\{<\mid=\}^*)$
u	: decreasing mode	$(> \{>\mid=\}^*)$

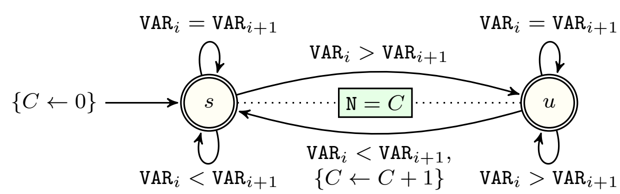


Figure 5.792: Automaton of the `valley` constraint

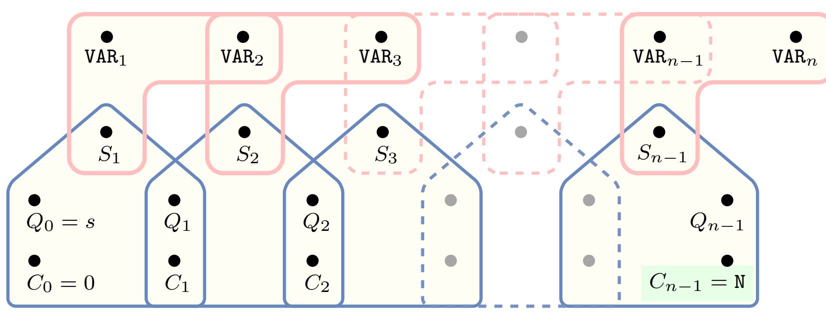


Figure 5.793: Hypergraph of the reformulation corresponding to the automaton of the `valley` constraint (since all states of the automaton are accepting there is no restriction on the last variable Q_{n-1})

Glue matrix where $\overrightarrow{C}$ and $\overleftarrow{C}$ resp. represent the counter value C at the end of a prefix and at the end of the corresponding reverse suffix that partitions the sequence `VARIABLES`.

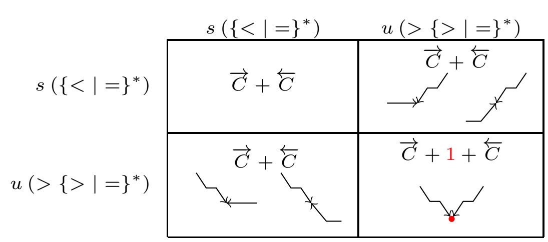


Figure 5.794: Glue matrix of the `valley` constraint

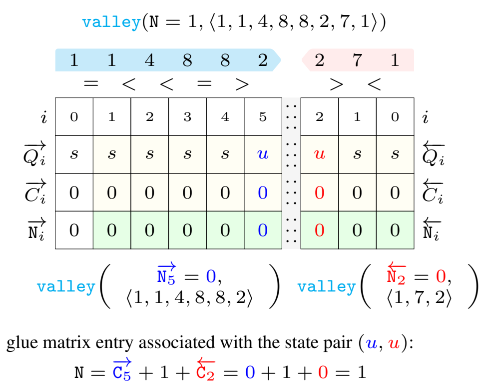


Figure 5.795: Illustrating the use of the state pair (u, u) of the glue matrix for linking N with the counters variables obtained after reading the prefix $1, 1, 4, 8, 8, 2$ and corresponding suffix $2, 7, 1$ of the sequence $1, 1, 4, 8, 8, 2, 7, 1$; note that the suffix $2, 7, 1$ (in pink) is proceed in reverse order; the left (resp. right) table shows the initialisation (for $i = 0$) and the evolution (for $i > 0$) of the state of the automaton and its counter C upon reading the prefix $1, 1, 4, 8, 8, 2$ (resp. the reverse suffix $1, 7, 2$).