5.287 nvalue

DESCRIPTION LINKS GRAPH AUTOMATON

Origin [303]

Constraint nvalue(NVAL, VARIABLES)

Synonyms cardinality_on_attributes_values, values.

Arguments

NVAL	:	dvar
VARIABLES	:	collection(var−dvar)

Restrictions

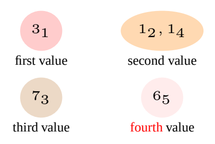
required(VARIABLES, var)
NVAL $\geq$ min(1, |VARIABLES|)
NVAL $\leq$ |VARIABLES|
NVAL $\leq$range(VARIABLES.var)

Purpose NVAL is the number of distinct values taken by the variables of the collection VARIABLES.

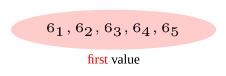
Example

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

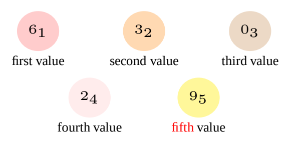
- The first nvalue constraint holds since its first argument NVAL $= 4$ is set to the number of distinct values occurring within the collection $\langle 3, 1, 7, 1, 6\rangle$.



- The second nvalue constraint holds since its first argument NVAL $= 1$ is set to the number of distinct values occurring within the collection $\langle 6, 6, 6, 6, 6\rangle$.



- The third nvalue constraint holds since its first argument NVAL $= 5$ is set to the number of distinct values occurring within the collection $\langle 6, 3, 0, 2, 9\rangle$.



All solutions Figure 5.616 gives all solutions to the following non ground instance of the nvalue constraint: $N \in [1, 2]$, $V_1 \in [2, 4]$, $V_2 \in [1, 2]$, $V_3 \in [2, 4]$, nvalue(N, $\langle V_1, V_2, V_3\rangle$).

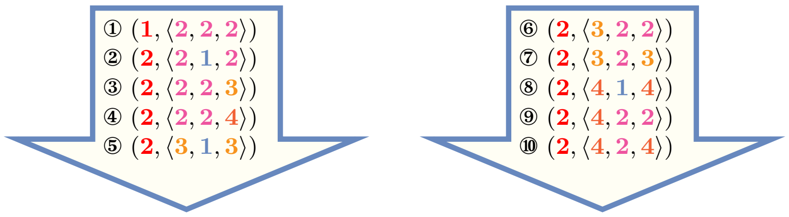


Figure 5.616: All solutions corresponding to the non ground example of the `nvalue` constraint of the **All solutions** slot

Typical

`NVAL` > 1
`NVAL` $<$ |`VARIABLES`|
|`VARIABLES`| > 1

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: `NVAL` determined by `VARIABLES`.
- Contractible wrt. `VARIABLES` when `NVAL` $= 1$ and |`VARIABLES`| > 0.
- Contractible wrt. `VARIABLES` when `NVAL` $=$ |`VARIABLES`|.

Usage

The `nvalue` constraint allows relaxing the `alldifferent` constraint by restricting its first argument `NVAL` to be close, but not necessarily equal, to the number of variables of the `VARIABLES` collection.

A classical example from the early 1850s is the *dominating queens* chess puzzle problem: Place a number of queens on an n by n chessboard in such a way that all cells of the chessboard are either attacked by a queen or are occupied by a queen. A queen can attack all cells located on the same column, on the same row or on the same diagonal. Part (A) of Figure 5.617 illustrates a set of five queens which together attack all of the cells of an 8 by 8 chessboard. The *dominating queens* problem can be modelled by just one `nvalue` constraint:

- We first label the different cells of the chessboard from 1 to n^2.
- We then associate to each cell c of the chessboard a domain variable. Its initial domain is set to the labels of the cells that can attack cell c. For instance, in the context of an 8 by 8 chessboard, the initial domain of V_{29} will be set to {2,5,8,11,13,15,20..22,25..32,36..38,43,45,47,50,53,56,57,61} (see the green cells of part (B) of Figure 5.617).
- Finally, we post the constraint `nvalue`$(Q, \langle$`var` $- V_1,$ `var` $- V_2, \ldots,$ `var` $- V_{n^2}\rangle)$ where Q is a domain variable in $[1, n^2]$ that gives the total number of queens used for controlling all cells of the chessboard. For the solution depicted by Part (A) of Figure 5.617, the label in each cell of Part (C) of Figure 5.617 gives the value assigned to the corresponding variable. Note that, since a given cell can be attacked by several queens, we have also other assignments corresponding to the solution depicted by Part (A) of Figure 5.617.

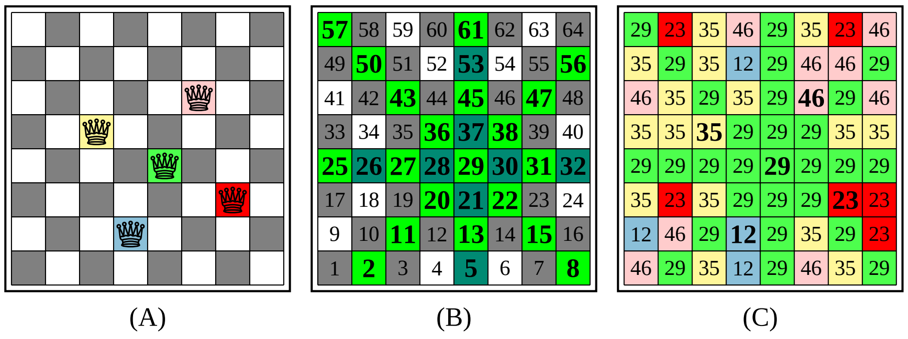


Figure 5.617: Modelling the *dominating queens* problem with a single `nvalue` constraint; (A) a solution to the dominating queens problem, (B) the initial domain (in bold) of the variable associated with cell 29: in a solution the value j assigned to the variable associated with cell i represents the label of the cell attacking cell i (i.e. in a solution one of the selected queens is located on cell j), (C) the value of each cell in the model with one single `nvalue` constraint corresponding to the solution depicted in (A).

The `nvalue` constraint occurs also in many practical applications. In the context of timetabling one wants to set up a limit on the maximum number of activity types it is possible to perform. For frequency allocation problems, one optimisation criterion is to minimise the number of distinct frequencies that you use all over the entire network.

The `nvalue` constraint generalises several constraints like:

- `alldifferent`(`VARIABLES`): in order to get the `alldifferent` constraint, one has to set `NVAL` to the total number of variables.
- `not_all_equal`(`VARIABLES`): in order to get the `not_all_equal` constraint, one has to set the minimum value of `NVAL` to 2.

Remark This constraint appears in [303, page 339] under the name of *Cardinality on Attributes Values*. The `nvalue` constraint is called `values` in **JaCoP** (`http://www.jacop.eu/`). A constraint called `k_diff` enforcing that a set of variables takes at least k distinct values appears in the PhD thesis of J.-C. Régin [341].

It was shown in [69] that, finding out whether a `nvalue` constraint has a solution or not is NP-hard. This was achieved by reduction from 3-SAT. In the same article, it is also shown, by reduction from minimum hitting set cardinality, that computing a sharp lower bound on `NVAL` is NP-hard.

Both reformulations of the `coloured_cumulative` constraint and of the `coloured_cumulatives` constraint use the `nvalue` constraint.

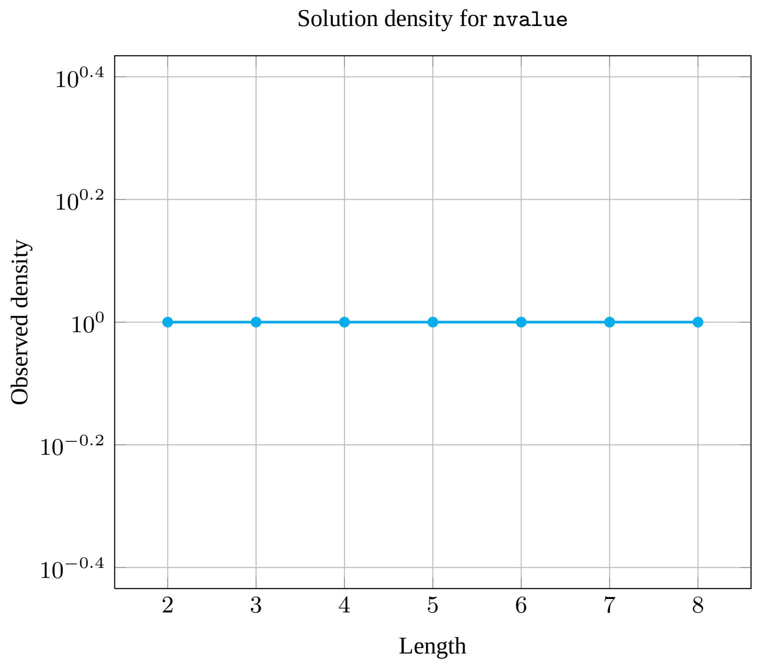
Algorithm A first filtering algorithm for the `nvalue` constraint was described in [27]. Assuming that the minimum value of variable `NVAL` is not constrained at all, two algorithms that both achieve bound-consistency were provided one year later in [40]. Under the same assumption, algorithms that partially take into account holes in the domains of the variables of the `VARIABLES` collection are described in [40, 62].

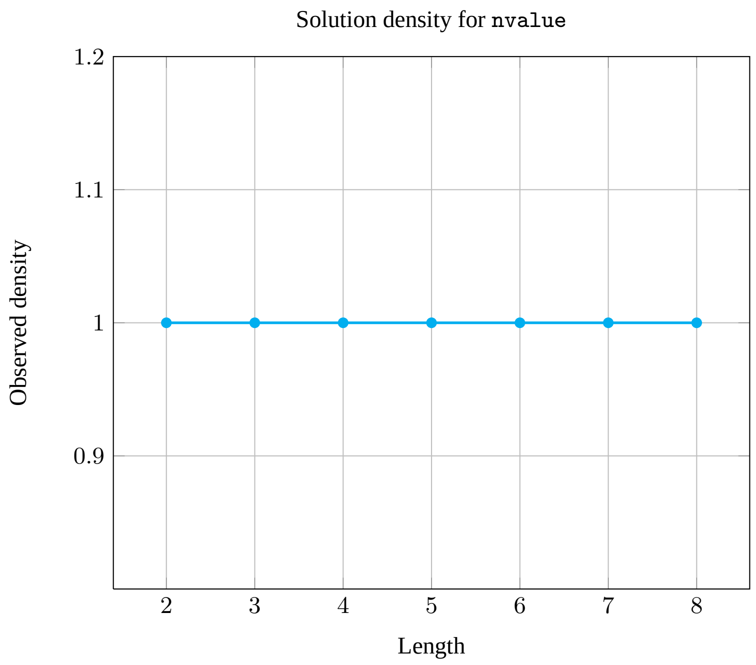
Reformulation A model, involving linear inequalities constraints, preserving bound-consistency was introduced in [72].

Counting

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

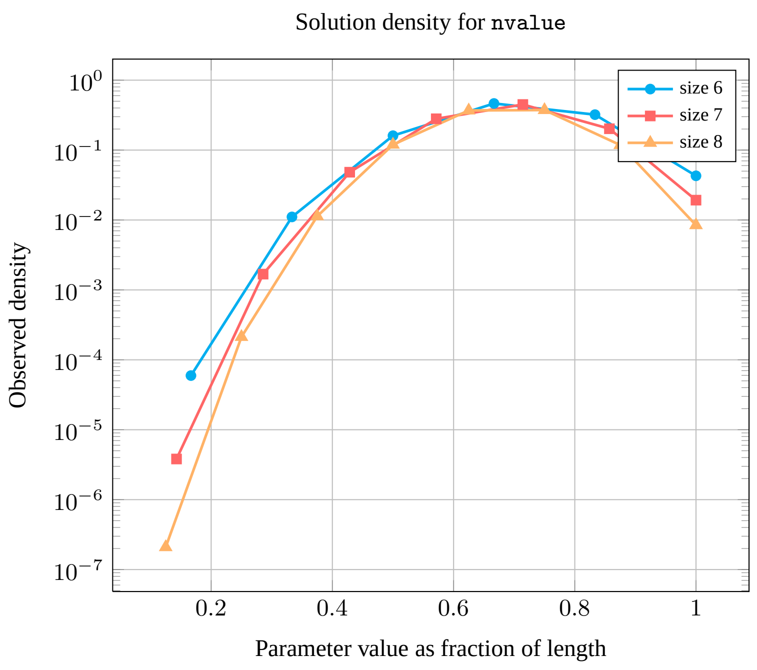
Number of solutions for `nvalue`: domains $0..n$

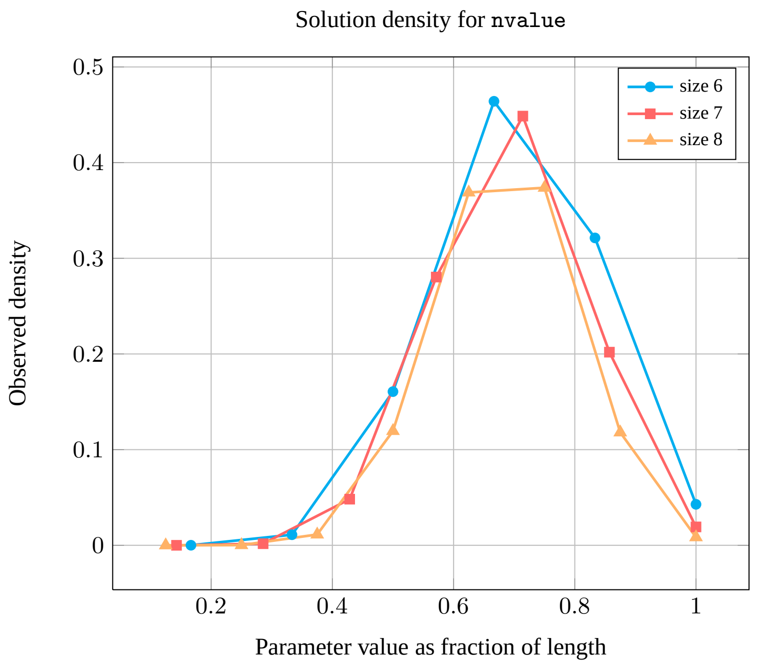




Length (n)		2	3	4	5	6	7	8
Total		9	64	625	7776	117649	2097152	43046721
Parameter value	1	3	4	5	6	7	8	9
	2	6	36	140	450	1302	3528	9144
	3	-	24	360	3000	18900	101136	486864
	4	-	-	120	3600	54600	588000	5143824
	5	-	-	-	720	37800	940800	15876000
	6	-	-	-	-	5040	423360	16087680
	7	-	-	-	-	-	40320	5080320
	8	-	-	-	-	-	-	362880

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





Systems `nvalues` in **Gecode**, `nvalue` in **MiniZinc**, `nvalue` in **SICStus**.

Used in `track`.

See also

assignment dimension added: `assign_and_nvalues`.

common keyword: `among`, `among_diff_0`, `count`, `global_cardinality`, `max_nvalue`, `min_nvalue` *(counting constraint)*, `nvalues_except_0` *(counting constraint,number of distinct values)*.

cost variant: `sum_of_weights_of_distinct_values` *(introduce a weight for each value and replace number of distinct values by sum of weights associated with distinct values)*.

generalisation: `nclass` *(*`variable` *replaced by* `variable` $\in$ `partition`*)*, `nequivalence` *(*`variable` *replaced by* `variable` mod `constant`*)*, `ninterval` *(*`variable` *replaced by* `variable`/`constant`*)*, `npair` *(*`variable` *replaced by* `pair` *of* `variables`*)*, `nvalues` *(replace an equality with the number of distinct values by a comparison with the number of distinct values)*, `nvector` *(variable replaced by vector)*.

implied by: `increasing_nvalue`.

implies: `atleast_nvalue` *(*$=$ `NVAL` *replaced by* $\geq$ `NVAL`*)*, `atmost_nvalue` *(*$=$ `NVAL` *replaced by* $\leq$ `NVAL`*)*.

related: `balance` *(restriction on how balanced an assignment is)*, `coloured_cumulative` *(restrict number of distinct colours on each maximum clique of the interval graph associated with the tasks)*, `coloured_cumulatives` *(restrict number of distinct colours on each maximum clique of the interval graph associated with the tasks assigned to the same machine)*, `increasing_nvalue_chain`, `k_alldifferent` *(necessary condition for two overlapping* `alldifferent` *constraints)*, `soft_alldifferent_var`.

shift of concept: `nvalue_on_intersection`.

soft variant: `nvalues_except_0` *(value* 0 *is ignored)*.

specialisation: `all_equal` *(enforce to have one single value)*, `alldifferent` *(enforce a number of distinct values equal to the number of variables)*, `not_all_equal` *(enforce to have at least two distinct values)*.

uses in its reformulation: `consecutive_values`, `cycle`, `min_n`.

Keywords

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

complexity: 3-SAT, minimum hitting set cardinality.

constraint arguments: pure functional dependency.

constraint type: counting constraint, value partitioning constraint.

filtering: bound-consistency, convex bipartite graph.

final graph structure: strongly connected component, equivalence.

modelling: number of distinct equivalence classes, number of distinct values, functional dependency.

problems: domination.

puzzles: dominating queens.

Cond. implications

`nvalue`(`NVAL`, `VARIABLES`)
  with `increasing`(`VARIABLES`)
 implies `increasing_nvalue`(`NVAL`, `VARIABLES`).

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

Graph model Parts (A) and (B) of Figure 5.618 respectively show the initial and final graph associated with the first example of the **Example** slot. Since we use the **NSCC** graph property we show the different strongly connected components of the final graph. Each strongly connected component corresponds to a value that is assigned to some variables of the `VARIABLES` collection. The 4 following values 1, 3, 6 and 7 are used by the variables of the `VARIABLES` collection.

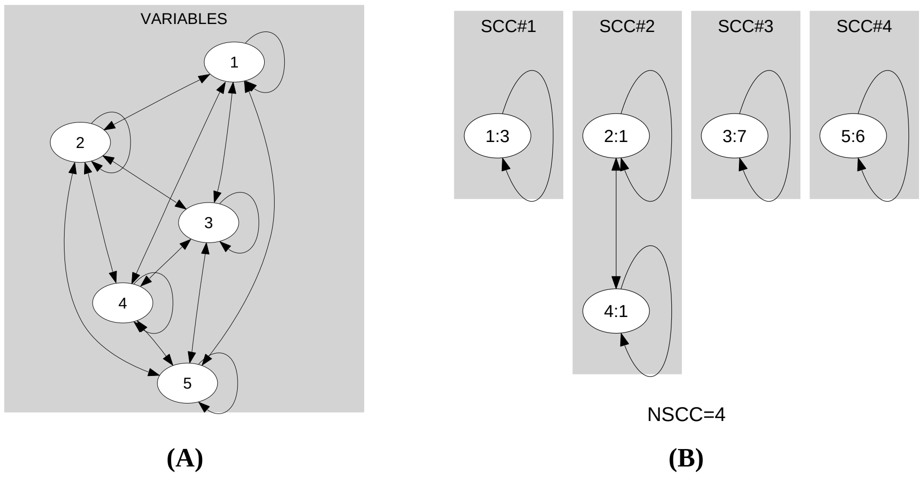


Figure 5.618: Initial and final graph of the `nvalue` constraint

Automaton Figure 5.619 depicts the automaton associated with the `nvalue` constraint. To each item of the collection `VARIABLES` corresponds a signature variable S_i that is equal to 0.

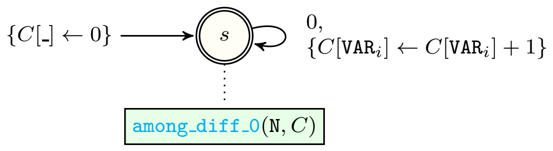


Figure 5.619: Automaton of the `nvalue` constraint

Quiz

EXERCISE 1 (checking whether a ground instance holds or not)[a]

A. Does the constraint `nvalue`$(0, \langle 0,0,0,0\rangle)$ hold?

B. Does the constraint `nvalue`$(3, \langle 1,2,3\rangle)$ hold?

C. Does the constraint `nvalue`$(3, \langle 1,2,3,3\rangle)$ hold?

[a]Hint: go back to the definition of `nvalue`.

EXERCISE 2 (finding all solutions)[a]

Give all the solutions to the constraint:

$$\begin{cases} N \in \{1,5\}, \\ V_1 \in [3,5], \quad V_2 \in [3,4], \quad V_3 \in [2,5], \\ V_4 \in [3,3], \quad V_5 \in [3,4], \quad V_6 \in [3,7], \\ \texttt{nvalue}(N, \langle V_1,V_2,V_3,V_4,V_5,V_6\rangle). \end{cases}$$

[a]Hint: identify the smallest and largest possible values of N, enumerate solutions in lexicographic order.

EXERCISE 3 (identifying infeasible values wrt the at most side)[a]

Identify all variable-value pairs (V_i, val) $(1 \le i \le 6)$, such that the following constraint has no solution when variable V_i is assigned value val:

$$\begin{cases} N \in [0,2], \\ V_1 \in [2,4], \quad V_2 \in [2,5], \quad V_3 \in [4,5], \\ V_4 \in [4,7], \quad V_5 \in [5,8], \quad V_6 \in [6,9], \\ \texttt{nvalue}(N, \langle V_1,V_2,V_3,V_4,V_5,V_6\rangle). \end{cases}$$

[a]Hint: are variables equivalent wrt a given value?

EXERCISE 4 (identifying infeasible variable-value pairs wrt the at least side)[a]

Identify all variable-value pairs (V_i, val) $(1 \leq i \leq 6)$, such that the following constraint has no solution when variable V_i is assigned value val:

$$\begin{cases} N \in [5,6], \\ V_1 \in [2,4], \quad V_2 \in [2,3], \quad V_3 \in [4,5], \\ V_4 \in [2,3], \quad V_5 \in [2,3], \quad V_6 \in [5,6], \\ \texttt{nvalue}(N, \langle V_1, V_2, V_3, V_4, V_5, V_6 \rangle). \end{cases}$$

[a]Hint: find out how to compute the maximum number of distinct values.

EXERCISE 5 (variable-based degree of violation)[a]

Compute the variable-based degree of violation[b] of the following constraints:

A. $\texttt{nvalue}(4, \langle 2, 2, 2, 2 \rangle)$,

B. $\texttt{nvalue}(3, \langle 3, 1, 5, 2, 3 \rangle)$.

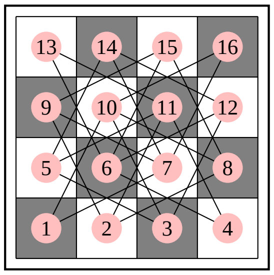
[a]Hint: take advantage of the functional dependency.
[b]Given a constraint for which all variables are fixed, the *variable-based degree of violation* is the minimum number of variables to assign differently in order to satisfy the constraint.

EXERCISE 6 (variations of dominating knights)[a]

A. Provide a model involving only one `nvalue` constraint for showing that the cardinality of the dominating set[b] of the knight graph of a 4 by 4 chessboard does not exceed 4.

B. Show how to modify your model for also considering the fact that each knight must be protected by at least one other knight. Show that the number of required knights does not exceed 6.

[a]Hint: model the knight graph with a set of variables; in a domination problem whats matters for each vertex v is which vertices attack v.
[b]Given a graph G a *dominating set* $\mathcal{D}$ is a subset of the vertices of G such that every vertex of G either belongs to $\mathcal{D}$ or is adjacent to a vertex of G.



4×4 knight graph

SOLUTION TO EXERCISE 1

A. *No, since* $\langle 0,0,0,0\rangle$ *contains just one distinct value (and not* 0 *distinct values as stated by the first argument).*

B. *Yes, since* $\langle 1,2,3\rangle$ *contains* 3 *distinct values as stated by the first argument.*

C. *Yes, since* $\langle 1,2,3,3\rangle$ *contains* 3 *distinct values as stated by the first argument.*

SOLUTION TO EXERCISE 2

the seven solutions

$N, \langle V_1, V_2, V_3, V_4, V_5, V_6\rangle$

① $(1, \langle 3,3,3,3,3,3\rangle)$
② $(5, \langle 5,3,2,3,4,6\rangle)$
③ $(5, \langle 5,3,2,3,4,7\rangle)$
④ $(5, \langle 5,4,2,3,3,6\rangle)$
⑤ $(5, \langle 5,4,2,3,3,7\rangle)$
⑥ $(5, \langle 5,4,2,3,4,6\rangle)$
⑦ $(5, \langle 5,4,2,3,4,7\rangle)$

A. *Value* 3 *being the only common value to variables* V_1, V_2, V_3, V_4, V_5, V_6, *we get a single solution where* N *is set to* 1, *i.e. solution* ①.

B. *A matching of cardinality* 5 *is given by* $V_1 = 5$, $V_2 = 3$, $V_3 = 2$, $V_4 = 3$, $V_5 = 4$, $V_6 = 6$. *It is maximum since variables* V_1, V_2, V_3, V_4, V_5 *have to be assigned one of the four values* 2, 3, 4 *and* 5, *and since values* 6 *and* 7 *can only be assigned to variable* V_6. *In any maximum matching we have that:*

(a) *Since variable* V_6 *is the only variable that can be assigned values* 6 *or* 7, *we have* $V_6 = 6$ *or* $V_6 = 7$.

(b) *Since variable* V_3 *is the only variable that can be assigned value* 2, *we have* $V_3 = 2$.

(c) *Now that* V_3 *is assigned value* 2 *and that* V_6 *is assigned values* 6 *or* 7, *variable* V_1 *is the only variable that can be assigned value* 5, *we have* $V_1 = 5$.

Finally combining the fact that variables V_2, V_5 *have to be assigned a distinct value in* $\{3,4\}$ *and variable* V_6 *a value in* $\{6,7\}$ *we obtain the remaining six solutions* ②, ③, ④, ⑤, ⑥, ⑦.

SOLUTION TO EXERCISE 3

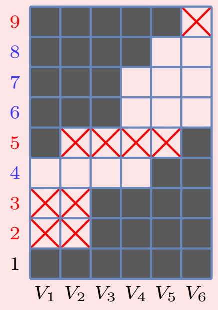
The constraint forces that at most two distinct values are assigned to variables $V_1, V_2, \ldots, V_6$, i.e. there is no restriction coming from N on the minimum number of distinct values. In this context, a value val assigned to one of the variables V_i $(1 \leq i \leq 6)$ can be assigned to any other variable V_i without increasing the number of distinct values. Consequently a value val that is not removed (resp. removed) from a variable V_i $(1 \leq i \leq 6)$ can also not be removed (resp. removed) from a variable V_j $(j \neq i, 1 \leq j \leq 6)$. Let us successively study the values that can not be removed and the values that can be removed from $V_1, V_2, \ldots, V_6$.

A. [*FEASIBLE VALUES*]

Consider the three solutions

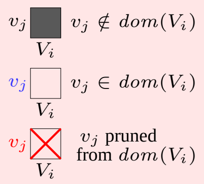
nvalue$(2, \langle 4, 4, 4, 4, 6, 6\rangle)$,
nvalue$(2, \langle 4, 4, 4, 4, 7, 7\rangle)$,
nvalue$(2, \langle 4, 4, 4, 4, 8, 8\rangle)$.

All values used in the previous solutions (i.e., values 4, 6, 7, 8) can not be removed from $V_1, V_2, \ldots, V_6$.



B. [*INFEASIBLE VALUES*]

We now show that 2 cannot be assigned to any variable. If 2 can be used then we assign 2 to all variables that have 2 in their domains, i.e., V_1 and V_2. Now in order not to exceed two distinct values, the remaining variables V_3, V_4, V_5, V_6 must have a value in common, which is not the case. We can show in the same way that values 3, 5 and 9 cannot be assigned to any variable.



v_j V_i $v_j \notin dom(V_i)$
v_j V_i $v_j \in dom(V_i)$
v_j V_i v_j pruned from $dom(V_i)$

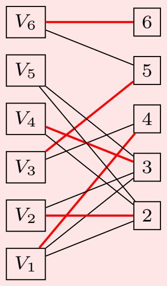
Finally since $V_1, V_2, \ldots, V_6$ do not have any value in common, N can only be equal to 2.

SOLUTION TO EXERCISE 4

The constraint forces that at least five distinct values are assigned to variables $V_1, V_2, \ldots, V_6$, i.e. there is no restriction coming from N on the maximum number of distinct values. Consequently identifying infeasible variable-value pairs is equivalent to finding edges that do not belong to any matching[a] of cardinality greater than or equal to the minimum value of N in the variable-value bipartite graph $\mathcal{G}(V, E)$ associated with the `nvalue` *constraint (the vertices V of $\mathcal{G}$ are defined by the variables $V_1, V_2, \ldots, V_6$ and by the values $2, 3, \ldots, 6$, while the edges E are defined by the pairs (V_i, val), $(1 \leq i \leq 6)$ such that $val \in dom(V_i)$).*

A. [*MAXIMUM MATCHING*]

The solution nvalue$(5, \langle 4, 2, 5, 3, 2, 6\rangle)$ corresponds to a matching of cardinality 5 shown in red on the variable-value graph. This matching is maximum since $|dom(V_1) \cup dom(V_2) \cup \cdots \cup dom(V_6)| = 5$. Therefore N can only be equal to 5.

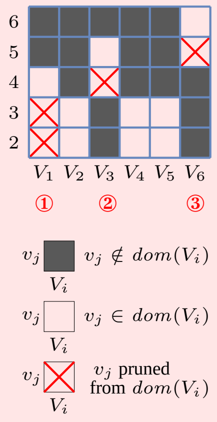


B. [*INFEASIBLE EDGES*]

① *Since $|dom(V_2) \cup dom(V_4)| = 2$, V_1 must be assigned value 4 in any maximum matching.*

② *Since V_1 must be assigned value 4 in any maximum matching and since $dom(V_3) = \{4, 5\}$, V_3 must be assigned value 5 in any maximum matching.*

③ *Since V_3 must be assigned value 5 in any maximum matching and since $dom(V_6) = \{5, 6\}$, V_6 must be assigned value 6 in any maximum matching.*


6
5
4
3
2
V_1 V_2 V_3 V_4 V_5 V_6
①
②
③
v_j V_i $v_j \notin dom(V_i)$
v_j V_i $v_j \in dom(V_i)$
v_j V_i v_j pruned from $dom(V_i)$

C. *Finally, V_2, V_4, V_5 must be assigned two distinct values from $\{2, 3\}$ in any maximum matching.*

[a] A *matching* of a graph $\mathcal{G}$ is a set of edges of $\mathcal{G}$ such that no two edges have a vertex in common.

SOLUTION TO EXERCISE 5

For a violated `nvalue` *constraint it is always possible to change a single variable to get a feasible solution. This is done by setting the first argument of the* `nvalue` *constraint to the number of distinct values occurring in the second argument.*

A. *The degree of violation is equal to* 1 *since the first argument needs to be set to* 1 *in order to obtain a solution.*

$$\texttt{nvalue}(\overset{1}{\sqcap}\!\!\!\!\!\mathbf{4}, \langle 2, 2, 2, 2\rangle)$$

B. *The degree of violation is equal to* 1 *since the first argument needs to be set to* 4 *in order to obtain a solution.*

$$\texttt{nvalue}(\overset{4}{\mathbf{3}}, \langle 3, 1, 5, 2, 3\rangle)$$

Note that in this example we have other possibilities such as

$$\texttt{nvalue}(3, \langle 3, 1, \overset{1}{\mathbf{5}}, 2, 3\rangle)$$

SOLUTION TO EXERCISE 6

A. *Each vertex of the* 4×4 *knight graph is represented by a variable whose domain is set to the labels of its adjacent vertices as well as to its own label. Consequently we get the following* 16 *variables with their corresponding initial domains:*

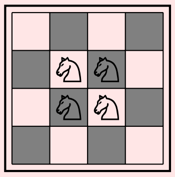
$V_1 \in \{1,7,10\}$, $V_2 \in \{2,8,9,11\}$, $V_3 \in \{3,5,10,12\}$,
$V_4 \in \{4,6,11\}$, $V_5 \in \{3,5,11,14\}$, $V_6 \in \{4,6,12,13,15\}$,
$V_7 \in \{1,7,9,14,16\}$, $V_8 \in \{2,8,10,15\}$, $V_9 \in \{2,7,9,15\}$,
$V_{10} \in \{1,3,8,10,16\}$, $V_{11} \in \{2,4,5,11,13\}$, $V_{12} \in \{3,6,12,14\}$,
$V_{13} \in \{6,11,13\}$, $V_{14} \in \{5,7,12,14\}$, $V_{15} \in \{6,8,9,15\}$,
$V_{16} \in \{7,10,16\}$.

We introduce a variable $N \in \{1,2,3,4\}$ *that provides the number of knights actually used and state the following constraint:*

$$\text{nvalue}(N, \langle V_1,V_2,V_3,V_4,V_5,V_6,V_7,V_8,V_9,\\ V_{10},V_{11},V_{12},V_{13},V_{14},V_{15},V_{16}\rangle)$$

In the previous constraint the assignment $V_i = j$ *means that a knight is located on vertex* j *and that vertex* j *attacks vertex* i. *Consequently the total number of distinct values in* $\langle V_1, V_2, \dots, V_{16}\rangle$ *is equal to the total number of used knights. The assignment*

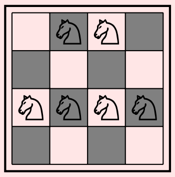
$N = 4$,
$V_1 = 7$, $V_2 = 11$, $V_3 = 10$, $V_4 = 6$,
$V_5 = 11$, $V_6 = 6$, $V_7 = 7$, $V_8 = 10$,
$V_9 = 7$, $V_{10} = 10$, $V_{11} = 11$, $V_{12} = 6$,
$V_{13} = 6$, $V_{14} = 7$, $V_{15} = 6$, $V_{16} = 7$



corresponds to the solution depicted on the right.

B. *Since a knight cannot protect itself, we only need to remove from the initial domain of each variable the label corresponding to its cell. The assignment*

$N = 6$,
$V_1 = 7$, $V_2 = 8$, $V_3 = 5$, $V_4 = 6$,
$V_5 = 14$, $V_6 = 15$, $V_7 = 14$, $V_8 = 15$,
$V_9 = 7$, $V_{10} = 8$, $V_{11} = 5$, $V_{12} = 6$,
$V_{13} = 6$, $V_{14} = 5$, $V_{15} = 6$, $V_{16} = 7$



corresponds to the solution depicted on the right.