

5.60 coloured_cumulative

	DESCRIPTION	LINKS	GRAPH
Origin	Derived from <code>cumulative</code> and <code>nvalues</code> .		
Constraint	<code>coloured_cumulative(TASKS, LIMIT)</code>		
Synonym	<code>colored_cumulative</code> .		
Arguments	$\text{TASKS} : \text{collection} \left(\begin{array}{l} \text{origin-dvar,} \\ \text{duration-dvar,} \\ \text{end-dvar,} \\ \text{colour-dvar} \end{array} \right)$ $\text{LIMIT} : \text{int}$		
Restrictions	<code>require_at_least(2, TASKS, [origin, duration, end])</code> <code>required(TASKS, colour)</code> <code>TASKS.duration ≥ 0</code> <code>TASKS.origin ≤ TASKS.end</code> <code>LIMIT ≥ 0</code>		
Purpose	<p>Consider the set \mathcal{T} of tasks described by the <code>TASKS</code> collection. The <code>coloured_cumulative</code> constraint enforces that, at each point in time, the number of distinct colours of the set of tasks that overlap that point, does not exceed a given limit. A task overlaps a point i if and only if (1) its origin is less than or equal to i, and (2) its end is strictly greater than i. For each task of \mathcal{T} it also imposes the constraint <code>origin + duration = end</code>.</p>		
Example	$\left(\left\langle \begin{array}{llll} \text{origin} - 1 & \text{duration} - 2 & \text{end} - 3 & \text{colour} - 1, \\ \text{origin} - 2 & \text{duration} - 9 & \text{end} - 11 & \text{colour} - 2, \\ \text{origin} - 3 & \text{duration} - 10 & \text{end} - 13 & \text{colour} - 3, \\ \text{origin} - 6 & \text{duration} - 6 & \text{end} - 12 & \text{colour} - 2, \\ \text{origin} - 7 & \text{duration} - 2 & \text{end} - 9 & \text{colour} - 3 \end{array} \right\rangle, 2 \right)$		
	<p>Figure 5.129 shows the solution associated with the example. Each rectangle of the figure corresponds to a task of the <code>coloured_cumulative</code> constraint. Tasks that have their colour attribute set to 1, 2 and 3 are respectively coloured in yellow, blue and pink. The <code>coloured_cumulative</code> constraint holds since at each point in time we do not have more than <code>LIMIT = 2</code> distinct colours.</p>		
Typical	<code> TASKS > 1</code> <code>range(TASKS.origin) > 1</code> <code>range(TASKS.duration) > 1</code> <code>range(TASKS.end) > 1</code> <code>range(TASKS.colour) > 1</code> <code>LIMIT <nval(TASKS.colour)</code>		

Symmetries

- Items of TASKS are [permutable](#).
- One and the same constant can be [added](#) to the `origin` and `end` attributes of all items of TASKS.
- All occurrences of two distinct values of `TASKS.colour` can be [swapped](#); all occurrences of a value of `TASKS.colour` can be [renamed](#) to any unused value.
- `LIMIT` can be [increased](#).

Usage

Useful for scheduling problems where a machine can only proceed in parallel a maximum number of tasks of distinct type. This condition cannot be modelled by the classical [cumulative](#) constraint.

Reformulation

The `coloured_cumulative` constraint can be expressed in term of a set of reified constraints and of `|TASKS| nvalue` constraints:

1. For each pair of tasks `TASKS[i]`, `TASKS[j]` ($i, j \in [1, |TASKS|]$) of the `TASKS` collection we create a variable C_{ij} which is set to the colour of task `TASKS[j]` if task `TASKS[j]` overlaps the origin attribute of task `TASKS[i]`, and to the colour of task `TASKS[i]` otherwise:
 - If $i = j$:
 - $C_{ij} = \text{TASKS}[i].\text{colour}$.
 - If $i \neq j$:
 - $C_{ij} = \text{TASKS}[i].\text{colour} \vee C_{ij} = \text{TASKS}[j].\text{colour}$.
 - $((\text{TASKS}[j].\text{origin} \leq \text{TASKS}[i].\text{origin} \wedge \text{TASKS}[j].\text{end} > \text{TASKS}[i].\text{origin}) \wedge (C_{ij} = \text{TASKS}[j].\text{colour})) \vee ((\text{TASKS}[j].\text{origin} > \text{TASKS}[i].\text{origin} \vee \text{TASKS}[j].\text{end} \leq \text{TASKS}[i].\text{origin}) \wedge (C_{ij} = \text{TASKS}[i].\text{colour}))$
2. For each task `TASKS[i]` ($i \in [1, |TASKS|]$) we create a variable N_i which gives the number of distinct colours associated to the tasks that overlap the origin of task `TASKS[i]` (`TASKS[i]` overlaps its own origin) and we impose N_i to not exceed the maximum number of distinct colours `LIMIT` allowed at each instant:
 - $N_i \geq 1 \wedge N_i \leq \text{LIMIT}$.
 - `nvalue`($N_i, \langle C_{i1}, C_{i2}, \dots, C_{i|TASKS|} \rangle$).

See also

[assignment dimension added: coloured_cumulatives](#).

common keyword: `cumulative`, `track(resource constraint)`.

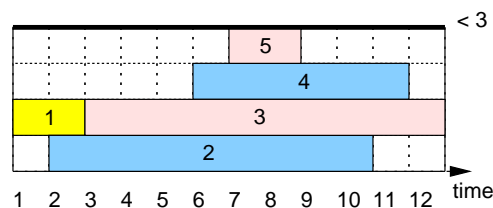


Figure 5.129: A coloured cumulative solution with at most two distinct colours in parallel

related: *nvalue*.

specialisation: *disjoint_tasks* (a colour is assigned to each collection of tasks of constraint *disjoint_tasks* and a limit of one single colour is enforced).

used in graph description: *nvalues*.

Keywords

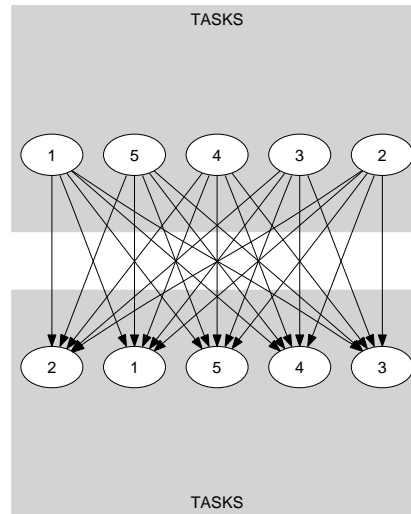
characteristic of a constraint: *coloured*.

constraint type: *scheduling constraint*, *resource constraint*, *temporal constraint*.

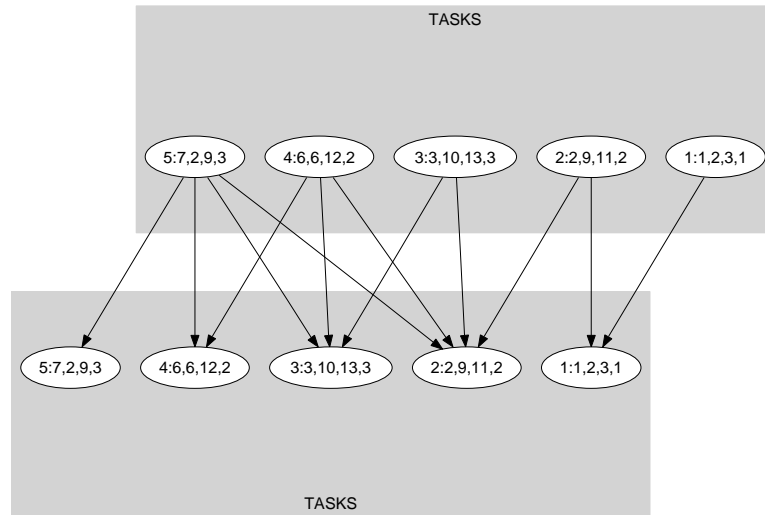
filtering: *compulsory part*.

modelling: *number of distinct values*, *zero-duration task*.

Arc input(s)	TASKS
Arc generator	$SELF \mapsto \text{collection}(\text{tasks})$
Arc arity	1
Arc constraint(s)	$\text{tasks.origin} + \text{tasks.duration} = \text{tasks.end}$
Graph property(ies)	$\overline{\text{NARC}} = \text{TASKS} $
<hr/>	
Arc input(s)	TASKS TASKS
Arc generator	$PRODUCT \mapsto \text{collection}(\text{tasks1}, \text{tasks2})$
Arc arity	2
Arc constraint(s)	<ul style="list-style-type: none"> • $\text{tasks1.duration} > 0$ • $\text{tasks2.origin} \leq \text{tasks1.origin}$ • $\text{tasks1.origin} < \text{tasks2.end}$
Graph class	<ul style="list-style-type: none"> • ACYCLIC • BIPARTITE • NO_LOOP
Sets	$SUC \mapsto \left[\begin{array}{l} \text{source}, \\ \text{variables} - \text{col} \left(\begin{array}{l} \text{VARIABLES} - \text{collection}(\text{var} - \text{dvar}), \\ [\text{item}(\text{var} - \text{TASKS.colour})] \end{array} \right) \end{array} \right]$
Constraint(s) on sets	$n\text{values}(\text{variables}, \leq, \text{LIMIT})$
<hr/>	
Graph model	<p>Same as cumulative, except that we use another constraint for computing the resource consumption at each time point.</p> <p>Parts (A) and (B) of Figure 5.130 respectively show the initial and final graph associated with the second graph constraint of the Example slot. On the one hand, each source vertex of the final graph can be interpreted as a time point. On the other hand the successors of a source vertex correspond to those tasks that overlap that time point. The coloured_cumulative constraint holds since for each successor set \mathcal{S} of the final graph the number of distinct colours of the tasks in \mathcal{S} does not exceed the LIMIT 2.</p>
Signature	<p>Since TASKS is the maximum number of vertices of the final graph of the first graph constraint we can rewrite $\overline{\text{NARC}} = \text{TASKS}$ to $\overline{\text{NARC}} \geq \text{TASKS}$. This leads to simplify $\overline{\text{NARC}}$ to $\overline{\text{NARC}}$.</p>



(A)



(B)

Figure 5.130: Initial and final graph of the coloured_cumulative constraint

20000128

631