

5.61 coloured_cumulatives

	DESCRIPTION	LINKS	GRAPH
Origin	Derived from <code>cumulatives</code> and <code>nvalues</code> .		
Constraint	<code>coloured_cumulatives(TASKS, MACHINES)</code>		
Synonym	<code>colored_cumulatives</code> .		
Arguments	$\begin{array}{l} \text{TASKS} \quad : \quad \text{collection} \left(\begin{array}{l} \text{machine-dvar,} \\ \text{origin-dvar,} \\ \text{duration-dvar,} \\ \text{end-dvar,} \\ \text{colour-dvar} \end{array} \right) \\ \text{MACHINES} \quad : \quad \text{collection}(\text{id-int, capacity-int}) \end{array}$		
Restrictions	<pre>required(TASKS, [machine, colour]) require_at_least(2, TASKS, [origin, duration, end]) TASKS.duration ≥ 0 TASKS.origin ≤ TASKS.end required(MACHINES, [id, capacity]) distinct(MACHINES, id) MACHINES.capacity ≥ 0</pre>		
Purpose	<p>Consider a set \mathcal{T} of tasks described by the <code>TASKS</code> collection. The <code>coloured_cumulatives</code> constraint enforces for each machine m of the <code>MACHINES</code> collection the following condition: at each point in time p, the numbers of distinct colours of the set of tasks that both overlap that point p and are assigned to machine m does not exceed the capacity of machine m. 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. It also imposes for each task of \mathcal{T} the constraint <code>origin + duration = end</code>.</p>		
Example	$\left(\begin{array}{l} \text{machine-1 origin-6 duration-6 end-12 colour-1,} \\ \text{machine-1 origin-2 duration-9 end-11 colour-2,} \\ \text{machine-2 origin-7 duration-3 end-10 colour-2,} \\ \text{machine-1 origin-1 duration-2 end-3 colour-1,} \\ \text{machine-2 origin-4 duration-5 end-9 colour-2,} \\ \text{machine-1 origin-3 duration-10 end-13 colour-1} \\ \langle \text{id-1 capacity-2, id-2 capacity-1} \rangle \end{array} \right)$		

Figure 5.131 shows the solution associated with the example. Each rectangle of the figure corresponds to a task of the `coloured_cumulatives` constraint. Tasks that have their colour attribute set to 1 and 2 are respectively coloured in blue and pink. The `coloured_cumulatives` constraint holds since for machine 1 we have at most two distinct colours in parallel (which is the maximum capacity for machine 1), while for machine 2 we have no more than one single colour in parallel (which is actually the maximum capacity for machine 2).

Typical

```

|TASKS| > 1
range(TASKS.machine) > 1
range(TASKS.origin) > 1
range(TASKS.duration) > 1
range(TASKS.end) > 1
range(TASKS.colour) > 1
TASKS.duration > 0
|MACHINES| > 1
MACHINES.capacity > 0
MACHINES.capacity <nval(TASKS.colour)
|TASKS| > |MACHINES|

```

Symmetries

- Items of TASKS are [permutable](#).
- Items of MACHINES are [permutable](#).
- MACHINES.capacity can be [increased](#).
- All occurrences of two distinct values in TASKS.machine or MACHINES.id can be [swapped](#); all occurrences of a value in TASKS.machine or MACHINES.id can be [renamed](#) to any unused value.

Usage

Useful for scheduling problems where several machines are available and where you have to assign each task to a specific machine. In addition each machine can only proceed in parallel a maximum number of tasks of distinct types.

Reformulation

The `coloured_cumulatives` 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 both tasks are assigned to the same machine and if task $TASKS[j]$ overlaps the origin attribute of task $TASKS[i]$, and to the colour of task $TASKS[i]$ otherwise:
 - If $i = j$:
 - $C_{ij} = TASKS[i].colour$.
 - If $i \neq j$:
 - $C_{ij} = TASKS[i].colour \vee C_{ij} = TASKS[j].colour$.

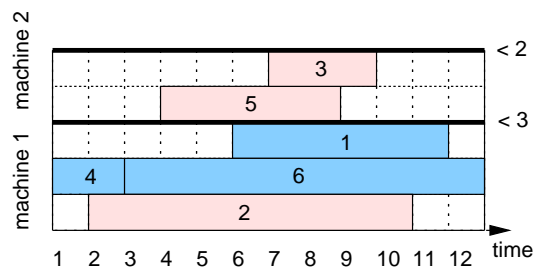


Figure 5.131: Assignment of the tasks on the two machines

$$\begin{aligned}
& - ((\text{TASKS}[j].\text{machine} = \text{TASKS}[i].\text{machine} \wedge \\
& \quad \text{TASKS}[j].\text{origin} \leq \text{TASKS}[i].\text{origin} \wedge \\
& \quad \text{TASKS}[j].\text{end} > \text{TASKS}[i].\text{origin}) \wedge (C_{ij} = \text{TASKS}[j].\text{colour})) \vee \\
& ((\text{TASKS}[j].\text{machine} \neq \text{TASKS}[i].\text{machine} \vee \\
& \quad \text{TASKS}[j].\text{origin} > \text{TASKS}[i].\text{origin} \vee \\
& \quad \text{TASKS}[j].\text{end} \leq \text{TASKS}[i].\text{origin}) \wedge (C_{ij} = \text{TASKS}[i].\text{colour}))
\end{aligned}$$

2. For each task $\text{TASKS}[i]$ ($i \in [1, |\text{TASKS}|]$) we create a variable N_i which gives the number of distinct colours associated to the tasks that both are assigned to the same machine as task $\text{TASKS}[i]$ and overlap the origin of task $\text{TASKS}[i]$ ($\text{TASKS}[i]$ overlaps its own origin) and we impose N_i to not exceed the maximum number of distinct colours LIMIT allowed at each instant:

- $N_i \geq 1 \wedge N_i \leq \text{LIMIT}$.
- $\text{nvalue}(N_i, \langle C_{i1}, C_{i2}, \dots, C_{i|\text{TASKS}|} \rangle)$.

See also

assignment dimension removed: `coloured_cumulative` (machine attribute removed), `cumulative` (machine attribute removed and number of distinct colours replaced by sum of task heights).

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

related: `nvalue`.

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`, `assignment dimension`, `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)	$\underline{NARC} = \text{TASKS} $

For all items of MACHINES:

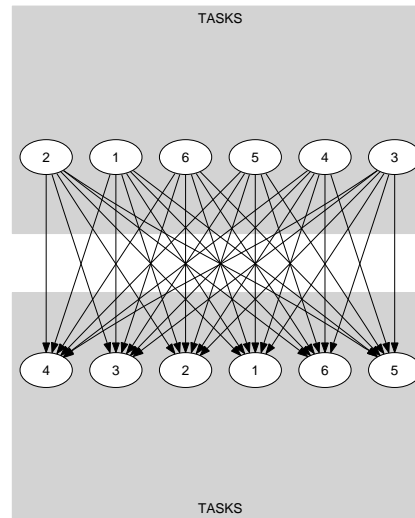
Arc input(s)	TASKS TASKS
Arc generator	$PRODUCT \mapsto \text{collection}(\text{tasks1}, \text{tasks2})$
Arc arity	2
Arc constraint(s)	<ul style="list-style-type: none"> • $\text{tasks1.machine} = \text{MACHINES.id}$ • $\text{tasks1.machine} = \text{tasks2.machine}$ • $\text{tasks1.duration} > 0$ • $\text{tasks2.origin} \leq \text{tasks1.origin}$ • $\text{tasks1.origin} < \text{tasks2.end}$
Graph class	<ul style="list-style-type: none"> • ACYCLIC • BIPARTITE • NO_LOOP
Sets	$SUCC \mapsto \left[\begin{array}{l} \text{source}, \\ \text{variables} - \text{col} \left(\begin{array}{l} \text{VARIABLES} - \text{collection}(\text{var} - \text{dvar}), \\ [\text{item}(\text{var} - \text{TASKS.colour})] \end{array} \right) \end{array} \right]$
Constraint(s) on sets	$\underline{nvalues}(\text{variables}, \leq, \text{MACHINES.capacity})$

Graph model

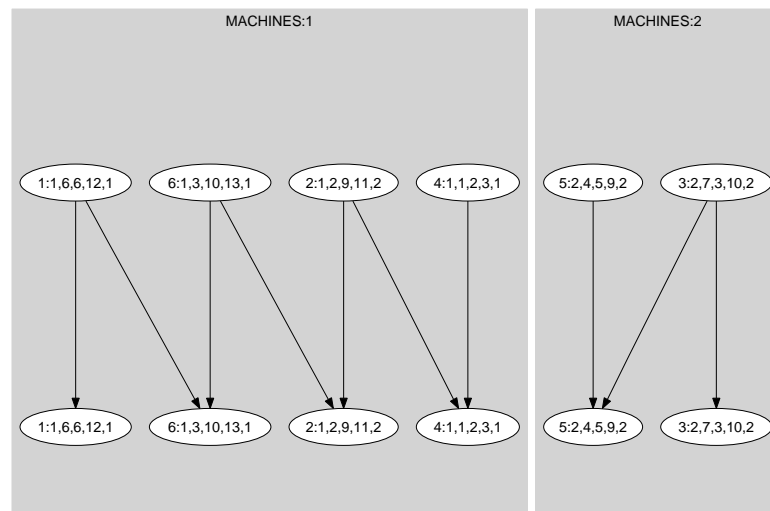
Parts (A) and (B) of Figure 5.132 respectively shows the initial and final graph associated with machines 1 and 2 involved in the **Example** slot. On the one hand, each source vertex of the final graph can be interpreted as a time point p on a specific machine m . On the other hand the successors of a source vertex correspond to those tasks that both overlap that time point p and are assigned to machine m . The `coloured_cumulatives` constraint holds since for each successor set \mathcal{S} of the final graph the number of distinct colours in \mathcal{S} does not exceed the capacity of the machine corresponding to the time point associated with \mathcal{S} .

Signature

Since TASKS is the maximum number of vertices of the final graph of the first graph constraint we can rewrite $\underline{NARC} = |\text{TASKS}|$ to $\underline{NARC} \geq |\text{TASKS}|$. This leads to simplify \underline{NARC} to \overline{NARC} .



(A)



(B)

Figure 5.132: Initial and final graph of the coloured_cumulatives constraint

20000128

637