5.98 cumulative_product

DESCRIPTION LINKS GRAPH

Origin

Derived from cumulative.

Constraint

cumulative_product(TASKS, LIMIT)

Arguments

$$\begin{array}{lll} \texttt{TASKS} & : & \texttt{collection}\left(\begin{array}{l}\texttt{origin}-\texttt{dvar},\\ \texttt{duration}-\texttt{dvar},\\ \texttt{end}-\texttt{dvar},\\ \texttt{height}-\texttt{dvar}\end{array}\right) \\ \texttt{LIMIT} & : & \texttt{int} \end{array}$$

Restrictions

require_at_least(2, TASKS, [origin, duration, end])
required(TASKS, height)
TASKS.duration ≥ 0
TASKS.origin $\leq$ TASKS.end
TASKS.height ≥ 1
LIMIT ≥ 0

Purpose

Consider a set $\mathcal{T}$ of tasks described by the TASKS collection. The cumulative_product constraint forces that at each point in time, the product of the heights 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. It also imposes for each task of $\mathcal{T}$ the constraint origin + duration = end.

Example

$$\left(\left\langle\begin{array}{llll}\texttt{origin}-1 & \texttt{duration}-3 & \texttt{end}-4 & \texttt{height}-1,\\ \texttt{origin}-2 & \texttt{duration}-9 & \texttt{end}-11 & \texttt{height}-2,\\ \texttt{origin}-3 & \texttt{duration}-10 & \texttt{end}-13 & \texttt{height}-1,\\ \texttt{origin}-6 & \texttt{duration}-6 & \texttt{end}-12 & \texttt{height}-1,\\ \texttt{origin}-7 & \texttt{duration}-2 & \texttt{end}-9 & \texttt{height}-3\end{array}\right\rangle, 6\right)$$

Figure 5.220 shows the solution associated with the example. To each task of the cumulative_product constraint corresponds a set of rectangles coloured with the same colour: the sum of the lengths of the rectangles corresponds to the duration of the task, while the height of the rectangles (i.e., all the rectangles associated with a task have the same height) corresponds to the height of the task. The profile corresponding to the product of the heights of the tasks that overlap a given point is depicted by a thick red line. The cumulative_product constraint holds since at each point in time the product of the heights of the tasks that overlap that point is not strictly greater than the upper limit 6 enforced by the last argument of the cumulative_product constraint.

TASKS

①	`o` − 1	`d` − 3	`e` − 4	`h` − 1
②	`o` − 2	`d` − 9	`e` − 11	`h` − 2
③	`o` − 3	`d` − 10	`e` − 13	`h` − 1
④	`o` − 6	`d` − 6	`e` − 12	`h` − 1
⑤	`o` − 7	`d` − 2	`e` − 9	`h` − 3

(`o` for `origin`, `d` for `duration`, `e` for `end`, `h` for `height`)

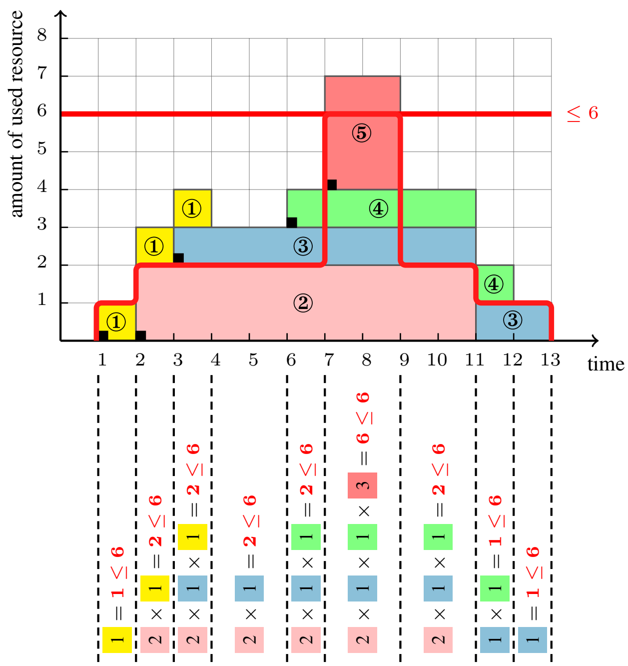


Figure 5.220: Resource consumption profile in red corresponding to the product of the heights of the five tasks of the **Example** slot

Typical

$|\texttt{TASKS}| > 1$
$\texttt{range}(\texttt{TASKS.origin}) > 1$
$\texttt{range}(\texttt{TASKS.duration}) > 1$
$\texttt{range}(\texttt{TASKS.end}) > 1$
$\texttt{range}(\texttt{TASKS.height}) > 1$
$\texttt{TASKS.duration} > 0$
$\texttt{LIMIT} < \texttt{prod}(\texttt{TASKS.height})$

Symmetries

- Items of `TASKS` are permutable.
- `TASKS.height` can be decreased to any value ≥ 0.
- One and the same constant can be added to the `origin` and `end` attributes of all items of `TASKS`.
- `LIMIT` can be increased.

Arg. properties

Contractible wrt. `TASKS`.

Reformulation

The `cumulative_product` constraint can be expressed in term of a set of reified constraints and of $|\texttt{TASKS}|$ constraints of the form $h_1 \cdot h_2 \cdot \dots \cdot h_{|\texttt{TASKS}|} \leq l$:

1. For each pair of tasks $\texttt{TASKS}[i], \texttt{TASKS}[j]$ $(i, j \in [1, |\texttt{TASKS}|])$ of the `TASKS` collection we create a variable H_{ij} which is set to the height of task $\texttt{TASKS}[j]$ if task $\texttt{TASKS}[j]$ overlaps the origin attribute of task $\texttt{TASKS}[i]$, and to 1 otherwise:
   - If $i = j$:
     - $H_{ij} = \texttt{TASKS}[i].\texttt{height}$.
   - If $i \neq j$:
     - $H_{ij} = \texttt{TASKS}[j].\texttt{height} \vee H_{ij} = 1$.
     - $((\texttt{TASKS}[j].\texttt{origin} \leq \texttt{TASKS}[i].\texttt{origin} \wedge \texttt{TASKS}[j].\texttt{end} > \texttt{TASKS}[i].\texttt{origin}) \wedge (H_{ij} = \texttt{TASKS}[j].\texttt{height})) \vee$
       $((\texttt{TASKS}[j].\texttt{origin} > \texttt{TASKS}[i].\texttt{origin} \vee \texttt{TASKS}[j].\texttt{end} \leq \texttt{TASKS}[i].\texttt{origin}) \wedge (H_{ij} = 1))$
2. For each task $\texttt{TASKS}[i]$ $(i \in [1, |\texttt{TASKS}|])$ we impose a constraint of the form $H_{i1} \cdot H_{i2} \cdot \dots \cdot H_{i|\texttt{TASKS}|} \leq \texttt{LIMIT}$.

See also

common keyword: `cumulative` *(resource constraint)*.

used in graph description: `product_ctr`.

Keywords

characteristic of a constraint: product.

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

filtering: compulsory part.

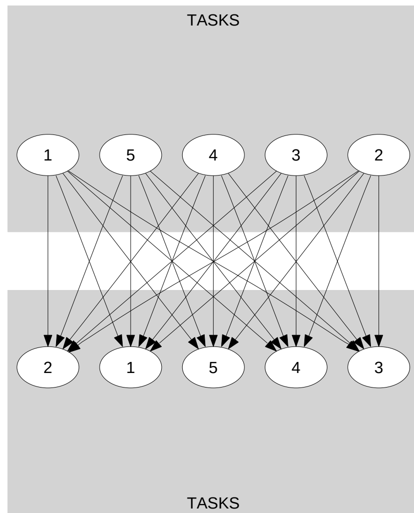
modelling: zero-duration task.

Arc input(s)	TASKS
Arc generator	$SELF \mapsto$ collection(tasks)
Arc arity	1
Arc constraint(s)	tasks.origin + tasks.duration = tasks.end
Graph property(ies)	**NARC**= |TASKS|

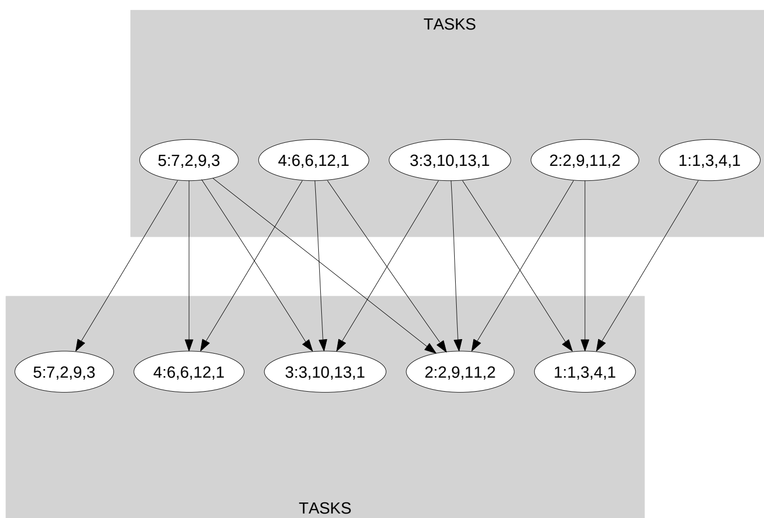
Arc input(s)	TASKS TASKS
Arc generator	$PRODUCT \mapsto$ collection(tasks1, tasks2)
Arc arity	2
Arc constraint(s)	• tasks1.duration > 0 • tasks2.origin ≤ tasks1.origin • tasks1.origin < tasks2.end
Graph class	• ACYCLIC • BIPARTITE • NO_LOOP
Sets	SUCC $\mapsto$ [source, variables − col (VARIABLES−collection(var−dvar), [item(var − ITEMS.height)])]
Constraint(s) on sets	product_ctr(variables, ≤, LIMIT)

Graph model Parts (A) and (B) of Figure 5.221 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 cumulative_product constraint holds since for each successor set $\mathcal{S}$ of the final graph the product of the heights of the tasks in $\mathcal{S}$ does not exceed the limit LIMIT $= 6$.

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



(A)



(B)

Figure 5.221: Initial and final graph of the `cumulative_product` constraint