5.86 cumulative_with_level_of_priority

DESCRIPTION LINKS GRAPH

Origin H. Simonis

Constraint cumulative_with_level_of_priority(TASKS, PRIORITIES)

Arguments

TASKS : collection(priority−int, origin−dvar, duration−dvar, end−dvar, height−dvar)

PRIORITIES : collection(id−int, capacity−int)

Restrictions

required(TASKS, [priority, height])
require_at_least(2, TASKS, [origin, duration, end])
TASKS.priority ≥ 1
TASKS.priority $\leq$ |PRIORITIES|
TASKS.duration ≥ 0
TASKS.origin $\leq$ TASKS.end
TASKS.height ≥ 0
required(PRIORITIES, [id, capacity])
PRIORITIES.id ≥ 1
PRIORITIES.id $\leq$ |PRIORITIES|
increasing_seq(PRIORITIES, id)
increasing_seq(PRIORITIES, capacity)

Purpose

Consider a set $\mathcal{T}$ of tasks described by the TASKS collection where each task has a given priority chosen in the range $[1, \text{PRIORITIES}]$. Let $\mathcal{T}_i$ denote the subset of tasks of $\mathcal{T}$ that all have a priority less than or equal to i. For each set $\mathcal{T}_i$, the cumulative_with_level_of_priority constraint enforces that at each point in time, the cumulated height 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. Finally, it also imposes for each task of $\mathcal{T}$ the constraint origin + duration = end.

Example

$$\left(\begin{array}{l}\left\langle\begin{array}{lllll}\texttt{priority}-1 & \texttt{origin}-1 & \texttt{duration}-2 & \texttt{end}-3 & \texttt{height}-1,\\ \texttt{priority}-1 & \texttt{origin}-2 & \texttt{duration}-3 & \texttt{end}-5 & \texttt{height}-1,\\ \texttt{priority}-1 & \texttt{origin}-5 & \texttt{duration}-2 & \texttt{end}-7 & \texttt{height}-2,\\ \texttt{priority}-2 & \texttt{origin}-3 & \texttt{duration}-2 & \texttt{end}-5 & \texttt{height}-2,\\ \texttt{priority}-2 & \texttt{origin}-6 & \texttt{duration}-3 & \texttt{end}-9 & \texttt{height}-1\end{array}\right\rangle,\\ \langle \texttt{id}-1\ \texttt{capacity}-2, \texttt{id}-2\ \texttt{capacity}-3\rangle\end{array}\right)$$

Figure 5.174 shows the cumulated profile associated with both levels of priority. To each task of the cumulative_with_level_of_priority constraint corresponds a set of rectangles containing the same number (i.e., the position of the task within the TASKS collection): 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 resource consumption of the task. Tasks that have a priority of 1 are coloured in pink, while tasks that have a priority of 2 are coloured in blue. The `cumulative_with_level_of_priority` constraint holds since:

- At each point in time the cumulated resource consumption profile of the tasks of priority 1 does not exceed the upper capacity 2 enforced by the first item of the `PRIORITIES` collection.
- At each point in time the cumulated resource consumption profile of the tasks of priority 1 and 2 does not exceed the upper capacity 3 enforced by the second item of the `PRIORITIES` collection.

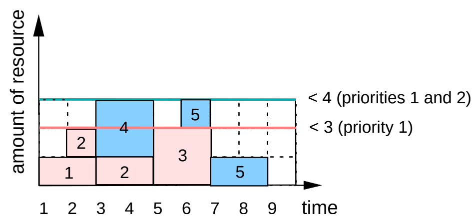


Figure 5.174: Resource consumption profile according to both levels of priority

Typical

$|$`TASKS`$| > 1$
range(`TASKS.priority`) > 1
range(`TASKS.origin`) > 1
range(`TASKS.duration`) > 1
range(`TASKS.end`) > 1
range(`TASKS.height`) > 1
`TASKS.duration` > 0
`TASKS.height` > 0
$|$`PRIORITIES`$| > 1$
`PRIORITIES.capacity` > 0
`PRIORITIES.capacity` $<$ sum(`TASKS.height`)
$|$`TASKS`$| > |$`PRIORITIES`$|$

Symmetries

- Items of `TASKS` are permutable.
- `TASKS.priority` can be increased to any value $\leq |$`PRIORITIES`$|$.
- `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`.
- `PRIORITIES.capacity` can be increased.

Usage

The `cumulative_with_level_of_priority` constraint was suggested by problems from the telecommunication area where one has to ensure different levels of quality of service. For this purpose the capacity of a transmission link is splitted so that a given percentage is reserved to each level. In addition we have that, if the capacities allocated to levels $1, 2, \ldots, i$ is not completely used, then level $i+1$ can use the corresponding spare capacity.

Remark

The `cumulative_with_level_of_priority` constraint can be modelled by a conjunction of `cumulative` constraints. As shown by the next example, the consistency for all variables of the `cumulative` constraints does not implies consistency for the corresponding `cumulative_with_level_of_priority` constraint. The following `cumulative_with_level_of_priority` constraint

$$\left(\begin{array}{l}\left\langle\begin{array}{llll}\texttt{priority}-1 & \texttt{origin}-o_1 & \texttt{duration}-2 & \texttt{height}-2,\\ \texttt{priority}-1 & \texttt{origin}-o_2 & \texttt{duration}-2 & \texttt{height}-1,\\ \texttt{priority}-2 & \texttt{origin}-o_3 & \texttt{duration}-1 & \texttt{height}-3\end{array}\right\rangle,\\ \left\langle\begin{array}{ll}\texttt{id}-1 & \texttt{capacity}-2,\\ \texttt{id}-2 & \texttt{capacity}-3\end{array}\right\rangle\end{array}\right)$$

where the domains of o_1, o_2 and o_3 are respectively equal to $\{1,2,3\}$, $\{1,2,3\}$ and $\{1,2,3,4\}$ corresponds to the following conjunction of `cumulative` constraints

$$\texttt{cumulative}\left(\left\langle\begin{array}{lll}\texttt{origin}-o_1 & \texttt{duration}-2 & \texttt{height}-2,\\ \texttt{origin}-o_2 & \texttt{duration}-2 & \texttt{height}-1\end{array}\right\rangle,2\right)$$

$$\texttt{cumulative}\left(\left\langle\begin{array}{lll}\texttt{origin}-o_1 & \texttt{duration}-2 & \texttt{height}-2,\\ \texttt{origin}-o_2 & \texttt{duration}-2 & \texttt{height}-1,\\ \texttt{origin}-o_3 & \texttt{duration}-1 & \texttt{height}-3\end{array}\right\rangle,3\right)$$

Even if the `cumulative` constraint could achieve arc-consistency, the previous conjunction of `cumulative` constraints would not detect the fact that there is no solution.

See also

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

used in graph description: `sum_ctr`.

Keywords

characteristic of a constraint: derived collection.

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

modelling: zero-duration task.

Derived Collection

$$\texttt{col}\left(\begin{array}{l}\texttt{TIME\_POINTS}-\texttt{collection}(\texttt{idp}-\texttt{int},\texttt{duration}-\texttt{dvar},\texttt{point}-\texttt{dvar}),\\ \left[\begin{array}{l}\texttt{item}\left(\begin{array}{l}\texttt{idp}-\texttt{TASKS.priority},\\ \texttt{duration}-\texttt{TASKS.duration},\\ \texttt{point}-\texttt{TASKS.origin}\end{array}\right),\\ \texttt{item}\left(\begin{array}{l}\texttt{idp}-\texttt{TASKS.priority},\\ \texttt{duration}-\texttt{TASKS.duration},\\ \texttt{point}-\texttt{TASKS.end}\end{array}\right)\end{array}\right]\end{array}\right)$$

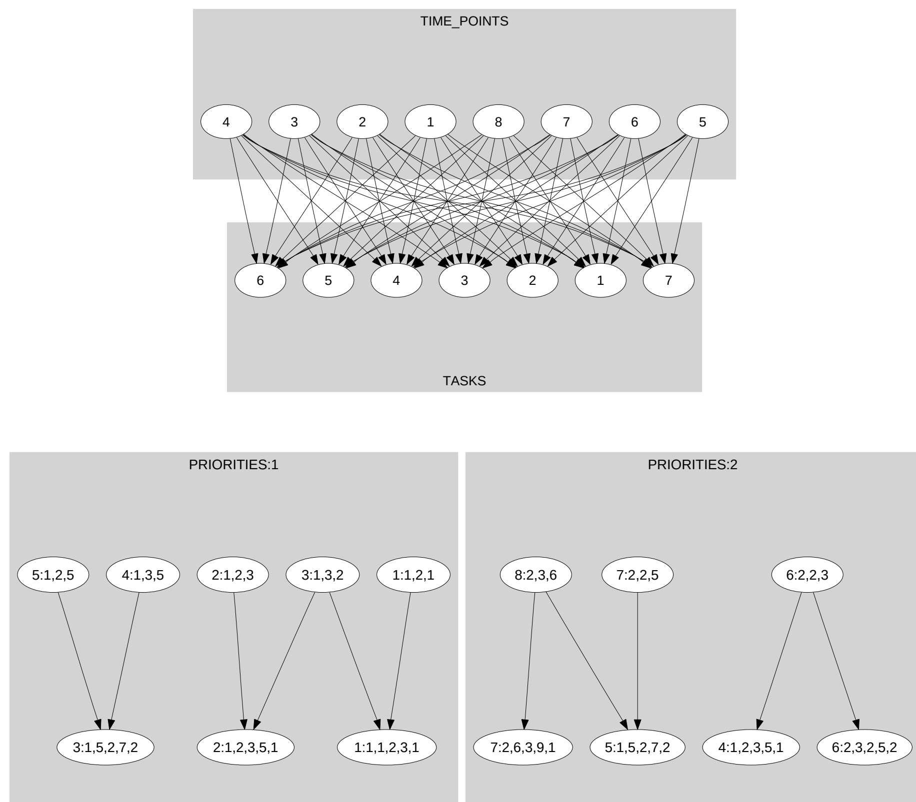
Arc input(s)	TASKS
Arc generator	*SELF*↦collection(tasks)
Arc arity	1
Arc constraint(s)	tasks.origin + tasks.duration = tasks.end
Graph property(ies)	**NARC**= |TASKS|

For all items of PRIORITIES:

Arc input(s)	TIME_POINTS TASKS
Arc generator	*PRODUCT*↦collection(time_points, tasks)
Arc arity	2
Arc constraint(s)	• time_points.idp = PRIORITIES.id • time_points.idp ≥ tasks.priority • time_points.duration > 0 • tasks.origin ≤ time_points.point • time_points.point < tasks.end
Graph class	• ACYCLIC • BIPARTITE • NO_LOOP
Sets	SUCC ↦ [source, variables − col(VARIABLES−collection(var−dvar), [item(var − TASKS.height)])]
Constraint(s) on sets	sum_ctr(variables, ≤, PRIORITIES.capacity)

Graph model

Within the context of the second graph constraint, part (A) of Figure 5.175 shows the initial graphs associated with priorities 1 and 2 of the **Example** slot. Part (B) of Figure 5.175 shows the corresponding final graphs associated with priorities 1 and 2. On the one hand, each source vertex of the final graph can be interpreted as a time point p. On the other hand the successors of a source vertex correspond to those tasks that both overlap that time point p and have a priority less than or equal to a given level. The cumulative_with_level_of_priority constraint holds since for each successor set $\mathcal{S}$ of the final graph the sum of the height of the tasks in $\mathcal{S}$ is less than or equal to the capacity associated with a given level of priority.



(A)

(B)

Figure 5.175: Initial and final graph of the `cumulative_with_level_of_priority` constraint

Signature

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