## 5.97 cumulative_convex

DESCRIPTION LINKS GRAPH

**Origin**

Derived from cumulative

**Constraint**

cumulative_convex(TASKS, LIMIT)

**Type**

POINTS : collection(var−dvar)

**Arguments**

TASKS : collection(points − POINTS, height−dvar)
LIMIT : int

**Restrictions**

required(POINTS, var)
$|$POINTS$| > 0$
required(TASKS, [points, height])
TASKS.height $\geq 0$
LIMIT $\geq 0$

**Purpose**

Cumulative scheduling constraint or scheduling under resource constraints. Consider a set $\mathcal{T}$ of tasks described by the TASKS collection where each task is defined by:

- A *set of distinct points* depicting the time interval where the task is actually running: the smallest and largest coordinates of these points respectively give the first and last instant of that time interval.
- A *height* that depicts the resource consumption used by the task from its first instant to its last instant.

The cumulative_convex 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 $i$.

**Example**

$$\left( \left\langle \begin{array}{ll} \texttt{points} - \langle 2,1,5 \rangle & \texttt{height} - 1, \\ \texttt{points} - \langle 4,5,7 \rangle & \texttt{height} - 2, \\ \texttt{points} - \langle 14,13,9,11,10 \rangle & \texttt{height} - 2 \end{array} \right\rangle, 3 \right)$$

Figure 5.217 shows the cumulated profile associated with the example. To each set of points defining a task corresponds a rectangle. The height of each rectangle represents the resource consumption of the associated task. The cumulative_convex constraint holds since at each point in time we do not have a cumulated resource consumption strictly greater than the upper limit 3 enforced by the last argument of the cumulative_convex constraint.

**Typical**

$|$TASKS$| > 1$
TASKS.height $> 0$
LIMIT $<$sum(TASKS.height)

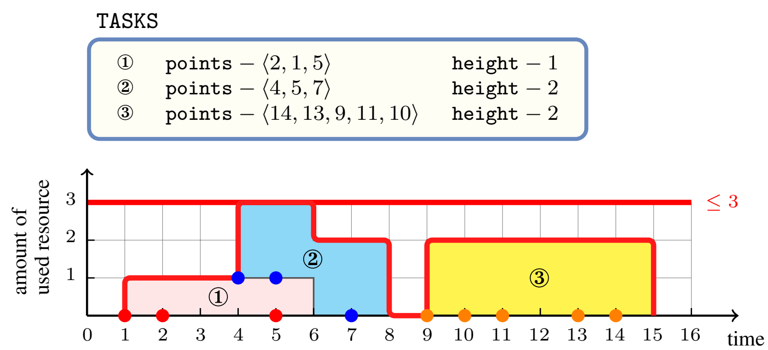



Figure 5.217: Points defining the three tasks of the **Example** slot and corresponding resource consumption profile (note that the vertical position of a task does not really matter but is only used for displaying the contribution of a task to the resource consumption profile)

**Symmetries**

- Items of `TASKS` are permutable.
- Items of `TASKS.points` are permutable.
- `TASKS.height` can be decreased to any value $\geq 0$.
- `LIMIT` can be increased.

**Arg. properties**

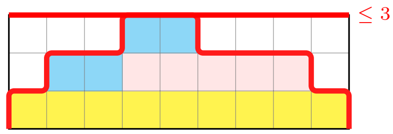
Contractible wrt. `TASKS`.

**Usage**

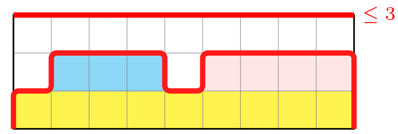
A natural use of the `cumulative_convex` constraint corresponds to problems where a task is defined as the convex hull of a set of distinct points $P_1, \ldots, P_n$ that are not initially fixed. Note that, by explicitly introducing a start $S$ and an end $E$ variables, and by using a `minimum`$(S, \langle \mathtt{var} - P_1, \ldots, \mathtt{var} - P_n \rangle)$ and a `maximum`$(E, \langle \mathtt{var} - P_1, \ldots, \mathtt{var} - P_n \rangle)$ constraints, one could replace the `cumulative_convex` constraint by a `cumulative` constraint. However this hinders propagation.

As a concrete example of use of the `cumulative_convex` constraint we present a constraint model for a well-known pattern-sequencing problem [166] (also known to be equivalent to the graph path-width [263] problem) that is based on a single `cumulative_convex` constraint. The *pattern sequencing problem* can be described as follows: Given a 0-1 matrix in which each column $j$ $(1 \leq j \leq p)$ corresponds to a product required by the customers and each row $i$ $(1 \leq i \leq c)$ corresponds to the order of a particular customer (The entry $c_{ij}$ is equal to 1 if and only if customer $i$ has ordered some quantity of product $j$.), the objective is to find a permutation of the products such that the maximum number of open orders at any point in the sequence is minimised. Order $i$ is *open* at point $k$ in the production sequence if there is a product required in order $i$ that appears at or before position $k$ in the sequence and also a product that appears at or after position $k$ in the sequence.

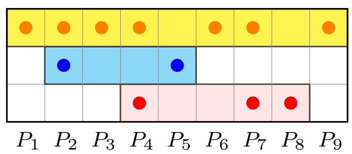
Before giving the constraint model, let us first provide an instance of the pattern-sequencing problem. Consider the matrix $\mathcal{M}_1$ depicted by part (A1) of Fig. 5.218. Part (A2) gives its corresponding *cumulated* matrix $\mathcal{M}_2$ obtained by setting to 1 each 0 of $\mathcal{M}_1$ that is both preceded and followed by a 1. Part (A3) depicts the corresponding solution in term of the `cumulative_convex` constraint: to each row of the matrix $\mathcal{M}_1$ corresponds a task



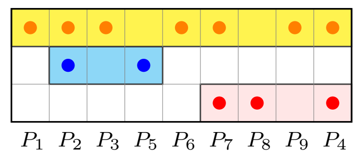

(A4)




(B4)




(A3)




(B3)

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| $P_1$ | $P_2$ | $P_3$ | $P_4$ | $P_5$ | $P_6$ | $P_7$ | $P_8$ | $P_9$ |

(A2)

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| $P_1$ | $P_2$ | $P_3$ | $P_5$ | $P_6$ | $P_7$ | $P_8$ | $P_9$ | $P_4$ |

(B2)

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| $P_1$ | $P_2$ | $P_3$ | $P_4$ | $P_5$ | $P_6$ | $P_7$ | $P_8$ | $P_9$ |

(A1)

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| $P_1$ | $P_2$ | $P_3$ | $P_5$ | $P_6$ | $P_7$ | $P_8$ | $P_9$ | $P_4$ |

(B1)

Figure 5.218: An input matrix for the pattern sequencing problem (A1), its corresponding cumulated matrix (A2), a view in term of tasks (A3) and the corresponding cumulative profile (A4); a second matrix (B1) where column 4 of (A1) is put at the rightmost position.

defined as the convex hull of the different 1 located on that row. Finally part (A4) gives the cumulated profile associated with part (A3), namely the number of 1 in each column of $\mathcal{M}_2$. The cost 3 of this solution is equal to the maximum number of 1 in the columns of the *cumulated* matrix $\mathcal{M}_2$. As shown by parts (B1-B4), we can get a lower cost of 2 by pushing the fourth column to the rightmost position.

The idea of the model is to associate to each row (i.e., customer) $i$ of the *cumulated* matrix a *stack task* that starts at the first 1 on row $i$ and ends at the last 1 of row $i$ (i.e., the task corresponds to the convex hull of the different 1 located on row $i$). Then the cost of a solution is simply the maximum height on the corresponding cumulated profile.

For each column $j$ of the 0-1 matrix initially given there is a variable $V_j$ ranging from 1 to the number of columns $p$. The value of $V_j$ gives the position of column $j$ in a solution. We put all the stack tasks in a `cumulative_convex` constraint, telling that each stack task uses one unit of the resource during all it execution. Since we want to have the same model for different limits on the maximum number of open stacks, and since all variables $V_1, V_2, \ldots, V_p$ have to be distinct, we have an extra dummy task characterised as the convex hull of $V_1, V_2, \ldots, V_p$. This extra dummy task has a height $H$ that has to be maximised. For the matrix depicted by (A1) of Fig. 5.218 we pass to the `cumulative_convex` constraint the following collection of tasks:

$$\left\langle \begin{array}{ll} \texttt{points} - \langle P_1, P_2, P_3, P_4, P_6, P_7, P_9 \rangle & \texttt{height} - 1, \\ \texttt{points} - \langle P_2, P_5 \rangle & \texttt{height} - 1, \\ \texttt{points} - \langle P_4, P_7, P_8 \rangle & \texttt{height} - 1, \\ \texttt{points} - \langle P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9 \rangle & \texttt{height} - 0 \end{array} \right\rangle$$

**Algorithm**

A first natural way to handle the `cumulative_convex` constraint is to accumulate the compulsory part [250] of the different tasks in a profile and to prune according to this profile. We give the main ideas for computing the compulsory part of a task and for pruning a task according to the profile of compulsory parts.

**Compulsory part of a task** Given a task $T$ characterised as the convex hull of a set of distinct points $P_1, P_2, \ldots, P_k$ the compulsory part of $T$ corresponds to the, possibly empty, interval $[s_T, e_T]$ where:

- $s_T$ is the largest value $v$ such that, when all variables $P_1, P_2, \ldots, P_k$ are greater than or equal to $v$, all variables $P_1, P_2, \ldots, P_k$ can still take distinct values.
- $e_T$ is the smallest value $v$ such that, when all variables $P_1, P_2, \ldots, P_k$ are less than or equal to $v$, all variables $P_1, P_2, \ldots, P_k$ can still take distinct values.

**Pruning according to the profile of compulsory parts** Given two instants $i$ and $j$ $(i < j)$ and a task $T$ characterised as the convex hull of a set of distinct points $P_1, P_2, \ldots, P_k$, assume that $T$ cannot overlap $i$ and $j$ since this would lead exceeding `LIMIT`, the second argument of the `cumulative_convex` constraint. Furthermore assume that, when all variables $P_1, P_2, \ldots, P_k$ are both greater than $i$ and less than $j$, all variables $P_1, P_2, \ldots, P_k$ cannot take distinct values. Then all values of $[i+1, j-1]$ can be removed from variables $P_1, P_2, \ldots, P_k$.

**See also**

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

**used in graph description:** `alldifferent`, `between_min_max`, `sum_ctr`.

**Keywords**

**characteristic of a constraint:** convex.

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

**filtering:** compulsory part.

**problems:** pattern sequencing.

**Derived Collection**

$$\texttt{col}\left(\begin{array}{l}\texttt{INSTANTS}-\texttt{collection}(\texttt{instant}-\texttt{dvar}),\\ [\texttt{item}(\texttt{instant}-\texttt{TASKS.points.var})]\end{array}\right)$$

| | |
|---|---|
| **Arc input(s)** | TASKS |
| **Arc generator** | $SELF$ $\mapsto$ `collection`(tasks) |
| **Arc arity** | 1 |
| **Arc constraint(s)** | `alldifferent`(tasks.points) |
| **Graph property(ies)** | **NARC**$=$ \|TASKS\| |

| | |
|---|---|
| **Arc input(s)** | INSTANTS TASKS |
| **Arc generator** | $PRODUCT$ $\mapsto$ `collection`(instants, tasks) |
| **Arc arity** | 2 |
| **Arc constraint(s)** | `between_min_max`(instants.instant, tasks.points) |
| **Graph class** | • ACYCLIC<br>• BIPARTITE<br>• NO_LOOP |

**Sets**

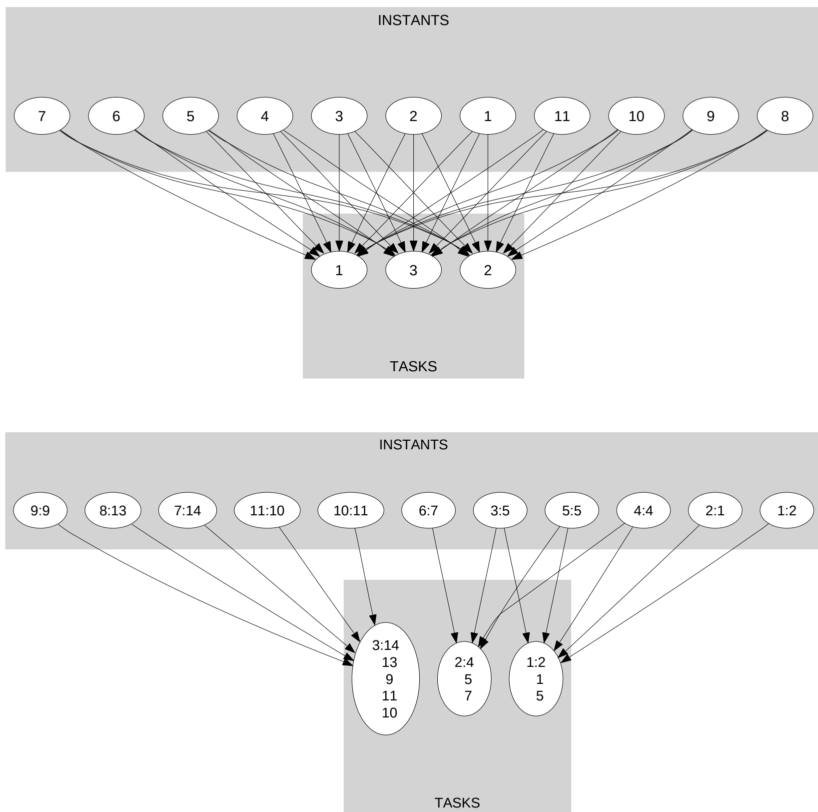
$$\text{SUCC} \mapsto \left[\begin{array}{l}\texttt{source},\\ \texttt{variables}-\texttt{col}\left(\begin{array}{l}\texttt{VARIABLES}-\texttt{collection}(\texttt{var}-\texttt{dvar}),\\ [\texttt{item}(\texttt{var}-\texttt{TASKS.height})]\end{array}\right)\end{array}\right]$$

**Constraint(s) on sets** `sum_ctr`(variables, $\leq$, LIMIT)

**Graph model**

The first graph constraint forces for each task that the set of points defining its time interval are all distinct. The second graph constraint makes sure for each time point $t$, that the cumulated heights of the tasks that overlap $t$ does not exceed the limit of the resource.

Parts (A) and (B) of Figure 5.219 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 corresponding to a point used in the definitions of the different tasks. On the other hand, the successors of a source vertex correspond to those tasks that overlap a given time point. The `cumulative_convex` constraint holds since, for each successor set $\mathcal{S}$ of the final graph, the sum of the heights of the tasks in $\mathcal{S}$ does not exceed the limit $\texttt{LIMIT} = 3$.




**(A)**

**(B)**

Figure 5.219: Initial and final graph of the `cumulative_convex` constraint