

**5.297 sliding\_time\_window\_sum**

	DESCRIPTION	LINKS	GRAPH
<b>Origin</b>	Derived from <code>sliding_time_window</code> .		
<b>Constraint</b>	<code>sliding_time_window_sum(WINDOW_SIZE, LIMIT, TASKS)</code>		
<b>Arguments</b>	WINDOW_SIZE : <code>int</code> LIMIT : <code>int</code> TASKS : <code>collection(origin-dvar, end-dvar, npoint-dvar)</code>		
<b>Restrictions</b>	WINDOW_SIZE > 0 LIMIT ≥ 0 <code>required(TASKS, [origin, end, npoint])</code> TASKS.origin ≤ TASKS.end TASKS.npoint ≥ 0		
<b>Purpose</b>	For any time window of size WINDOW_SIZE, the sum of the points of the tasks of the collection TASKS that overlap that time window do not exceed a given limit LIMIT.		
<b>Example</b>	$\left( 9, 16, \left\langle \begin{array}{lll} \text{origin} - 10 & \text{end} - 13 & \text{npoint} - 2, \\ \text{origin} - 5 & \text{end} - 6 & \text{npoint} - 3, \\ \text{origin} - 6 & \text{end} - 8 & \text{npoint} - 4, \\ \text{origin} - 14 & \text{end} - 16 & \text{npoint} - 5, \\ \text{origin} - 2 & \text{end} - 4 & \text{npoint} - 6 \end{array} \right\rangle \right)$		
	The lower part of Figure 5.534 indicates the different tasks on the time axis. Each task is drawn as a rectangle with its corresponding identifier in the middle. Finally the upper part of Figure 5.534 shows the different time windows and the respective contribution of the tasks in these time windows. A line with two arrows depicts each time window. The two arrows indicate the start and the end of the time window. At the right of each time window we give its occupation. Since this occupation is always less than or equal to the limit 16, the <code>sliding_time_window_sum</code> constraint holds.		
<b>Symmetries</b>	<ul style="list-style-type: none"> <li>• WINDOW_SIZE can be <code>decreased</code>.</li> <li>• LIMIT can be <code>increased</code>.</li> <li>• Items of TASKS are <code>permutable</code>.</li> <li>• TASKS.npoint can be <code>decreased</code> to any value ≥ 0.</li> <li>• One and the same constant can be <code>added</code> to the <code>origin</code> and <code>end</code> attributes of all items of TASKS.</li> </ul>		
<b>Usage</b>	This constraint may be used for timetabling problems in order to put an upper limit on the cumulated number of points in a shift.		
<b>Reformulation</b>	The <code>sliding_time_window_sum</code> constraint can be expressed in term of a set of $ \text{TASKS} ^2$ reified constraints and of $ \text{TASKS} $ <code>linear inequalities</code> constraints:		

1. For each pair of tasks  $\text{TASKS}[i], \text{TASKS}[j]$  ( $i, j \in [1, |\text{TASKS}|]$ ) of the  $\text{TASKS}$  collection we create a variable  $Point_{ij}$  which is set to  $\text{TASKS}[j].\text{npoint}$  if  $\text{TASKS}[j]$  intersects the time window  $\mathcal{W}_i$  of size  $\text{WINDOW\_SIZE}$  that starts at instant  $\text{TASKS}[i].\text{origin}$ , or 0 otherwise:
  - If  $i = j$  (i.e.,  $\text{TASKS}[i]$  and  $\text{TASKS}[j]$  coincide):
    - $Point_{ij} = \text{TASKS}[i].\text{npoint}$ .
  - If  $i \neq j$  and  $\text{TASKS}[j].\text{end} < \text{TASKS}[i].\text{origin}$  (i.e.,  $\text{TASKS}[j]$  for sure ends before the time window  $\mathcal{W}_i$ ):
    - $Point_{ij} = 0$ .
  - If  $i \neq j$  and  $\text{TASKS}[j].\text{origin} > \text{TASKS}[i].\text{origin} + \text{WINDOW\_SIZE} - 1$  (i.e.,  $\text{TASKS}[j]$  for sure starts after the time window  $\mathcal{W}_i$ ):
    - $Point_{ij} = 0$ .
  - Otherwise (i.e.,  $\text{TASKS}[j]$  can potentially overlap the time window  $\mathcal{W}_i$ ):
    - $Point_{ij} = \min(1, \max(0, \min(\text{TASKS}[i].\text{origin} + \text{WINDOW\_SIZE}, \text{TASKS}[j].\text{end}) - \max(\text{TASKS}[i].\text{origin}, \text{TASKS}[j].\text{origin}))) \cdot \text{TASKS}[j].\text{npoint}$ .
2. For each task  $\text{TASKS}[i]$  ( $i \in [1, |\text{TASKS}|]$ ) we create a linear inequality constraint  $Point_{i1} + Point_{i2} + \dots + Point_{i|\text{TASKS}|} \leq \text{LIMIT}$ .

**See also**

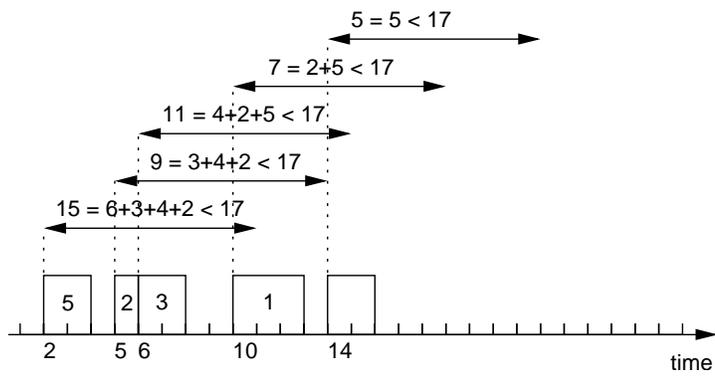
**related:** [sliding\\_time\\_window](#) (sum of the points of intersecting tasks with sliding time window replaced by sum of intersections of tasks with sliding time window).

**used in graph description:** [sum\\_ctr](#).

**Keywords**

**characteristic of a constraint:** [time window](#), [sum](#).

**constraint type:** [sliding sequence constraint](#), [temporal constraint](#).

Figure 5.534: Time windows of the `sliding_time_window_sum` constraint

<b>Arc input(s)</b>	TASKS
<b>Arc generator</b>	<i>SELF</i> $\mapsto$ <code>collection(tasks)</code>
<b>Arc arity</b>	1
<b>Arc constraint(s)</b>	<code>tasks.origin <math>\leq</math> tasks.end</code>
<b>Graph property(ies)</b>	$\overline{\text{NARC}} =  \text{TASKS} $
<b>Arc input(s)</b>	TASKS
<b>Arc generator</b>	<i>CLIQUE</i> $\mapsto$ <code>collection(tasks1, tasks2)</code>
<b>Arc arity</b>	2
<b>Arc constraint(s)</b>	<ul style="list-style-type: none"> <li>• <code>tasks1.end <math>\leq</math> tasks2.end</code></li> <li>• <code>tasks2.origin - tasks1.end &lt; WINDOW_SIZE - 1</code></li> </ul>
<b>Sets</b>	$\text{SUCC} \mapsto \left[ \begin{array}{l} \text{source,} \\ \text{variables} - \text{col} \left( \begin{array}{l} \text{VARIABLES} - \text{collection}(\text{var} - \text{dvar}), \\ [\text{item}(\text{var} - \text{TASKS.npoint})] \end{array} \right) \end{array} \right]$
<b>Constraint(s) on sets</b>	<code>sum_ctr(variables, <math>\leq</math>, LIMIT)</code>

**Graph model**

We generate an arc from a task  $t_1$  to a task  $t_2$  if task  $t_2$  does not end before the end of task  $t_1$  and if task  $t_2$  intersects the time window that starts at the last instant of task  $t_1$ . Each set generated by SUCC corresponds to all tasks that intersect in time the time window that starts at instant  $\text{end} - 1$ , where  $\text{end}$  is the end of a given task.

Parts (A) and (B) of Figure 5.535 respectively show the initial and final graph associated with the **Example** slot. In the final graph, the successors of a given task  $t$  correspond to the set of tasks that both do not end before the end of task  $t$ , and intersect the time window that starts at the  $\text{end} - 1$  of task  $t$ .

**Signature**

Consider the first graph constraint. Since we use the *SELF* arc generator on the TASKS collection the maximum number of arcs of the final graph is equal to  $|\text{TASKS}|$ . Therefore we can rewrite  $\overline{\text{NARC}} = |\text{TASKS}|$  to  $\overline{\text{NARC}} \geq |\text{TASKS}|$  and simplify  $\overline{\overline{\text{NARC}}}$  to  $\overline{\text{NARC}}$ .

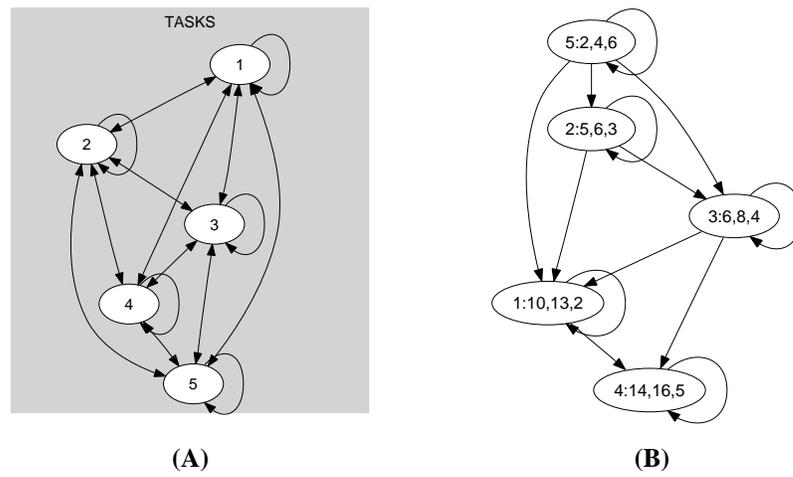


Figure 5.535: Initial and final graph of the `sliding_time_window_sum` constraint

20030820

1547