5.199 interval_and_sum

DESCRIPTION LINKS GRAPH AUTOMATON

Origin

Derived from cumulative.

Constraint

interval_and_sum(SIZE_INTERVAL, TASKS, LIMIT)

Arguments

SIZE_INTERVAL	:	int
TASKS	:	collection(origin−dvar, height−dvar)
LIMIT	:	int

Restrictions

SIZE_INTERVAL > 0
required(TASKS, [origin, height])
TASKS.origin ≥ 0
TASKS.height ≥ 0
LIMIT ≥ 0

Purpose

A maximum resource capacity constraint: We have to fix the origins of a collection of tasks in such a way that, for all the tasks that are allocated to the same interval, the sum of the heights does not exceed a given capacity. All the intervals we consider have the following form: $[k\cdot\texttt{SIZE\_INTERVAL}, k\cdot\texttt{SIZE\_INTERVAL}+\texttt{SIZE\_INTERVAL}-1]$, where k is an integer.

Example

$$\left(5,\left\langle\begin{array}{ll}\texttt{origin}-1 & \texttt{height}-2,\\ \texttt{origin}-10 & \texttt{height}-2,\\ \texttt{origin}-10 & \texttt{height}-3,\\ \texttt{origin}-4 & \texttt{height}-1\end{array}\right\rangle,5\right)$$

Figure 5.462 shows the solution associated with the example. The constraint interval_and_sum holds since the sum of the heights of the tasks that are located in the same interval does not exceed the limit 5. Each task t is depicted by a rectangle r associated with the interval to which the task t is assigned. The rectangle r is labelled with the position of t within the items of the TASKS collection. The origin of task t is represented by a small black square located within its corresponding rectangle r. Finally, the height of a rectangle r is equal to the height of the task t to which it corresponds.

Typical

SIZE_INTERVAL > 1
$|$TASKS$| > 1$
range(TASKS.origin) > 1
range(TASKS.height) > 1
LIMIT $<$ sum(TASKS.height)

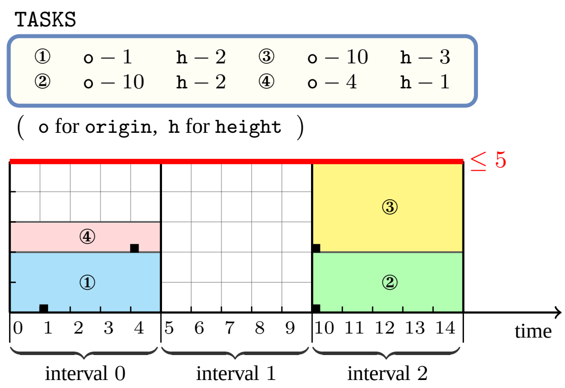


Figure 5.462: The `interval_and_sum` solution to the **Example** slot with the use of each interval

Symmetries

- Items of `TASKS` are permutable.
- One and the same constant can be added to the `origin` attribute of all items of `TASKS`.
- An occurrence of a value of `TASKS.origin` that belongs to the k-th interval, of size `SIZE_INTERVAL`, can be replaced by any other value of the same interval.
- `TASKS.height` can be decreased to any value ≥ 0.
- `LIMIT` can be increased.

Arg. properties

Contractible wrt. `TASKS`.

Usage

This constraint can be use for timetabling problems. In this context the different intervals are interpreted as morning and afternoon periods of different consecutive days. We have a capacity constraint for all tasks that are assigned to the same morning or afternoon of a given day.

Reformulation

Let K denote the index of the last possible interval where the tasks can be assigned: $K = \lfloor \frac{\max_{i\in[1,|\texttt{TASKS}|]}(\texttt{TASKS}[i].\texttt{origin})+\texttt{SIZE\_INTERVAL}-1}{\texttt{SIZE\_INTERVAL}} \rfloor$. The $\texttt{interval\_and\_sum}(\texttt{SIZE\_INTERVAL}, \texttt{TASKS}, \texttt{LIMIT})$ constraint can be expressed in term of a set of reified constraints and of K arithmetic constraints (i.e., `scalar_product` constraints).

1. For each task $\texttt{TASKS}[i]$ ($i \in [1, |\texttt{TASKS}|]$) and for each interval $[k \cdot \texttt{SIZE\_INTERVAL}, k \cdot \texttt{SIZE\_INTERVAL} + \texttt{SIZE\_INTERVAL} - 1]$ ($k \in [0, K]$) we create a 0-1 variable B_{ik} that will be set to 1 if and only if the origin of task $\texttt{TASKS}[i]$ is assigned within interval $[k \cdot \texttt{SIZE\_INTERVAL}, k \cdot \texttt{SIZE\_INTERVAL} + \texttt{SIZE\_INTERVAL} - 1]$:
$$B_{ik} \Leftrightarrow \texttt{TASKS}[i].\texttt{origin} \geq k \cdot \texttt{SIZE\_INTERVAL} \wedge \texttt{TASKS}[i].\texttt{origin} \leq k \cdot \texttt{SIZE\_INTERVAL} + \texttt{SIZE\_INTERVAL} - 1$$
2. Finally, for each interval $[k \cdot \texttt{SIZE\_INTERVAL}, k \cdot \texttt{SIZE\_INTERVAL} + \texttt{SIZE\_INTERVAL} - 1]$ ($k \in [0, K]$), we impose the sum $\texttt{TASKS}[1].\texttt{height} \cdot$

B_{1k} + $\texttt{TASKS}[2].\texttt{height} \cdot B_{2k} + \cdots + \texttt{TASKS}[|\texttt{TASKS}|].\texttt{height} \cdot B_{|\texttt{TASKS}|k}$ to not exceed the maximum allowed capacity LIMIT.

See also

assignment dimension removed: `sum_ctr` *(assignment dimension corresponding to intervals is removed).*

related: `interval_and_count` *(`sum_ctr` constraint replaced by `among_low_up`).*

used in graph description: `sum_ctr`.

Keywords

application area: assignment.

characteristic of a constraint: automaton, automaton with array of counters.

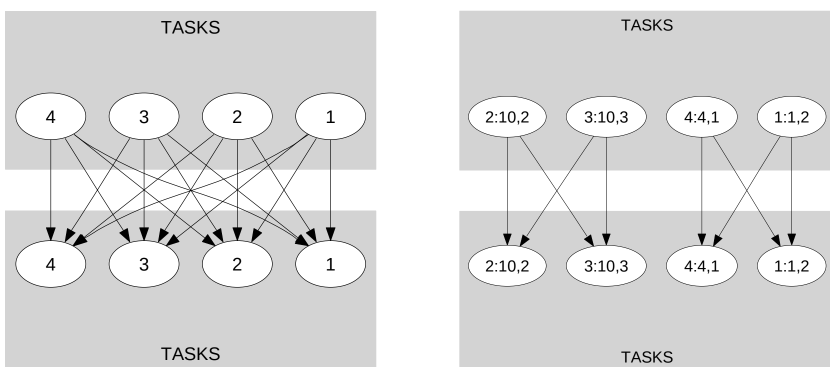
constraint type: timetabling constraint, resource constraint, temporal constraint.

modelling: assignment dimension, interval.

Arc input(s)	TASKS TASKS
Arc generator	$PRODUCT \mapsto$ collection(tasks1, tasks2)
Arc arity	2
Arc constraint(s)	tasks1.origin/SIZE_INTERVAL = tasks2.origin/SIZE_INTERVAL
Sets	SUCC $\mapsto$ [source, variables − col (VARIABLES−collection(var−dvar), [item(var − TASKS.height)])]
Constraint(s) on sets	sum_ctr(variables, $\leq$, LIMIT)

Graph model We use a bipartite graph where each class of vertices corresponds to the different tasks of the TASKS collection. There is an arc between two tasks if their origins belong to the same interval. Finally we enforce a sum_ctr constraint on each set $\mathcal{S}$ of successors of the different vertices of the final graph. This put a restriction on the maximum value of the sum of the height attributes of the tasks of $\mathcal{S}$.

Parts (A) and (B) of Figure 5.463 respectively show the initial and final graph associated with the **Example** slot. Each connected component of the final graph corresponds to items that are all assigned to the same interval.



(A) **(B)**

Figure 5.463: Initial and final graph of the interval_and_sum constraint

Automaton Figure 5.464 depicts the automaton associated with the `interval_and_sum` constraint. To each item of the collection `TASKS` corresponds a signature variable S_i that is equal to 1.

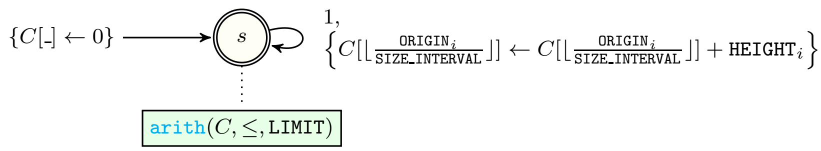


Figure 5.464: Automaton of the `interval_and_sum` constraint