5.160 geost_time

DESCRIPTION LINKS

Origin Generalisation of diffn.

Constraint geost_time(K, DIMS, OBJECTS, SBOXES)

Types

$$\begin{array}{lll}
\text{VARIABLES} & : & \text{collection}(\text{v}-\text{dvar}) \\
\text{INTEGERS} & : & \text{collection}(\text{v}-\text{int}) \\
\text{POSITIVES} & : & \text{collection}(\text{v}-\text{int})
\end{array}$$

Arguments

$$\begin{array}{lll}
\text{K} & : & \text{int} \\
\text{DIMS} & : & \text{sint} \\
\text{OBJECTS} & : & \text{collection}\left(\begin{array}{l}\text{oid}-\text{int},\\ \text{sid}-\text{dvar},\\ \text{x}-\text{VARIABLES},\\ \text{start}-\text{dvar},\\ \text{duration}-\text{dvar},\\ \text{end}-\text{dvar}\end{array}\right) \\
\text{SBOXES} & : & \text{collection}(\text{sid}-\text{int}, \text{t}-\text{INTEGERS}, \text{l}-\text{POSITIVES})
\end{array}$$

Restrictions

$|\text{VARIABLES}| \geq 1$
$|\text{INTEGERS}| \geq 1$
$|\text{POSITIVES}| \geq 1$
$\text{required}(\text{VARIABLES}, \text{v})$
$|\text{VARIABLES}| = \text{K}$
$\text{required}(\text{INTEGERS}, \text{v})$
$|\text{INTEGERS}| = \text{K}$
$\text{required}(\text{POSITIVES}, \text{v})$
$|\text{POSITIVES}| = \text{K}$
$\text{POSITIVES.v} > 0$
$\text{K} \geq 0$
$\text{DIMS} \geq 0$
$\text{DIMS} < \text{K}$
$\text{distinct}(\text{OBJECTS}, \text{oid})$
$\text{required}(\text{OBJECTS}, [\text{oid}, \text{sid}, \text{x}])$
$\text{require\_at\_least}(2, \text{OBJECTS}, [\text{start}, \text{duration}, \text{end}])$
$\text{OBJECTS.oid} \geq 1$
$\text{OBJECTS.oid} \leq |\text{OBJECTS}|$
$\text{OBJECTS.sid} \geq 1$
$\text{OBJECTS.sid} \leq |\text{SBOXES}|$
$\text{OBJECTS.duration} \geq 0$
$|\text{SBOXES}| \geq 1$
$\text{required}(\text{SBOXES}, [\text{sid}, \text{t}, \text{l}])$
$\text{SBOXES.sid} \geq 1$
$\text{SBOXES.sid} \leq |\text{SBOXES}|$
$\text{do\_not\_overlap}(\text{SBOXES})$

Purpose

Holds if (1) the difference between the end in time and the start in time of each object is equal to its duration in time, and if (2) for each pair of objects (O_i, O_j), $i < j$, O_i and O_j do not overlap with respect to a set of dimensions depicted by `DIMS` as well as to the time axis. Note that an object with duration zero can never overlap any other object. O_i and O_j are objects that take a shape among a set of shapes. Each *shape* is defined as a finite set of shifted boxes, where each shifted box is described by a box in a `K`-dimensional space at a given offset (from the origin of the shape) with given sizes that are all strictly greater than 0. More precisely, a *shifted box* is an entity defined by its shape id `sid`, shift offset `t`, and sizes `l`. Then, a shape is defined as the union of shifted boxes sharing the same shape id. An *object* is an entity defined by its unique object identifier `oid`, shape id `sid` and origin `x`.

An object O_i *does not overlap* an object O_j with respect to a set of dimensions depicted by `DIMS` as well as to the time axis if and only if:

- The start in time of O_i is greater than or equal to the end in time of O_j.
- The start in time of O_j is greater than or equal to the end in time of O_i.
- For all shifted box s_i associated with O_i and for all shifted box s_j associated with O_j there exists a dimension $d \in$ `DIMS` such that the start of s_i in dimension d is greater than or equal to the end of s_j in dimension d, or the start of s_j in dimension d is greater than or equal to the end of s_i in dimension d.

Example

$$\left(\begin{array}{l}
2, \{0,1\}, \\
\left\langle\begin{array}{llllll}
\texttt{oid}-1 & \texttt{sid}-1 & \texttt{x}-\langle 1,2\rangle & \texttt{start}-0 & \texttt{duration}-1 & \texttt{end}-1, \\
\texttt{oid}-2 & \texttt{sid}-5 & \texttt{x}-\langle 2,1\rangle & \texttt{start}-0 & \texttt{duration}-1 & \texttt{end}-1, \\
\texttt{oid}-3 & \texttt{sid}-8 & \texttt{x}-\langle 4,1\rangle & \texttt{start}-0 & \texttt{duration}-1 & \texttt{end}-1
\end{array}\right\rangle, \\
\left\langle\begin{array}{lll}
\texttt{sid}-1 & \texttt{t}-\langle 0,0\rangle & \texttt{l}-\langle 2,1\rangle, \\
\texttt{sid}-1 & \texttt{t}-\langle 0,1\rangle & \texttt{l}-\langle 1,2\rangle, \\
\texttt{sid}-1 & \texttt{t}-\langle 1,2\rangle & \texttt{l}-\langle 3,1\rangle, \\
\texttt{sid}-2 & \texttt{t}-\langle 0,0\rangle & \texttt{l}-\langle 3,1\rangle, \\
\texttt{sid}-2 & \texttt{t}-\langle 0,1\rangle & \texttt{l}-\langle 1,3\rangle, \\
\texttt{sid}-2 & \texttt{t}-\langle 2,1\rangle & \texttt{l}-\langle 1,1\rangle, \\
\texttt{sid}-3 & \texttt{t}-\langle 0,0\rangle & \texttt{l}-\langle 2,1\rangle, \\
\texttt{sid}-3 & \texttt{t}-\langle 1,1\rangle & \texttt{l}-\langle 1,2\rangle, \\
\texttt{sid}-3 & \texttt{t}-\langle -2,2\rangle & \texttt{l}-\langle 3,1\rangle, \\
\texttt{sid}-4 & \texttt{t}-\langle 0,0\rangle & \texttt{l}-\langle 3,1\rangle, \\
\texttt{sid}-4 & \texttt{t}-\langle 0,1\rangle & \texttt{l}-\langle 1,1\rangle, \\
\texttt{sid}-4 & \texttt{t}-\langle 2,1\rangle & \texttt{l}-\langle 1,3\rangle, \\
\texttt{sid}-5 & \texttt{t}-\langle 0,0\rangle & \texttt{l}-\langle 2,1\rangle, \\
\texttt{sid}-5 & \texttt{t}-\langle 1,1\rangle & \texttt{l}-\langle 1,1\rangle, \\
\texttt{sid}-5 & \texttt{t}-\langle 0,2\rangle & \texttt{l}-\langle 2,1\rangle, \\
\texttt{sid}-6 & \texttt{t}-\langle 0,0\rangle & \texttt{l}-\langle 3,1\rangle, \\
\texttt{sid}-6 & \texttt{t}-\langle 0,1\rangle & \texttt{l}-\langle 1,1\rangle, \\
\texttt{sid}-6 & \texttt{t}-\langle 2,1\rangle & \texttt{l}-\langle 1,1\rangle, \\
\texttt{sid}-7 & \texttt{t}-\langle 0,0\rangle & \texttt{l}-\langle 3,2\rangle, \\
\texttt{sid}-8 & \texttt{t}-\langle 0,0\rangle & \texttt{l}-\langle 2,3\rangle
\end{array}\right\rangle
\end{array}\right)$$

Parts (A), (B) and (C) of Figure 5.350 respectively represent the potential shapes associated with the three objects of the example. Part (D) shows the position of the three objects of the example, where the first, second and third objects were respectively assigned shapes 1, 5 and 8. The coordinates of the leftmost lowest corner of each object are stressed in bold. The `geost_time` constraint holds since the three objects do not overlap: even though the time intervals associated with each object overlap (i.e., they are in fact identical), their corresponding shapes do not overlap (i.e., see part (D) if Figure 5.350).

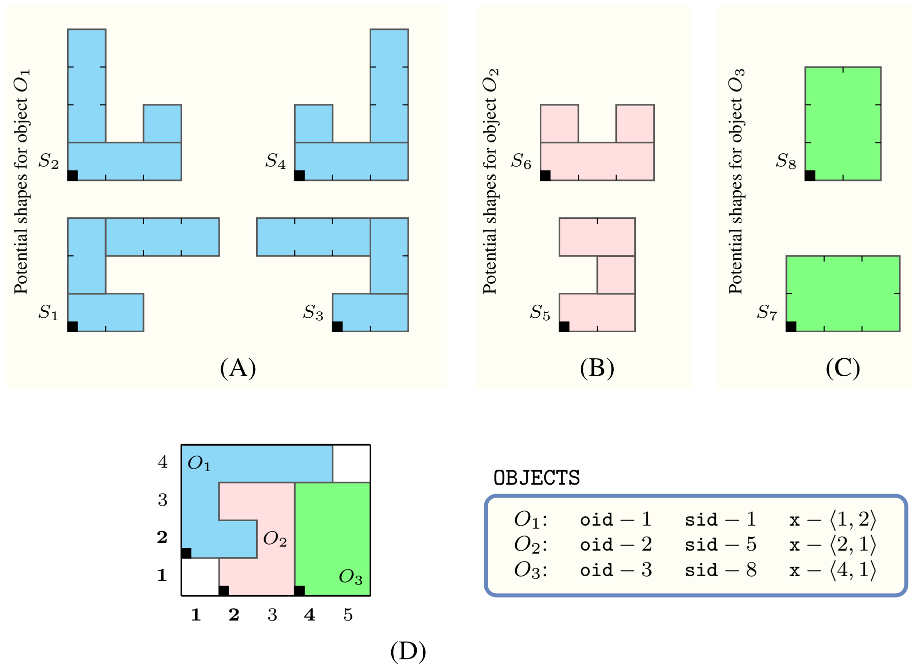

Figure 5.350: (D) The three non-overlapping objects O_1, O_2, O_3 of the **Example** slot respectively assigned shapes S_1, S_5, S_8; (A), (B), (C) shapes S_1, S_2, S_3, S_4, S_5, S_6, S_7 and S_8 are respectively made up from 3, 3, 3, 3, 3, 3, 1 and 1 disjoint shifted box.

Typical $|\texttt{OBJECTS}| > 1$

Symmetries

- Items of `OBJECTS` are permutable.
- Items of `SBOXES` are permutable.
- Items of `OBJECTS.x`, `SBOXES.t` and `SBOXES.l` are permutable (*same permutation used*).
- `SBOXES.l.v` can be decreased to any value ≥ 1.
- One and the same constant can be added to the `start` and `end` attributes of all items of `OBJECTS`.

Usage The `geost_time` constraint allows to model directly a large number of placement problems. Figure 5.351 sketches ten typical use of the `geost_time` constraint:

- The first case (A) corresponds to a non-overlapping constraint among three segments (or three tasks in disjunction).

- The second, third and fourth cases (B,C,D) correspond to a non-overlapping constraint between rectangles where (B) and (C) are special cases where the sizes of all rectangles in the second dimension are equal to 1; this can be interpreted as a *machine assignment problem* where each rectangle corresponds to a non-pre-emptive task that has to be placed in time and assigned to a specific machine so that no two tasks assigned to the same machine overlap in time. In Part (B) the duration of each task is fixed, while in Part (C) the duration depends on the machine to which the task is actually assigned. This dependence can be expressed by the `element` constraint, which specifies the dependence between the shape variable and the assignment variable of each task.
- The fifth case (E) corresponds to a non-overlapping constraint between rectangles where each rectangle can have two orientations. This is achieved by associating with each rectangle two shapes of respective sizes $l \cdot h$ and $h \cdot l$. Since their orientation is not initially fixed, an `element_lesseq` constraint can be used for enforcing the three rectangles to be included within the bounding box defined by the origin's coordinates $1, 1$ and sizes $8, 3$.
- The sixth case (F) corresponds to a non-overlapping constraint between more complex objects where each object is described by a given set of rectangles.
- The seventh case (G) describes a rectangle placement problem where one has to first assign each rectangle to a strip so that all rectangles that are assigned to the same strip do not overlap.
- The eighth case (H) corresponds to a non-overlapping constraint between parallelepipeds.
- The ninth case (I) can be interpreted as a non-overlapping constraint between parallelepipeds that are assigned to the same container. The first dimension corresponds to the identifier of the container, while the next three dimensions are associated with the position of a parallelepiped inside a container.
- Finally the tenth case (J) describes a rectangle placement problem over three consecutive time-slots: rectangles assigned to the same time-slot should not overlap in time. We initially start with the three rectangles 1, 2 and 3. Rectangle 3 is no more present at instant 2 (the arrow $\downarrow$ within rectangle 3 at time 1 indicates that rectangle 3 will disappear at the next time-point), while rectangle 4 appears at instant 2 (the arrow $\uparrow$ within rectangle 4 at time 2 denotes the fact that the rectangle 4 appears at instant 2). Finally rectangle 2 disappears at instant 3 and is replaced by rectangle 5.

Algorithm A sweep-based filtering algorithm for this constraint is described in [38]. Unlike previous sweep filtering algorithms which move a line for finding a feasible position for the origin of an object, this algorithm performs a recursive traversal of the multidimensional placement space. It explores all points of the domain of the origin of the object under focus, one by one, in increasing lexicographic order, until a point is found that is not infeasible for any non-overlapping constraints. To make the search efficient, instead of moving each time to the successor point, the search is arranged so that it skips points that are known to be infeasible for some non-overlapping constraint.

Systems geost in **Choco**, geost in **JaCoP**.

See also **common keyword:** `diffn`, `non_overlap_sboxes` *(geometrical constraint,non-overlapping)*, `visible` *(geometrical constraint,sweep)*.

specialisation: `geost` *(*`temporal dimension` *removed)*.

Keywords

constraint type: decomposition, timetabling constraint, predefined constraint.

filtering: sweep.

geometry: geometrical constraint, non-overlapping.

modelling: assignment dimension, assignment to the same set of values, assigning and scheduling tasks that run in parallel, disjunction.

modelling exercises: assignment to the same set of values, assigning and scheduling tasks that run in parallel.

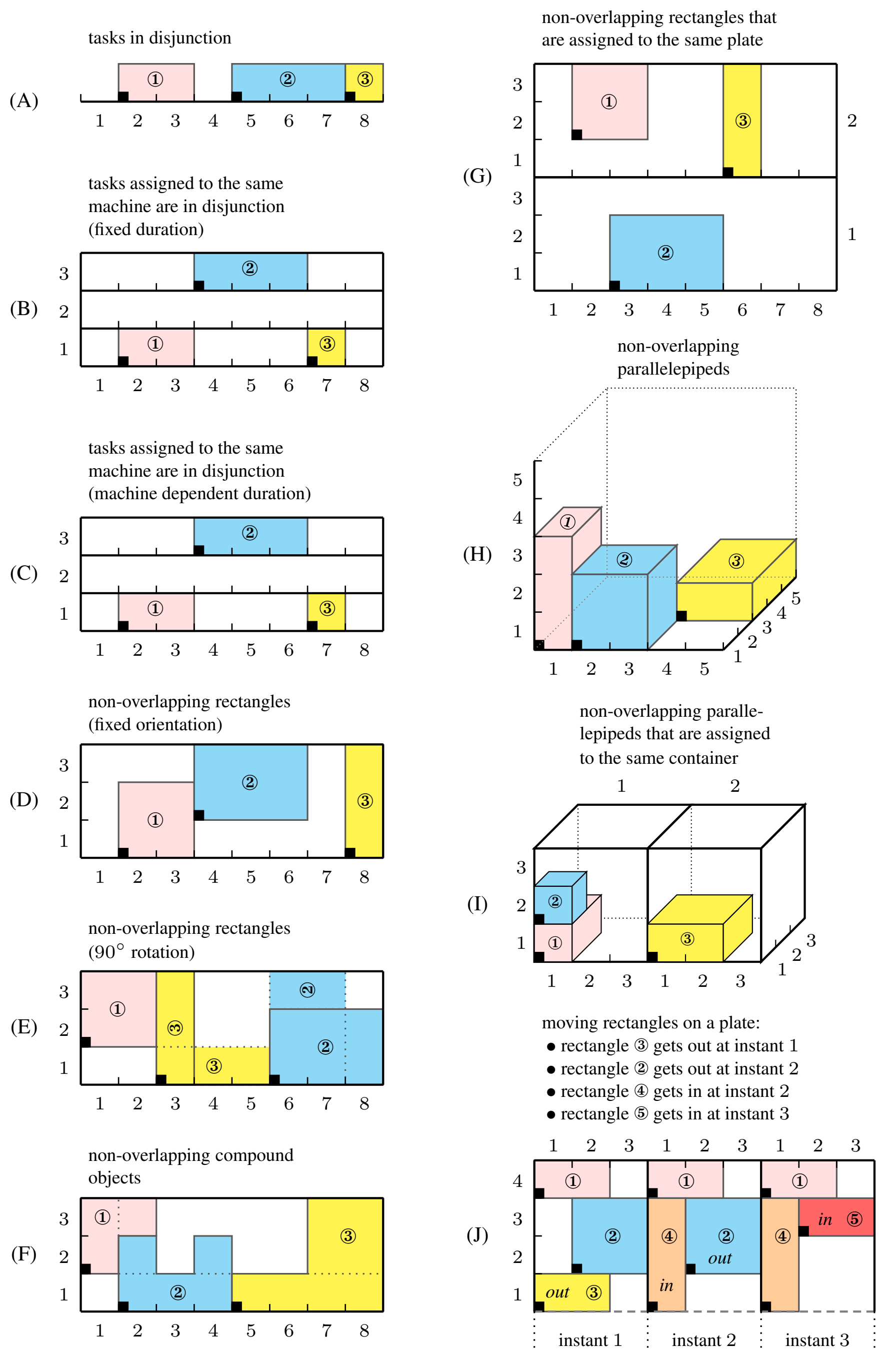

Figure 5.351: Ten typical examples of use of the `geost_time` constraint (ground instances)