5.134 geost

DESCRIPTION LINKS

Origin Generalisation of diffn.

Constraint geost(K, OBJECTS, SBOXES)

Types

```
VARIABLES  :  collection(v−dvar)
INTEGERS   :  collection(v−int)
POSITIVES  :  collection(v−int)
```

Arguments

```
K          :  int
OBJECTS    :  collection(oid−int, sid−dvar, x − VARIABLES)
SBOXES     :  collection(sid−int, t − INTEGERS, l − POSITIVES)
```

Restrictions

```
required(VARIABLES, v)
|VARIABLES| = K
required(INTEGERS, v)
|INTEGERS| = K
required(POSITIVES, v)
|POSITIVES| = K
POSITIVES.v > 0
K > 0
required(OBJECTS, [oid, sid, x])
OBJECTS.oid ≥ 1
OBJECTS.oid ≤ |OBJECTS|
OBJECTS.sid ≥ 1
OBJECTS.sid ≤ |SBOXES|
required(SBOXES, [sid, t, l])
SBOXES.sid ≥ 1
SBOXES.sid ≤ |SBOXES|
```

Purpose

Holds if, 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 $\{1, 2, \ldots, \mathtt{K}\}$. 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. 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 the set of dimensions $\{1, 2, \ldots, \mathtt{K}\}$ if and only if 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 $d \in \{1, 2, \ldots, \mathtt{K}\}$ 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, \left\langle \begin{array}{lll} \texttt{oid}-1 & \texttt{sid}-1 & \texttt{x}-\langle 1,2\rangle, \\ \texttt{oid}-2 & \texttt{sid}-5 & \texttt{x}-\langle 2,1\rangle, \\ \texttt{oid}-3 & \texttt{sid}-8 & \texttt{x}-\langle 4,1\rangle \end{array} \right\rangle, \\
\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, \\
\left\langle \begin{array}{lll} \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, \end{array} \right\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}
\end{array}\right)$$

Parts (A), (B) and (C) of Figure 5.266 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` constraint holds since the three objects do not overlap (i.e., see part (D) if Figure 5.266).

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.

Usage

The `geost` constraint allows to model directly a large number of placement problems.

Remark

In the two-dimensional case, when rectangles heights are all equal to one and when rectangles starts in the first dimension are all fixed, the `geost` constraint can be rewritten as a `k_alldifferent` constraint corresponding to a system of `alldifferent` constraints derived from the maximum cliques of the corresponding interval graph.

Algorithm

A sweep-based filtering algorithm for this constraint is described in [34]. 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

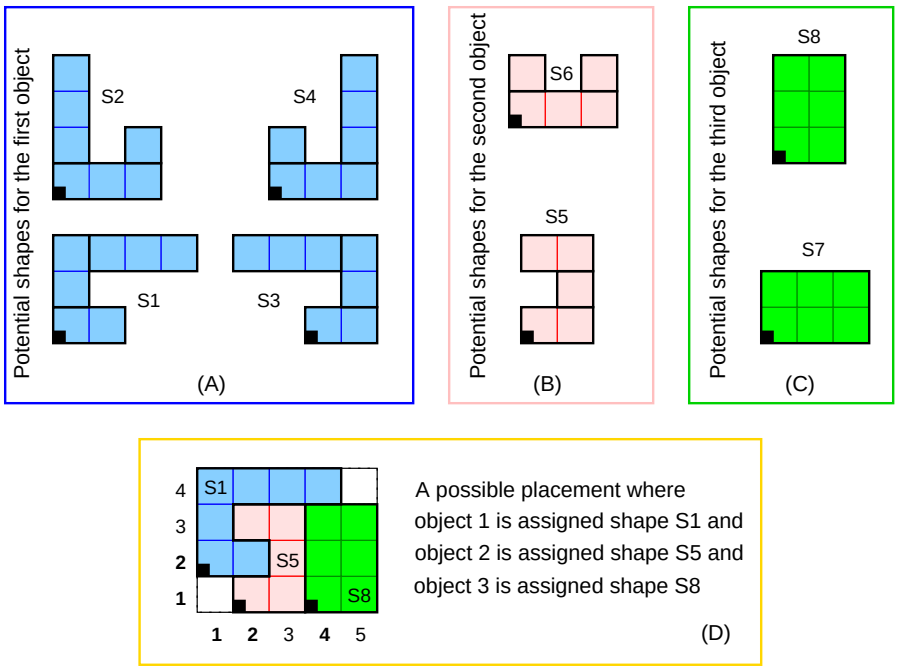

Figure 5.266: The three objects of the example

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.

Within the context of breaking symmetries six different ways of integrating within `geost` a chain of lexicographical ordering constraints like `lex_chain_less` for enforcing a lexicographic ordering on the origin coordinates of identical objects, are described in [2].

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

See also **common keyword:** `calendar` *(scheduling with machine choice, calendars and preemption)*, `diffn` *(geometrical constraint,non-overlapping)*, `lex_chain_less`, `lex_chain_lesseq` *(symmetry)*, `non_overlap_sboxes` *(geometrical constraint,non-overlapping)*, `visible` *(geometrical constraint,sweep)*.

generalisation: `geost_time` *(*`temporal dimension` *added to* `geometrical dimensions`*)*.

specialisation: `k_alldifferent` *(when rectangles heights are all equal to* 1 *and rectangles starts in the first dimension are all fixed)*, `lex_alldifferent` *(*`object` *replaced by* `vector`*)*.

Keywords **application area:** floor planning problem.

combinatorial object: pentomino.

constraint arguments: business rules.

constraint type: decomposition, timetabling constraint, predefined constraint, relaxation.

filtering: sweep.

geometry: geometrical constraint, non-overlapping.

heuristics: heuristics for two-dimensional rectangle placement problems.

modelling: scheduling with machine choice, calendars and preemption, disjunction, assignment dimension, assigning and scheduling tasks that run in parallel, assignment to the same set of values, relaxation dimension.

modelling exercises: scheduling with machine choice, calendars and preemption, assigning and scheduling tasks that run in parallel, assignment to the same set of values, relaxation dimension.

problems: strip packing, two-dimensional orthogonal packing, pallet loading.

puzzles: squared squares, packing almost squares, Partridge, pentomino, Shikaku, smallest square for packing consecutive dominoes, smallest square for packing rectangles with distinct sizes, smallest rectangle area, Conway packing problem.

symmetry: symmetry.