## 5.100 diffn

DESCRIPTION LINKS GRAPH

**Origin** [37]

**Constraint** diffn(ORTHOTOPES)

**Synonyms** disjoint, disjoint1, disjoint2, diff2.

**Type** ORTHOTOPE : collection(ori−dvar, siz−dvar, end−dvar)

**Argument** ORTHOTOPES : collection(orth − ORTHOTOPE)

**Restrictions**
$|\texttt{ORTHOTOPE}| > 0$
require_at_least(2, ORTHOTOPE, [ori, siz, end])
$\texttt{ORTHOTOPE.siz} \geq 0$
$\texttt{ORTHOTOPE.ori} \leq \texttt{ORTHOTOPE.end}$
required(ORTHOTOPES, orth)
same_size(ORTHOTOPES, orth)

**Purpose** Generalised multi-dimensional non-overlapping constraint: Holds if, for each pair of orthotopes $(O_1, O_2)$, $O_1$ and $O_2$ do not overlap. Two orthotopes do not overlap if there exists at least one dimension where their projections do not overlap.

**Example**

$$\left(\begin{array}{l} \texttt{orth} - \left\langle \begin{array}{lll} \texttt{ori}-2 & \texttt{siz}-2 & \texttt{end}-4, \\ \texttt{ori}-1 & \texttt{siz}-3 & \texttt{end}-4 \end{array} \right\rangle, \\ \left\langle \begin{array}{l} \texttt{orth} - \left\langle \begin{array}{lll} \texttt{ori}-4 & \texttt{siz}-4 & \texttt{end}-8, \\ \texttt{ori}-3 & \texttt{siz}-3 & \texttt{end}-6 \end{array} \right\rangle, \\ \texttt{orth} - \left\langle \begin{array}{lll} \texttt{ori}-9 & \texttt{siz}-2 & \texttt{end}-11, \\ \texttt{ori}-4 & \texttt{siz}-3 & \texttt{end}-7 \end{array} \right\rangle \end{array} \right\rangle \end{array}\right)$$

Figure 5.202 represents the respective position of the three rectangles of the example. The coordinates of the leftmost lowest corner of each rectangle are stressed in bold. The diffn constraint holds since the three rectangles do not overlap.

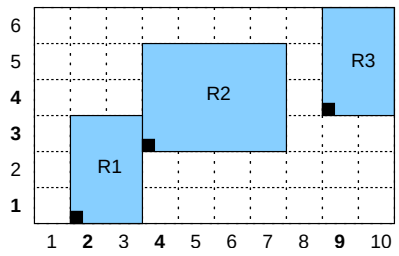


Figure 5.202: The three rectangles of the example

**Typical**

$|\texttt{ORTHOTOPE}| > 1$
$\texttt{ORTHOTOPE.siz} > 0$
$|\texttt{ORTHOTOPES}| > 1$

**Symmetries**

- Items of `ORTHOTOPES` are permutable.
- Items of `ORTHOTOPES.orth` are permutable (*same permutation used*).
- `ORTHOTOPES.orth.siz` can be decreased to any value $\geq 0$.
- One and the same constant can be added to the `ori` and `end` attributes of all items of `ORTHOTOPES.orth`.

**Usage**

The `diffn` constraint occurs in placement and scheduling problems. It was for instance used for scheduling problems where one has to both assign each non-preemptive task to a resource and fix its origin so that two tasks, which are assigned to the same resource, do not overlap. When the resource is a set of persons to which non-preemptive tasks have to be assigned this corresponds to so called *timetabling problems*. A second practical application from the area of the design of memory-dominated embedded systems [365] can be found in [366]. Together with arithmetic and `cumulative` constraints, the `diffn` constraint was used in [364] for packing more complex shapes such as angles. Figure 5.203 illustrates the angle packing problem on an instance involving 10 angles taken from [364].

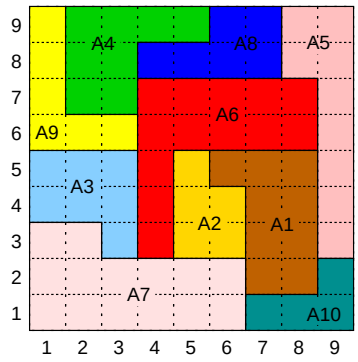


Figure 5.203: A solution for the angle packing problem of items A1 $= [2, 4, 3, 1]$, A2 $= [2, 2, 1, 3]$, A3 $= [1, 3, 3, 2]$, A4 $= [2, 1, 4, 3]$, A5 $= [1, 7, 2, 2]$, A6 $= [1, 2, 5, 5]$, A7 $= [6, 2, 2, 3]$, A8 $= [4, 2, 2, 1]$, A9 $= [3, 1, 1, 4]$, A10 $= [3, 2, 1, 1]$.

One other packing problem attributed to S. Golomb is to find the smallest square that can contain the set of consecutive squares from $1 \times 1$ up to $n \times n$ so that these squares do not overlap each other (see the smallest rectangle area problem).

**Remark**

When we have segments (respectively rectangles) the `diffn` constraint is referenced under the name `disjoint1` (respectively `disjoint2`) in **SICStus Prolog** [90]. When we have rectangles the `diffn` constraint is also called `diff2` in **JaCoP**.

It was shown in [368, page 137] that, finding out whether a non-overlapping constraint between a set of rectangles has a solution or not is NP-hard. This was achieved by reduction from sequencing with release times and deadlines.

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 `diffn` 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**

Checking whether a `diffn` constraint for which all variables are fixed is satisfied or not is related to the Klee's measure problem: given a collection of axis-aligned multi-dimensional boxes, how quickly can one compute the volume of their union. Then the `diffn` constraint holds if the volume of the union is equal to the sum of the volumes of the different boxes.

A first possible method for filtering is to use constructive disjunction. The idea is to try out each alternative of a disjunction (e.g., given two orthotopes $o_1$ and $o_2$ that should not overlap, we successively assume for each dimension that $o_1$ finishes before $o_2$, and that $o_2$ finishes before $o_1$) and to remove values that were pruned in all alternatives. For the two-dimensional case of `diffn` a second possible solution used in [328] is to represent explicitly the two-dimensional domain of the origin of each rectangle by a quadtree [333] and to accumulate all forbidden regions within this data structure. As for conventional domain variables, a failure occurs when a two-dimensional domain get empty. A third possible filtering algorithm based on sweep is described in [30].

The thesis of J. Nelissen [262] considers the case where all rectangles have the same size and can be rotated from 90 degrees (i.e., the pallet loading problem.). For the $n$-dimensional case of `diffn` a filtering algorithm handling the fact that two objects do not overlap is given in [40].

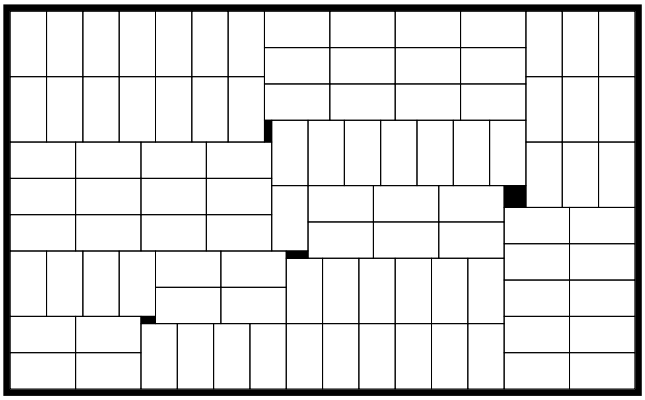

Figure 5.204: A hard instance from [262, page 165]: A solution for packing 99 rectangles of size $5 \times 9$ into a rectangle of size $86 \times 52$

Extensions of the non-overlapping constraint to polygons and to more complex shapes are respectively described in [40] and in [323]. Specialised propagation algorithms for the squared squares problem [77] (based on the fact that no waste is permitted) are given in [160] and in [159].

The `cumulative` constraint can be used as a necessary condition for the `diffn` constraint. Figure 5.205 illustrates this point for the two-dimensional case. A first (respectively second) `cumulative` constraint is obtained by forgetting the $y$-coordinate (respectively the

$x$-coordinate) of the origin of each rectangle occurring in a `diffn` constraint. Parts (B) and (C) respectively depict the cumulated profiles associated with the projection of the rectangles depicted by part (A) on the $x$ and $y$ axes. The `cumulative` constraint is a necessary but not sufficient condition for the two-dimensional case of the `diffn` constraint. Figure 5.206 illustrates this point on an example taken from [70] where one has to place the 8 rectangles R1, R2, R3, R4, R5, R6, R7, R8 of respective size $5 \times 2$, $8 \times 2$, $6 \times 1$, $5 \times 1$, $2 \times 1$, $3 \times 1$, $2 \times 2$ and $1 \times 2$ in a big rectangle of size $12 \times 4$. As shown by Figure 5.206 there is a cumulative solution where R8 is splitted in two parts but M. Hujter proves in [195] that their is no solution where no rectangle is splitted.

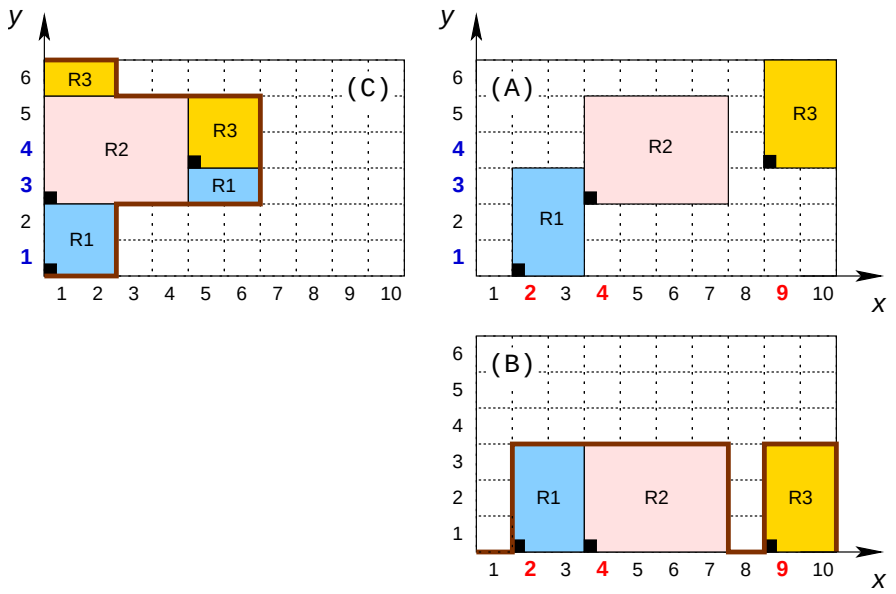


Figure 5.205: Looking from the perspective of the `cumulative` constraint in a two-dimensional rectangles placement problem

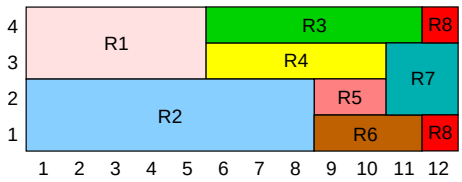


Figure 5.206: Illustrating the necessary but not sufficient placement condition

In the context of $n$ parallelepipeds that have to be packed [165, 233] within a box of sizes $X \times Y \times Z$ one can proceed as follows for stating three `cumulative` constraints. The $i^{th}$ $(i \in [1, n])$ parallelepiped is described by the following attributes:

- $ox_i$, $oy_i$, $oz_i$ $(i \in [1, n])$ the coordinates of its origin on the $x$, $y$ and $z$-axes.
- $sx_i$, $sy_i$, $sz_i$ $(i \in [1, n])$ its sizes on the $x$, $y$ and $z$-axes.

- $px_i$, $py_i$, $pz_i$ ($i \in [1, n]$) the surfaces of its projections on the planes $yz$, $xz$, and $xy$ respectively equal to $sy_i sz_i$, $sx_i sz_i$, and $sx_i sy_i$.
- $v_i$ its volume (equal to $sx_i sy_i sz_i$).

For the placement of $n$ parallelepipeds we get the following necessary conditions that respectively correspond to three `cumulative` constraints on the planes $yz$, $xz$, and $xy$:

$$\begin{cases} \forall i \in [1, X] : \sum_{j|ox_j \leq i \leq ox_j + sx_j - 1} px_j \leq YZ \\ \forall i \in [1, Y] : \sum_{j|oy_j \leq i \leq oy_j + sy_j - 1} py_j \leq XZ \\ \forall i \in [1, Z] : \sum_{j|oz_j \leq i \leq oz_j + sz_j - 1} pz_j \leq XY \end{cases}$$

**Reformulation**

Based on the fact that two orthotopes do not overlap if there exists at least one dimension where their projections do not overlap one can reformulate the `diffn(ORTHOTOPES)` constraint as a disjunction of inequalities between the origin and the end attributes. In addition one has to link the origin, the size and the end attributes of each orthotope in each dimension.

If we consider the example described in the **Example** slot we get the following reformulation:

- $4 = 2+2$ (*link between the origin, size and end in dimension* $1$ *of the first orthotope*),
- $4 = 1+3$ (*link between the origin, size and end in dimension* $2$ *of the first orthotope*),
- $8 = 4 + 4$ (*link between the origin, size and end in dimension* $1$ *of the second orthotope*),
- $6 = 3 + 3$ (*link between the origin, size and end in dimension* $2$ *of the second orthotope*),
- $11 = 9 + 2$ (*link between the origin, size and end in dimension* $1$ *of the third orthotope*),
- $7 = 4+3$ (*link between the origin, size and end in dimension* $2$ *of the third orthotope*),
- $4 \leq 4 \vee 8 \leq 2 \vee 4 \leq 3 \vee 6 \leq 1$ (*non-overlapping between the first and second orthotopes*),
- $4 \leq 9 \vee 11 \leq 2 \vee 4 \leq 4 \vee 7 \leq 1$ (*non-overlapping between the first and third orthotopes*),
- $8 \leq 9 \vee 11 \leq 4 \vee 6 \leq 4 \vee 7 \leq 3$ (*non-overlapping between the second and third orthotopes*).

**Systems**

`geost` in **Choco**, `diff2` in **JaCoP**, `diff` in **JaCoP**, `disjoint` in **JaCoP**, `disjointconditional` in **JaCoP**.

**Used in**

`diffn_column`, `diffn_include`, `place_in_pyramid`.

**See also**

**common keyword:** calendar *(scheduling with machine choice, calendars and preemption)*, diffn_column, diffn_include *(geometrical constraint,orthotope)*, geost, geost_time, non_overlap_sboxes *(geometrical constraint,non-overlapping)*, visible *(geometrical constraint)*.

**implied by:** orths_are_connected.

**implies:** cumulative *(implies one* cumulative *constraint for each dimension)*.

**related:** cumulative_two_d *(*cumulative_two_d *is a necessary condition for* diffn*: forget one dimension when the number of dimensions is equal to* 3*)*, lex_chain_less, lex_chain_lesseq *(lexicographic ordering on the origins of* tasks, rectangles, . . .*)*, two_orth_column, two_orth_include.

**specialisation:** all_min_dist *(*orthotope *replaced by* line segment*, of same length)*, alldifferent *(*orthotope *replaced by* variable*)*, cumulatives *(*orthotope *replaced by* task *with* machine assignment *and* origin *attributes)*, disjunctive *(orthotope replaced by* task *of* heigth 1*)*, k_alldifferent *(when rectangles heights are all equal to* 1 *and rectangles starts in the first dimension are all fixed)*, lex_alldifferent *(*orthotope *replaced by* vector*)*.

**used in graph description:** orth_link_ori_siz_end, two_orth_do_not_overlap.

**Keywords**

**application area:** floor planning problem.

**characteristic of a constraint:** core.

**combinatorial object:** pentomino.

**complexity:** sequencing with release times and deadlines.

**constraint arguments:** business rules.

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

**filtering:** Klee measure problem, sweep, quadtree, compulsory part, constructive disjunction, SAT.

**geometry:** geometrical constraint, orthotope, polygon, non-overlapping.

**heuristics:** heuristics for two-dimensional rectangle placement problems.

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

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

**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.

| | |
|---|---|
| **Arc input(s)** | ORTHOTOPES |
| **Arc generator** | $SELF \mapsto$ collection(orthotopes) |
| **Arc arity** | 1 |
| **Arc constraint(s)** | orth_link_ori_siz_end(orthotopes.orth) |
| **Graph property(ies)** | NARC$=$ \|ORTHOTOPES\| |

| | |
|---|---|
| **Arc input(s)** | ORTHOTOPES |
| **Arc generator** | $CLIQUE(\neq) \mapsto$ collection(orthotopes1, orthotopes2) |
| **Arc arity** | 2 |
| **Arc constraint(s)** | two_orth_do_not_overlap(orthotopes1.orth, orthotopes2.orth) |
| **Graph property(ies)** | NARC$=$ \|ORTHOTOPES\| $*$ \|ORTHOTOPES\| $-$ \|ORTHOTOPES\| |

**Graph model** The diffn constraint is expressed by using two graph constraints:

- The first graph constraint enforces for each dimension and for each orthotope the link between the corresponding ori, siz and end attributes.
- The second graph constraint imposes each pair of distinct orthotopes to not overlap.

Parts (A) and (B) of Figure 5.207 respectively show the initial and final graph associated with the second graph constraint of the **Example** slot. Since we use the NARC graph property, the arcs of the final graph are stressed in bold.

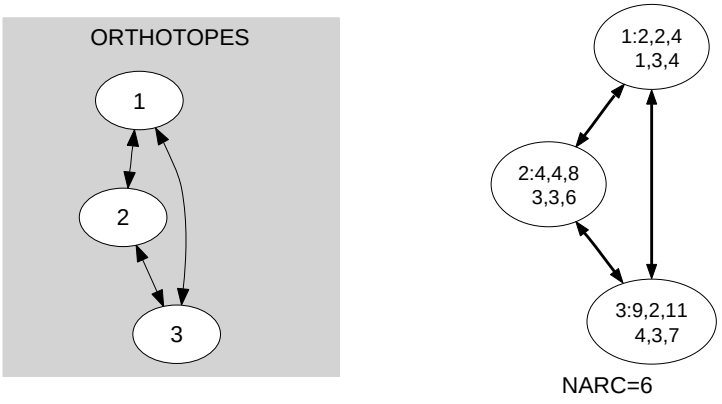


Figure 5.207: Initial and final graph of the diffn constraint

**Signature** Since |ORTHOTOPES| is the maximum number of vertices of the final graph of the first graph constraint we can rewrite NARC $=$ |ORTHOTOPES| to NARC $\geq$ |ORTHOTOPES|. This leads to simplify $\overline{\mathbf{NARC}}$ to $\overline{\mathbf{NARC}}$.

Since we use the $CLIQUE(\neq)$ arc generator on the ORTHOTOPES collection, |ORTHOTOPES| · |ORTHOTOPES| − |ORTHOTOPES| is the maximum number of vertices of the final graph of the second graph constraint. Therefore we can rewrite $\mathbf{NARC} = |\text{ORTHOTOPES}| \cdot |\text{ORTHOTOPES}| - |\text{ORTHOTOPES}|$ to $\mathbf{NARC} \geq |\text{ORTHOTOPES}| \cdot |\text{ORTHOTOPES}| - |\text{ORTHOTOPES}|$. Again, this leads to simplify $\overline{\mathbf{NARC}}$ to $\overline{\mathbf{NARC}}$.