

5.118 **diffn**

	DESCRIPTION	LINKS	GRAPH
Origin	[41]		
Constraint	diffn(ORTHOTOPES)		
Synonyms	disjoint, disjoint1, disjoint2, diff2.		
Type	ORTHOTOPE : collection(ori-dvar, siz-dvar, end-dvar)		
Argument	ORTHOTOPES : collection(orth - ORTHOTOPE)		
Restrictions	ORTHOTOPE > 0 require_at_least(2, ORTHOTOPE, [ori, siz, end]) ORTHOTOPE.siz ≥ 0 ORTHOTOPE.ori ≤ 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 one of the orthotopes has zero size or if there exists at least one dimension where their projections do not overlap.		

Example	$\left(\left\langle \begin{array}{l} \text{orth} - \langle \text{ori} - 2 \text{ siz} - 2 \text{ end} - 4, \text{ori} - 1 \text{ siz} - 2 \text{ end} - 3 \rangle, \\ \text{orth} - \langle \text{ori} - 4 \text{ siz} - 4 \text{ end} - 8, \text{ori} - 2 \text{ siz} - 2 \text{ end} - 4 \rangle, \\ \text{orth} - \langle \text{ori} - 6 \text{ siz} - 5 \text{ end} - 11, \\ \text{ori} - 5 \text{ siz} - 2 \text{ end} - 7 \rangle \end{array} \right\rangle \right)$
---------	---

Figure 5.277 represents the 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 as explained in Figure 5.278.

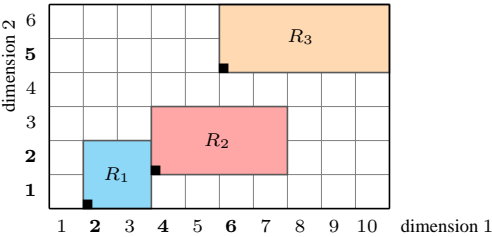


Figure 5.277: Illustration of the **Example** slot: the three rectangles

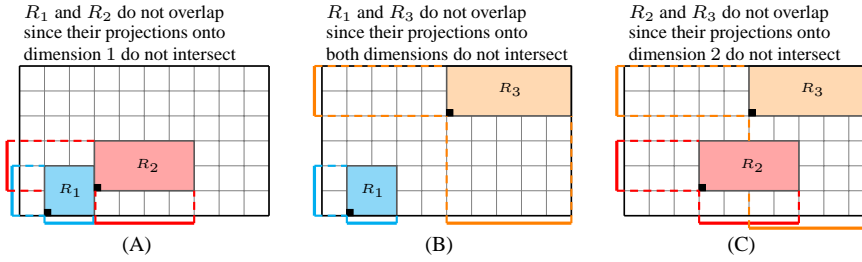


Figure 5.278: Illustration of the **Example** slot: the reasons (A), (B), (C) why the pairs of rectangles (R_1, R_2) , (R_1, R_3) , (R_2, R_3) do not overlap

All solutions

Figure 5.279 gives all solutions to the following non ground instance of the `diffn` constraint:

```

 $X_1 \in [1, 3], EX_1 \in [1, 9], Y_1 \in [1, 3], EY_1 \in [1, 9],$ 
 $X_2 \in [1, 3], EX_2 \in [1, 9], Y_2 \in [2, 3], EY_2 \in [1, 9],$ 
 $X_3 \in [1, 2], EX_3 \in [1, 9], Y_3 \in [1, 4], EY_3 \in [1, 9],$ 
 $X_4 \in [1, 3], EX_4 \in [1, 9], Y_4 \in [1, 3], EY_4 \in [1, 9],$ 
diffn(( $\langle X_1 \ 2 \ EX_1, Y_1 \ 3 \ EY_1 \rangle, \langle X_2 \ 3 \ EX_2, Y_2 \ 2 \ EY_2 \rangle,$ 
 $\langle X_3 \ 1 \ EX_3, Y_3 \ 4 \ EY_3 \rangle, \langle X_4 \ 4 \ EX_4, Y_4 \ 1 \ EY_4 \rangle$ )).

```

Typical

```

|ORTHOTOPE| > 1
ORTHOTOPE.siz > 0
|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 ≥ 0 .
- One and the same constant can be [added](#) to the `ori` and `end` attributes of all items of `ORTHOTOPES.orth`.

Arg. properties

[Contractible](#) wrt. `ORTHOTOPES`.

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 [402] can be found in [403]. Together with arithmetic and [cumulative](#) constraints, the `diffn` constraint was used in [401] for packing more complex shapes such as angles. Figure 5.280 illustrates the angle packing problem on an instance involving 10 angles taken from [401].

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

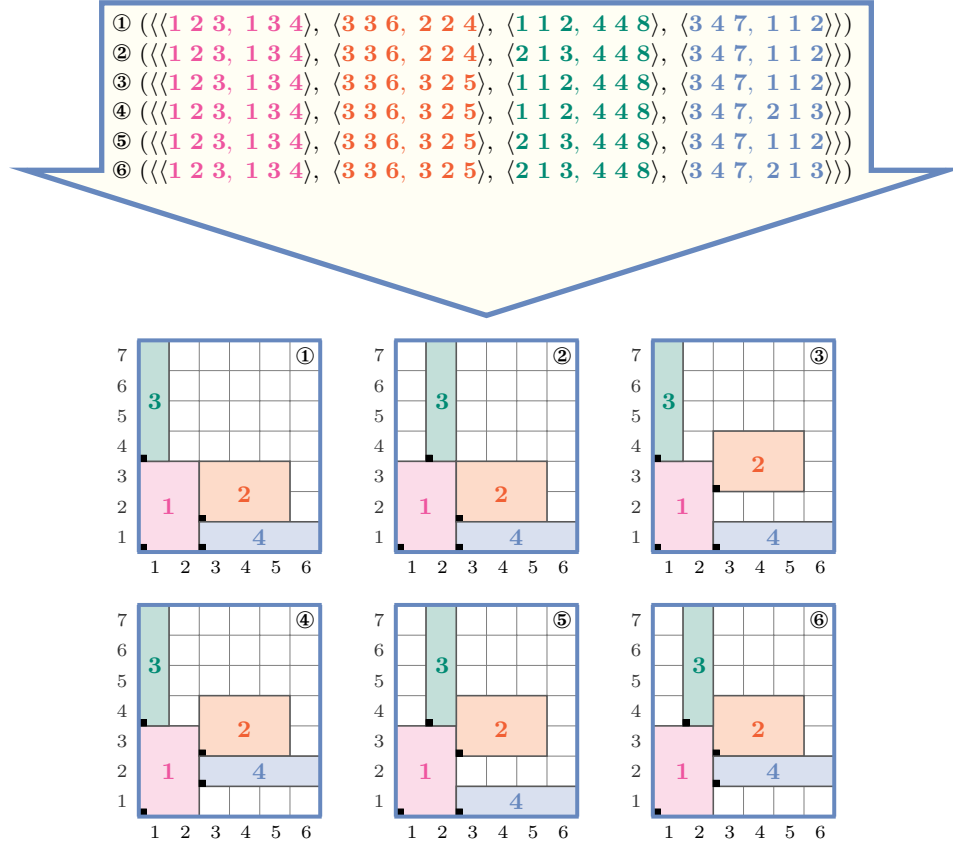


Figure 5.279: All solutions corresponding to the non ground example of the `diffn` constraint of the **All solutions** slot

Remark

Unlike the definition of the **Purpose** slot the original paper [41] introducing the `diffn` constraint imposes all `orthotopes` sizes to be different from 0. But it is convenient to allow variable sizes which can be assigned value 0 to model the fact that an orthotope can be skipped.

When we have segments (respectively rectangles) the `diffn` constraint is referenced under the name `disjoint1` (respectively `disjoint2`) in **SICStus Prolog** [99]. When we have rectangles the `diffn` constraint is also called `diff2` in **JaCoP**. In **MiniZinc** (<http://www.minizinc.org/>) the `diffn` constraint considers only rectangles.

It was shown in [406, 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.

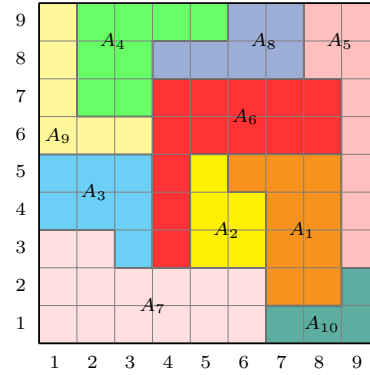


Figure 5.280: A solution for the angle packing problem of items $A_1 = [2, 4, 3, 1]$, $A_2 = [2, 2, 1, 3]$, $A_3 = [1, 3, 3, 2]$, $A_4 = [2, 1, 4, 3]$, $A_5 = [1, 7, 2, 2]$, $A_6 = [1, 2, 5, 5]$, $A_7 = [6, 2, 2, 3]$, $A_8 = [4, 2, 2, 1]$, $A_9 = [3, 1, 1, 4]$, $A_{10} = [3, 2, 1, 1]$.

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 non zero size orthotopes 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 [361] is to represent explicitly the two-dimensional domain of the origin of each rectangle by a [quadtree](#) [367] 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 [32].

The thesis of J. Nelissen [292] 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 [44].

Extensions of the non-overlapping constraint to polygons and to more complex shapes are respectively described in [44] and in [356]. Specialised propagation algorithms for the [squared squares](#) problem [85] (based on the fact that no waste is permitted) are given in [181] and in [180].

The [cumulative](#) constraint can be used as a *necessary condition* for the `diffn` constraint. Figure 5.282 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.



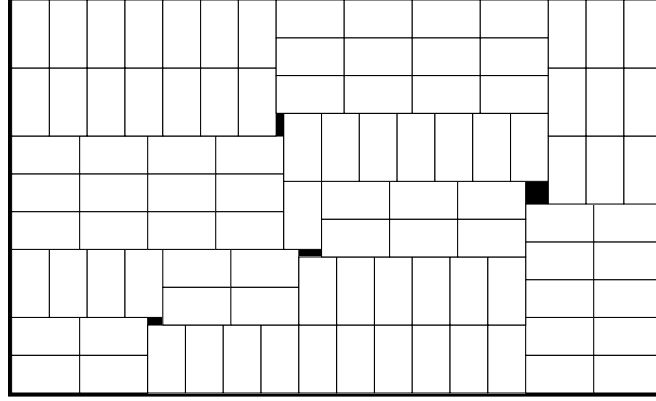



Figure 5.281: A hard instance from [292, page 165]: A solution for packing 99 rectangles of size 5×9 into a rectangle of size 86×52

The **cumulative** constraint is a necessary *but not sufficient* condition for the two-dimensional case of the **diffn** constraint. Figure 5.283 illustrates this point on an example taken from [77] where one has to place the 8 rectangles $R_1, R_2, R_3, R_4, R_5, R_6, R_7, R_8$ 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 1×2 in a big rectangle of size 12×4 . As shown by Figure 5.283 there is a cumulative solution where R_8 is split in two parts but M. Hujter proves in [221] that there is no solution where no rectangle is split. 

In the context of n parallelepipeds that have to be packed [187, 261] within a box of sizes $X \times Y \times Z$ one can proceed as follows for stating three **cumulative** constraints. The i^{th} (with $i \in [1, n]$) parallelepiped is described by the following attributes:

- ox_i, oy_i, oz_i (with $i \in [1, n]$) the coordinates of its origin on the x, y and z -axes.
- sx_i, sy_i, sz_i (with $i \in [1, n]$) its sizes on the x, y and z -axes.
- px_i, py_i, pz_i (with $i \in [1, n]$) the surfaces of its projections onto 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:

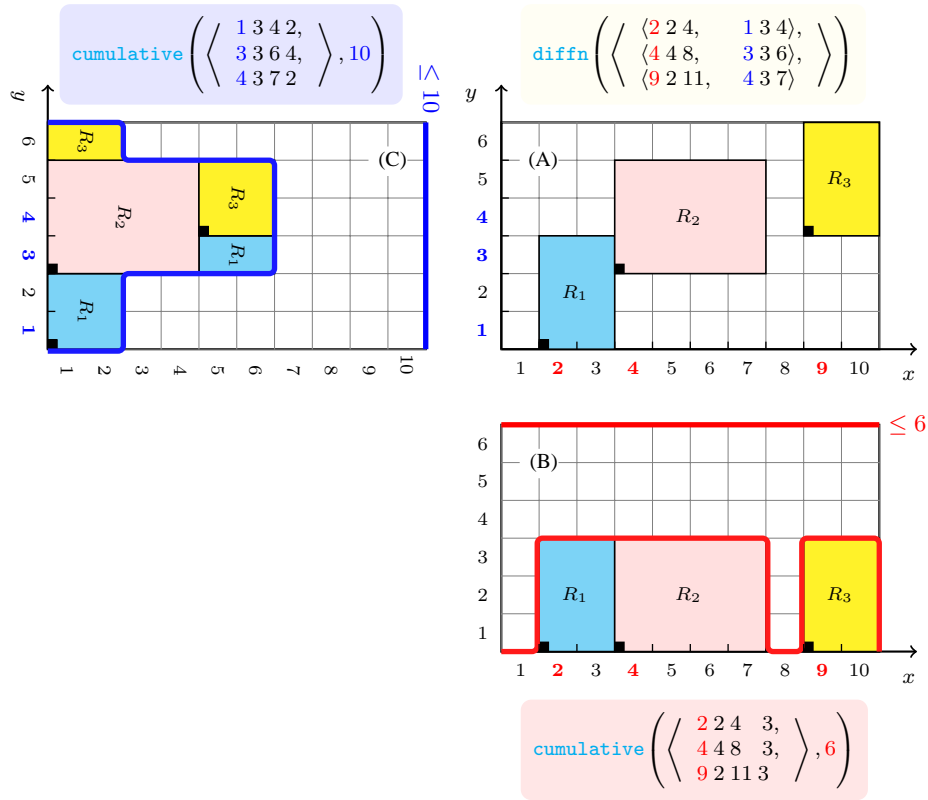


Figure 5.282: Looking from the perspective of the cumulative constraint in a two-dimensional rectangles placement problem: projecting the three rectangles of (A) on the x axis (B) and on the y axis (C)

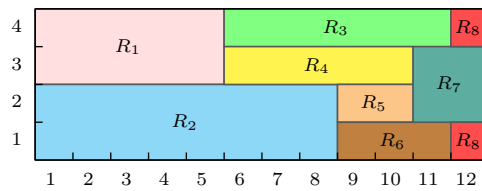


Figure 5.283: Illustrating the necessary but not sufficient placement condition: the rectangle R_8 is split in two parts

- $4 = 2 + 2$ (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**, `nooverlap` in **Gecode**, `diff2` in **JaCoP**, `diff` in **JaCoP**, `disjoint` in **JaCoP**, `disjointconditional` in **JaCoP**, `diffn` in **MiniZinc**.

Used in

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

See also

common keyword: `calendar` (*multi-site employee scheduling with calendar constraints, 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 height 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, multi-site employee scheduling with calendar constraints, 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, multi-site employee scheduling with calendar constraints, 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	$SELF \mapsto \text{collection}(\text{orthotopes})$
Arc arity	1
Arc constraint(s)	$\text{orth_link_ori_siz_end}(\text{orthotopes.orth})$
Graph property(ies)	$\overline{NARC} = \text{ORTHOTOPES} $
Arc input(s)	ORTHOTOPES
Arc generator	$CLIQUE(\neq) \mapsto \text{collection}(\text{orthotopes1}, \text{orthotopes2})$
Arc arity	2
Arc constraint(s)	$\text{two_orth_do_not_overlap}(\text{orthotopes1.orth}, \text{orthotopes2.orth})$
Graph property(ies)	$\overline{NARC} = \text{ORTHOTOPES} * \text{ORTHOTOPES} - \text{ORTHOTOPES} $

Graph model

The diffn constraint is expressed by using two graph constraints:

- The first graph constraint forces 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 5.284 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.

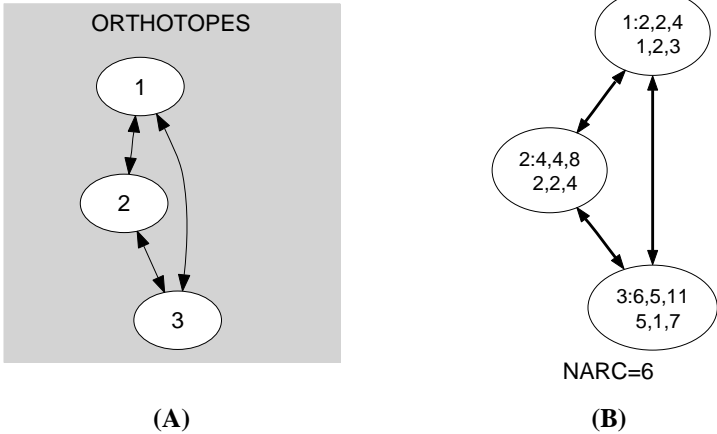


Figure 5.284: Initial and final graph of the diffn constraint

Signature

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

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

Quiz

EXERCISE 1 (checking whether a ground instance holds or not)^a

- A. Does the constraint $\text{diffn}(\langle\langle 1 \ 1 \ 1, 2 \ 2 \ 4 \rangle, \langle 5 \ 1 \ 6, 4 \ 1 \ 5 \rangle\rangle)$ hold?
- B. Does the constraint $\text{diffn}(\langle\langle 2 \ 2 \ 4, 3 \ 2 \ 5 \rangle, \langle 4 \ 3 \ 7, 4 \ 2 \ 6 \rangle, \langle 8 \ 2 \ 10, 2 \ 3 \ 5 \rangle\rangle)$ hold?
- C. Does the constraint $\text{diffn}(\langle\langle 2 \ 2 \ 4, 2 \ 2 \ 4 \rangle, \langle 4 \ 5 \ 9, 4 \ 2 \ 6 \rangle, \langle 8 \ 2 \ 10, 2 \ 3 \ 5 \rangle\rangle)$ hold?
- D. Does the constraint $\text{diffn}(\langle\langle 3 \ 2 \ 5 \rangle, \langle 4 \ 0 \ 4 \rangle, \langle 6 \ 3 \ 9 \rangle\rangle)$ hold?

^aHint: go back to the definition of diffn .

EXERCISE 2 (finding all solutions)^a

Give all the solutions to the constraint:

$$\left\{ \begin{array}{llll} OX_1 \in [1, 5], & EX_1 \in [1, 5], & OY_1 \in [1, 5], & EY_1 \in [1, 5], \\ OX_2 \in [1, 5], & EX_2 \in [1, 5], & OY_2 \in [1, 5], & EY_2 \in [1, 5], \\ OX_3 \in [1, 5], & EX_3 \in [1, 5], & OY_3 \in [1, 5], & EY_3 \in [1, 5], \\ \text{diffn} \left(\left\langle \begin{array}{ll} \langle OX_1 \ 1 \ EX_1, & OY_1 \ 3 \ EY_1 \rangle, \\ \langle OX_2 \ 4 \ EX_2, & OY_2 \ 1 \ EY_2 \rangle, \\ \langle OX_3 \ 3 \ EX_3, & OY_3 \ 3 \ EY_3 \rangle \end{array} \right\rangle \right) \end{array} \right\}.$$

^aHint: consider rectangles by decreasing surface and focus on the coordinates of their origins (OX_3, OY_3) , (OX_2, OY_2) and (OX_1, OY_1) ; enumerate solutions in lexicographic order of (OX_3, OY_3) .

EXERCISE 3 (finding the unique solution)^a

Find the unique solution to the constraint:

$$\left\{ \begin{array}{l} \begin{array}{llll} OX_1 \in [1, 8], & EX_1 \in [1, 8], & OY_1 \in [1, 8], & EY_1 \in [1, 8], \\ OX_2 \in [1, 8], & EX_2 \in [1, 8], & OY_2 \in [1, 8], & EY_2 \in [1, 8], \\ OX_3 \in [1, 8], & EX_3 \in [1, 8], & OY_3 \in [1, 8], & EY_3 \in [1, 8], \\ OX_4 \in [1, 8], & EX_4 \in [1, 8], & OY_4 \in [1, 8], & EY_4 \in [1, 8], \\ OX_5 \in [1, 8], & EX_5 \in [1, 8], & OY_5 \in [1, 8], & EY_5 \in [1, 8], \end{array} \\ \text{diffn} \left(\left\langle \begin{array}{ll} \langle OX_1 \ 2 \ EX_1, & OY_1 \ 5 \ EY_1 \rangle, \\ \langle OX_2 \ 5 \ EX_2, & OY_2 \ 1 \ EY_2 \rangle, \\ \langle OX_3 \ 2 \ EX_3, & OY_3 \ 4 \ EY_3 \rangle, \\ \langle OX_4 \ 4 \ EX_4, & OY_4 \ 2 \ EY_4 \rangle, \\ \langle OX_5 \ 3 \ EX_5, & OY_5 \ 3 \ EY_5 \rangle, \\ \langle 1 \ 3 \ 4, & 1 \ 3 \ 4 \rangle \end{array} \right\rangle \right) \end{array} \right.$$

^aHint: reason on whose compulsory parts of the projections of the rectangles onto the x and y axes.

EXERCISE 4 (degrees of violation for non-overlapping)^aA. Give the variable-based degree of violation^b of the constraint

$$\text{diffn} \left(\left\langle \begin{array}{ll} \langle 3 \ 2 \ 5, & 4 \ 2 \ 6 \rangle, \\ \langle 4 \ 5 \ 9, & 5 \ 2 \ 7 \rangle, \\ \langle 6 \ 2 \ 8, & 3 \ 5 \ 8 \rangle, \\ \langle 7 \ 1 \ 8, & 2 \ 4 \ 6 \rangle \end{array} \right\rangle \right).$$

B. Give the decomposition-based degree of violation^c of the same constraint.

C. In the decomposition-based degree of violation each violated binary constraint contributes +1 to the degree of violation, which does not consider how much two orthotopes overlap. For the same constraint give the *overlap decomposition-based degree of violation* where the degree of violation of a binary no-overlap constraint is equal to the overlap between the corresponding orthotopes.

D. Given two fixed orthotopes of the **diffn** constraint, propose a formula for computing how much these orthotopes overlap.

^aHint: focus on the rectangles that overlap; by changing one coordinate of a rectangle one can move it (A); count how many rectangles overlap (B); count how much each pair of rectangles overlaps (C).

^bGiven a constraint for which all variables are fixed, the *variable-based degree of violation* is the minimum number of variables to assign differently in order to satisfy the constraint.

^cGiven a constraint that can be decomposed in a conjunction of binary constraints, the *decomposition-based degree of violation* is the number of binary constraints that do not hold.

SOLUTION TO EXERCISE 1

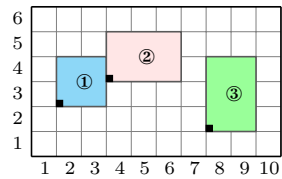
A. No, since the first rectangle $\langle 1 \ 1 \ 1, 2 \ 2 \ 4 \rangle$ is not well formed, i.e., $1 + 1 \neq 1$.

B. Yes, since the three rectangles do not overlap:

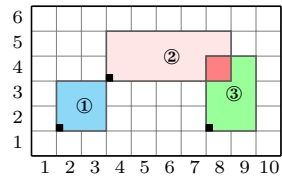
- Rectangles ① and ② do not overlap since their projections onto the x axis do not overlap,
- Rectangles ② and ③ do not overlap since their projections onto the x axis do not overlap,
- Rectangles ① and ③ do not overlap since their projections onto the x axis do not overlap.

C. No, since rectangles ② and ③ overlap, i.e., their projections onto the x and y axes both overlap.

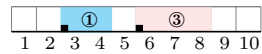
D. Yes, since line segments ① and ③ do not overlap and since the size of line segment ② is zero.



B.



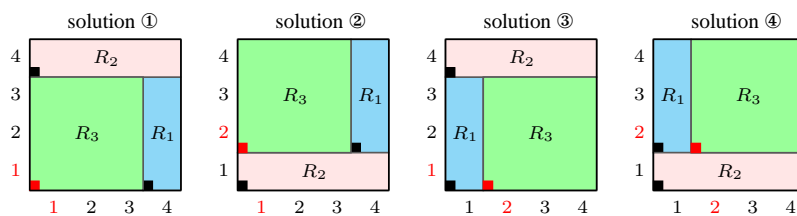
C.



D.

SOLUTION TO EXERCISE 2*the four solutions (once R_3 is fixed, R_2 and R_1 are also fixed)*

- $\langle \langle OX_1 1EX_1, OY_1 3EY_1 \rangle, \langle OX_2 4EX_2, OY_2 1EY_2 \rangle, \langle OX_3 3EX_3, OY_3 3EY_3 \rangle \rangle$
 ① $(\langle \langle 4 \ 1 \ 5, 1 \ 3 \ 4 \rangle, \langle 1 \ 4 \ 5, 4 \ 1 \ 5 \rangle, \langle 1 \ 3 \ 4, 1 \ 3 \ 4 \rangle \rangle)$
 ② $(\langle \langle 4 \ 1 \ 5, 2 \ 3 \ 5 \rangle, \langle 1 \ 4 \ 5, 1 \ 1 \ 2 \rangle, \langle 1 \ 3 \ 4, 2 \ 3 \ 5 \rangle \rangle)$
 ③ $(\langle \langle 1 \ 1 \ 2, 1 \ 3 \ 4 \rangle, \langle 1 \ 4 \ 5, 4 \ 1 \ 5 \rangle, \langle 2 \ 3 \ 5, 1 \ 3 \ 4 \rangle \rangle)$
 ④ $(\langle \langle 1 \ 1 \ 2, 2 \ 3 \ 5 \rangle, \langle 1 \ 4 \ 5, 1 \ 1 \ 2 \rangle, \langle 2 \ 3 \ 5, 2 \ 3 \ 5 \rangle \rangle)$



SOLUTION TO EXERCISE 3

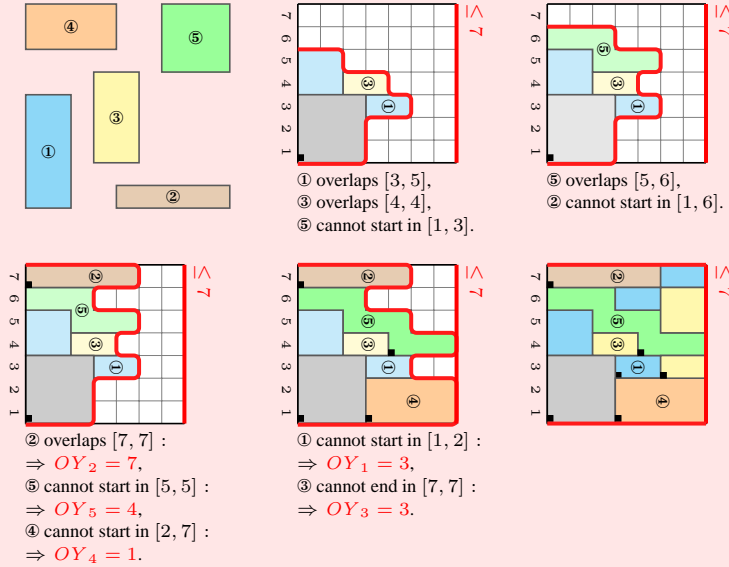
We go through the following reasoning steps:

A. [SELECTING THE x OR y AXIS FOR REASONING]

Among the rectangles that have the largest side lengths, i.e., $R_1 : 2 \times 5$ and $R_2 : 5 \times 1$, we select the rectangle that has the largest surface, i.e., R_1 . Since the largest size of R_1 is located in dimension y we first choose to reason on the compulsory parts of the projections of the rectangles onto the y axis.

B. [REASONING ON THE PROJECTIONS ONTO THE y AXIS]

We focus on the projections of the rectangles onto the y axis and gradually build the cumulated profile of their compulsory parts. Fixed (respectively not completely fixed) projections use a saturated (respectively unsaturated) colour. The gray square corresponds to an initially fixed square.

**C. [FILTERING A RECTANGLE WRT ONE OTHER RECTANGLE]**

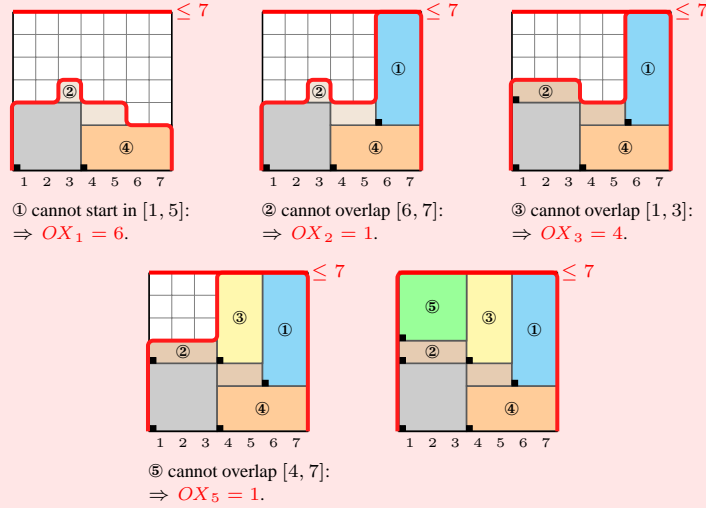
A no-overlap constraint between two rectangles can be represented as a disjunction with four alternatives:

- on the x axis the first rectangle ends before the start of the second rectangle,
- on the x axis the second rectangle ends before the start of the first rectangle,
- on the y axis the first rectangle ends before the start of the second rectangle,
- on the y axis the second rectangle ends before the start of the first rectangle.

If we consider the fixed 3×3 gray square and the rectangle R_4 we have $4 \leq OX_4 \vee EX_4 \leq 1 \vee 4 \leq 1 \vee 3 \leq 1$. The part $4 \leq 1 \vee 3 \leq 1$ does not hold. Since the minimum value of EX_4 is equal to 5 the inequality $EX_4 \leq 1$ does also not hold. Consequently $4 \leq OX_4$ must hold and the minimum value of OX_4 is equal to 4. Moreover since the maximum value of OX_4 is 4 we have $OX_4 = 4$.

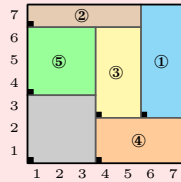
SOLUTION TO EXERCISE 3 (continued)**D.** [REASONING ON THE PROJECTIONS ONTO THE x AXIS]

We focus on the projections of the rectangles onto the x axis and gradually build the cumulated profile of their compulsory parts. Fixed (respectively not completely fixed) projections use a saturated (respectively unsaturated) colour.

**E.** [PUTTING THINGS TOGETHER]

For any pair of distinct rectangles i, j we check

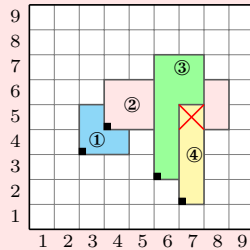
$EX_i \leq OX_j \vee EX_j \leq OX_i \vee EY_i \leq OY_j \vee EY_j \leq OY_i$, i.e., there exists at least one dimension where the projections of the two rectangles do not overlap. We obtain the following unique solution.



SOLUTION TO EXERCISE 4

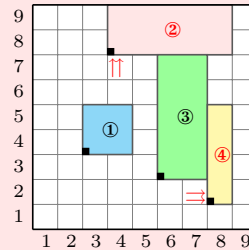
- A.** Since point $(7, 5)$ is included in the three rectangles ②, ③, ④, (see Figure (A1)), we need to modify the attributes of at least two rectangles. Figure (A2) shows a solution where only rectangles ② and ④ are translated. Therefore the variable-based degree of violation is equal to 2.

$(7, 5) \in \text{②, ③, ④} \Rightarrow$
has to move at least
two rectangles



(A1)

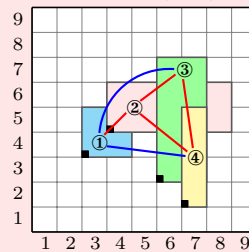
A solution with only
two translations: ②, ④



(A2)

- B.** Figure (B) shows the constraint network associated with the decomposition where each vertex corresponds to a rectangle and each edge to a binary no-overlap constraint between two rectangles. Edges where the corresponding binary constraint holds (respectively does not hold) are coloured in blue (in red). Consequently the decomposition-based degree of violation is equal to 4 (i.e., each pair (①, ②) , (②, ③) , (②, ④) , (③, ④) corresponds to two overlapping rectangles).

Clique graph associated
with the decomposition
with 3 violations (in red)

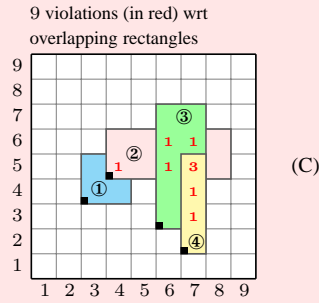


(B)

SOLUTION TO EXERCISE 4 (continued)

C. As illustrated by Figure (C), the overlap decomposition-based degree of violation is equal to 9 since:

- the overlap between rectangles ① and ② is equal to 1,
- the overlap between rectangles ② and ③ is equal to 4,
- the overlap between rectangles ② and ④ is equal to 1,
- the overlap between rectangles ③ and ④ is equal to 3,
- the other pairs of rectangles do not overlap.



D. Given two orthotopes i and j , defined by their end and their origin in each dimension, their overlap is defined by

$$\prod_{d=1}^{|\text{ORTHOTOPE}|} \max(0, \min(\text{end}_{i,d}, \text{end}_{j,d}) - \max(\text{ori}_{i,d}, \text{ori}_{j,d})).$$

20000128

1061