

A Generic Geometrical Constraint Kernel in Space and Time for Handling Polymorphic k -Dimensional Objects

N. Beldiceanu¹, M. Carlsson², E. Poder¹, R. Sadek¹, and C. Truchet³

¹ **École des Mines de Nantes, LINA FRE CNRS 2729, FR-44307, France**
`{Nicolas.Beldiceanu, Emmanuel.Poder, Rida.Sadek}@emn.fr`

² **SICS, P.O. Box 1263, SE-164 29 Kista, Sweden**
`Mats.Carlsson@sics.se`

³ **Université de Nantes, LINA FRE CNRS 2729, FR-44322, Nantes, France**
`Charlotte.Truchet@univ-nantes.fr`

Introduction

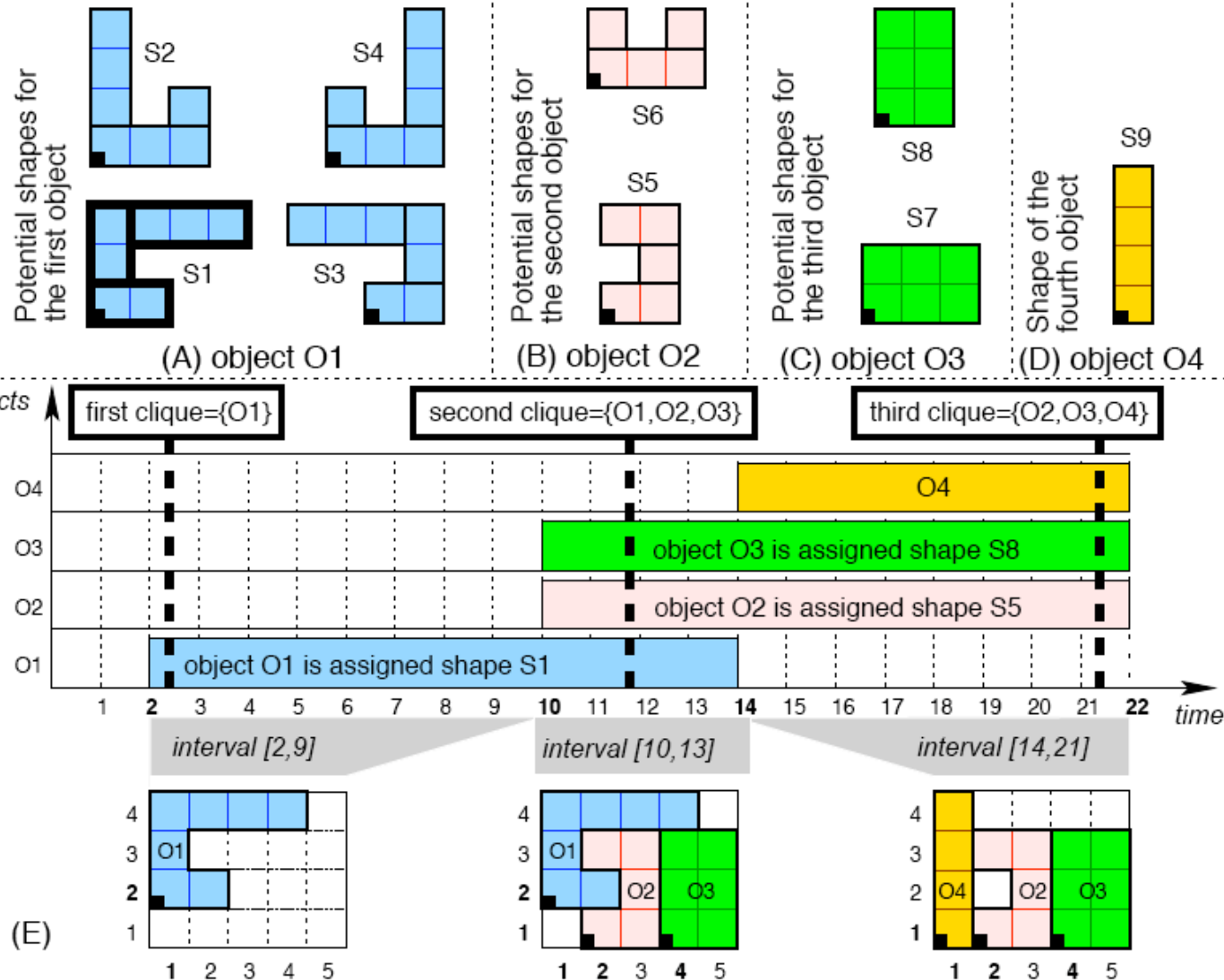
External Geometrical Constraints

Internal Geometrical Constraints

The Propagation Kernel

A First Evaluation

A Generic Placement Kernel: *geost*



A Generic Placement Kernel: *geost*

```
geost(2,
```

Number of dimensions

```
[object(1,1,[1,2], 2,12,14),
 object(2,5,[2,1],10,12,22),
 object(3,8,[4,1],10,12,22),
 object(4,9,[1,1],14, 8,22)],
```

Object Id, Shape Id, Origin, Start, Duration, End

Objects

Additional attributes (type, weight, customer, ...) can eventually be added

```
sbox(1,[0,0],[2,1]), sbox(1,[0,1],[1,2]), sbox(1,[1,2],[3,1]),
sbox(2,[0,0],[3,1]), sbox(2,[0,1],[1,3]), sbox(2,[2,1],[1,1]),
sbox(3,[0,0],[2,1]), sbox(3,[1,1],[1,2]), sbox(3,[2,2],[3,1]),
sbox(4,[0,0],[3,1]), sbox(4,[0,1],[1,1]), sbox(4,[2,1],[1,3]),
sbox(5,[0,0],[2,1]), sbox(5,[1,1],[1,1]), sbox(5,[0,2],[2,1]),
sbox(6,[0,0],[3,1]), sbox(6,[0,1],[1,1]), sbox(6,[2,1],[1,1]),
sbox(7,[0,0],[3,2]),
sbox(8,[0,0],[2,3]),
sbox(9,[0,0],[1,4]),
```

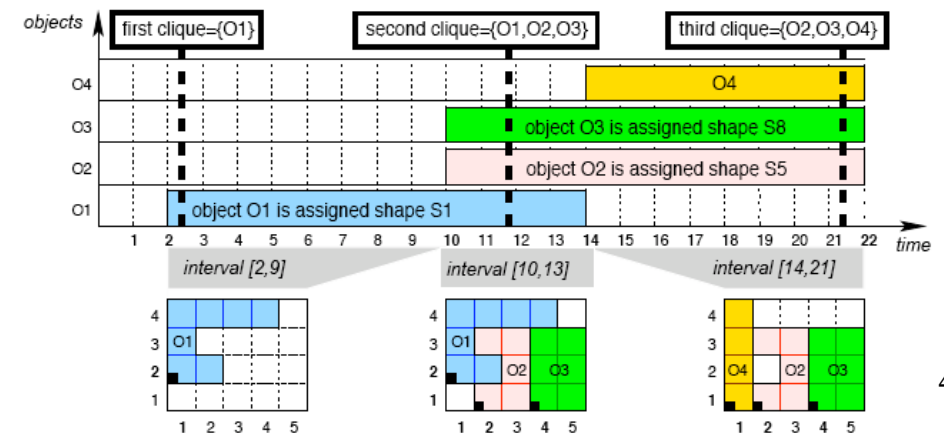
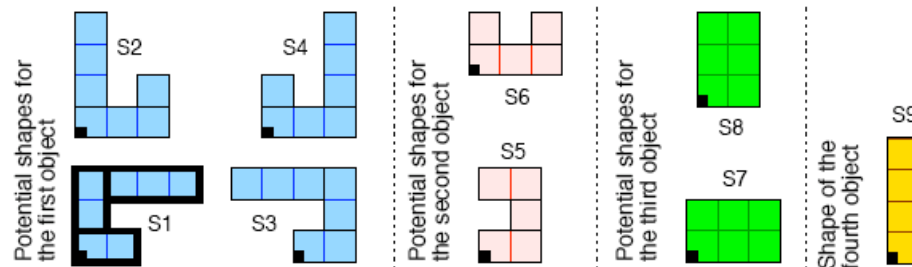
Potential shapes, where a shape is defined by a set of sboxes sharing the same shape id

```
[non-overlapping([0,1],[1,2,3,4]),included([0,1],[1,2,3,4],[1,1],[5,4])]
```

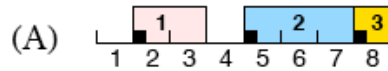
Attributes Objects

Attributes Objects Box

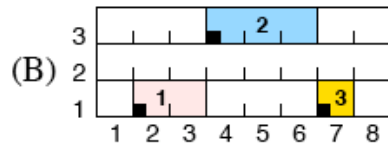
List of external constraints



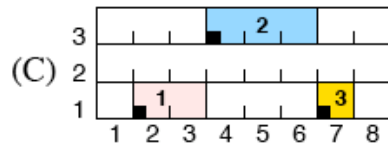
Applications



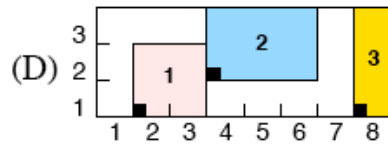
(A) disjunctive



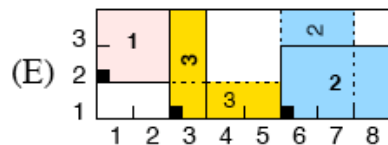
(B) machine assignment



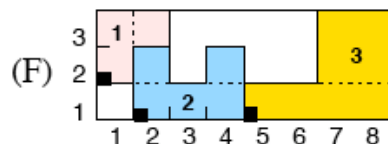
(C) machine assignment
(machine dependant duration)



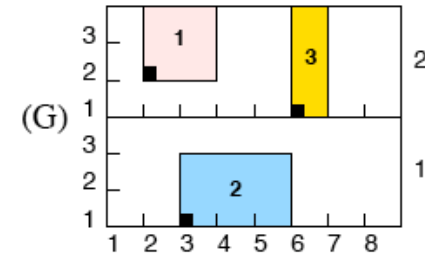
(D) 2D non-overlapping
(fixed orientation)



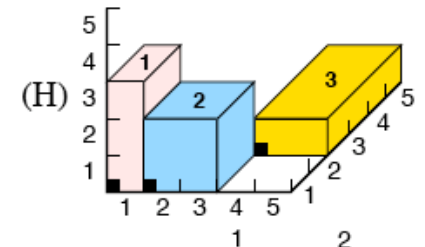
(E) 2D non-overlapping
(90° rotation)



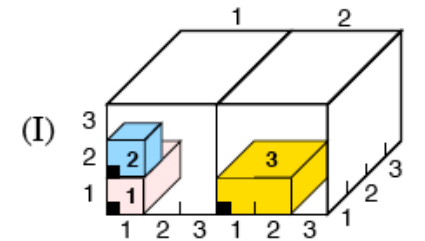
(F) 2D non-overlapping
(irregular shapes)



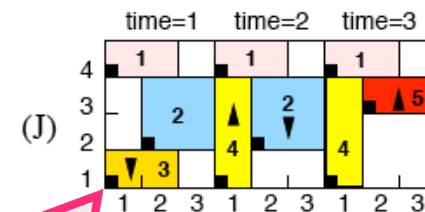
(G) 2D non-overlapping
and assignment



(H) 3D non-overlapping



(I) 3D non-overlapping
and assignment



(J) pick-up delivery

```
geost(2, [object(1,1,[1,4]),1,3,4), object(2,2,[2,2]),1,2,3), object(3,1,[1,1]),1,1,2),
object(4,3,[1,1]),2,2,4), object(5,1,[2,3]),3,1,4)],
[sbox(1,[0,0],[2,1]), sbox(2,[0,0],[2,2]), sbox(3,[0,0],[1,3])],
[non-overlapping([0,1],[1,2,3,4,5])])
```

Mixing Constraints on Several Dimensions

EXAMPLE OF PROBLEM

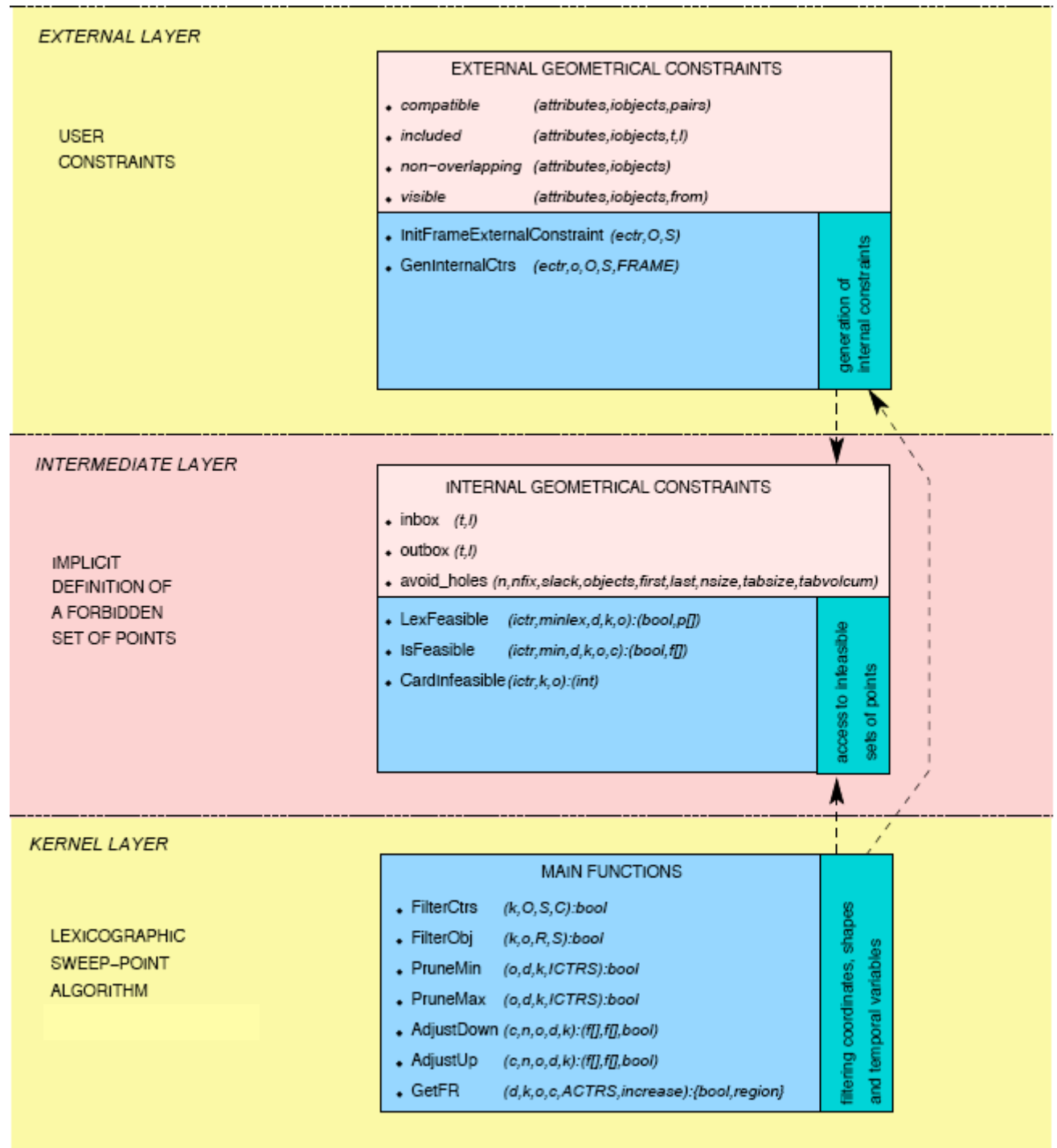
Input: A set of parallelepipeds P and a subset P' of P

Constraints: (1) all parallelepipeds of P should not overlap
(2) no parallelepipeds of P' should be piled

Solution with *geost*: *non-overlapping*([0, 1, 2], \mathcal{P})

non-overlapping([0, 1], \mathcal{P}')

Overall Architecture



Introduction

External Geometrical Constraints

Internal Geometrical Constraints

The Propagation Kernel

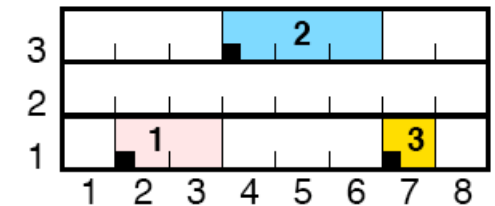
A First Evaluation

Example of External Constraint: *compatible*

Define the **possible pairs** for two given attributes

EXAMPLE

```
geost(2, [object(1,1,[2,1],0,1,1),object(2,4,[4,3],0,1,1),object(3,5,[7,1],0,1,1)],
        [shape(1,[0,0],[2,1]),shape(2,[0,0],[3,1]),
         shape(3,[0,0],[2,1]),shape(4,[0,0],[3,1]),
         shape(5,[0,0],[1,1]),shape(6,[0,0],[2,1]),
         non-overlapping([0,1],[1,2,3]), compatible([sid,1],[1,2,3],[1-1,2-2,3-2,4-3,5-1,6-2])
```



Define the compatibility between the **shape id** and the **origin in dimension 1**
(i.e., *which duration should we have according to the machine to which a task is assigned*)

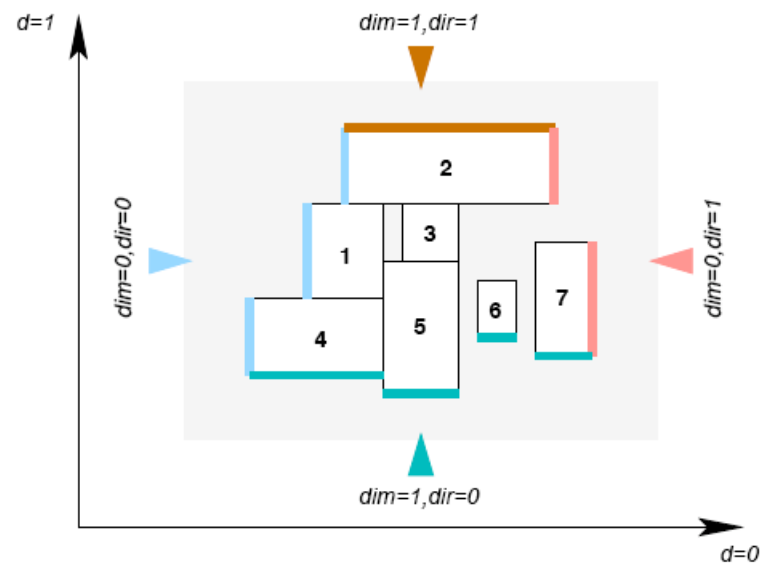
- shape 1 can only be used on machine 1
- shape 2 can only be used on machine 2
- shape 3 can only be used on machine 2
- shape 4 can only be used on machine 3
- shape 5 can only be used on machine 1
- shape 6 can only be used on machine 2

Example of External Constraint: *visible*

IDEA

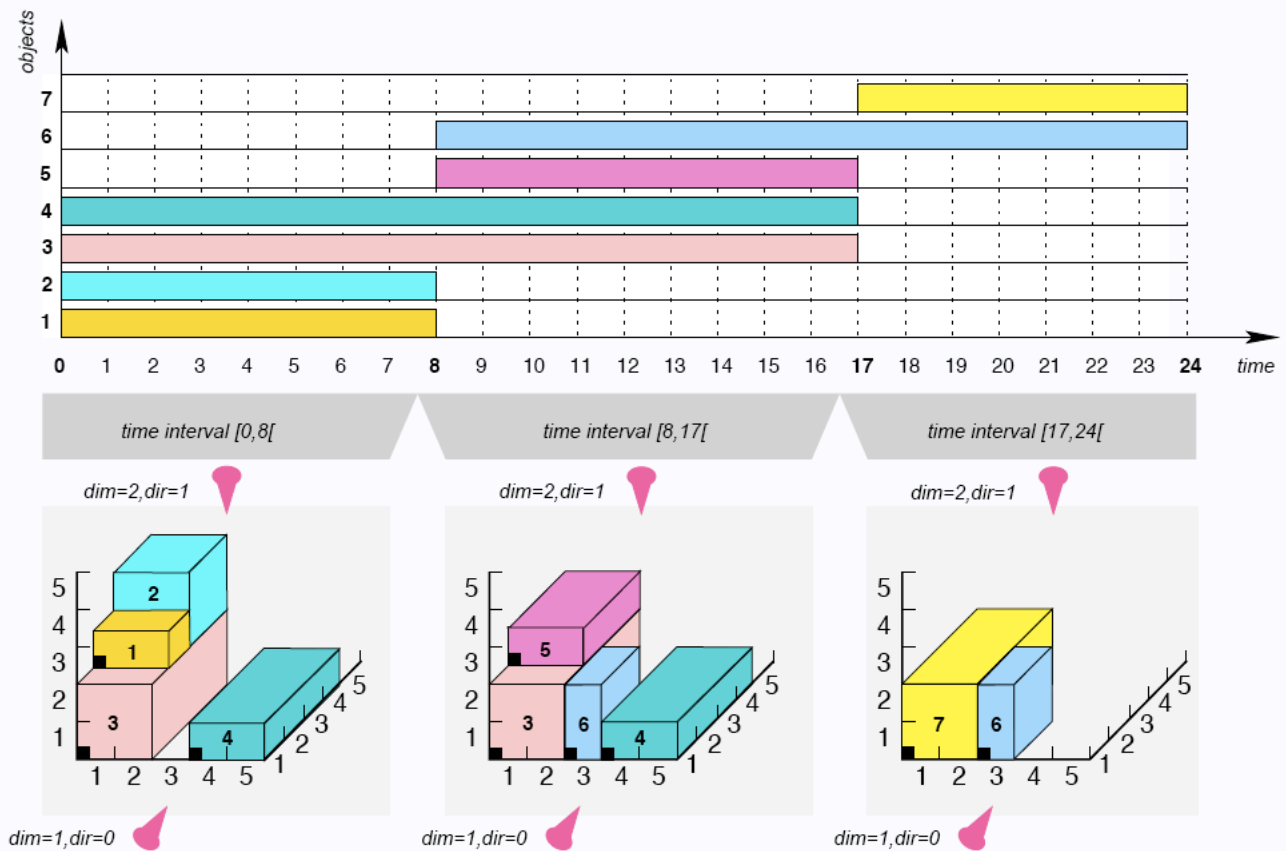
Given a set of potential observations places P , and given for each box a set of visible faces, the **visible** constraint specifies that at least one visible face of each box should be entirely visible from at least one observation place of P at the start and end(-1) time associated to the box.

**Completely visible faces
from a set of observations points**



Application of *visible*: pick-up delivery

```
geost(3, [object(1,1,[1,2,3], 0, 8, 8), object(2,2,[1,3,3],0,8, 8), object(3,3,[1,1,1],0,17,17),
object(4,4,[4,1,1], 0,17,17), object(5,5,[1,2,3],8,9,17), object(6,6,[3,1,1],8,12,24),
object(7,3,[1,1,1],17, 7,24)],
[shape(1,[0,0,0],[2,1,1],face- [<1,0>,<2,1>]), shape(2,[0,0,0],[2,2,2],face- [<1,0>,<2,1>]),
shape(3,[0,0,0],[2,4,2],face- [<1,0>,<2,1>]), shape(4,[0,0,0],[2,4,1],face- [<1,0>,<2,1>]),
shape(5,[0,0,0],[2,3,1],face- [<1,0>,<2,1>]), shape(6,[0,0,0],[1,2,2],face- [<1,0>,<2,1>])],
[non-overlapping([0,1,2],[1,2,3,4,5,6,7]),
visible([0,1,2,face],[1,2,3,4,5,6,7],[<1,0>]), visible([0,1,2,face],[1,2,3,4,5,6,7],[<2,1>])])
```



Introduction

External Geometrical Constraints

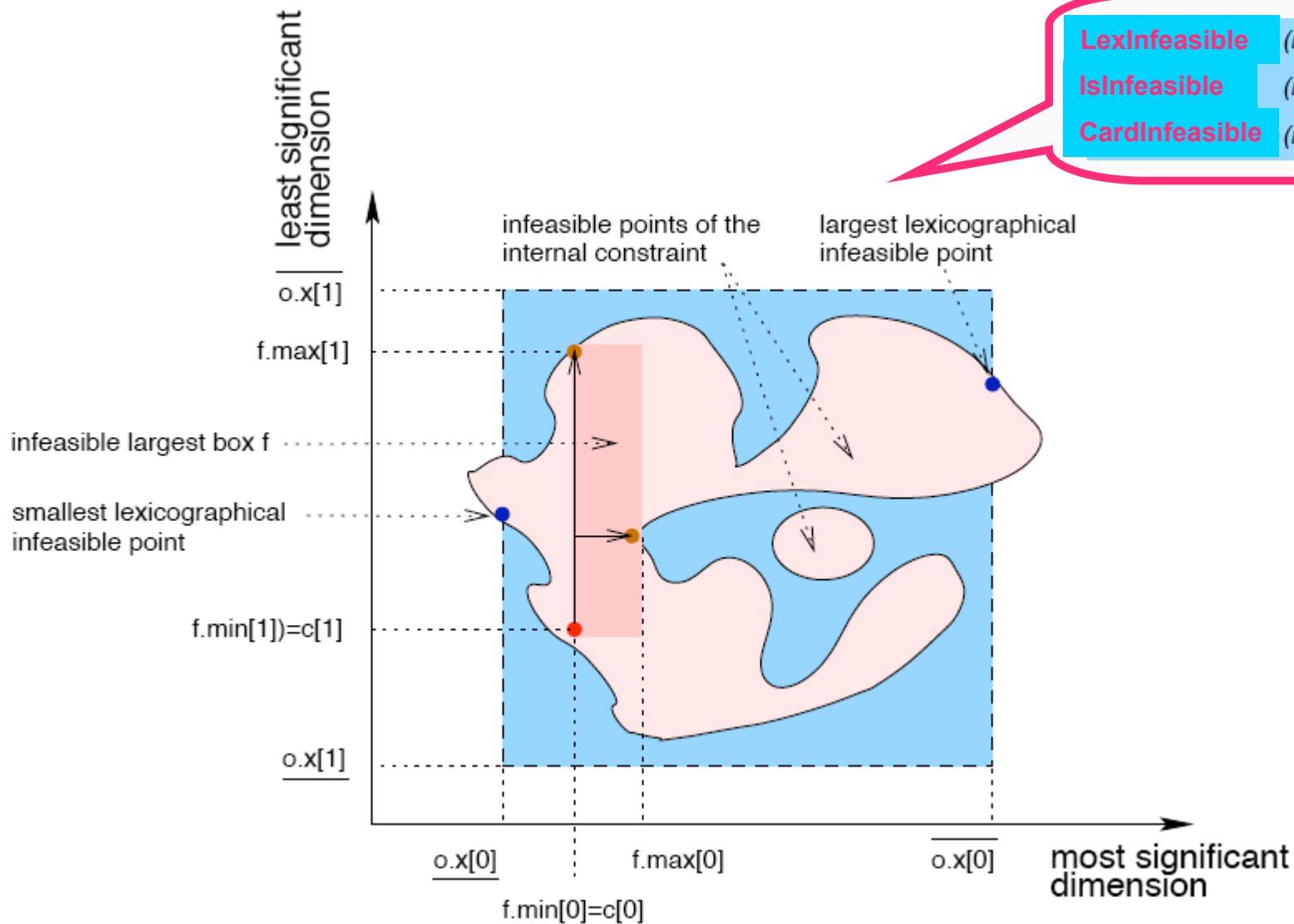
Internal Geometrical Constraints

The Propagation Kernel

A First Evaluation

Intermediate Layer

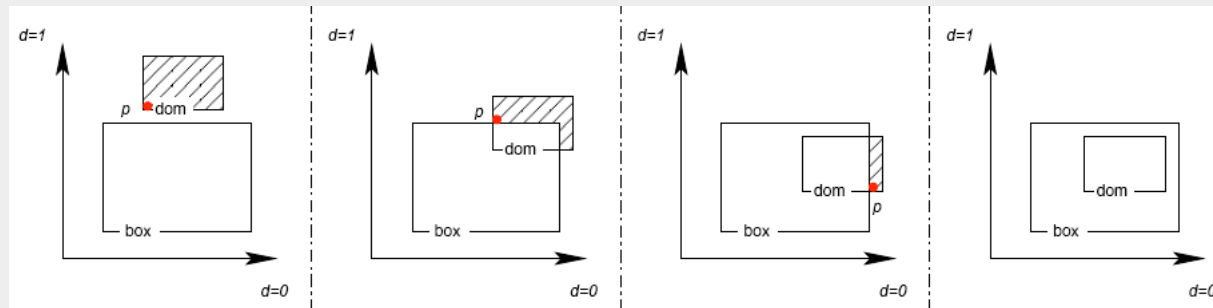
SERVICES ASSOCIATED TO AN INTERNAL CONSTRAINT (i.e., a set of forbidden points)



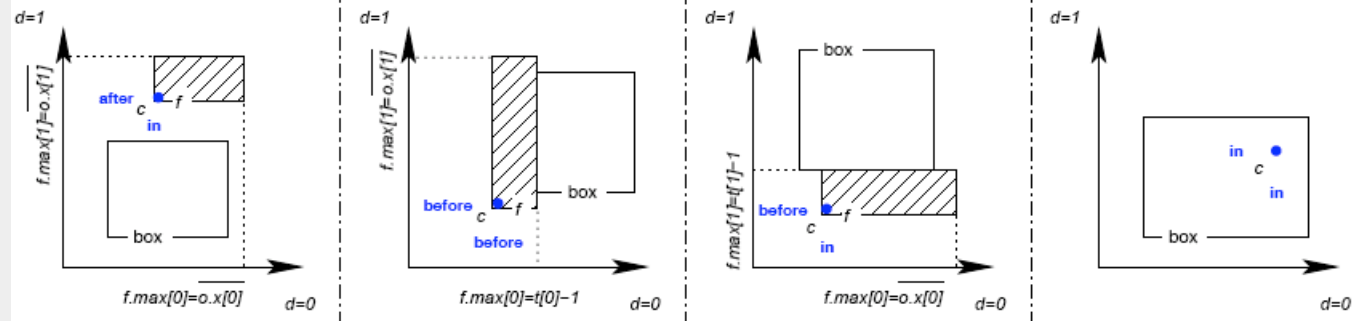
Example of Internal Constraint: *inbox*

The inbox constraint The $\text{inbox}(t, l)$ constraint (according to an object o of the *geost* constraint) is an internal constraint which enforces the point $o.x$ to be located inside the shifted box defined by its shift offset $t[d]$, $0 \leq d < k$, and sizes $l[d]$, $0 \leq d < k$ (i.e., $\forall d \in [0, k - 1] : t[d] \leq o.x[d] \leq t[d] + l[d] - 1$).

LexInfeasible



IsFeasible



Introduction

External Geometrical Constraints

Internal Geometrical Constraints

The Propagation Kernel

A First Evaluation

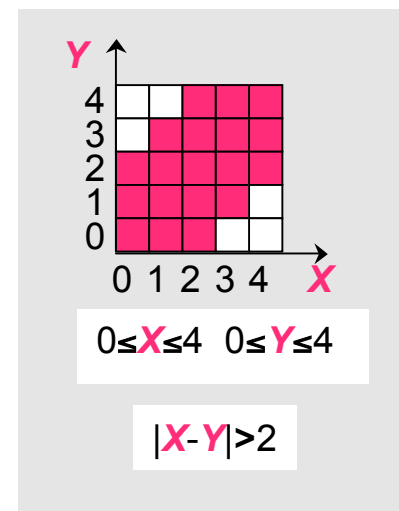
Communication between Constraints

SLOGAN OF CONSTRAINT PROGRAMMING

Constraints **communicate only** via the domains of their shared variables.

AN OTHER APPROACH

A constraint can be assimilated as a **set of forbidden points**, each variable corresponding to a dimension



PROBLEM: hard to **aggregate** sets of forbidden points associated to **different** constraints !!!

SOLUTION: set of forbidden points associated to different constraint should communicate everything is handled in an implicit way (**lazy evaluation**)

Sweep Algorithms in Computational Geometry

Standard technique for coming up with **efficient** algorithms

- Computational geometry, an introduction

[Preparata & Shamos, 1985]

- Computational Geometry, Algorithms and Applications

[Berg, Kreveld, Overmars & Schwarzkopf, 1997]

- Géométrie algorithmique

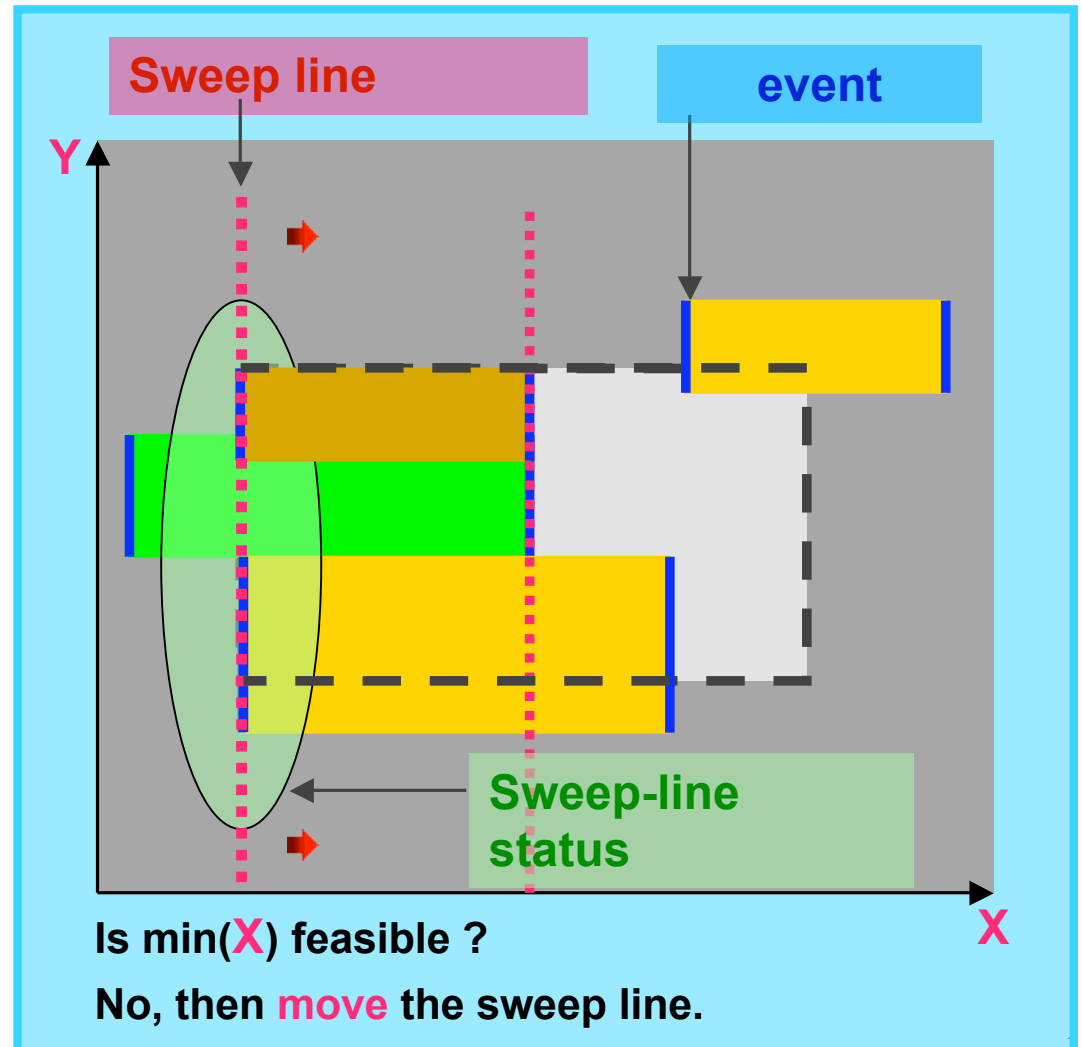
[Boissonnat & Yvinec, 1995]

Basic Idea of the Sweep Algorithm (in dimension 2)

Accumulates forbidden regions

$\left\{ \begin{array}{l} \text{CTR}_1(X, Y, \dots) \\ \text{CTR}_2(X, Y, \dots) \\ \dots \\ \text{CTR}_n(X, Y, \dots) \end{array} \right.$

sharing 2 given variables X and Y



Question: How to Generalize to k Dimensions ?

Key problem with the sweep-line status:

don't want to use a multi-dimensional data structure since it just kills scalability

Geometric Kernel : a Lexicographic Sweep-Point Algorithm

VARIABLES

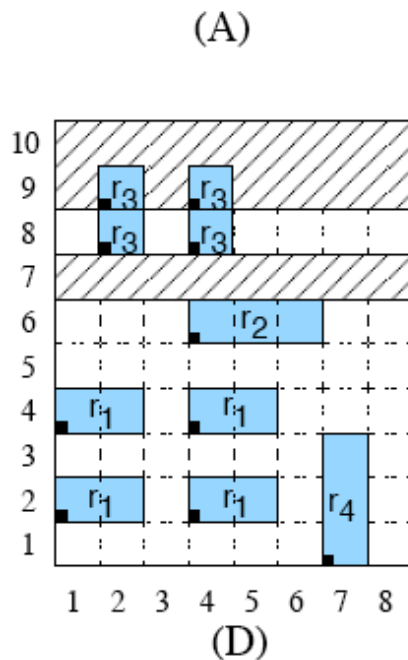
x1 in 1..4, y1 in 2..4
 x2 in 4..4, y2 in 6..6
 x3 in 2..4, y3 in 8..9
 x4 in 7..7, y4 in 1..1
 x5 in 1..8, y5 in 1..8, y5 <= 7

EXTERNAL CONSTRAINT (non-overlapping)

```
geost( [object(1,1,[x1,y1],0,1,1),object(2,2,[x2,y2],0,1,1),
        object(3,3,[x3,y3],0,1,1),object(4,4,[x4,y4],0,1,1),
        object(5,5,[x5,y5],0,1,1)],
        [shape(1,[0,2],[0,1]),shape(2,[0,3],[0,1]),shape(3,[0,1],[0,1]),
        shape(4,[0,1],[0,3]),shape(5,[0,5],[0,4])],
        [non-overlapping([0,1],[1,2,3,4,5])])
```

INTERNAL CONSTRAINTS GENERATED

FOR FILTERING THE ORIGIN OF THE FIFTH OBJECT, i.e. (x5,y5) (ICTRS)
 ctr1: outbox([1,1],[2,2]) ctr3: outbox([1,8],[2,1])
 ctr2: outbox([1,3],[6,4]) ctr4: outbox([3,1],[5,3])
 ctr5: outbox([1,7],[8,1])

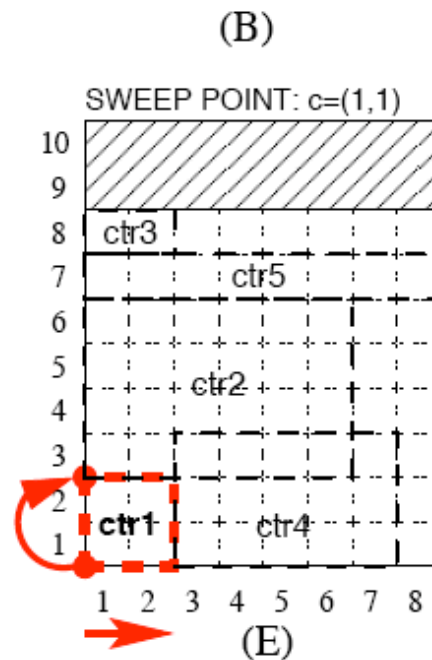


DCTRS
(delayed internal constraint)

- ctr1
- ctr2
- ctr5
- ctr3
- ctr4

ACTRS
(active internal constraint)

-



DCTRS

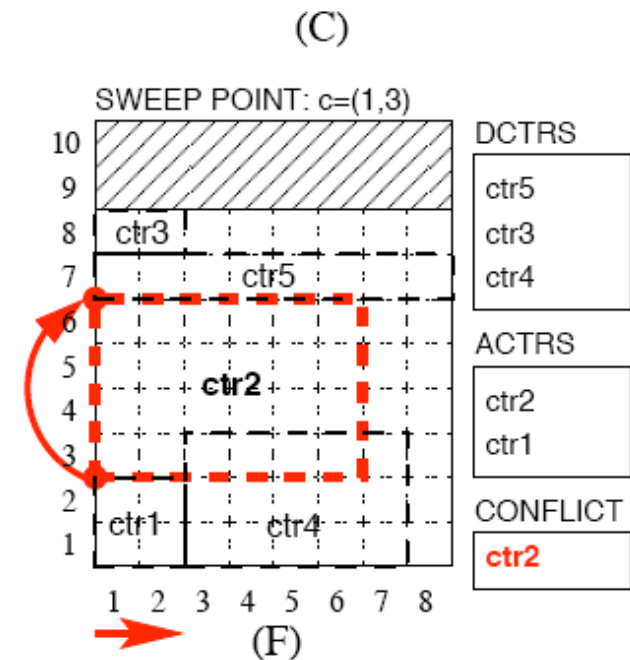
- ctr2
- ctr5
- ctr3
- ctr4

ACTRS

- ctr1

CONFLICT

- ctr1



DCTRS

- ctr5
- ctr3
- ctr4

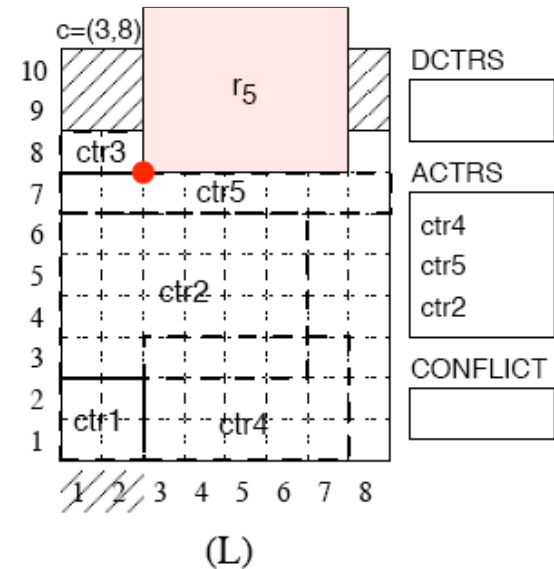
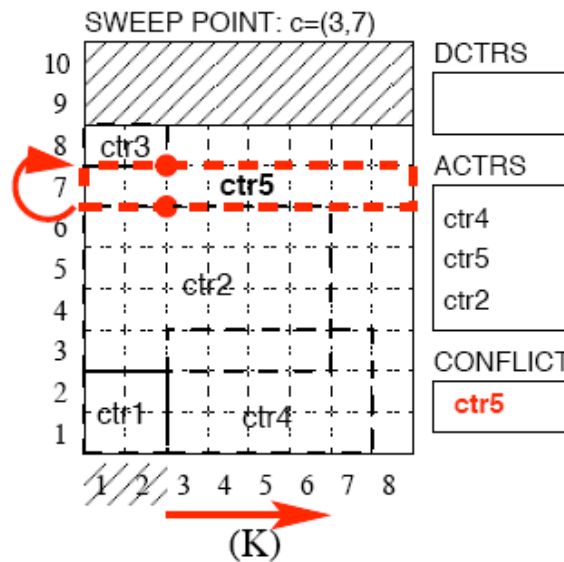
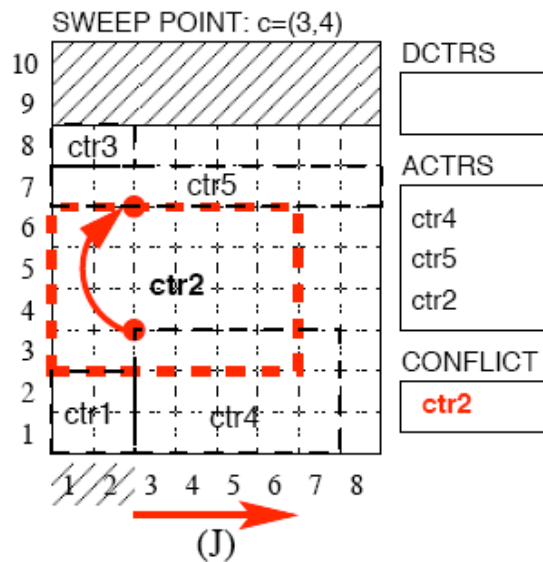
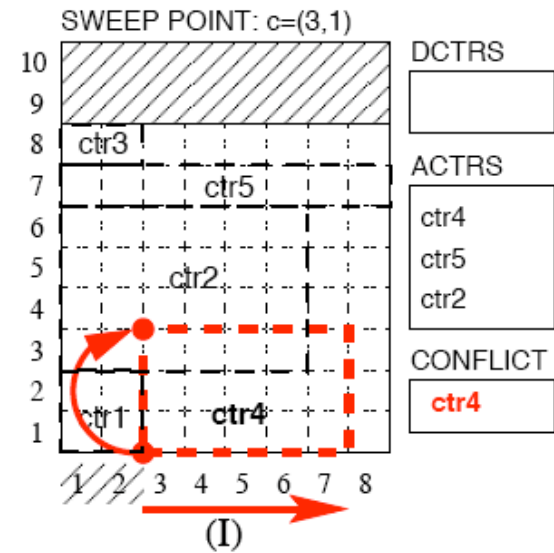
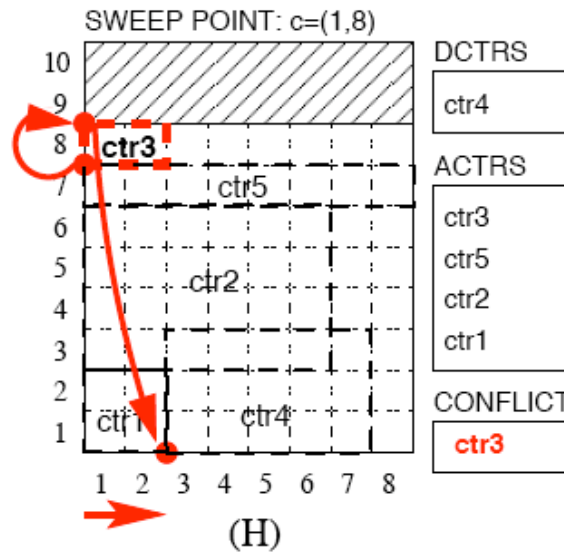
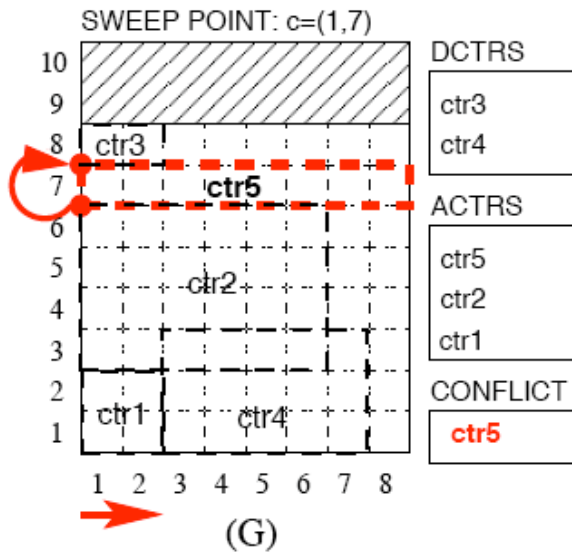
ACTRS

- ctr2
- ctr1

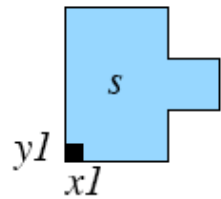
CONFLICT

- ctr2

Geometric Kernel : a Lexicographic Sweep-Point Algorithm

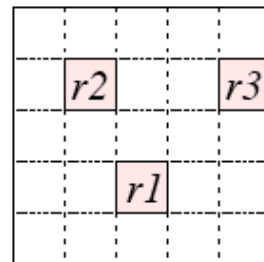


Where Splitting Objects Kills Propagation



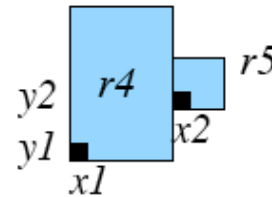
Shape to place within (B) so that it does not overlap rectangles $r1, r2, r3$

(A)



Placement space where to put s

(B)



Decomposing s in two rectangles $r4$ and $r5$

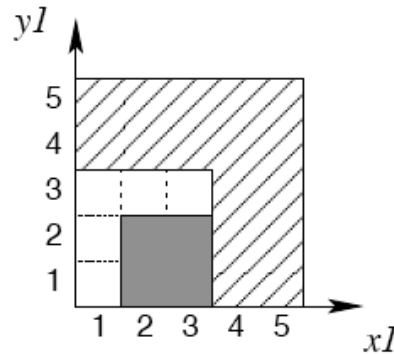
(C)

- ctr1: $x1+2=x2$
- ctr2: $y1+1=y2$
- ctr3: $r1$ and $r4$ do not overlap
- ctr4: $r2$ and $r4$ do not overlap
- ctr5: $r3$ and $r4$ do not overlap
- ctr6: $r1$ and $r5$ do not overlap
- ctr7: $r2$ and $r5$ do not overlap
- ctr8: $r3$ and $r5$ do not overlap

Constraints of the problem

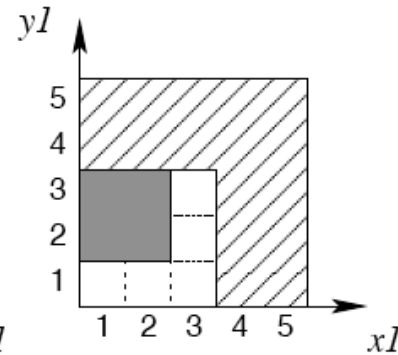
(D)

Where Splitting Objects Kills Propagation



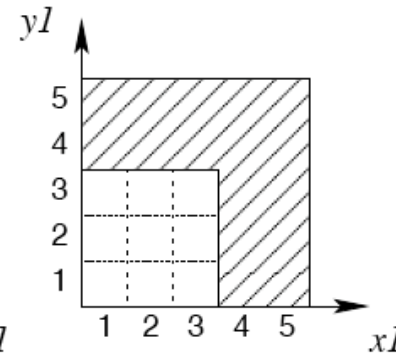
Forbidden pairs of values for (x_1, y_1) according to ctr3

(E)



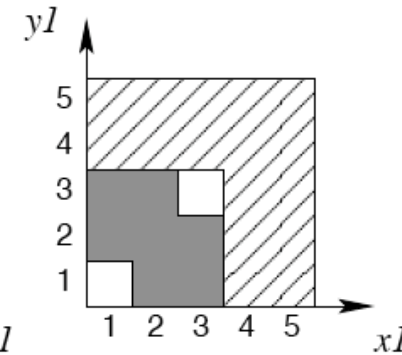
Forbidden pairs of values for (x_1, y_1) according to ctr4

(F)



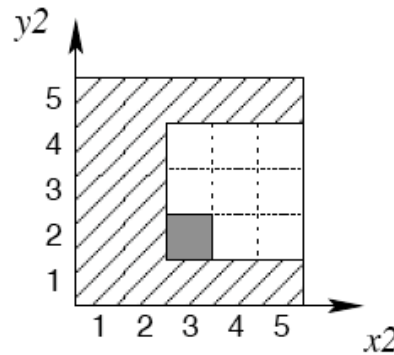
Forbidden pairs of values for (x_1, y_1) according to ctr5

(G)



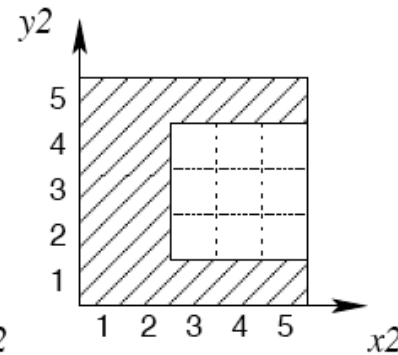
Forbidden pairs of values for (x_1, y_1) according to ctr3, ctr4 and ctr5

(H)



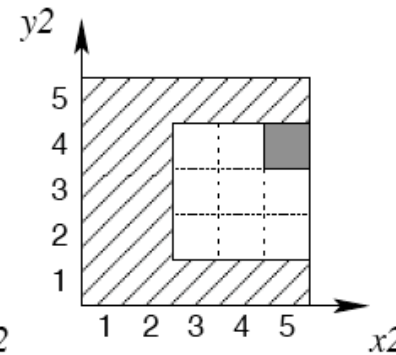
Forbidden pairs of values for (x_2, y_2) according to ctr6

(I)



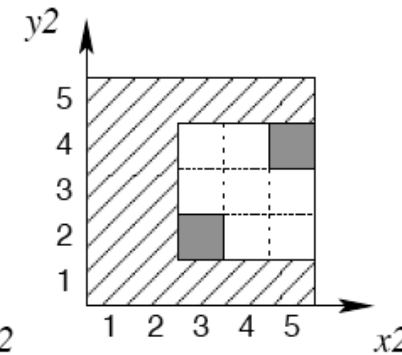
Forbidden pairs of values for (x_2, y_2) according to ctr7

(J)



Forbidden pairs of values for (x_2, y_2) according to ctr8

(K)



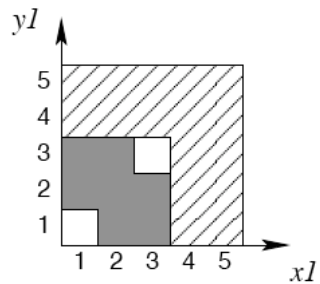
Forbidden pairs of values for (x_2, y_2) according to ctr6, ctr7 and ctr8

(L)

Where Splitting Objects Kills Propagation

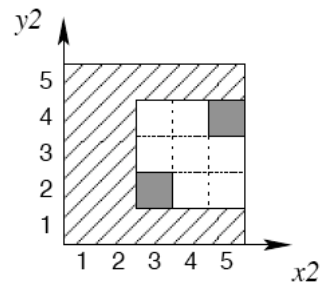
QUESTION :

How to combine information from $(x1,y1)$ and $(x2,y2)$?



Forbidden pairs of values for $(x1,y1)$ according to ctr3, ctr4 and ctr5

(H)



Forbidden pairs of values for $(x2,y2)$ according to ctr6, ctr7 and ctr8

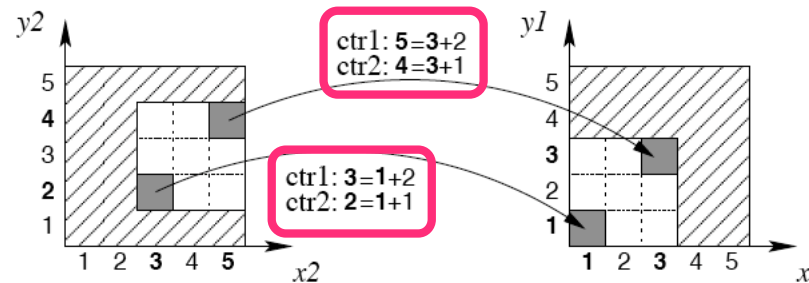
(L)

$$\text{ctr1: } x1+2=x2$$

$$\text{ctr2: } y1+1=y2$$

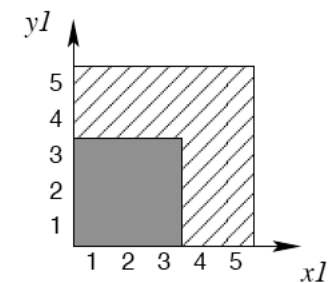
ANSWER :

Combining the infeasible points for $(x1,y1)$ and $(x2,y2)$ ONLY possible if ctr1 and ctr2 are integrated within the sweep process !



Transmission of the forbidden pairs of values for $(x2,y2)$ to forbidden pairs of values for $(x1,y1)$ through the external constraints ctr1 and ctr2 is not done if ctr1 and ctr2 are not integrated within the sweep algorithm

(M)



Overall set of forbidden pair of values for $(x1,y1)$

(N)

Introduction

External Geometrical Constraints

Internal Geometrical Constraints

The Propagation Kernel

A First Evaluation

A First Evaluation (April 2007) with a focus on non-overlapping

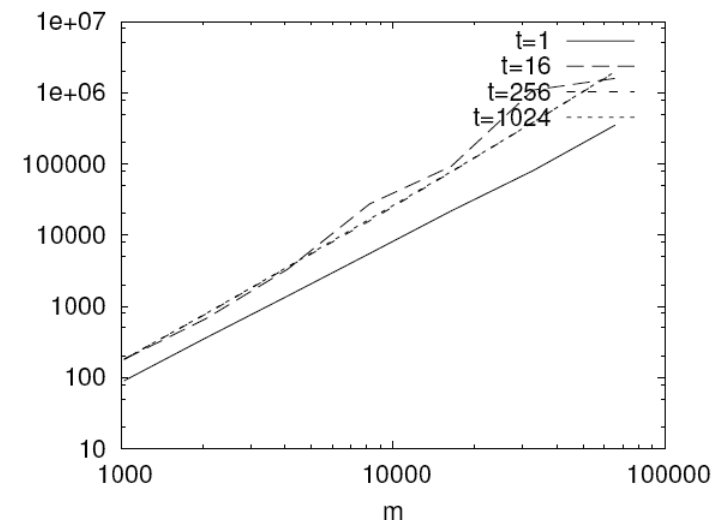
- **Scalability** on **loosely constrained problems** (20% spare space)
- **Tight** placement problems (**0% spare space**)
 - Perfect squared squares
 - 3D pentominoes [Colmerauer, Gilletta 99]
- **State of the art OR** for **2D orthogonal packing** [Clautiaux, Carlier, Jouglet 07]

Evaluation on SICStus Prolog 4 compiled with gcc-02 version 4.0.2 on a 3GHz Pentium IV

Loosely Constrained Problems

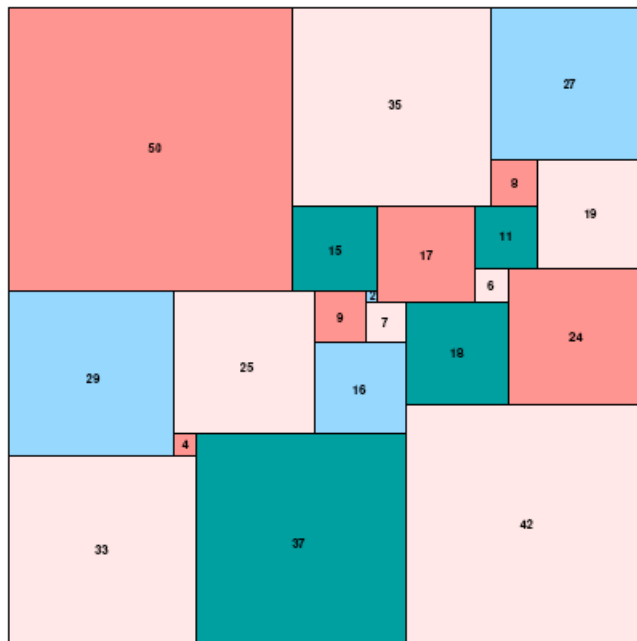
- Search first solution for random problem instances of m k -dimensional boxes for k in $\{2,3,4\}$ involving t in $\{1,16,256,1024\}$ distinct types of boxes, and m in $\{1024,2048,\dots,65536\}$.
- The number of **1.048.576** variables in *geost* was reached (*first time in a constraint solver in a backtracking environment !*).
- Can typically pack **1024 2D, 3D and 4D distinct boxes** in at most **200 msec**.
- Worst time, **13694 sec**, obtained for packing **262.144 4D parallelepipeds** corresponding to **1024 distinct types**, with a memory consumption of **351MB**.
- Approach sensible to the number of distinct types of boxes

Results for $k=2$ for various t and m (time in msec)

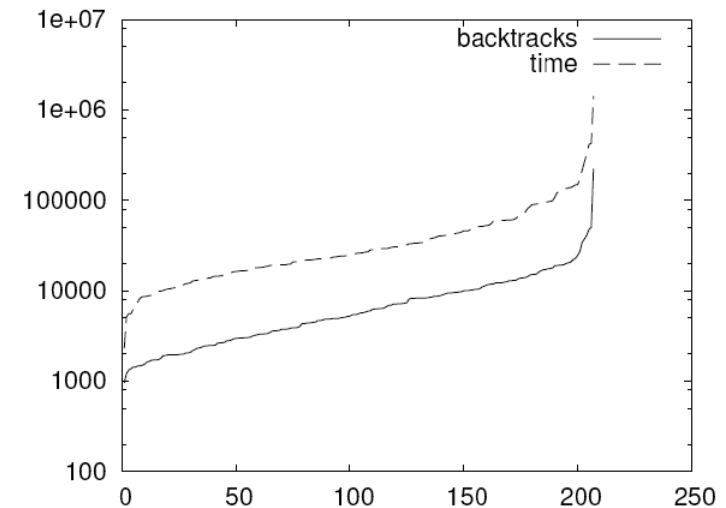


Tight Placement Problems: Perfect squared squares

- Time and number of backtracks for **exploring all the search space** without breaking any symmetry (207 instances).
- **159** problems were completely solved within **60 seconds**.
- The maximum time of **1635 seconds** was spent on **problem 48**.



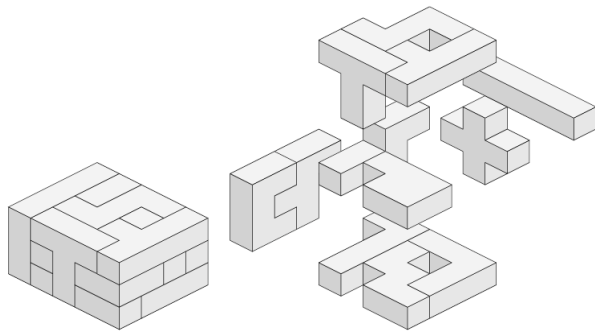
Example: problem 1



Results for the 207 instances
(time in msec)

Tight Placement Problems: 3D Pentominoes

- Time and number of backtracks for **exploring all the search space** without breaking any symmetry (7 instances).



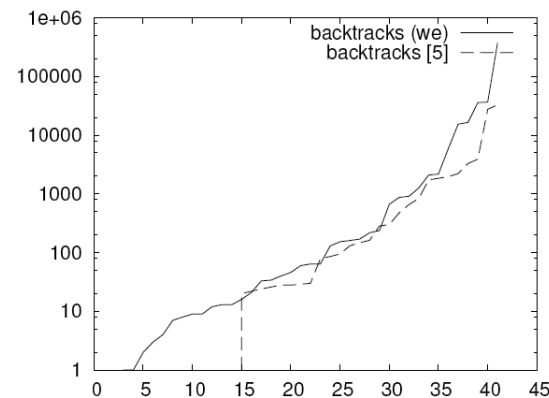
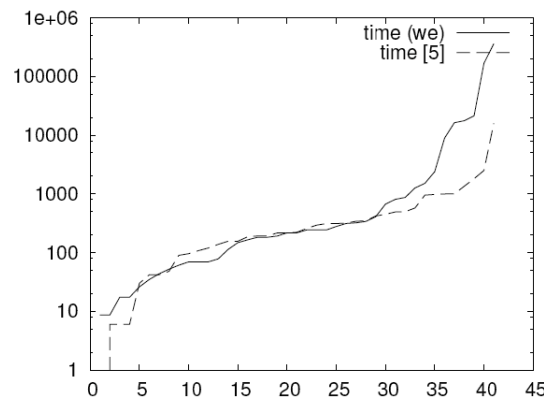
Example: 12 pentominoes in 5x4x3

configuration	backtracks (1st)	time (1st)	backtracks (all)	time (all)	solutions
$20 \times 3 \times 1$	1434	1840	47381	49740	8
$15 \times 4 \times 1$	290	560	888060	939060	1472
$12 \times 5 \times 1$	1594	1850	3994455	4112870	4040
$10 \times 6 \times 1$	111	260	9688985	10726810	9356
$10 \times 3 \times 2$	1267	2370	1203511	1778980	96
$6 \times 5 \times 2$	157	730	n/a	n/a	n/a
$5 \times 4 \times 3$	3567	14930	n/a	n/a	n/a

3D pentomino packing instances (time in msec)

State of the Art OR for 2D Orthogonal Packing

- Feasibility problem that consists of determining whether a set of rectangles that cannot be rotated can be packed or not into a rectangle of fixed size (use symmetries).
- 41 instances involving between 10 and 23 rectangles (slack vary from 0% to 20%).
- All the instances were solved (18 instances are much easier for Clautiaux 07, 18 instances are much easier for us).



Comparing time (time in msec) and number of backtracks
In each curve the instances are ordered by increasing y value

Conclusion

Once again, use the **sweep** idea: *quite simple, but powerful !*

- The overall architecture was designed in order to allow to integrate additional constraints **without modifying** the kernel.
- Can directly handle objects that **move in time**.
- When propagating on one object, consider **all constraints** involving that object.
- Scale better (**one million integer variables** in a standard constraint system: compatible with **backtracking**).

One last observation:

*disjunctive, cumulative, non-overlapping constraints should all be integrated within **one single global constraint** since:*

- (1) they all correspond to related nested dynamic sub problems,*
- (2) allow to get better propagation,*
- (3) allow to reuse code.*

More information (*kernel, benchmarks*)

A Generic Geometrical Constraint Kernel in Space and Time for Handling Polymorphic k -Dimensional Objects

N. Beldiceanu¹, M. Carlsson², E. Poder¹, R. Sadek¹, and C. Truchet³

¹ École des Mines de Nantes, LINA FRE CNRS 2729, FR-44307 Nantes, France
{Nicolas.Beldiceanu, Emmanuel.Poder, Rida.Sadek}@emn.fr

² SICS, P.O. Box 1263, SE-164 29 Kista, Sweden
Mats.Carlsson@sics.se

³ Université de Nantes, LINA FRE CNRS 2729, FR-44322 Nantes, France
Charlotte.Truchet@univ-nantes.fr

SICS Technical Report T2007:08

ISSN: 1100-3154

ISRN: SICS-T-2007/08-SE

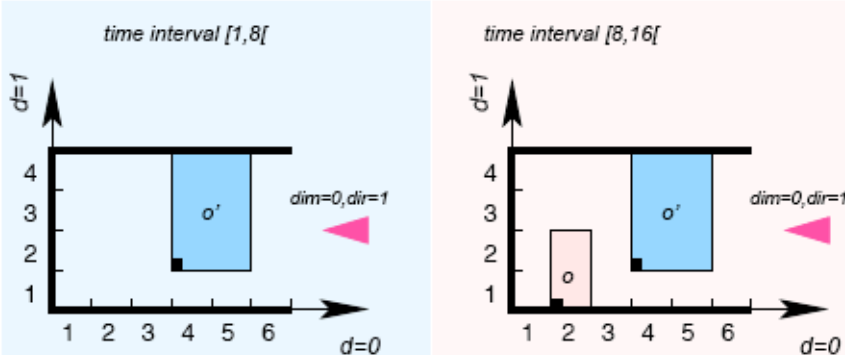
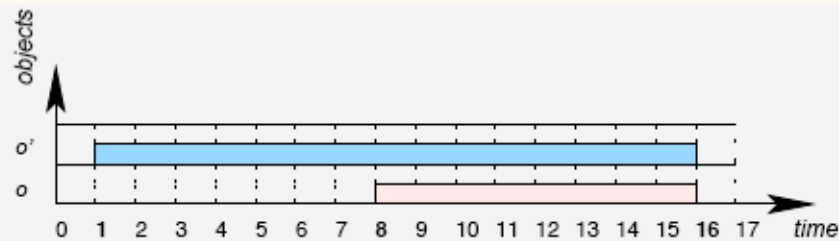
Acknowledgements

This research was conducted under European Union Sixth Framework Programme Contract FP6-034691 “Net-WMS”.

External *visible* Constraint: Examples

VIOLATED CONSTRAINT

```
geost(2, [object(o,1,[1,2],8,8,16),
          object(o',2,[4,2],1,15,16)],
       [shape(1,[0,0],[1,2],face-[<0,1>]),
        shape(2,[0,0],[2,3],face-[<0,1>]),
        [visible([0,1,face],[o,o'],[<0,1>])])
```

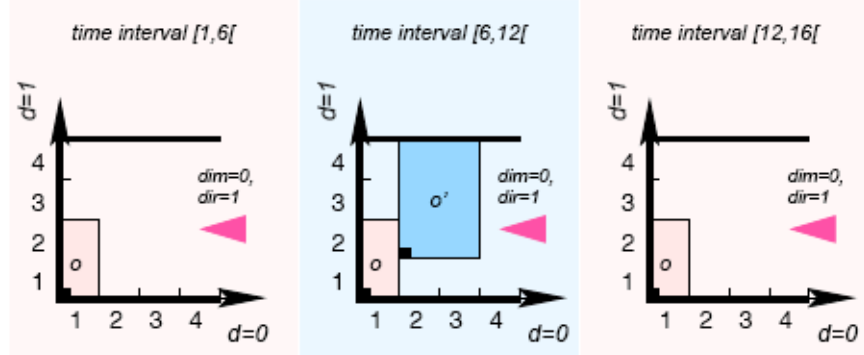
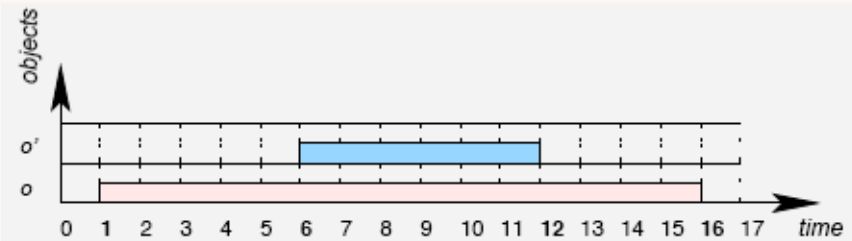


o is masked by *o'* according to $\langle \text{dim}=0, \text{dir}=1 \rangle$ since:

- (A) *o* and *o'* intersect in time,
- (B) *o* and *o'* intersect in dimension 1,
- (C) in dimension 0, *o'* starts after the end of *o*,
- (D) the start in time of *o* is located after the start in time of *o'*,
- (E) $\langle \text{dim}=0, \text{dir}=1 \rangle$ is a potentially visible face of *o*.

CONSTRAINT THAT HOLDS

```
geost(2, [object(o,1,[1,1],1,15,16),
          object(o',2,[2,2],6,6,12)],
       [shape(1,[0,0],[1,2],face-[<0,1>]),
        shape(2,[0,0],[2,3],face-[<0,1>]),
        [visible([0,1,face],[o,o'],[<0,1>])])
```

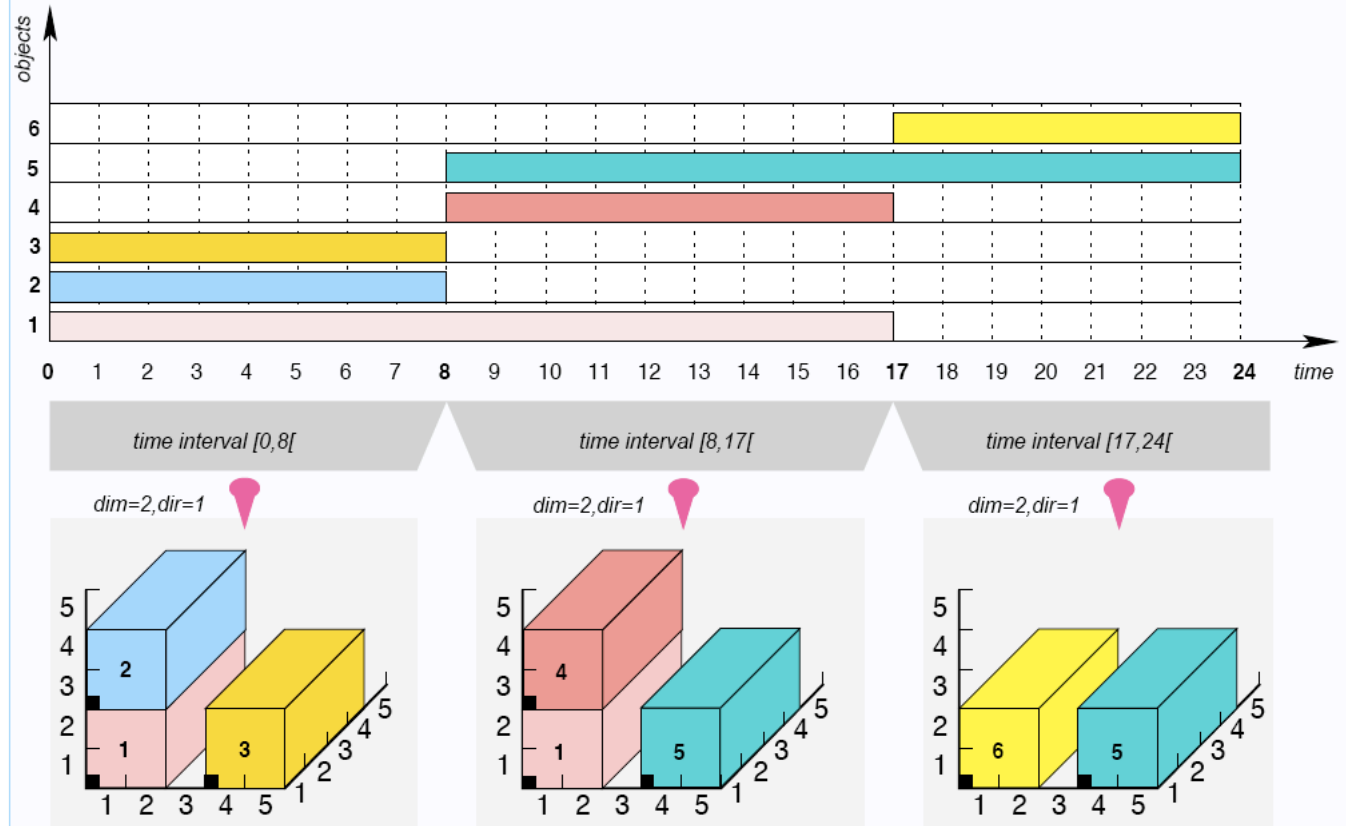


o is not masked by *o'* according to $\langle \text{dim}=0, \text{dir}=1 \rangle$ since:

- (A) Even though *o* and *o'* intersect in time,
- (B) and even though *o* and *o'* intersect in dimension 1,
- (C) and even though, in dimension 0, *o'* starts after the end of *o*,
- (E) and even though $\langle \text{dim}=0, \text{dir}=1 \rangle$ is a potentially visible face of *o*, condition (D) does not hold.

Application of *visible*: ship loading

```
geost(3, [object(1,1,[1,1,1],0,17,17), object(2,1,[1,1,3],0, 8, 8), object(3,1,[4,1,1], 0,8, 8),
object(4,1,[1,1,3],8, 9,17), object(5,1,[4,1,1],8,16,24), object(6,1,[1,1,1],17,7,24)],
[shape(1,[0,0,0],[2,4,2],face-<[2,1>]),
non-overlapping([0,1,2],[1,2,3,4,5,6]), visible([0,1,2,face],[1,2,3,4,5,6],[<2,1>]) )
```



Applications of Sweep Algorithms

Within the “**Geometry Literature Database**”, more than 100 references:

- Voronoi diagram
- Map overlay
- Nearest objects
- Triangulations
- Hidden surface removals
- Rectangles intersection
- Shortest path

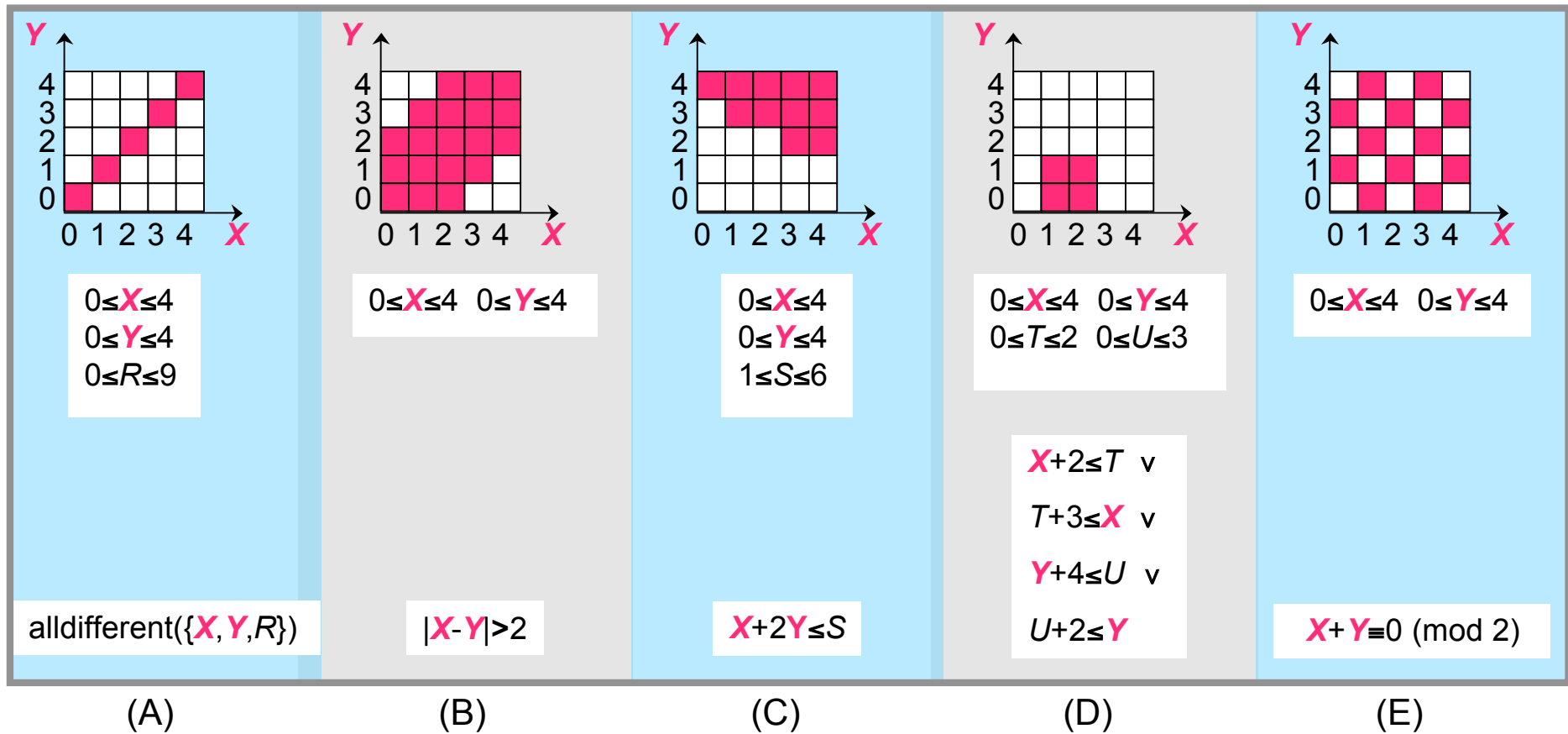
But not used very often within constraint programming !

Previous Uses of Sweep Algorithms within CP

Filtering for the following patterns:

- A **conjunction** of constraints **sharing** two variables
- **Cardinality** operator with **two shared variables**
- **Non-overlapping** between two **polygones**
- The multi-ressource **cumulatives** constraint

Representing a Constraint (Set of Forbidden Points)

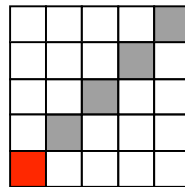


Example

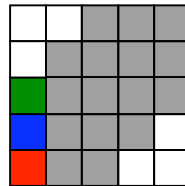
PROBLEM:

Adjust the minimum of X according to Y and all constraints:

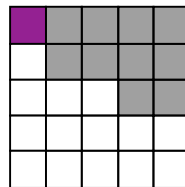
- $0 \leq X \leq 4$ $0 \leq Y \leq 4$
- $1 \leq S \leq 6$ $0 \leq T \leq 2$ $0 \leq U \leq 3$
- alldifferent($\{X, Y, R\}$)
- $|X - Y| > 2$
- $X + 2Y \leq S$
- $X + 2 \leq T \vee T + 3 \leq X \vee Y + 4 \leq U \vee U + 2 \leq Y$
- $X + Y \equiv 0 \pmod{2}$



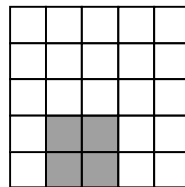
alldifferent($\{X, Y, R\}$)



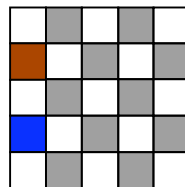
$|X - Y| > 2$



$X + 2Y \leq S$



$X + 2 \leq T \vee T + 3 \leq X \vee Y + 4 \leq U \vee U + 2 \leq Y$



$X + Y \equiv 0 \pmod{2}$

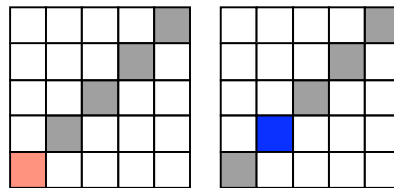


Example

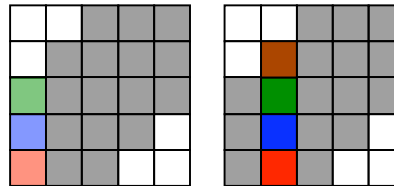
PROBLEM:

Adjust the minimum of X according to Y and all constraints:

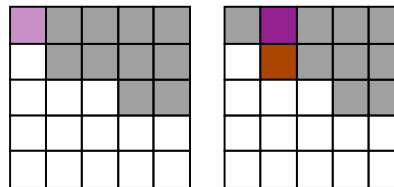
- $0 \leq X \leq 4$ $0 \leq Y \leq 4$
- $1 \leq S \leq 6$ $0 \leq T \leq 2$ $0 \leq U \leq 3$
- alldifferent($\{X, Y, R\}$)
- $|X - Y| > 2$
- $X + 2Y \leq S$
- $X + 2 \leq T \vee T + 3 \leq X \vee$
- $Y + 4 \leq U \vee U + 2 \leq Y$
- $X + Y \equiv 0 \pmod{2}$



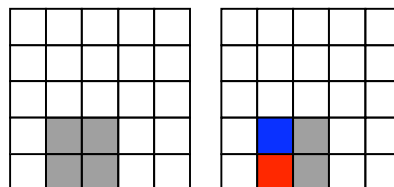
alldifferent($\{X, Y, R\}$)



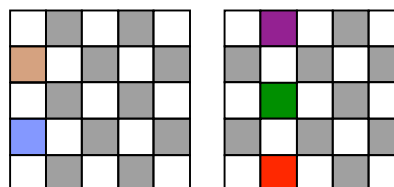
$|X - Y| > 2$



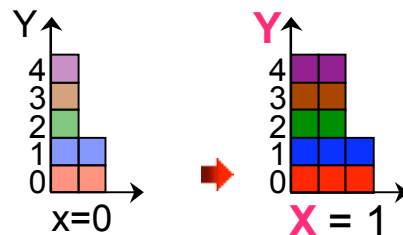
$X + 2Y \leq S$



$X + 2 \leq T \vee T + 3 \leq X \vee$
 $Y + 4 \leq U \vee U + 2 \leq Y$



$X + Y \equiv 0 \pmod{2}$

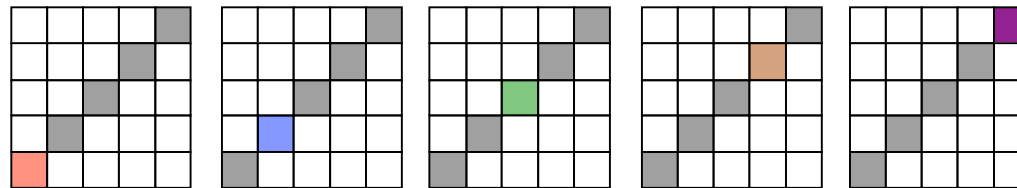


Example

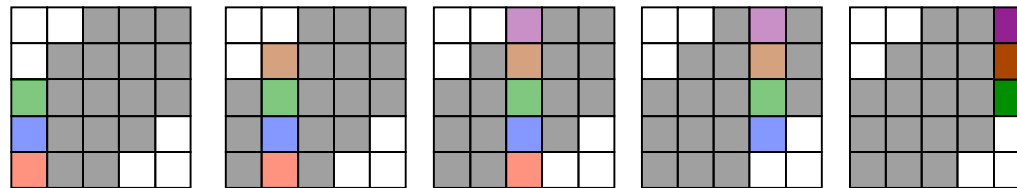
PROBLEM:

Adjust the minimum of X according to Y and all constraints:

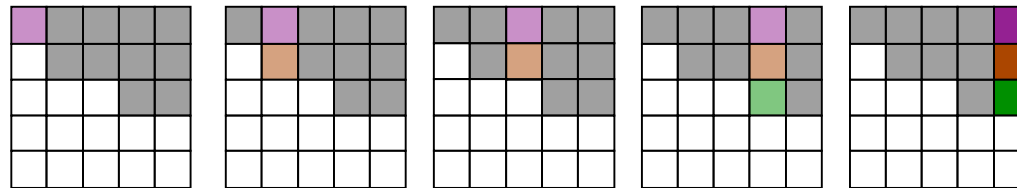
- $0 \leq X \leq 4$ $0 \leq Y \leq 4$
- $1 \leq S \leq 6$ $0 \leq T \leq 2$ $0 \leq U \leq 3$
- alldifferent($\{X, Y, R\}$)
- $|X - Y| > 2$
- $X + 2Y \leq S$
- $X + 2 \leq T \vee T + 3 \leq X \vee Y + 4 \leq U \vee U + 2 \leq Y$
- $X + Y \equiv 0 \pmod{2}$



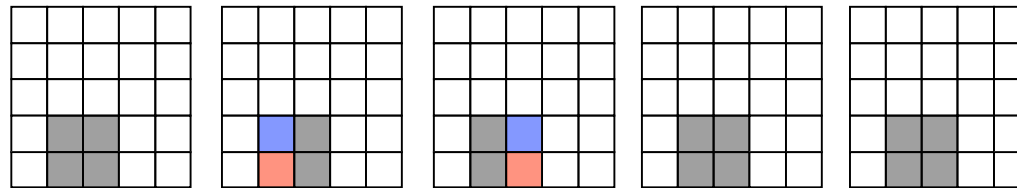
alldifferent($\{X, Y, R\}$)



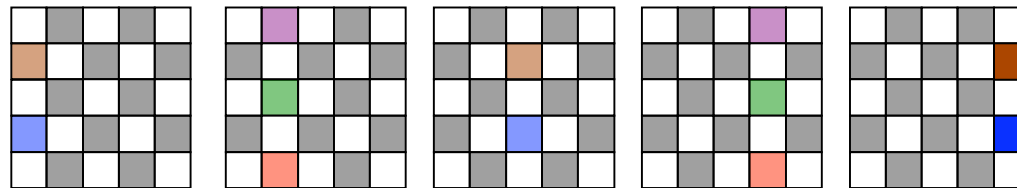
$|X - Y| > 2$



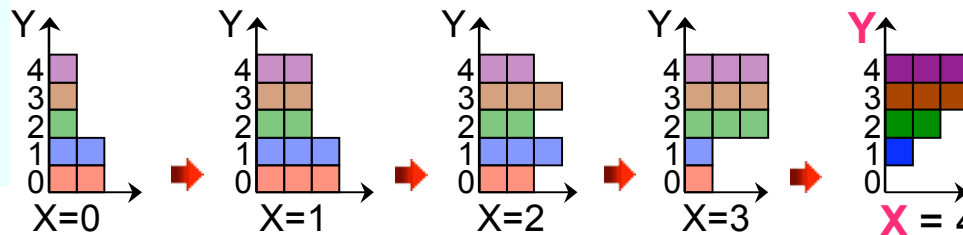
$X + 2Y \leq S$



$X + 2 \leq T \vee T + 3 \leq X \vee Y + 4 \leq U \vee U + 2 \leq Y$

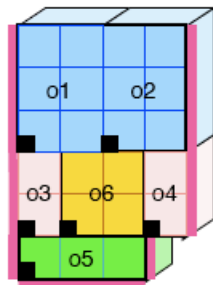
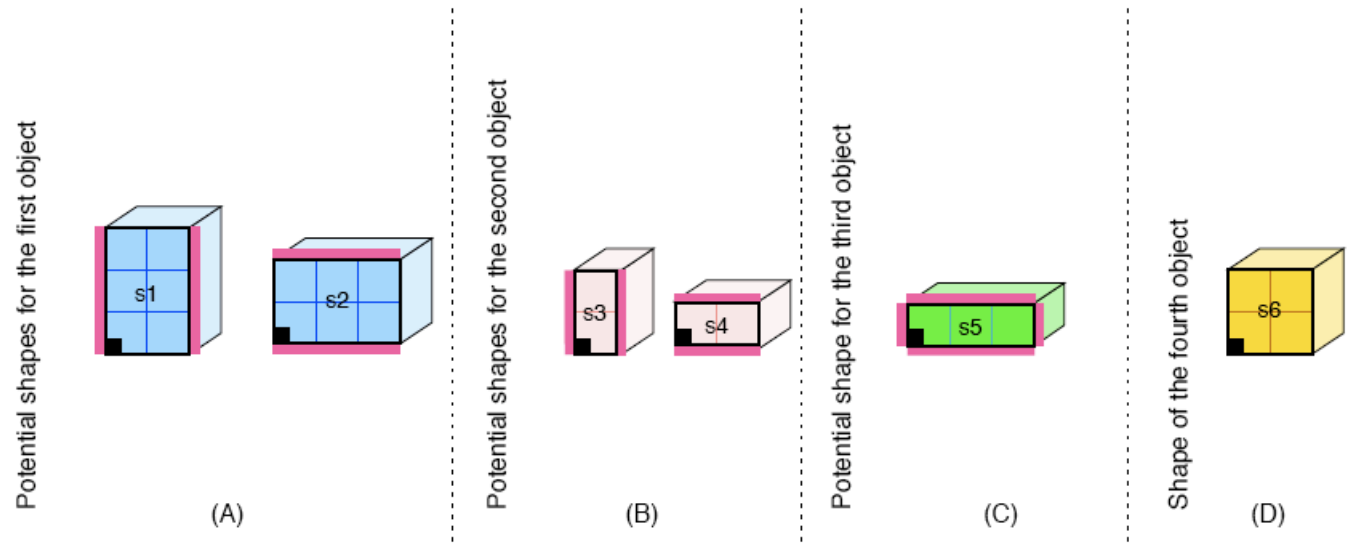


$X + Y \equiv 0 \pmod{2}$



Deduction:
 $X > 3$

Application of *visible*: pallet loading



```
geost(3,[object(o1,s1,[1,4,1],0,1,1),object(o2,s1,[3,4,1],0,1,1),object(o3,s2,[1,2,1],0,1,1),
object(o4,s2,[4,1,1],0,1,1),object(o5,s3,[1,1,1],0,1,1),object(o6,s4,[2,2,1],0,1,1)],
[shape(s1,[0,0,0],[2,3,1],face- [<0,0>,<0,1>]),shape(s2,[0,0,0],[3,2,1],face- [<1,0>,<1,1>]),
shape(s3,[0,0,0],[1,2,1],face- [<0,0>,<0,1>]),shape(s4,[0,0,0],[2,1,1],face- [<1,0>,<1,1>]),
shape(s5,[0,0,0],[3,1,1],face- [<0,0>,<0,1>,<1,0>,<1,1>]),
shape(s6,[0,0,0],[2,2,1],face- [])],
[non-overlapping([0,1,2],[o1,o2,o3,o4,o5,o6]),
visible([0,1,2,face],[o1,o2,o3,o4,o5],[<0,0>,<0,1>,<1,0>,<1,1>])])
```

(F)