Sweep Based Algorithms for Resource Scheduling

Nicolas Beldiceanu
EMN (CNRS/INRIA)
Nicolas.Beldiceanu@mines-nantes.fr

Master class CPAIOR 2012 (28 May)
INTRODUCTION

• Sweep Algorithms in Computational Geometry
• Main Ideas of Sweep Algorithms

CONSTRAINT CONSIDERED

• Used for cumulative resources (cumulative, cumulatives)
• Used for resources with choice (diffn, geost)

Sweeping over the Time Axis

• Basic scheme for cumulatives

Sweeping over the Placement Space

• Two dimensional case
• Multi dimensional case
• Points to remember

Other Sweep Based Filtering Algorithms

Conclusion
**Sweep Algorithms in Computational Geometry**

**Standard** technique in the design of **efficient** algorithms, described in:

- Computational geometry, an introduction  
  [Preparata & Shamos, 1985]

- Computational Geometry, Algorithms and Applications  
  [Berg, Kreveld, Overmars & Schwarzkopf, 1997]

- Géométrie algorithmique  
  [Boissonnat & Yvinec, 1995]
Main Ideas of Sweep Algorithms
(in the context of line segment intersection)

**GOAL:** the worst case complexity should also depend on the number of intersections

Steps of the sweep algorithm:

1. **Move** to the next event point
2. **Update** the sweep line status:
   - start events: ●
   - end events: □
Traditionally sweep algorithms are used for **checking** a property or for **computing** a quantity with respect to a set of **fixed** objects.

But in the context of constraint programming:

1. **The objects** (e.g., tasks, rectangles) are **not fixed**
   - Attributes are **variables** with a domain.
2. **Want to enforce a condition**
   - Detect infeasibility and **remove values** leading to failure.
INTRODUCTION
• Sweep Algorithms in Computational Geometry
• Main Ideas of Sweep Algorithms

CONSTRAINT CONSIDERED
• Used for cumulative resources (cumulative, cumulatives)
• Used for resources with choice (diffn, geost)

SWEEPING OVER THE TIME AXIS
• Basic scheme for cumulatives

SWEEPING OVER THE PLACEMENT SPACE
• Two dimensional case
• Multi dimensional case
• Points to remember

OTHER SWEEP BASED FILTERING ALGORITHMS

CONCLUSION
The *cumulative* Constraint

The **original cumulative** constraint
[Aggoun & Beldiceanu 92]:

- **Restrict** the resource consumption at each point in time.

![Cumulative Constraint Diagram](image)

The **generalized cumulatives** constraint
[Beldiceanu & Carlsson 01]:

- A **pool** of cumulative resources,
- Height of a task can be **negative**,
- Maximum or **minimum** resource consumption,
- Holds for time-points crossed by **at least one task**.

![Generalized Cumulative Constraint Diagram](image)
diffn/geost

(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
diffn/geost

machine assignment + relaxation
INTRODUCTION
• Sweep Algorithms in Computational Geometry
• Main Ideas of Sweep Algorithms

CONSTRAINT CONSIDERED
• Used for cumulative resources (cumulative, cumulatives)
• Used for resources with choice (diffn, geost)

SWEEPING OVER THE TIME AXIS
• Basic scheme for cumulatives

SWEEPING OVER THE PLACEMENT SPACE
• Two dimensional case
• Multi dimensional case
• Points to remember

OTHER SWEEP BASED FILTERING ALGORITHMS

CONCLUSION
The Sweep Algorithm: Fixed Tasks

Sweep line status:
- number of tasks which overlap the sweep line
- sum of the height

Event points:
- start of each task
- end of each task

Sweep line status:
- \( nb\_task=1 \)  \( sum\_height=2 \)
- \( nb\_task=2 \)  \( sum\_height=3 \)
- \( nb\_task=0 \)  \( sum\_height=0 \)
- \( nb\_task=1 \)  \( sum\_height=4 \)
- \( nb\_task=0 \)  \( sum\_height=0 \)

Minimum level to reach for those instants where there is at least one task

Current position of the sweep line: 3
More about sweep for cumulative scheduling by A. Letort in workshop in bin packing and placement constraints (BPPC’12)
INTRODUCTION
• Sweep Algorithms in Computational Geometry
• Main Ideas of Sweep Algorithms

CONSTRAINT CONSIDERED
• Used for cumulative resources (cumulative, cumulatives)
• Used for resources with choice (diffn, geost)

SWEEPING OVER THE TIME AXIS
• Basic scheme for cumulatives

SWEEPING OVER THE PLACEMENT SPACE
• Two dimensional case
• Multi dimensional case
• Points to remember

OTHER SWEEP BASED FILTERING ALGORITHMS

CONCLUSION
Forbidden Regions

**DEFINITION** *forbidden region* according to a constraint $\text{Ctr}$ and two variables $X,Y$ of $\text{Ctr}$:

Two intervals $\text{inf}_x..\text{sup}_x$ and $\text{inf}_y..\text{sup}_y$ such that:

For all $x$ in $\text{inf}_x..\text{sup}_x$,

$y$ in $\text{inf}_y..\text{sup}_y$: $\text{Ctr}$ with the assignment $X=x$ and $Y=y$ is false.

**EXAMPLE** forbidden regions of $\text{alldifferent}([X,Y,R])$ according to $X$ and $Y$:
Examples of **Forbidden Regions**

- **A**: $0 \leq X \leq 4$
- **B**: $0 \leq X \leq 4$, $0 \leq Y \leq 4$
- **C**: $0 \leq X \leq 4$, $0 \leq Y \leq 4$, $0 \leq R \leq 9$
- **D**: $0 \leq X \leq 4$, $0 \leq Y \leq 4$, $0 \leq T \leq 2$, $0 \leq U \leq 3$
- **E**: $0 \leq X \leq 4$, $0 \leq Y \leq 4$

- **alldifferent**($\{X, Y, R\}$)
- $|X-Y| > 2$
- $X+2Y \leq S$
- $X+2 \leq T$ $\lor$ $T+3 \leq X$ $\lor$ $Y+4 \leq U$ $\lor$ $U+2 \leq Y$
- $X+Y \equiv 0 \pmod{2}$
Basic **Idea** of Sweep Pruning

Accumulates forbidden regions that come from different constraints

\[
\begin{align*}
CTR_1(X,Y,...) \\
CTR_2(X,Y,...) \\
\ldots \ldots \\
CTR_n(X,Y,...)
\end{align*}
\]

involving **two** given variables $X$ and $Y$

Is min($X$) feasible?
No, so move the sweep-line.
For each \( y \in \text{dom}(Y) \):
- **number** of forbidden regions containing the point \((\Delta, y)\)

**Remove** a value \( \Delta \in \text{dom}(X) \) if:
- for all \( y \in \text{dom}(Y) \) the number of forbidden regions is \( > 0 \)
An Example

PROBLEM:
Adjust minimum of $X$ according to $Y$ and to all following constraints:

$$0 \leq X \leq 4 \quad 0 \leq Y \leq 4$$
$$1 \leq S \leq 6 \quad 0 \leq T \leq 2 \quad 0 \leq U \leq 3$$
$$\text{alldifferent}([X,Y,R])$$
$$|X-Y| > 2$$
$$X+2Y \leq S$$
$$X+2 \leq T \lor T+3 \leq X \lor Y+4 \leq U \lor U+2 \leq Y$$
$$X+Y \equiv 0 \pmod{2}$$
An Example

PROBLEM:
Adjust minimum of $X$ according to $Y$ and to all following constraints:

\[
\begin{align*}
0 \leq X & \leq 4 & 0 \leq Y & \leq 4 \\
1 \leq S & \leq 6 & 0 \leq T & \leq 2 & 0 \leq U & \leq 3 \\
alldifferent({X, Y, R}) \\
|X-Y| & > 2 \\
X+2Y & \leq S \\
X+2T & \vee T+3 \leq X & \vee Y+4 \leq U & \vee U+2 \leq Y \\
X+Y & \equiv 0 \pmod{2}
\end{align*}
\]

Deduction:
\[X > 3\]
Forbidden and Safe Regions (for relaxation)

**DEFINITION** forbidden region according to a constraint Ctr and 2 variables X, Y of Ctr:

Two intervals inf_x..sup_x and inf_y..sup_y such that:

∀ x ∈ inf_x..sup_x,

∀ y ∈ inf_y..sup_y: Ctr with the assignment X=x and Y=y is false.

**DEFINITION** safe region according to a constraint Ctr and 2 variables X, Y of Ctr:

Two intervals inf_x..sup_x and inf_y..sup_y such that:

∀ x ∈ inf_x..sup_x,

∀ y ∈ inf_y..sup_y: Ctr with the assignment X=x and Y=y is true.

**EXAMPLE**

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

0 ≤ X ≤ 4
0 ≤ Y ≤ 4
1 ≤ S ≤ 6
X + 2Y ≤ S
INTRODUCTION
• Sweep Algorithms in Computational Geometry
• Main Ideas of Sweep Algorithms

CONTRAINT CONSIDERED
• Used for cumulative resources \((\text{cumulative, cumulatives})\)
• Used for resources with choice \((\text{diffn, geost})\)

SWEEPING OVER THE TIME AXIS
• Basic scheme for cumulatives

SWEEPING OVER THE PLACEMENT SPACE
• Two dimensional case
• Multi dimensional case
• Points to remember

OTHER SWEEP BASED FILTERING ALGORITHMS

CONCLUSION
Question: How to Generalize to $k$ Dimensions?

Key problem with the sweep-line status: don't want to use a multi-dimensional data structure for storing the forbidden regions since it just kills scalability

A Lexicographic Sweep-Point Algorithm
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])

(D) DCTRS (delayed internal constraint)
ctr1 ctr2
ctr5
ctr4

(E) SWEEP POINT: c=(1,1)
DCTRS
ctr2
ctr5
ctr3
ctr4

(F) SWEEP POINT: c=(1,3)
ACTRS
ctr1

DCTRS
ctr5
ctr3
ctr1

ACTRS
ctr1

CONFLICT
ctr1

CONFLICT
ctr2
A Lexicographic Sweep-Point Algorithm

- SWEEP POINT: c=(1,7)
  - DCTRS:
    - ctrl3
    - ctrl4
  - ACTRS:
    - ctrl5
    - ctrl2
    - ctrl1
  - CONFLICT:
    - ctrl5

- SWEEP POINT: c=(1,8)
  - DCTRS:
    - ctrl4
  - ACTRS:
    - ctrl3
    - ctrl5
    - ctrl4
    - ctrl2
    - ctrl1
  - CONFLICT:
    - ctrl5

- SWEEP POINT: c=(3,1)
  - DCTRS:
    - ctrl4
    - ctrl5
  - ACTRS:
    - ctrl4
    - ctrl5
    - ctrl4
    - ctrl2
  - CONFLICT:
    - ctrl4

- SWEEP POINT: c=(3,4)
  - DCTRS:
    - ctrl3
  - ACTRS:
    - ctrl4
    - ctrl5
    - ctrl2
    - ctrl1
  - CONFLICT:
    - ctrl2

- SWEEP POINT: c=(3,7)
  - DCTRS:
    - ctrl3
    - ctrl5
  - ACTRS:
    - ctrl4
    - ctrl5
    - ctrl2
    - ctrl1
  - CONFLICT:
    - ctrl5

- SWEEP POINT: c=(3,8)
  - DCTRS:
    - ctrl5
  - ACTRS:
    - ctrl4
    - ctrl5
    - ctrl2
  - CONFLICT:
    - ctrl5
Forbidden box wrt. an unfeasible point

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

LexInfeasible \( (ictr, minlex, d, k, o) : (bool, p[]) \)
IsInfeasible \( (ictr, min, d, k, o, c) : (bool, f[]) \)
CardInfeasible \( (ictr, k, o) : (int) \)
Working area for a forbidden box

**IDEA:** In some dimensions, $B_i$ is **restricted** by the previous forbidden boxes.

**Additional requirement:**

(8) $B_i$ should be contained in the working area (in grey on the example)
INTRODUCTION
• Sweep Algorithms in Computational Geometry
• Main Ideas of Sweep Algorithms

CONSTRAINT CONSIDERED
• Used for cumulative resources (cumulative, cumulatives)
• Used for resources with choice (diftn, geost)

SWEEPING OVER THE TIME AXIS
• Basic scheme for cumulatives

SWEEPING OVER THE PLACEMENT SPACE
• Two dimensional case
• Multi dimensional case
• Points to remember

OTHER SWEEP BASED FILTERING ALGORITHMS

CONCLUSION
Points to Remember

An other way to look a constraint propagation:

- Rather than projecting on a single variable project on several variables

- Aggregate forbidden tuples coming from different constraints
  (without using an explicit constraint store like MDD for instance)
Points to Remember

- A combination of a generic method (*the sweep itself*) and structure (*computing quickly forbidden regions for specific constraints*)

- Practical efficiency achieved by:
  - **Light** data structure (*points*)
  - Forbidden regions are *not explicitly represented*, but
  - Forbidden regions *computed on demand* (*lazy evaluation*)
  - Incrementality achieved by witness, source and target, domination between objects
Points to Know

- Not a polynomial filtering algorithm if number of dimension is variable
  
  (potentially an exponential number of jumps)

- But works well in practice under following assumption
  
  (when consider a limited number of dimensions, e.g. 8)

- On a clique of non-overlapping constraint do the following relaxation
INTRODUCTION

• Sweep Algorithms in Computational Geometry
• Main Ideas of Sweep Algorithms

CONSTRAINT CONSIDERED

• Used for cumulative resources \((cumulative, \ cumulatives)\)
• Used for resources with choice \((diffn, \ geost)\)

SWEEPING OVER THE TIME AXIS

• Basic scheme for cumulatives

SWEEPING OVER THE PLACEMENT SPACE

• Two dimensional case
• Multi dimensional case
• Points to remember

OTHER SWEEP BASED FILTERING ALGORITHMS

CONCLUSION
Other use/extension of sweep algorithms in CP

- cardinality constraints
- forbidden regions combining non-overlapping and symmetry breaking
- assignment of tasks with same origin (sweep synchronisation
  with incremental bipartite matching while sweeping)
- non-intersection between polygons
- compiling placement rules to code that generate forbidden regions
- within the context of continuous constraints (computes forbidden box
  associated to a numerical constraint)
INTRODUCTION
• Sweep Algorithms in Computational Geometry
• Main Ideas of Sweep Algorithms

CONSTRAINT CONSIDERED
• Used for cumulative resources (*cumulative, cumulatives*)
• Used for resources with choice (*diffn, geost*)

SWEEPING OVER THE TIME AXIS
• Basic scheme for cumulatives

SWEEPING OVER THE PLACEMENT SPACE
• Two dimensional case
• Multi dimensional case
• Points to remember

OTHER SWEEP BASED FILTERING ALGORITHMS

CONCLUSION
• Yet another use of sweep algorithms
  (sweep algorithms were already used for a lot of different purpose)

• Combine generality (forbidden, safe regions) with a specific algorithm (sweep)
  (lead to a generic class of propagation algorithms)
Is it possible to get better filtering for well known scheduling problems and better estimation of the cost? *(job-shop, open-shop, …)*
Further Sources of Information

**papers**

- Sweep as a Generic Pruning Technique Applied to the Non-overlapping Rectangles Constraint. CP 2001: 377-391 (*2 dimensional sweep algorithm*)
- Sweep as a Generic Pruning Technique Applied to Constraint Relaxation. CP 2001 workshop
- Non-overlapping Constraints between Convex Polytopes. CP 2001: 392-407
- **A New Multi-resource cumulatives Constraint with Negative Heights. CP 2002: 63-79** (*sweeping over the time axis*)
- **A Generic Geometrical Constraint Kernel in Space and Time for Handling Polymorphic *k*-Dimensional Objects. CP 2007: 180-194** (*multi-dimensional sweep algorithm*)
- A Geometric Constraint over k-Dimensional Objects and Shapes Subject to Business Rules. CP 2008: 220-234
- Six Ways of Integrating Symmetries within Non-overlapping Constraints. CPAIOR 2009: 11-25
- Sweeping with Continuous Domains. CP 2010: 137-151
Global Constraint Catalog, 2nd Edition (revision a)
Available at: http://soda.swedish-ict.se/5195/

Keywords related to scheduling:
- resource constraints (page 295)
- scheduling constraints (page 301)
- temporal constraints (page 330)
- timetabling constraints (page 331)

Modelling exercises:
- some at page 144

Systems

- Implementation of different variants of the non-overlapping constraint
  (choco, jacop, sicstus)  
  \(\text{\textless\textless} \text{geost constraint}\)

- Implementation of the multi-resource \textit{cumulatives} constraint
  (choco, gecode, sicstus) 
  \(\text{\textless\textless} \text{cumulative(s) constraint}\)