A SCALABLE SWEEP ALGORITHM FOR THE *CUMULATIVE* CONSTRAINT

ARNAUD LETORT,[1] NICOLAS BELDICEANU,[1] AND MATS CARLSSON[2]

arnaud.letort@mines-nantes.fr, nicolas.beldiceanu@mines-nantes.fr, matsc@sics.se

[1]EMN, TASC (CNRS/INRIA)

[2]SICS

OCTOBER 9, 2012

MOTIVATIONS

- **Need to handle large scale problems.**

  [Panel of the Future of CP 2011]

- **(Multi-dimensional) bin-packing problems, in the context of cloud computing.**

  [Panel of the Future of CP 2011], [2012 Roadef Challenge]

- **Existing papers usually leave open the scalability issue.**

- **Time-Table constraint is a good candidate.**

  [Baptiste 2006, Samos] time-tabling used in ILOG Scheduler for scalability purpose

  [Vilim, 2011 CPAIOR]

OUTLINE

The *cumulative* Constraint

A Critical Analysis of the [CP2001] Sweep Algorithm

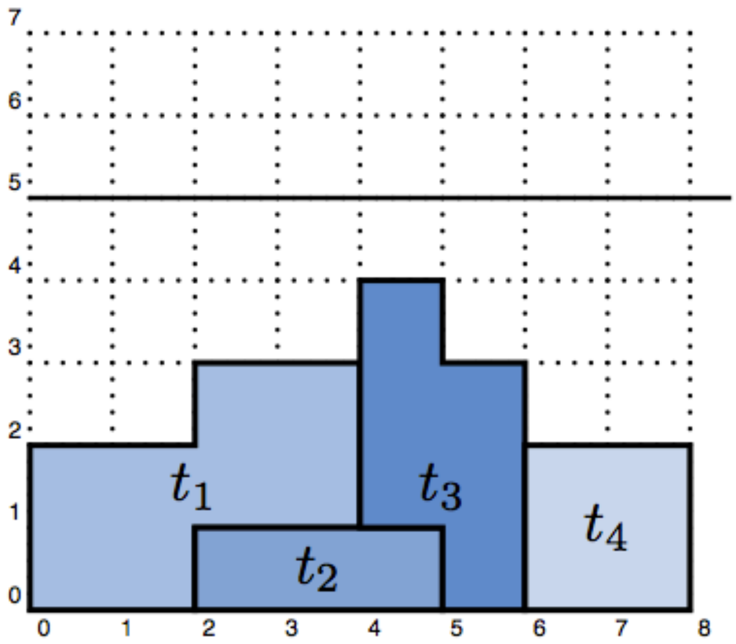
The Dynamic Sweep Algorithm

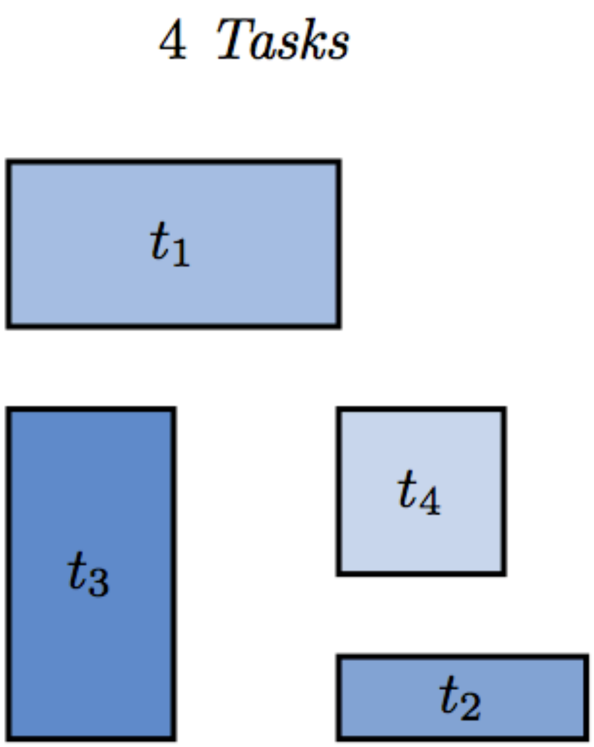
(using new dynamic events)

Evaluation

Conclusion

THE *CUMULATIVE* CONSTRAINT





OUTLINE

The *cumulative* Constraint

A Critical Analysis of the [CP2001] Sweep Algorithm

- Principle
- Illustration
- 4 Weaknesses

The Dynamic Sweep Algorithm

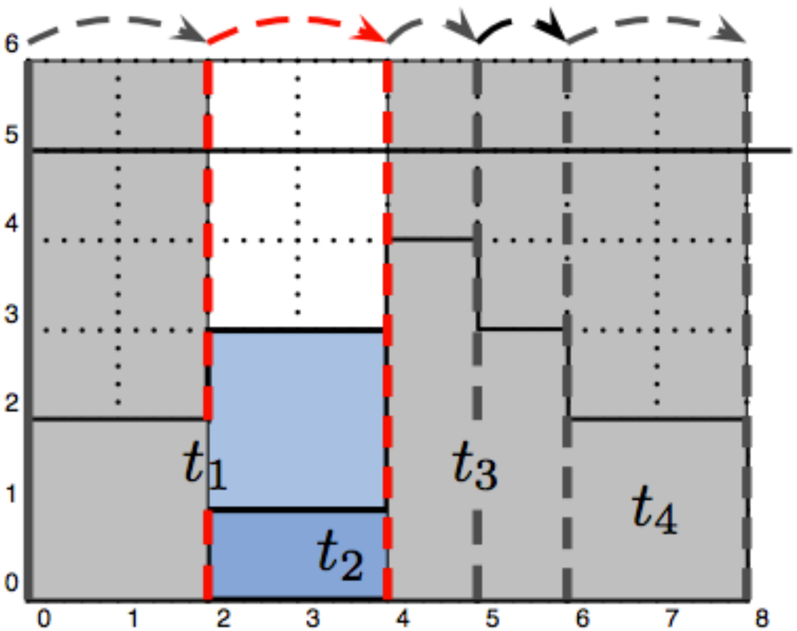
Evaluation

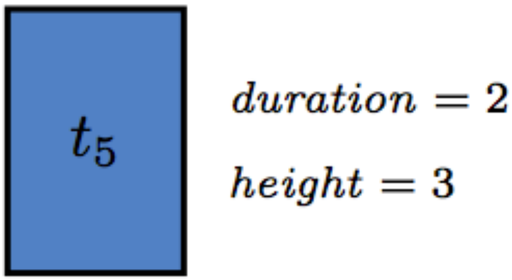
Conclusion

A CRITICAL ANALYSIS OF THE [CP2001] SWEEP ALGORITHM

(OVERVIEW)

The sweep-line “jumps” from event to event in order to build the cumulated profile and to perform checks and pruning.



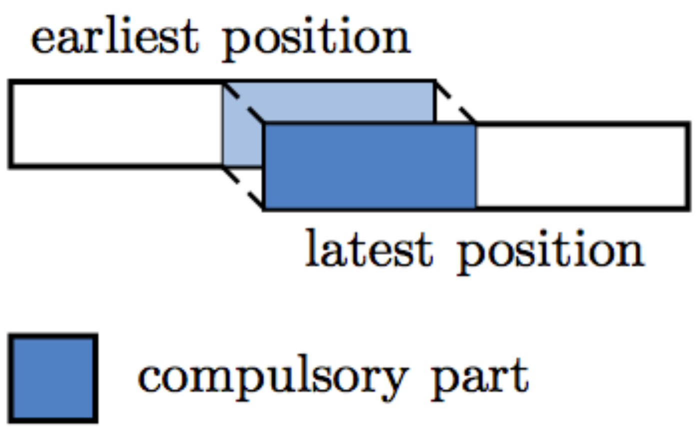


This task cannot overlap [2,4)

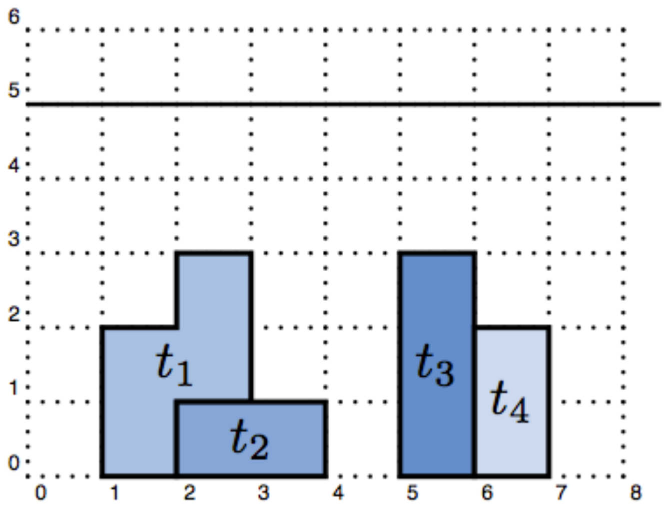
This task cannot start on [1,4)

A CRITICAL ANALYSIS OF THE [CP2001] SWEEP ALGORITHM
(PRINCIPLE: COMPULSORY PARTS)

Compulsory Part : the intersection of all the feasible instances of a task.

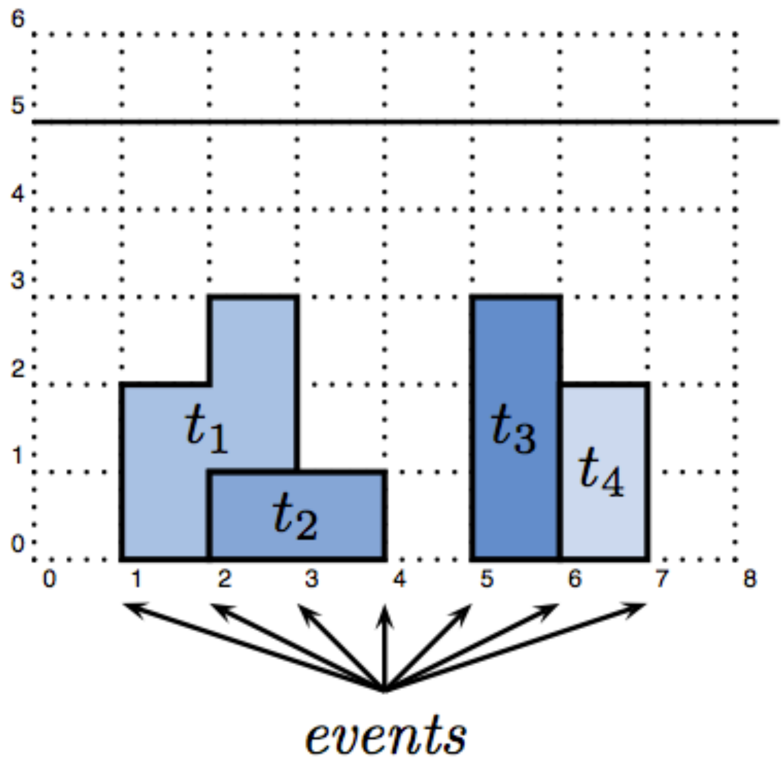


Cumulated Profile: the union of all the compulsory parts.



A CRITICAL ANALYSIS OF THE [CP2001] SWEEP ALGORITHM
(PRINCIPLE: EVENTS)

Event : a potential change of the height of the cumulated profile.
(i.e. start and end of compulsory part)



A CRITICAL ANALYSIS OF THE [CP2001] SWEEP ALGORITHM

(PRINCIPLE)

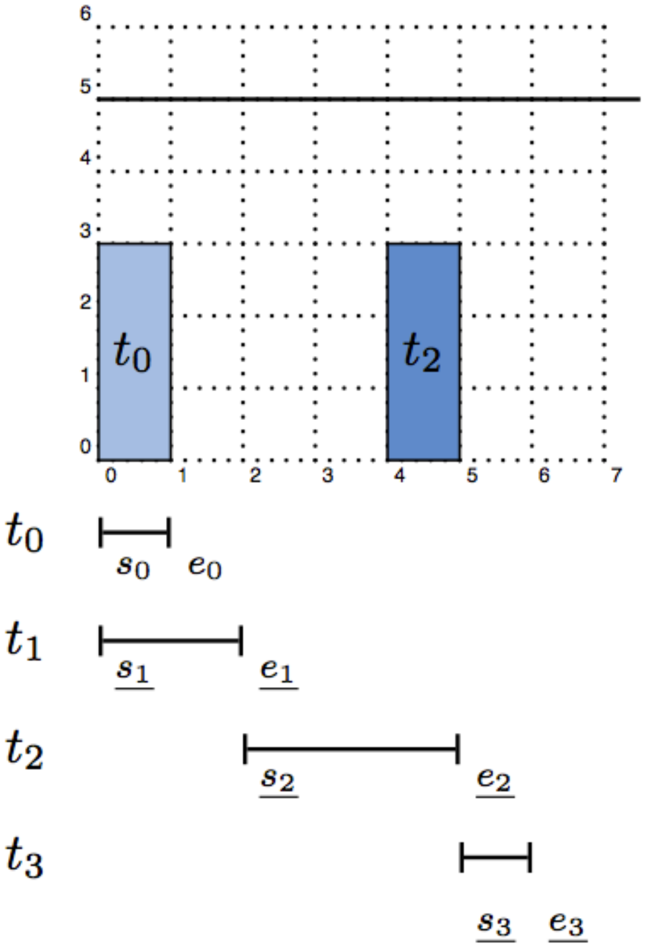
1. All events on the current sweep-line position are read (amount of available resource is updated).

2. The current and the next sweep-line positions define a sweep interval.

3. Scans all tasks that overlap the sweep interval. If the height of a task is greater than the available resource, an interval is removed from the start of the task.

A CRITICAL ANALYSIS OF THE [CP2001] SWEEP ALGORITHM

(ILLUSTRATION: INITIAL PROBLEM)



$t_0 : s_0 = 0,\ d_0 = 1,\ e_0 = 1,\ h_0 = 3$

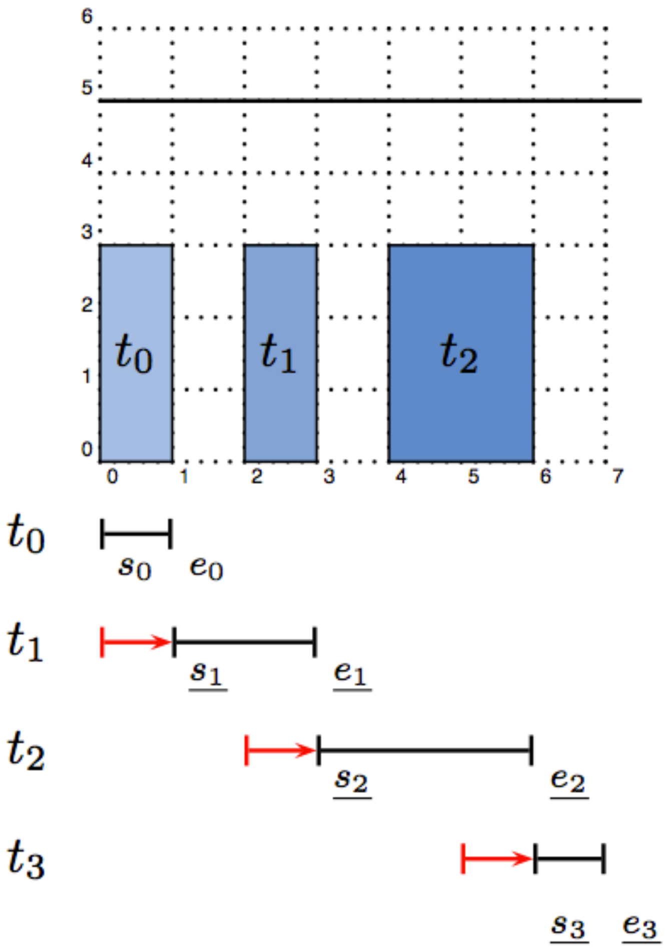
$t_1 : s_1 \in [0, 2],\ d_1 = 2,\ e_1 \in [2, 4],\ h_1 = 3$

$t_2 : s_2 \in [2, 4],\ d_2 = 3,\ e_2 \in [5, 7],\ h_2 = 3$

$t_3 : s_3 \in [5, 7],\ d_3 = 1,\ e_3 \in [6, 8],\ h_3 = 3$

A CRITICAL ANALYSIS OF THE [CP2001] SWEEP ALGORITHM

(ILLUSTRATION: AFTER A FOURTH SWEEP)



After 4 sweeps

$t_0 : s_0 = 0, d_0 = 1, e_0 = 1, h_0 = 3$

$t_1 : s_1 \in [1, 2], d_1 = 2, e_1 \in [3, 4], h_1 = 3$

$t_2 : s_2 \in [3, 4], d_2 = 3, e_2 \in [6, 7], h_2 = 3$

$t_3 : s_3 \in [6, 7], d_3 = 1, e_3 \in [7, 8], h_3 = 3$

A CRITICAL ANALYSIS OF THE [CP2001] SWEEP ALGORITHM
(4 WEAKNESSES)

1. **Too static**:

   Does not take into account the potential increase of the cumulated profile during a single sweep (see previous example).

2. **Often reaches its worst time complexity**:

   It needs to systematically re-scan all tasks that overlap the current sweep-line position to perform pruning. ($O(n^2)$)

3. **Creates holes in the domains**:

   A variable cannot just be compactly represented by its min/max values.

4. **Does not take advantage of the bin-packing**:

   The worst-case time complexity is left unchanged and is often reached.

OUTLINE

The *cumulative* Constraint

A Critical Analysis of the [CP2001] Sweep Algorithm

The Dynamic Sweep Algorithm

- Principle
- Illustration
- Property and Complexity
- Greedy Mode

Evaluation

Conclusion

THE DYNAMIC SWEEP ALGORITHM
(PRINCIPLE)

1. **A Dynamic sweep based algorithm**:

   It can directly take into account the increase of the cumulated profile during a single sweep.

2. **A "good" average time complexity**:

   Essential in order to handle large instances.

3. **Does not create holes in domains**:

   A variable can be compactly represented by its min/max values.

4. **Takes advantage of the bin-packing**:

   A better worst-case time complexity than for the cumulative.

THE DYNAMIC SWEEP ALGORITHM
(PRINCIPLE)

- Deal with **domain bounds**. [CP2012]
  (Creates holes in the domains. [CP2001]**)**

- Filter min and max values in **two distinct sweep stages**: *sweep_min* and *sweep_max*, speeds up the convergence to the fixpoint. [CP2012]

- New **dynamic** and **conditional events** [CP2012]
  (Too static [CP2001]**)**

- Use dedicated **data structures**. [CP2012]
  (Often reaches its worst time complexity [CP2001]**)**

THE DYNAMIC SWEEP ALGORITHM
(PRINCIPLE: NEW EVENTS)

- **Event related to the end of the compulsory part of a task is now dynamic.**
- **A conditional event is generated for each task initially without compulsory part.**

The adjustment of the earliest start of the task can induce the creation of a compulsory part.

↓

The conditional event is transformed into 2 events reflecting the new compulsory part.

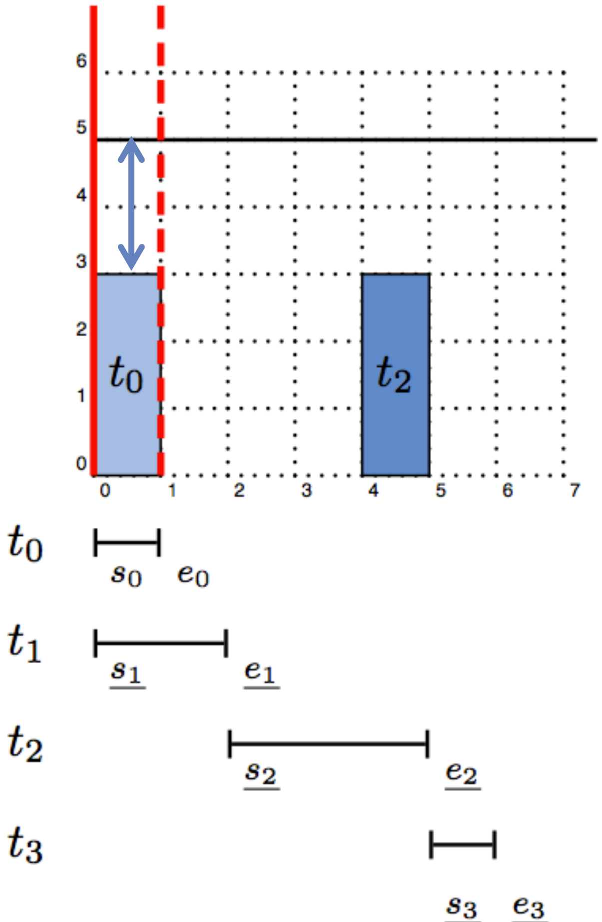
THE DYNAMIC SWEEP ALGORITHM
(PRINCIPLE: DATA STRUCTURES)

To partially avoid rescanning of all tasks:

- **A heap $h_{conflict}$ storing tasks in conflict with the current sweep interval. Tasks are ordered by increasing height.**

- **A heap h_{check} storing tasks not in conflict on the current sweep interval and for which the earliest start is not yet found. Tasks are ordered by decreasing height.**

THE DYNAMIC SWEEP ALGORITHM
(ILLUSTRATION)

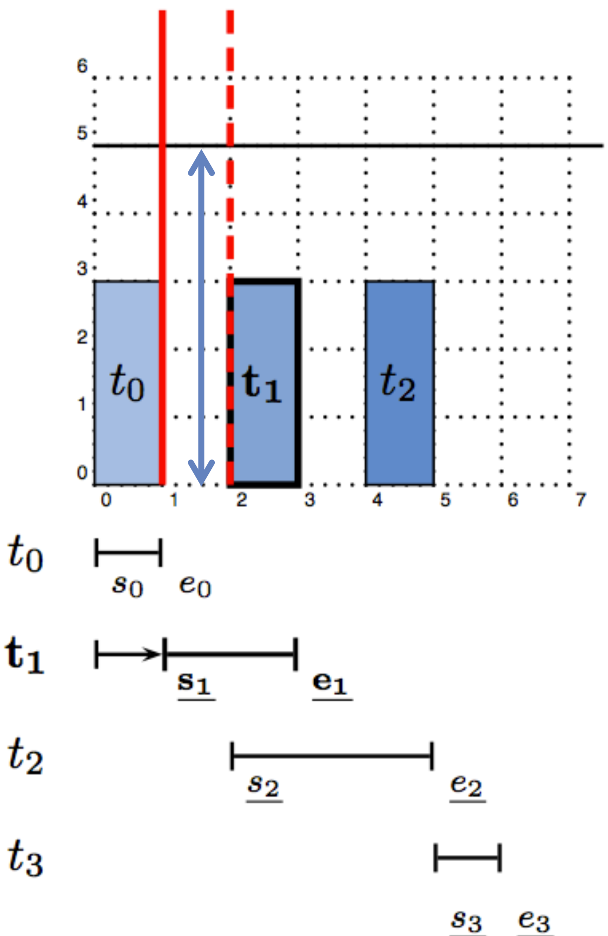


sweep interval = [0,1)
available resource = 2

h_1 (=3) **is greater than** the available resource (=2).

t_1 is added into $h_{conflict}$.

THE DYNAMIC SWEEP ALGORITHM
(ILLUSTRATION)



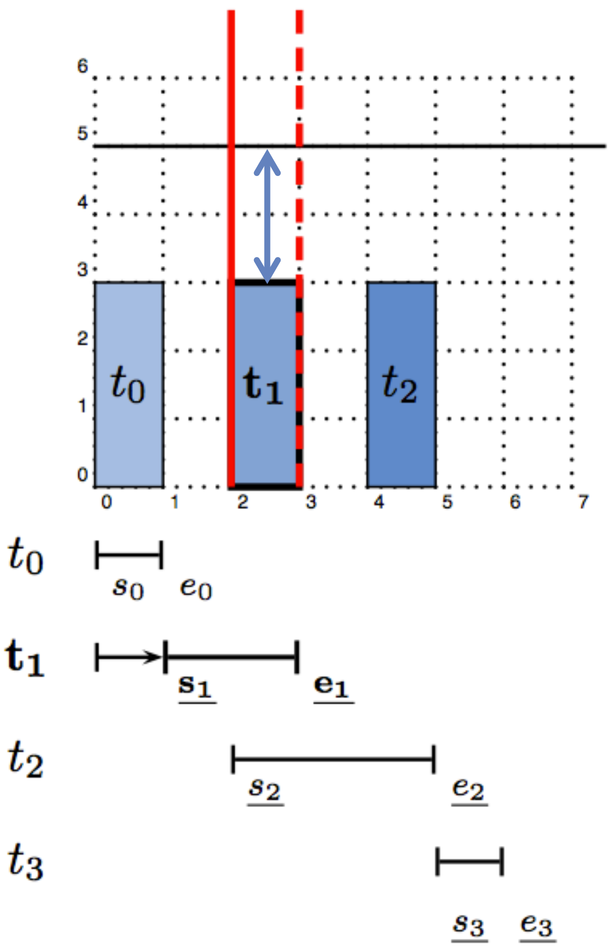
sweep interval = [1,2)

available resource = 5

Top task of $h_{conflict}$ (t_1) **is not greater than** the available resource. Consequently t_1 is removed from $h_{conflict}$ and added into h_{check}.

Earliest start of t_1 is adjusted to 1.
Its conditional event is transformed into
2 events reflecting its new compulsory part

THE DYNAMIC SWEEP ALGORITHM
(ILLUSTRATION)

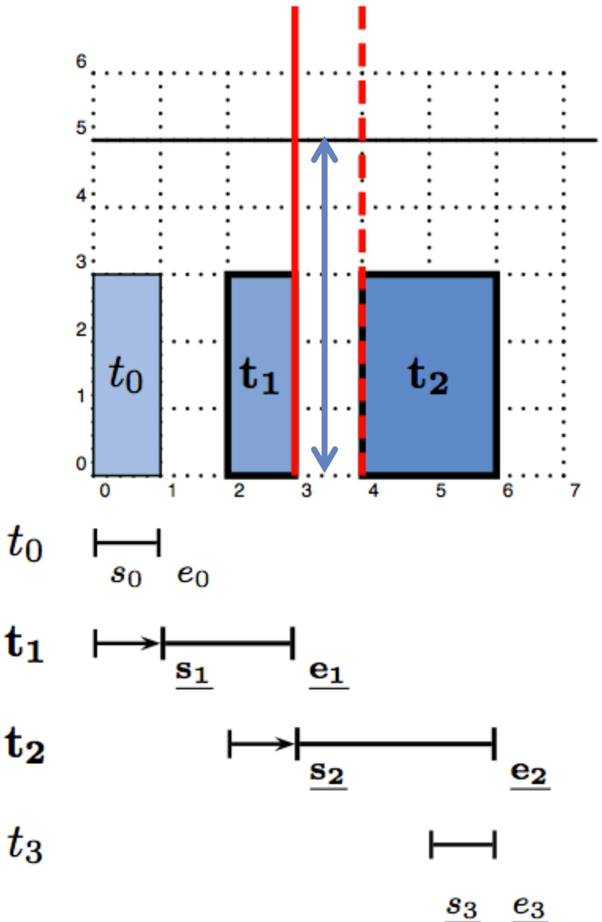


sweep interval = [2,3)
available resource = 2

h_2 (=3) **is greater than** the available resource (=2).

t_2 is added into $h_{conflict}$.

THE DYNAMIC SWEEP ALGORITHM
(ILLUSTRATION)



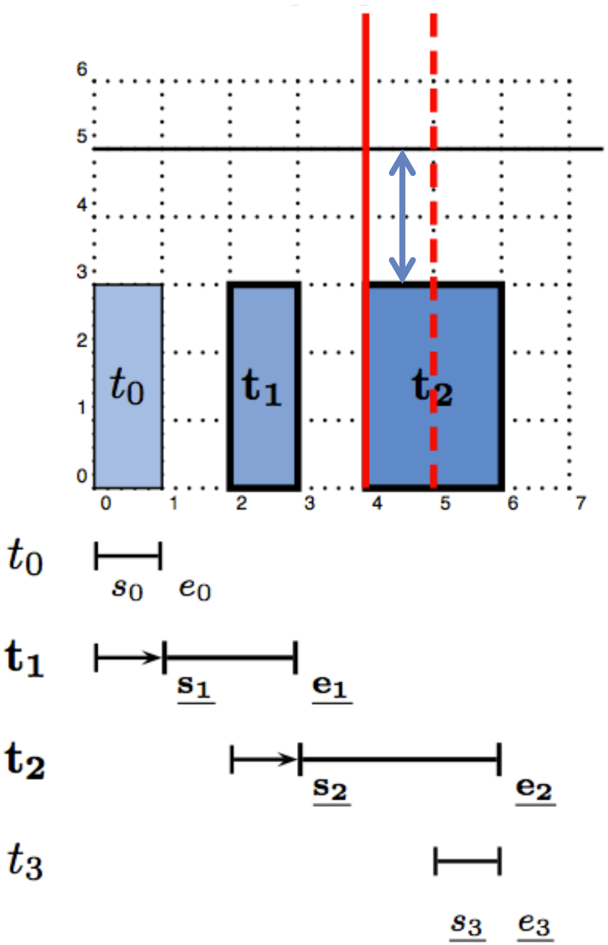
sweep interval = [3,4)

available resource = 5

Top task of $h_{conflict}$ (t_2) **is not greater than** the available resource. Consequently t_2 is removed from $h_{conflict}$ and added into h_{check}.

Earliest start of t_2 is adjusted to 3.
Event related to its end of compulsory part is pushed from 5 to 6.

THE DYNAMIC SWEEP ALGORITHM
(ILLUSTRATION)

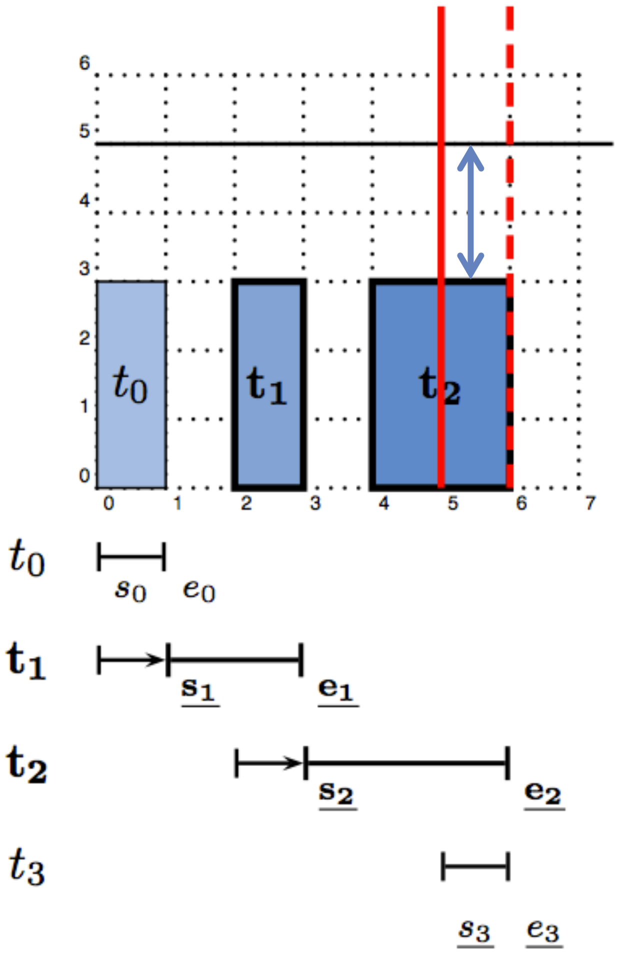


sweep interval = [4,5)

available resource = 2

Nothing to do.

THE DYNAMIC SWEEP ALGORITHM
(ILLUSTRATION)

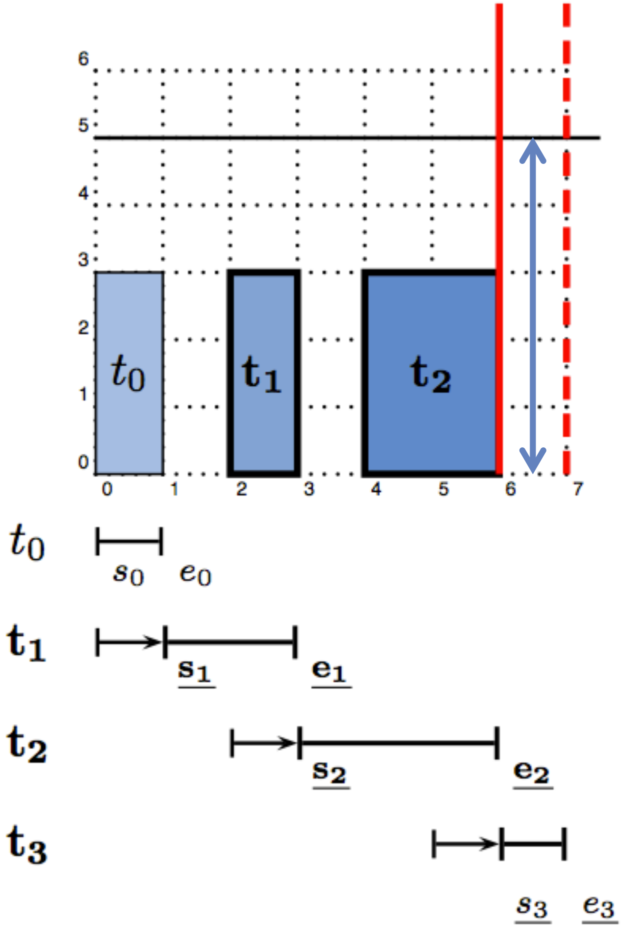


sweep interval = [5,6)
available resource = 2

h_3 (=3) **is greater than** the available resource (=2).

t_3 is added into $h_{conflict}$.

THE DYNAMIC SWEEP ALGORITHM
(ILLUSTRATION)



sweep interval = [6,7)
available resource = 5

Top task of $h_{conflict}$ (t_3) **is not greater than** the available resource. Consequently t_3 is removed from $h_{conflict}$.

Earliest start of t_3 is adjusted to 6. Nothing else to do.

THE DYNAMIC SWEEP ALGORITHM
(PROPERTY AND COMPLEXITY)

- **A worst-case time complexity of *$O(n^2 \log n)$* where *n* is the number of tasks.**

  There is a variant with a worst-case time complexity of $O(n^2)$,
  but the $O(n^2 \log n)$ version scales better.

- **Property after a call to *sweep_min:***

For any task *t* in *T*, one can schedule *t* at its earliest start without exceeding the resource limit wrt. the cumulated profile of *$T \backslash \{t\}$*.

THE DYNAMIC SWEEP ALGORITHM
(GREEDY MODE USING FILTERING)

Why ?

To handle larger (10 million tasks) instances in a CP solver.

How ?

It reuses the *sweep_min* part but directly fixes the start of the task rather than adjusting it. Then, the sweep-line is reset to this start and the process continues until all tasks get fixed or a resource overflow occurs.

OUTLINE

The *cumulative* Constraint

A Critical Analysis of the [CP2001] Sweep Algorithm

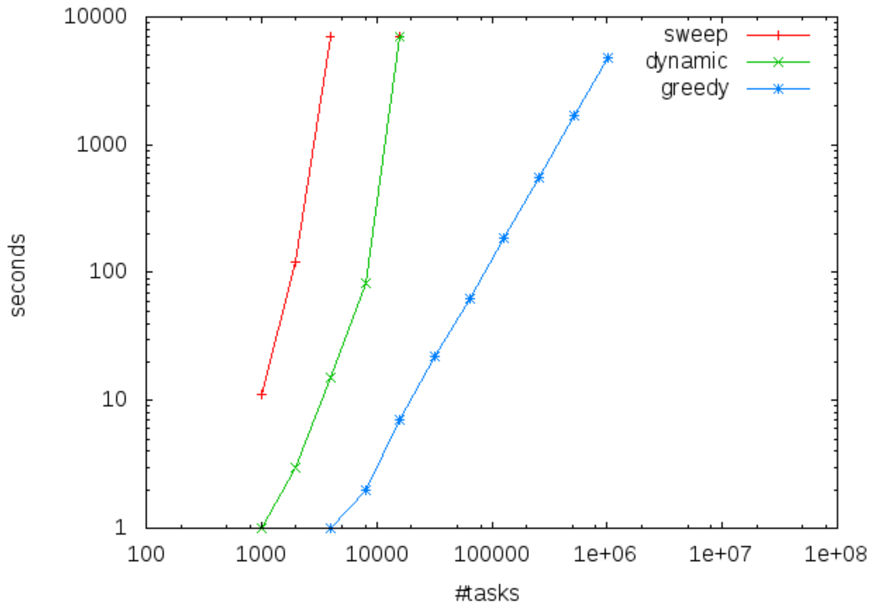
The Dynamic Sweep Algorithm

Evaluation

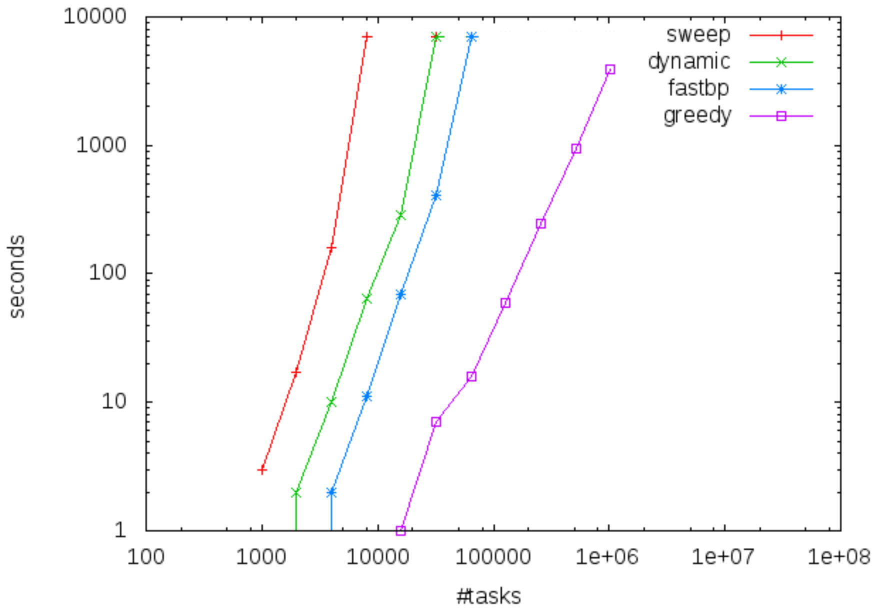
Conclusion

EVALUATION

Cumulative



Bin-Packing



- Random instances with a density close to 0.7.
- A **speedup increasing** with the number of tasks.
- The dynamic is **more robust** than the 2001 sweep wrt. different heuristics.
- The greedy mode could handle:

  1 million tasks in **12 minutes**

  up to **10^7 tasks** in ~8h (swap).

OUTLINE

The *cumulative* Constraint

A Critical Analysis of the [CP2001] Sweep Algorithm

The Dynamic Sweep Algorithm

Evaluation

Conclusion

CONCLUSION

- a **lean** sweep based filtering algorithm
- **dynamically** handle creation/extension of CP
- **faster** and **more scalable** than the 2001 sweep
- handle up to **10 million tasks** in greedy mode, **16000 tasks** in non-greedy mode.