LAAS CNRS

Resource Constraints in Scheduling

Emmanuel Hebrard

Content: constraint programming for scheduling

Content: constraint programming for scheduling

Part I: Propagation of resource constraints

- Constraint programming view of scheduling
- Main bibliographic sources
  - Constraint-based Scheduling (Baptiste, Le Pape, and Nuijten, 2001)
  - Petr Vilím's thesis (Vilím, 2007)

Content: constraint programming for scheduling

Part I: Propagation of resource constraints

- Constraint programming view of scheduling
- Main bibliographic sources
  - Constraint-based Scheduling (Baptiste, Le Pape, and Nuijten, 2001)
  - Petr Vilím's thesis (Vilím, 2007)

Part II: A very quick word on search

- My biased view

Content: constraint programming for scheduling

Part I: Propagation of resource constraints

- Constraint programming view of scheduling
- Main bibliographic sources
  - Constraint-based Scheduling (Baptiste, Le Pape, and Nuijten, 2001)
  - Petr Vilím's thesis (Vilím, 2007)

Part II: A very quick word on search

- My biased view

Part III: A glimpse of other types of resources

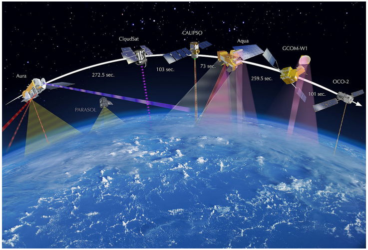
- Resources come in every form and shape, Rosetta/Philae example

What is scheduling?

- "Allocating scarce resources to activities over time" (Baker, 1974)

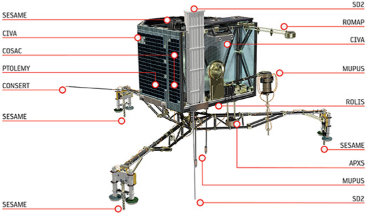
File download from observation satellites

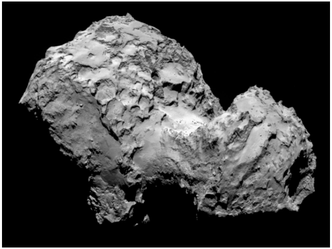
- Jobs: Files to transfer
- Resources:
  - Download channels: at most that many simultaneous downloads
  - Memory banks: cannot download two files stored on the same memory bank simultaneously
- Download as much data as possible within a given time window



Planning the mission of Philae on the comet 67P

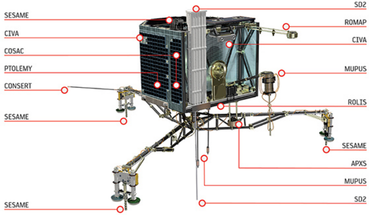
- Jobs: Scientific experiments
- Resources:
  - Batteries: threshold on the instant energy consumption
  - Memory: experiments produce data and transfers are possible only when Rosetta is visible
- Maximise the lifespan of the batterie





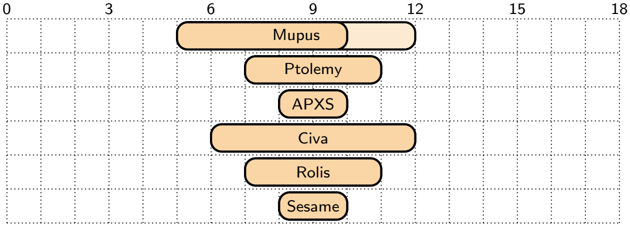
Planning the mission of Philae on the comet 67P

- Jobs: Scientific experiments
- Resources:
  - Batteries: threshold on the instant energy consumption
  - Memory: experiments produce data and transfers are possible only when Rosetta is visible
- Maximise the lifespan of the batterie



Notations

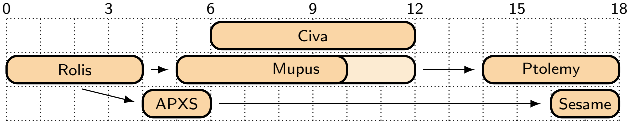
- A *task* i is represented as a rectangle
  - width represents processing time p_i (possibly variable)



Notations

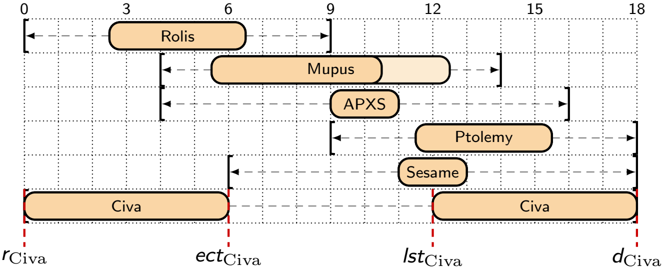
- A *task* i is represented as a rectangle
  - width represents processing time p_i (possibly variable)

- There can be *precedences* between start or end of tasks



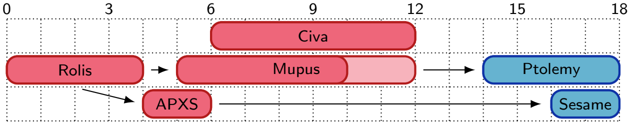
Notations

- A *task* i is represented as a rectangle
  - width represents processing time p_i (possibly variable)
  - its *domain* is given by its release date r_i and due date d_i
    - we denote $lst_i = d_i - p_i$ its latest start time and $ect_i = r_i + p_i$ its earliest completion time
- There can be *precedences* between start or end of tasks



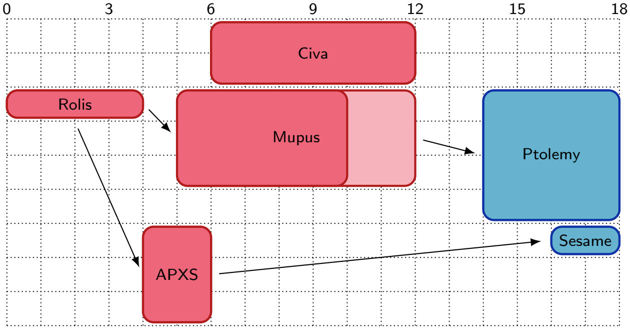
Notations

- A *task* i is represented as a rectangle
  - width represents processing time p_i (possibly variable)
  - its *domain* is given by its release date r_i and due date d_i
    - we denote $lst_i = d_i - p_i$ its latest start time and $ect_i = r_i + p_i$ its earliest completion time
- There can be *precedences* between start or end of tasks
- There can be *resources* required by some tasks



Notations

- A *task* i is represented as a rectangle
  - width represents processing time p_i (possibly variable)
  - height represents consumption c_i (possibly variable)
  - its *domain* is given by its release date r_i and due date d_i
    - we denote $lst_i = d_i - p_i$ its latest start time and $ect_i = r_i + p_i$ its earliest completion time
- There can be *precedences* between start or end of tasks
- There can be *resources* required by some tasks, with a given capacity C



Scheduling problem

- Assign a start time s_i and an completion time e_i to every task $i \in \mathcal{T}$ such that:
  - Precedence constraints are satisfied
  - Resource constraints are satisfied

Scheduling problem

- Assign a start time s_i and an completion time e_i to every task $i \in \mathcal{T}$ such that:
  - Precedence constraints are satisfied
  - Resource constraints are satisfied
- Objectives:
  - Minimize makespan ($\max\{e_i \mid i \in \mathcal{T}\}$)
  - Minimize total weighted tardiness ($\sum_{i \in \mathcal{T}} w_i \max(0, e_i - \delta_i)$)
  - ...

Scheduling problem

- Assign a start time s_i and an completion time e_i to every task $i \in \mathcal{T}$ such that:
  - Precedence constraints are satisfied
  - Resource constraints are satisfied
- Objectives:
  - Minimize makespan ($\max\{e_i \mid i \in \mathcal{T}\}$)
  - Minimize total weighted tardiness ($\sum_{i \in \mathcal{T}} w_i \max(0, e_i - \delta_i)$)
  - ...
- Within Constraint Programming: not so important
  - Handled by a constraint
  - Depend on start and end times of tasks

LAAS CNRS

Constraint vs Algorithm

- Constraints & models written using *variables* symbols (s_i, e_i)
- Algorithmic rules written using *domain* symbols (r_i,lst_i,ect_i,d_i)

Constraint vs Algorithm

- Constraints & models written using *variables* symbols (s_i, e_i)
- Algorithmic rules written using *domain* symbols (r_i,lst_i,ect_i,d_i)
- Example: precedence $i \prec j$
  - Constraint predicate:

$$e_i \leq s_j$$

Constraint vs Algorithm

- Constraints & models written using *variables* symbols (s_i, e_i)
- Algorithmic rules written using *domain* symbols (r_i,lst_i,ect_i,d_i)
- Example: precedence $i \prec j$
  - Constraint predicate:

$$e_i \leq s_j$$

  - Algorithmic rules:

$$ect_i > r_j \implies r_j = ect_i$$
$$lst_j < d_i \implies d_i = lst_j$$

Constraint vs Algorithm

- Constraints & models written using *variables* symbols (s_i, e_i)
- Algorithmic rules written using *domain* symbols (r_i,lst_i,ect_i,d_i)
- Example: precedence $i \prec j$
  - Constraint predicate:

$$e_i \leq s_j$$

  - Algorithmic rules:

$$ect_i > r_j \implies r_j = ect_i$$
$$lst_j < d_i \implies d_i = lst_j$$

- Notion of consistency: constraints are sufficient to define the result of propagation

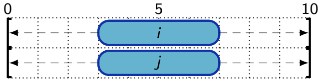
Bound consistency

Constraint C over variables $\mathcal{X}$

- C: predicate defining a relation in $\mathbb{N}^{|\mathcal{X}|}$

Ex: $i \prec j$

- Predicate: $e_i \leq s_j$



Bound consistency

Constraint C over variables $\mathcal{X}$

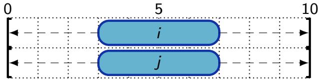
- C: predicate defining a relation in $\mathbb{N}^{|\mathcal{X}|}$

Bound support σ of x, t for C over $\mathcal{X}$

- $\sigma : \mathcal{X} \mapsto \mathbb{N}$ with $\sigma(x) = t$
- valid $\iff \forall x \in \mathcal{X}\ \min(x) \leq \sigma(x) \leq \max(x)$
- consistent $\iff C(\sigma(\mathcal{X}))$

Ex: $i \prec j$

- Predicate: $e_i \leq s_j$



- Ex.: no valid and consistent bound support for $e_i = 7$
  - consistent: $\langle e_i : 7, s_j : 7\rangle$
  - valid: $\langle e_i : 7, s_j : 6\rangle$

Bound consistency

Constraint C over variables $\mathcal{X}$

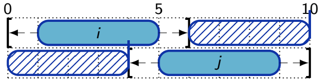
- C: predicate defining a relation in $\mathbb{N}^{|\mathcal{X}|}$

Bound support σ of x, t for C over $\mathcal{X}$

- $\sigma : \mathcal{X} \mapsto \mathbb{N}$ with $\sigma(x) = t$
- valid $\iff \forall x \in \mathcal{X} \; \min(x) \leq \sigma(x) \leq \max(x)$
- consistent $\iff C(\sigma(\mathcal{X}))$

Ex: $i \prec j$

- Predicate: $e_i \leq s_j$



- Ex.: no valid and consistent bound support for $e_i = 7$
  - consistent: $\langle e_i : 7, s_j : 7 \rangle$
  - valid: $\langle e_i : 7, s_j : 6 \rangle$

- Result of propagation algorithm is entailed by "*bound consistency on* $e_i \leq s_j$"
- Its complexity is not

Resource Constraint

- Very rich taxonomy of resources
- Focus on Renewable, discrete, non-interruptible, cumulative resource

A model of CumulativeResource

- Processing time: require the resources for at least p_i time $\qquad s_i + p_i \leq e_i \; \forall i$
- Non preemption: cannot be interrupted $\qquad s_i + p_i \geq e_i \; \forall i$
- Bounds: release and due dates $\qquad r_i \leq s_i \leq e_i \leq d_i \; \forall i$
- Resource capacity: additive, upper bounded resource usage $\qquad \sum_{s_i \leq t \leq e_i} c_i \leq C \; \forall t$

Resource Constraint

- Very rich taxonomy of resources
- Focus on Renewable, discrete, non-interruptible, cumulative resource

A model of CumulativeResource

- Processing time: require the resources for at least p_i time $s_i + p_i \leq e_i \; \forall i$
- Non preemption: cannot be interrupted $s_i + p_i \geq e_i \; \forall i$
- Bounds: release and due dates $r_i \leq s_i \leq e_i \leq d_i \; \forall i$
- Resource capacity: additive, upper bounded resource usage $\sum_{s_i \leq t \leq e_i} c_i \leq C \; \forall t$

- **File transfer**; memory bank: single-machine resource ($c_i = C = 1$)

Resource Constraint

- Very rich taxonomy of resources
- Focus on Renewable, discrete, non-interruptible, cumulative resource

A model of CumulativeResource

- Processing time: require the resources for at least p_i time $s_i + p_i \leq e_i \; \forall i$
- Non preemption: cannot be interrupted $s_i + p_i \geq e_i \; \forall i$
- Bounds: release and due dates $r_i \leq s_i \leq e_i \leq d_i \; \forall i$
- Resource capacity: additive, upper bounded resource usage $\sum_{s_i \leq t \leq e_i} c_i \leq C \; \forall t$

- **File transfer**; memory bank: single-machine resource ($c_i = C = 1$)
- **File transfer**; download channel: m-machine resource ($c_i = 1, C > 1$)

Resource Constraint

- Very rich taxonomy of resources
- Focus on Renewable, discrete, non-interruptible, cumulative resource

A model of CUMULATIVERESOURCE

- Processing time: require the resources for at least p_i time $s_i + p_i \leq e_i \ \forall i$
- Non preemption: cannot be interrupted $s_i + p_i \geq e_i \ \forall i$
- Bounds: release and due dates $r_i \leq s_i \leq e_i \leq d_i \ \forall i$
- Resource capacity: additive, upper bounded resource usage $\sum_{s_i \leq t \leq e_i} c_i \leq C \ \forall t$

- **File transfer**; memory bank: single-machine resource ($c_i = C = 1$)
- **File transfer**; download channel: m-machine resource ($c_i = 1, C > 1$)
- **Philae**; battery power threshold: cumulative resource ($c_i \geq 1, C > 1$)

Coping with NP-hardness

A model of CUMULATIVERESOURCE

$$
\begin{aligned}
s_i + p_i &\geq e_i && \forall i \\
s_i + p_i &\leq e_i && \forall i \\
r_i \leq s_i \leq e_i &\leq d_i && \forall i \\
\sum_{s_i \leq t \leq e_i} c_i &\leq C && \forall t
\end{aligned}
$$

- The problem CUMULATIVERESOURCE is strongly NP-hard
  - So is bound consistency, since a bound support is a valid schedule

Coping with NP-hardness

A model of CumulativeResource

$$
\begin{aligned}
s_i + p_i &\geq e_i \quad \forall i\\
s_i + p_i &\leq e_i \quad \forall i\\
r_i \leq s_i \leq e_i &\leq d_i \quad \forall i\\
\sum_{s_i \leq t \leq e_i} c_i &\leq C \quad \forall t
\end{aligned}
$$

- The problem CumulativeResource is strongly NP-hard
  - So is bound consistency, since a bound support is a valid schedule
- Relaxation: if the relaxed problem is unsatisfiable, so is the original problem

Coping with NP-hardness

A decomposition of CUMULATIVERESOURCE

$$
\begin{aligned}
s_i + p_i \geq e_i & \quad \forall i \\
s_i + p_i \leq e_i & \quad \forall i \\
r_i \leq s_i \leq e_i \leq d_i & \quad \forall i \\
\sum_{s_i \leq t \leq e_i} c_i \leq C & \quad \forall t
\end{aligned}
$$

- The problem CUMULATIVERESOURCE is strongly NP-hard
  - So is bound consistency, since a bound support is a valid schedule
- Relaxation: if the relaxed problem is unsatisfiable, so is the original problem
- Decomposition:
  - Enforcing bound consistency on CUMULATIVERESOURCE is NP-hard (global support)
  - Enforcing bound consistency on the model above is polynomial (local supports)

Time-tabling

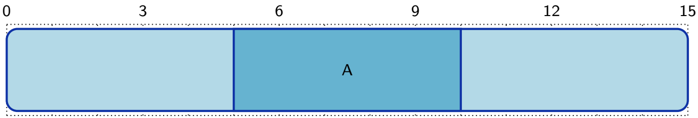
- Notion of "compulsory part" (Lahrichi, 1982)
  - Period in which the task must be in process

Time-tabling

- Notion of "compulsory part" (Lahrichi, 1982)
  - Period in which the task must be in process

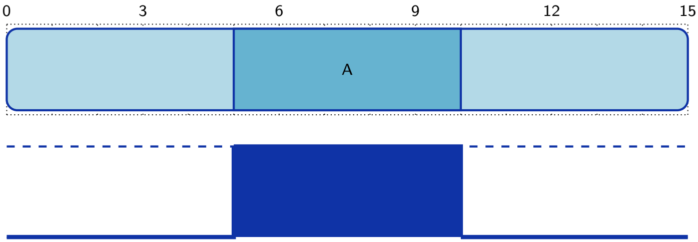
Time-tabling

- Notion of "compulsory part" (Lahrichi, 1982)
  - Period in which the task must be in process



Time-tabling

- Notion of “compulsory part” (Lahrichi, 1982)
  - Period in which the task must be in process
- Notion of “time-tables” (Le Pape, 1988), “resource profile” (Fox, 1990), “resource histogram” (Caseau and Laburthe, 1996)
  - Minimum usage of the resource over time



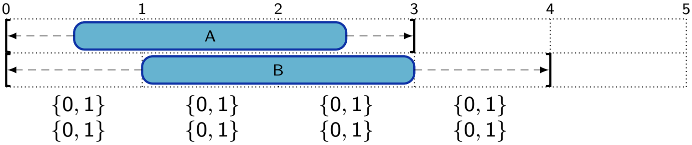
Time-tabling decomposition

- Boolean variables a_i^t standing for: "task i is in process at time t"
- Enforce bounds consistency on:

$$\forall i \quad s_i + p_i = e_i \quad \text{processing time \& non-preemption} \quad (1)$$

$$\forall t \quad \sum_{i \in \mathcal{T}} c_i a_i^t \leq C \quad \text{resource capacity} \quad (2)$$

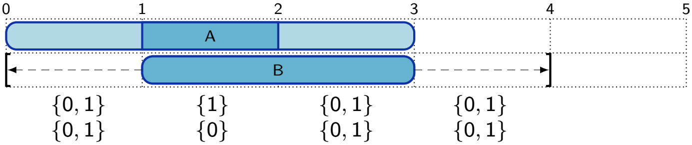
$$\forall i \forall t \quad a_i^t \iff s_i \leq t \wedge t < e_i \quad \text{channeling} \quad (3)$$



Time-tabling decomposition

- Boolean variables a_i^t standing for: "task i is in process at time t"
- Enforce bounds consistency on:

$\forall i$	$s_i + p_i = e_i$	processing time & non-preemption	(1)
$\forall t$	$\sum_{i \in \mathcal{T}} c_i a_i^t \leq C$	resource capacity	(2)
$\forall i \forall t$	$a_i^t \iff s_i \leq t \wedge t < e_i$	channeling	(3)

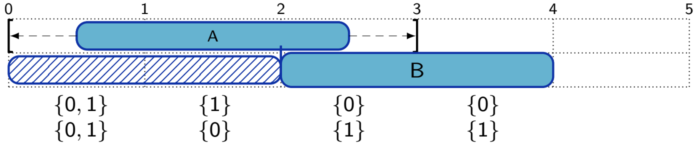


- $s_A \leq 1 \wedge 1 < e_A \Rightarrow a_A^1 = 1$ — A must be in process at $t = 1$
- $\Rightarrow a_B^1 = 0 \Rightarrow e_B \leq 1 \vee 1 < s_B$ — B cannot be in process at $t = 1$

Time-tabling decomposition

- Boolean variables a_i^t standing for: "task i is in process at time t"
- Enforce bounds consistency on:

$$\forall i \quad s_i + p_i = e_i \quad \text{processing time \& non-preemption} \quad (1)$$
$$\forall t \quad \sum_{i \in \mathcal{T}} c_i a_i^t \leq C \quad \text{resource capacity} \quad (2)$$
$$\forall i \forall t \quad a_i^t \iff s_i \leq t \wedge t < e_i \quad \text{channeling} \quad (3)$$



- $s_A \leq 1 \wedge 1 < e_A \Rightarrow a_A^1 = 1$ — A must be in process at $t = 1$
- $\Rightarrow a_B^1 = 0 \Rightarrow e_B \leq 1 \vee 1 < s_B$ — B cannot be in process at $t = 1$
- $\Rightarrow 1 < s_B$ — B cannot end earlier than $t = 2$ so it must start at $t \geq 2$

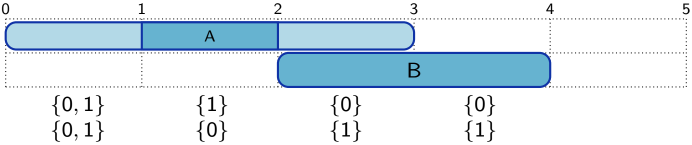
Time-tabling decomposition

- Boolean variables a_i^t standing for: "task i is in process at time t"
- Enforce bounds consistency on:

$$\forall i \quad s_i + p_i = e_i \quad \text{processing time \& non-preemption} \quad (1)$$

$$\forall t \quad \sum_{i \in \mathcal{T}} c_i a_i^t \leq C \quad \text{resource capacity} \quad (2)$$

$$\forall i \forall t \quad a_i^t \iff s_i \leq t \wedge t < e_i \quad \text{channeling} \quad (3)$$

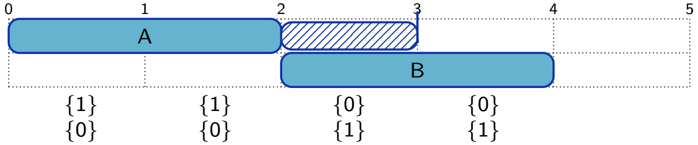


- $s_A \leq 1 \wedge 1 < e_A \Rightarrow a_A^1 = 1$ — A must be in process at $t = 1$
- $\Rightarrow a_B^1 = 0 \Rightarrow e_B \leq 1 \vee 1 < s_B$ — B cannot be in process at $t = 1$
- $\Rightarrow 1 < s_B$ — B cannot end earlier than $t = 2$ so it must start at $t \geq 2$

Time-tabling decomposition

- Boolean variables a_i^t standing for: "task i is in process at time t"
- Enforce bounds consistency on:

$$\begin{array}{llll} \forall i & s_i + p_i = e_i & \text{processing time \& non-preemption} & (1) \\ \forall t & \sum_{i \in \mathcal{T}} c_i a_i^t \leq C & \text{resource capacity} & (2) \\ \forall i \forall t & a_i^t \iff s_i \leq t \wedge t < e_i & \text{channeling} & (3) \end{array}$$

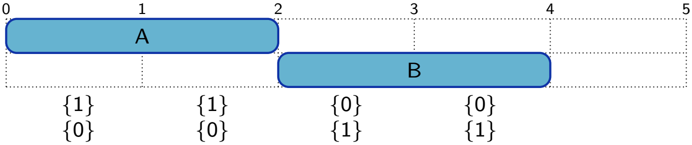


- $s_A \leq 1 \wedge 1 < e_A \Rightarrow a_A^1 = 1$ — A must be in process at $t = 1$
- $\Rightarrow a_B^1 = 0 \Rightarrow e_B \leq 1 \vee 1 < s_B$ — B cannot be in process at $t = 1$
- $\Rightarrow 1 < s_B$ — B cannot end earlier than $t = 2$ so it must start at $t \geq 2$

Time-tabling decomposition

- Boolean variables a_i^t standing for: "task i is in process at time t"
- Enforce bounds consistency on:

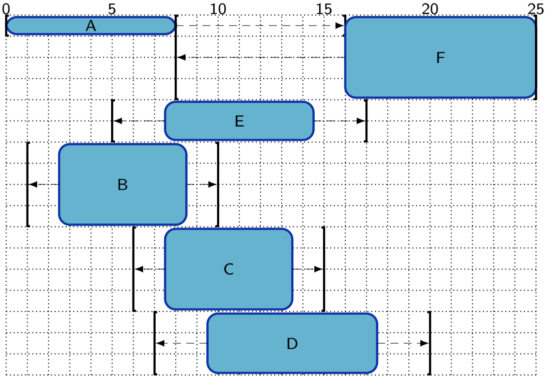
$$
\begin{array}{llll}
\forall i & s_i + p_i = e_i & \text{processing time \& non-preemption} & (1) \\
\forall t & \sum_{i \in \mathcal{T}} c_i a_i^t \leq C & \text{resource capacity} & (2) \\
\forall i \forall t & a_i^t \iff s_i \leq t \wedge t < e_i & \text{channeling} & (3)
\end{array}
$$



- $s_A \leq 1 \wedge 1 < e_A \Rightarrow a_A^1 = 1$ — A must be in process at $t = 1$
- $\Rightarrow a_B^1 = 0 \Rightarrow e_B \leq 1 \vee 1 < s_B$ — B cannot be in process at $t = 1$
- $\Rightarrow 1 < s_B$ — B cannot end earlier than $t = 2$ so it must start at $t \geq 2$

Time-tabling algorithm: satisfiability check

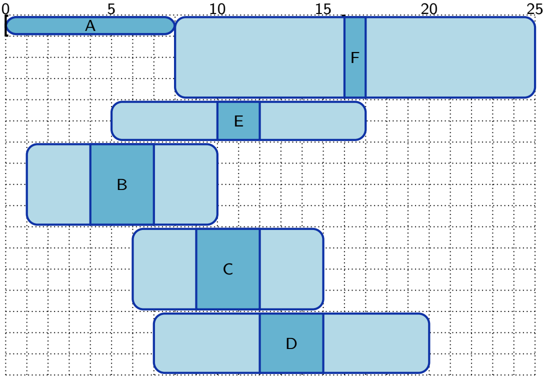
Resource profile



Time-tabling algorithm: satisfiability check

Resource profile

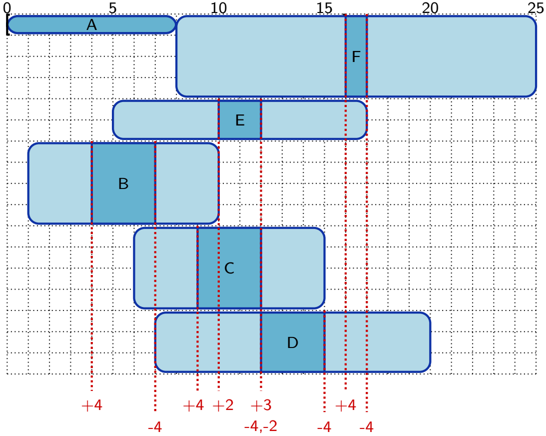
- Compute compulsory parts



Time-tabling algorithm: satisfiability check

Resource profile

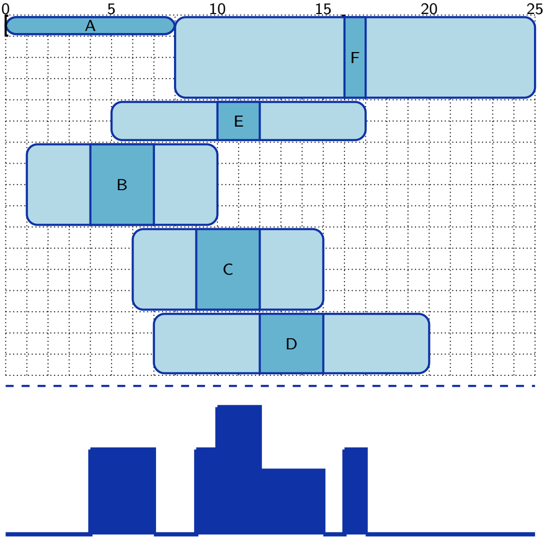
- Compute compulsory parts
- Sort "events" (start and end times of compulsory parts) ($O(n \log n)$)



Time-tabling algorithm: satisfiability check

Resource profile

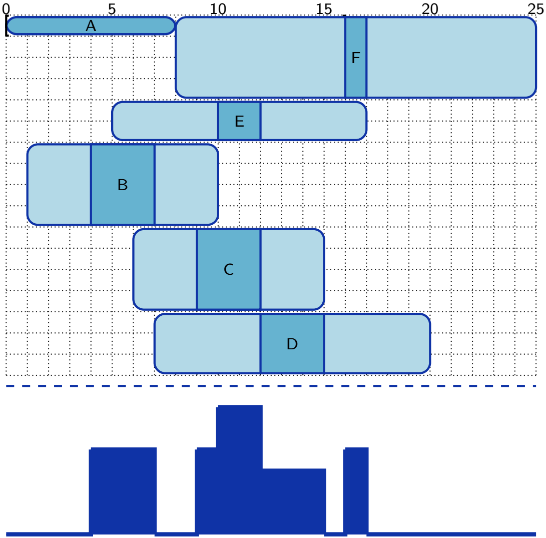
- Compute compulsory parts
- Sort “events” (start and end times of compulsory parts) ($O(n \log n)$)
- Process events to compute the profile P ($O(n)$)



Time-tabling algorithm: satisfiability check

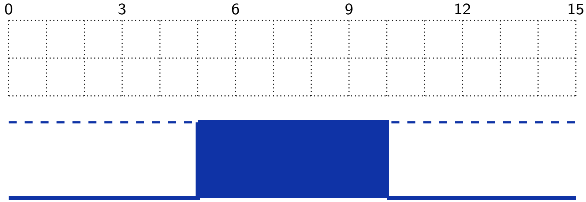
Resource profile

- Compute compulsory parts
- Sort "events" (start and end times of compulsory parts) ($O(n \log n)$)
- Process events to compute the profile P ($O(n)$)
- Sweep algorithm (Beldiceanu and Carlsson, 2001) (more general)



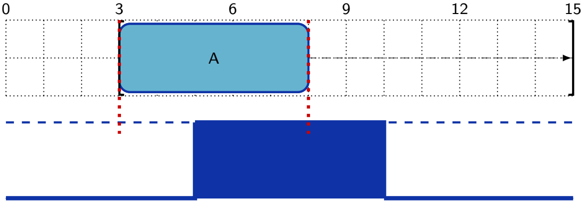
Time-tabling algorithm: bound propagator

- Assume $C = 3$ and consider the profile interval $[5, 10) : 2$



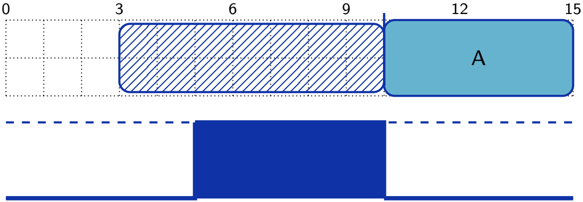
Time-tabling algorithm: bound propagator

- Assume $C = 3$ and consider the profile interval $[5, 10) : 2$
- A task A such that $[r_A, lst_A)$ overlaps with $[5, 10)$



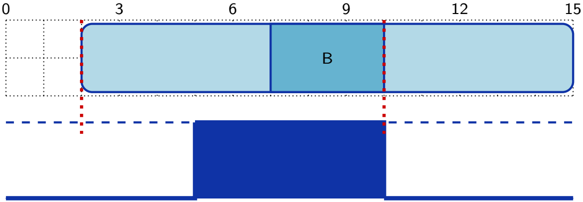
Time-tabling algorithm: bound propagator

- Assume $C = 3$ and consider the profile interval $[5, 10) : 2$
- A task A such that $[r_A, lst_A)$ overlaps with $[5, 10)$



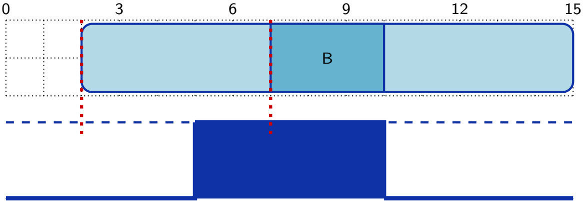
Time-tabling algorithm: bound propagator

- Assume $C = 3$ and consider the profile interval $[5, 10) : 2$
- A task B counted in the profile



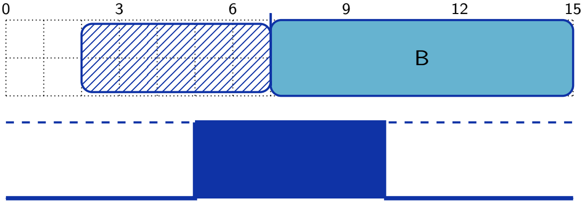
Time-tabling algorithm: bound propagator

- Assume $C = 3$ and consider the profile interval $[5, 10) : 2$
- A task B counted in the profile



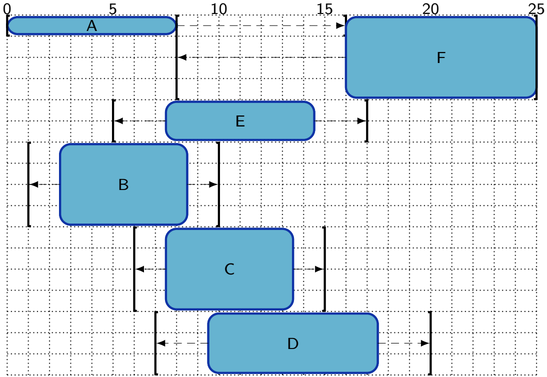
Time-tabling algorithm: bound propagator

- Assume $C = 3$ and consider the profile interval $[5, 10) : 2$
- A task B counted in the profile
- $[a, b)$ overlaps with $[r_i, \min(lst_i, ect_i)) \implies r_i = \min(b, lst_i)$



Time-tabling algorithm: bound propagator

- Interval $[a, b) : h$, Task i
  - overloaded iff $c_i + h > C$
  - relevant iff
    - $r_i < b$ and
    - $\min(ect_i, lst_i) > a$

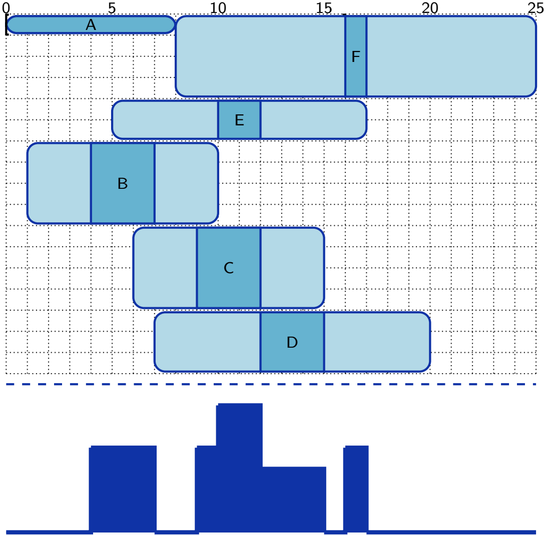


Time-tabling algorithm: bound propagator

- Interval $[a, b) : h$, Task i
  - overloaded iff $c_i + h > C$
  - relevant iff
    - $r_i < b$ and
    - $\min(ect_i, lst_i) > a$

Algorithm (Ouellet and Quimper, 2013)

- compute profile

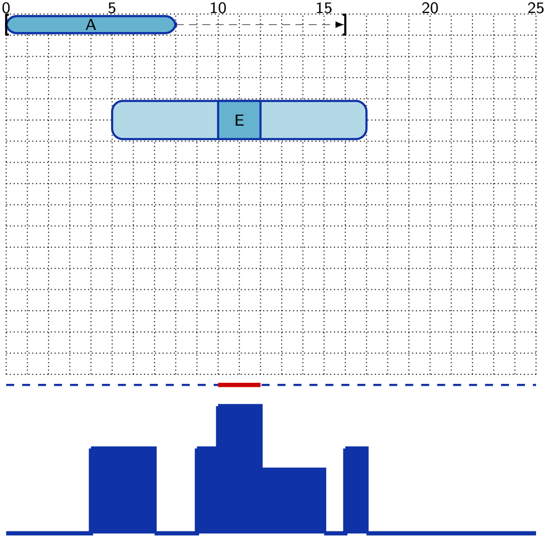


Time-tabling algorithm: bound propagator

- Interval $[a, b) : h$, Task i
  - overloaded iff $c_i + h > C$
  - relevant iff
    - $r_i < b$ and
    - $\min(ect_i, lst_i) > a$

Algorithm (Ouellet and Quimper, 2013)

- compute profile
- order chuncks by decreasing h
- for $i \in \mathcal{T}$ by increasing c_i:
  - add i's over. intervals to S

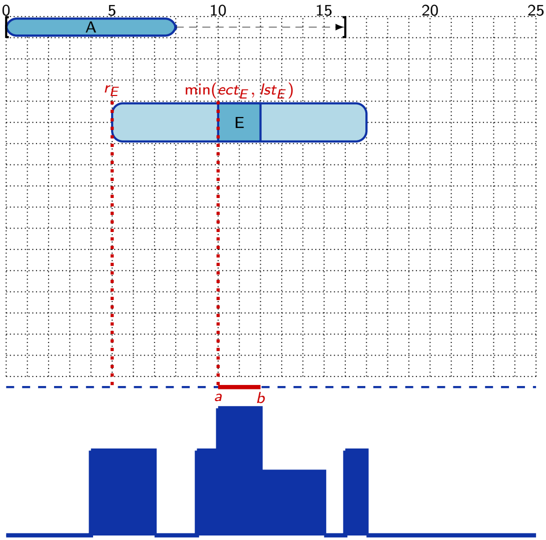


Time-tabling algorithm: bound propagator

- Interval $[a, b) : h$, Task i
  - overloaded iff $c_i + h > C$
  - relevant iff
    - $r_i < b$ and
    - $\min(ect_i, lst_i) > a$

Algorithm (Ouellet and Quimper, 2013)

- compute profile
- order chuncks by decreasing h
- for $i \in \mathcal{T}$ by increasing c_i:
  - add i's over. intervals to S
  - get i's relevant interval in S
    - $O(\log n)$ if S is an AVL tree

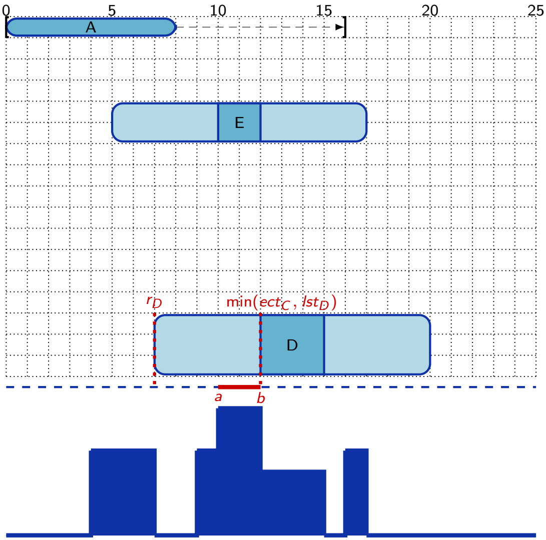


Time-tabling algorithm: bound propagator

- Interval $[a, b) : h$, Task i
  - overloaded iff $c_i + h > C$
  - relevant iff
    - $r_i < b$ and
    - $\min(ect_i, lst_i) > a$

Algorithm (Ouellet and Quimper, 2013)

- compute profile
- order chuncks by decreasing h
- for $i \in \mathcal{T}$ by increasing c_i:
  - add i's over. intervals to S
  - get i's relevant interval in S
    - $O(\log n)$ if S is an AVL tree

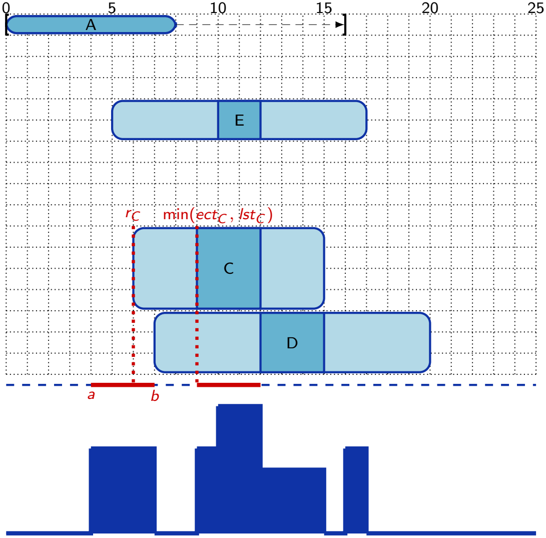


Time-tabling algorithm: bound propagator

- Interval $[a, b) : h$, Task i
  - overloaded iff $c_i + h > C$
  - relevant iff
    - $r_i < b$ and
    - $\min(ect_i, lst_i) > a$

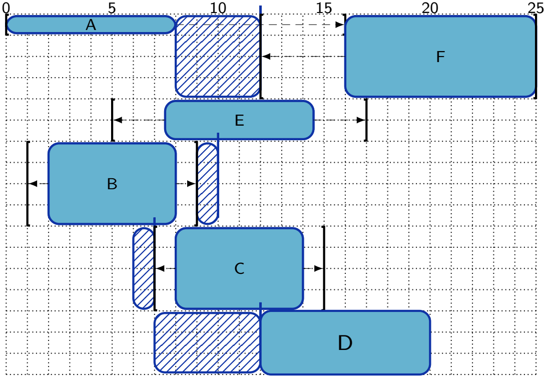
Algorithm (Ouellet and Quimper, 2013)

- compute profile
- order chuncks by decreasing h
- for $i \in \mathcal{T}$ by increasing c_i:
  - add i's over. intervals to S
  - get i's relevant interval in S
    - $O(\log n)$ if S is an AVL tree



Time-tabling algorithm: bound propagator

- Interval $[a, b) : h$, Task i
  - overloaded iff $c_i + h > C$
  - relevant iff
    - $r_i < b$ and
    - $\min(ect_i, lst_i) > a$



Algorithm (Ouellet and Quimper, 2013)

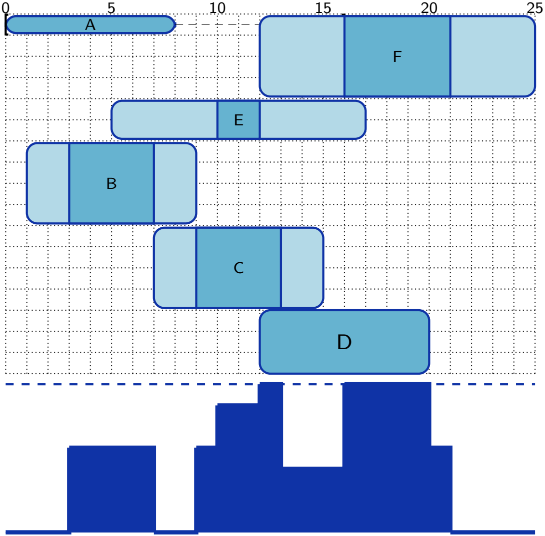
- compute profile
- order chuncks by decreasing h
- for $i \in \mathcal{T}$ by increasing c_i:
  - add i's over. intervals to S
  - get i's relevant interval in S
    - $O(\log n)$ if S is an AVL tree

Time-tabling algorithm: bound propagator

- Interval $[a, b)$: h, Task i
  - overloaded iff $c_i + h > C$
  - relevant iff
    - $r_i < b$ and
    - $\min(ect_i, lst_i) > a$

Algorithm (Ouellet and Quimper, 2013)

- compute profile
- order chuncks by decreasing h
- for $i \in \mathcal{T}$ by increasing c_i:
  - add i's over. intervals to S
  - get i's relevant interval in S
    - $O(\log n)$ if S is an AVL tree

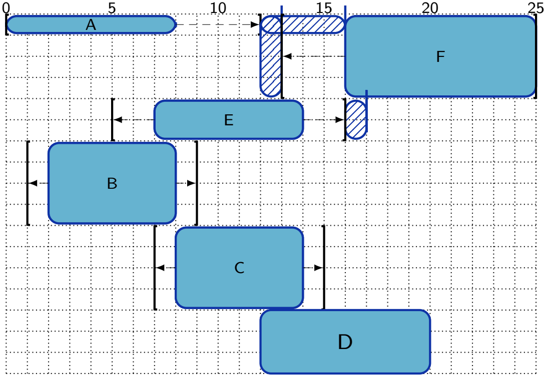


Time-tabling algorithm: bound propagator

- Interval $[a, b) : h$, Task i
  - overloaded iff $c_i + h > C$
  - relevant iff
    - $r_i < b$ and
    - $\min(ect_i, lst_i) > a$

Algorithm (Ouellet and Quimper, 2013)

- compute profile
- order chuncks by decreasing h
- for $i \in \mathcal{T}$ by increasing c_i:
  - add i's over. intervals to S
  - get i's relevant interval in S
    - $O(\log n)$ if S is an AVL tree

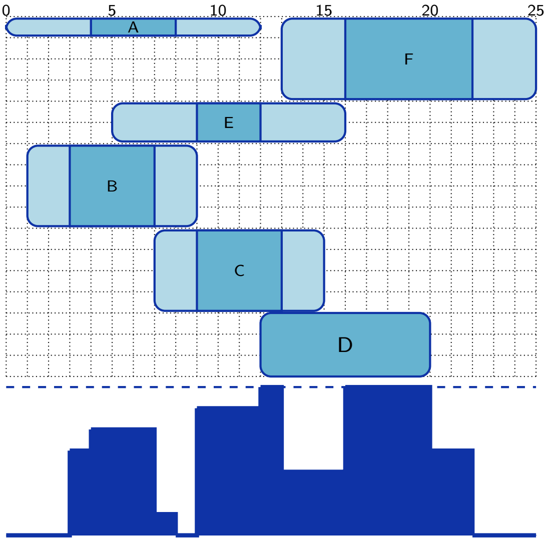


Time-tabling algorithm: bound propagator

- Interval $[a, b) : h$, Task i
  - overloaded iff $c_i + h > C$
  - relevant iff
    - $r_i < b$ and
    - $\min(ect_i, lst_i) > a$

Algorithm (Ouellet and Quimper, 2013)

- compute profile
- order chuncks by decreasing h
- for $i \in \mathcal{T}$ by increasing c_i:
  - add i's over. intervals to S
  - get i's relevant interval in S
    - $O(\log n)$ if S is an AVL tree

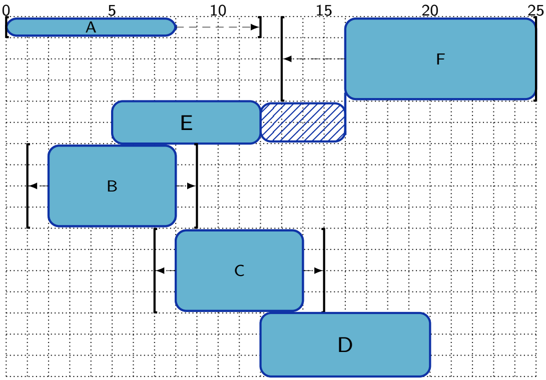


Time-tabling algorithm: bound propagator

- Interval $[a, b) : h$, Task i
  - overloaded iff $c_i + h > C$
  - relevant iff
    - $r_i < b$ and
    - $\min(ect_i, lst_i) > a$

Algorithm (Ouellet and Quimper, 2013)

- compute profile
- order chuncks by decreasing h
- for $i \in \mathcal{T}$ by increasing c_i:
  - add i's over. intervals to S
  - get i's relevant interval in S
    - $O(\log n)$ if S is an AVL tree

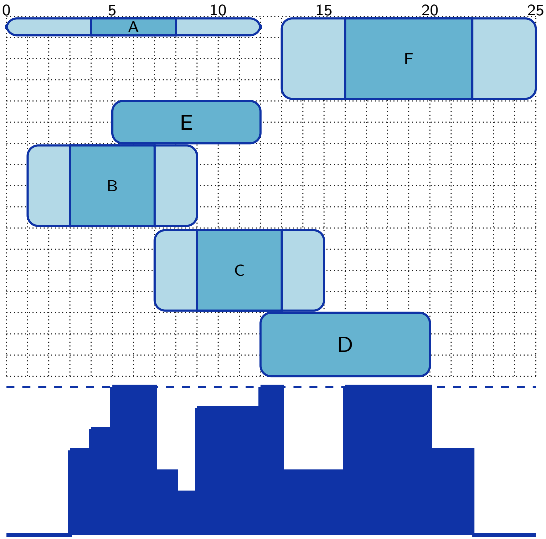


Time-tabling algorithm: bound propagator

- Interval $[a, b) : h$, Task i
  - overloaded iff $c_i + h > C$
  - relevant iff
    - $r_i < b$ and
    - $\min(ect_i, lst_i) > a$

Algorithm (Ouellet and Quimper, 2013)

- compute profile
- order chuncks by decreasing h
- for $i \in \mathcal{T}$ by increasing c_i:
  - add i's over. intervals to S
  - get i's relevant interval in S
    - $O(\log n)$ if S is an AVL tree



Time-tabling Algorithms

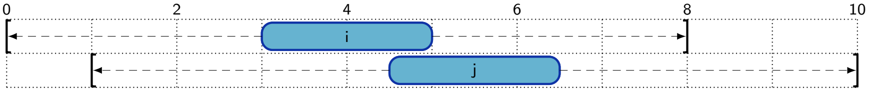
- Sweep algorithm (Beldiceanu and Carlsson, 2001)
- $O(n^2)$ synchornized sweep algoritm (**LetortEtAl12**)
- $O(n \log n)$ algorithm (Ouellet and Quimper, 2013)
- $O(n)$ algorithm (not pracical) and $O(n^2)$ efficient and simple algorithm (Gay, Hartert, and Schaus, 2015)

PySched – Available on GitHub

```python
outfile = open('tex/ex/timetabling.tex', 'w')

s = Schedule()

A = Task(s,duration=8,release=0,duedate=26,demand=1,label='A')
B = Task(s,duration=6,release=1,duedate=10,demand=4,label='B')
C = Task(s,duration=6,release=6,duedate=15,demand=4,label='C')
D = Task(s,duration=8,release=7,duedate=20,demand=3,label='D')
E = Task(s,duration=7,release=5,duedate=17,demand=2,label='E')
F = Task(s,duration=9,release=5,duedate=25,demand=4,label='F')

res = Resource(s, 'A', [A,B,C,D,E,F], capacity=7)

A << F

tt = Timetabling(res)

while True:
    s.save()
    if not tt.propagate():
        break

    s.latex(outfile, animated=True, precedences=False, pruning=True, offset=0, rows=[
    s.latex(outfile, animated=True, mandatory=True, profile=[res], precedences=False,
```

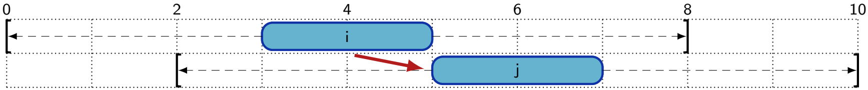
Disjunctive constraints

- Change the viewpoint (variables) from start times to precedences
- Notion of disjunctive graph (Roy and Sussman, 1964) central to (Carlier and Pinson, 1989)'s method
  - If *i* and *j* require the same exclusive resource



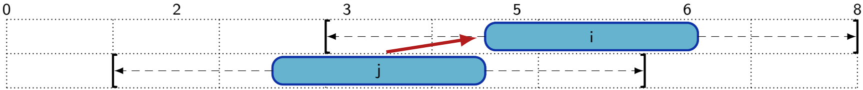
Disjunctive constraints

- Change the viewpoint (variables) from start times to precedences
- Notion of disjunctive graph (Roy and Sussman, 1964) central to (Carlier and Pinson, 1989)'s method
  - If i and j require the same exclusive resource, then either $i \prec j$



Disjunctive constraints

- Change the viewpoint (variables) from start times to precedences
- Notion of disjunctive graph (Roy and Sussman, 1964) central to (Carlier and Pinson, 1989)'s method
  - If i and j require the same exclusive resource, then either $i \prec j$ or $j \prec i$



Disjunctive decomposition (single-machine)

$$\forall i < j \in \mathcal{T}, \quad b_{ij} \iff e_i \leq s_j$$
$$b_{ij} \neq b_{ji}$$

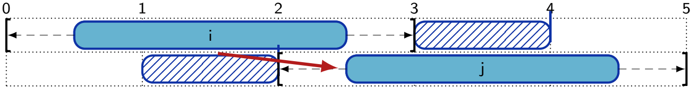
Disjunctive decomposition (single-machine)

$$\forall i < j \in \mathcal{T}, \quad b_{ij} \iff e_i \leq s_j$$

$$b_{ij} \neq b_{ji}$$

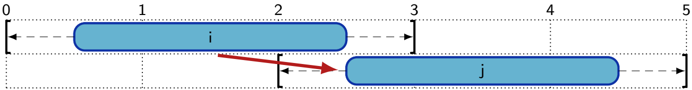
Disjunctive decomposition (single-machine)

$$\forall i < j \in \mathcal{T}, \quad b_{ij} \iff e_i \leq s_j$$
$$b_{ij} \neq b_{ji}$$



Disjunctive decomposition (single-machine)

$$\forall i < j \in \mathcal{T}, \quad b_{ij} \iff e_i \leq s_j$$
$$b_{ij} \neq b_{ji}$$



- Completeness of disjunctive propagation: deciding only b_{ij} variables is sufficient

Disjunctive algorithm

$$
\begin{aligned}
\forall i < j \in \mathcal{T}, \qquad [b_{ij}] &\implies i \prec j && \text{(post } i \prec j\text{'s propagator)}\\
\neg[b_{ij}] &\implies j \prec i && \text{(post } j \prec i\text{'s propagator)}\\
ect_j > lst_i &\implies [b_{ij}] &&\\
ect_i > lst_j &\implies \neg[b_{ij}] &&
\end{aligned}
$$

Disjunctive algorithm

$$\forall i < j \in \mathcal{T}, \quad [b_{ij}] \implies i \prec j \quad \text{(post } i \prec j\text{'s propagator)}$$

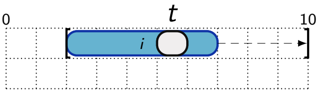
$$\neg[b_{ij}] \implies j \prec i \quad \text{(post } j \prec i\text{'s propagator)}$$

$$ect_j > lst_i \implies [b_{ij}]$$

$$ect_i > lst_j \implies \neg[b_{ij}]$$

Strictly stronger than Time-tabling

- Suppose $a_i^t = 0$ and $e_i > t$



Disjunctive algorithm

$$\forall i < j \in \mathcal{T}, \qquad [b_{ij}] \implies i \prec j \qquad \text{(post } i \prec j\text{'s propagator)}$$

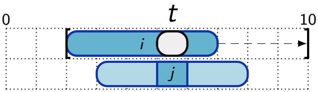
$$\neg[b_{ij}] \implies j \prec i \qquad \text{(post } j \prec i\text{'s propagator)}$$

$$ect_j > lst_i \implies [b_{ij}]$$

$$ect_i > lst_j \implies \neg[b_{ij}]$$

Strictly stronger than Time-tabling

- Suppose $a_i^t = 0$ and $e_i > t$
  - $\exists j$ s.t. $a_j^t = 1$

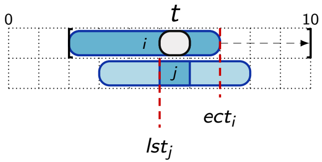


Disjunctive algorithm

$$\forall i < j \in \mathcal{T}, \quad [b_{ij}] \implies i \prec j \quad \text{(post } i \prec j\text{'s propagator)}$$
$$\neg[b_{ij}] \implies j \prec i \quad \text{(post } j \prec i\text{'s propagator)}$$
$$ect_j > lst_i \implies [b_{ij}]$$
$$ect_i > lst_j \implies \neg[b_{ij}]$$

Strictly stronger than Time-tabling

- Suppose $a_i^t = 0$ and $e_i > t$
  - $\exists j$ s.t. $a_j^t = 1$ hence $lst_j \leq t < ect_j$
  - Therefore, $lst_j < ect_i$ so b_{ij} must be false

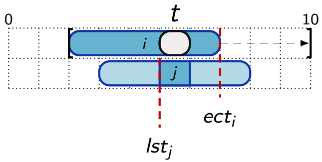


Disjunctive algorithm

$$\forall i < j \in \mathcal{T}, \quad [b_{ij}] \implies i \prec j \quad \text{(post } i \prec j\text{'s propagator)}$$
$$\neg[b_{ij}] \implies j \prec i \quad \text{(post } j \prec i\text{'s propagator)}$$
$$ect_j > lst_i \implies [b_{ij}]$$
$$ect_i > lst_j \implies \neg[b_{ij}]$$

Strictly stronger than Time-tabling

- Suppose $a_i^t = 0$ and $e_i > t$
  - $\exists j$ s.t. $a_j^t = 1$ hence $lst_j \leq t < ect_j$
  - Therefore, $lst_j < ect_i$ so b_{ij} must be false
  - and thus $j \prec i$ is posted



Generalisation to cumulative resources

- Either $i \prec j$, $j \prec i$ or $i \not\prec j \wedge j \not\prec i$
  - Easy change: $b_{ii} \neq b_{ji}$ becomes $\neg(b_{ij} \wedge b_{ji})$, but not complete anymore!

Generalisation to cumulative resources

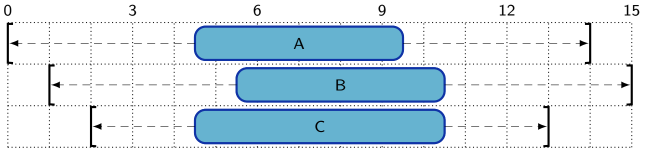
- Either $i \prec j$, $j \prec i$ or $i \nprec j \wedge j \nprec i$
  - Easy change: $b_{ii} \neq b_{ji}$ becomes $\neg(b_{ij} \wedge b_{ji})$, but not complete anymore!
- To keep the property that start times do not need to be set:
  - For every minimal subset S of tasks such that $\sum_{i \in S} c_i > C$, post:

$$\bigvee_{i \neq j \in S} b_{ij}$$

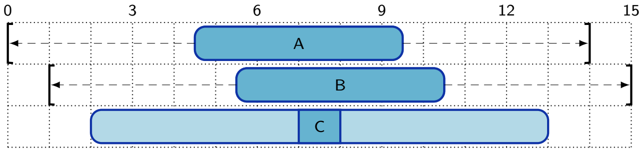
  - Not used in practice

Energetic reasoning

- A lot of different forms and flavours
  - Preemptive relaxation
  - Fully Elastic relaxation
  - Partialy Elastic relaxation
  - Edge finding
  - Energetic reasoning

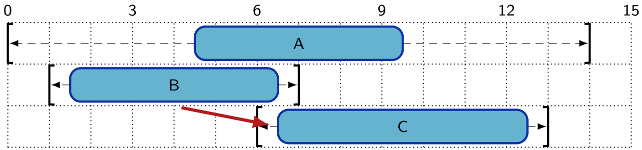


- A lot of different forms and flavours
  - Preemptive relaxation
  - Fully Elastic relaxation
  - Partialy Elastic relaxation
  - Edge finding
  - Energetic reasoning



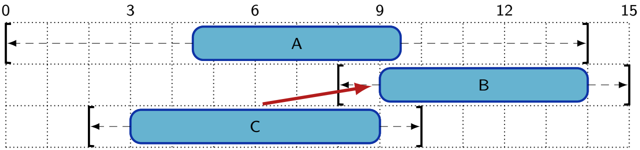
Energetic reasoning

- A lot of different forms and flavours
  - Preemptive relaxation
  - Fully Elastic relaxation
  - Partialy Elastic relaxation
  - Edge finding
  - Energetic reasoning

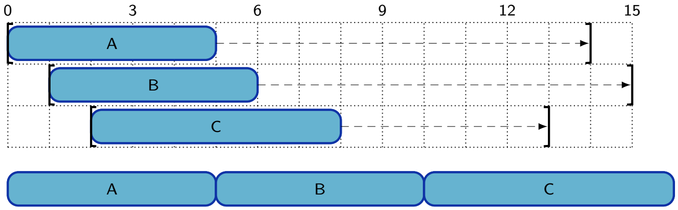


Energetic reasoning

- A lot of different forms and flavours
  - Preemptive relaxation
  - Fully Elastic relaxation
  - Partialy Elastic relaxation
  - Edge finding
  - Energetic reasoning

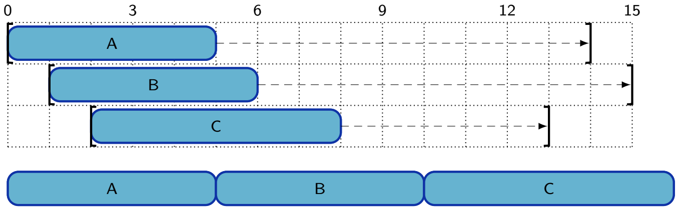


- A lot of different forms and flavours
  - Preemptive relaxation
  - Fully Elastic relaxation
  - Partialy Elastic relaxation
  - Edge finding
  - Energetic reasoning



Energetic reasoning

- A lot of different forms and flavours
  - Preemptive relaxation
  - Fully Elastic relaxation
  - Partialy Elastic relaxation
  - Edge finding
  - Energetic reasoning
- Basic idea: view a (set of) task(s) as fluid quantity (energy)



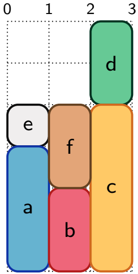
Preemptive relaxation (Federgruen and Groenevelt, 1986)

- Tasks can be interrupted (cut in vertical slices)

$$\begin{array}{rcl} \forall i \forall t \notin [r_i, d_i) & a_i^t = 0 & \text{bounds} \\ \forall t & \sum_i c_i a_i^t \leq C & \text{resource capacity} \\ \forall i & \sum_t a_i^t = p_i & \text{processing times} \end{array}$$

Properties of the preemptive relaxation

- NP-Hard, can encode BinPacking even with unit processing times
  - Capacity = number of bins
  - Consumption = item size
- Easy when $\forall i \; c_i = 1$: Maximum flow formulation
  - Gcc when $\forall i, \; p_i = 1$
  - AllDifferent when $\forall i, \; p_i = 1$ and $C = 1$



Fully elastic relaxation (Baptiste, Le Pape, and Nuijten, 1998)

- Tasks can be interrupted and resource usage is given by a total energy
- Replace Boolean "in process" a_i^t variables by integer "usage" u_i^t variables in $[0, c_i]$

$$
\begin{array}{rcl}
\forall i \forall t \notin [r_i, d_i) & u_i^t = 0 & \text{bounds} \\
\forall t & \sum_i u_i^t \leq C & \text{resource capacity} \\
\forall i & \sum_t u_i^t = p_i c_i & \text{energy}
\end{array}
$$

Properties of the fully elastic relaxation

- Polynomial
  - When $C = 1$, equivalent to preemptive relaxation and solved by Jackson Preemptive Schedule (Jackson, 1955) in $O(n \log n)$
  - When $C > 1$: equivalent reformulation to the $C = 1$ case
    - The horizon / time windows are multiplied by C
    - The processing time p_i is multiplied c_i

Partially elastic relaxation (Baptiste, Le Pape, and Nuijten, 1998)

- Strengthen the relaxation: constraint on energy of nested intervals

$\forall i \forall t \notin [r_i, d_i)$	$u_i^t = 0$	bounds
$\forall t$	$\sum_i u_i^t \leq C$	resource capacity
$\forall i$	$\sum_t u_i^t = p_i c_i$	energy
$\forall i \forall t \in [r_i, d_i)$	$\sum_{x<t} u_i^t \leq c_i(t - r_i)$	
$\forall i \forall t \in [r_i, d_i)$	$\sum_{x \geq t} u_i^t \leq c_i(d_i - t)$	

Properties of the partially elastic relaxation

- Equivalent to the SubsetSum bound (**Perregaard95**)
- Equivalent to the preemptive energetic reasoning (**LopezEtAl92**; Lopez, 1991)
- Algorithm in $O(n^2 \log n)$ (Baptiste, 1998)

Overload Checking decomposition

- The relaxation gives a satisfiability test but not a propagator
- Decomposition (Carlier, 1982):

$$\forall \Omega \subseteq \mathcal{T} \quad s_\Omega = \min(\{s_j \mid j \in \Omega\})$$
$$\forall \Omega \subseteq \mathcal{T} \quad e_\Omega = \max(\{e_j \mid j \in \Omega\})$$
$$\forall \Omega \subseteq \mathcal{T} \quad w_\Omega \leq C(e_\Omega - s_\Omega)$$

with $w_\Omega = \sum_{j \in \Omega} p_j c_j$

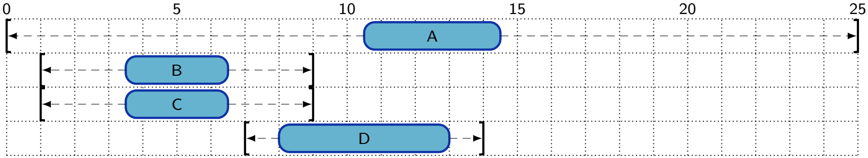
Theorem

Bound consistency on this decomposition fails
if and only if
the fully elastic decomposition is unsatisfiable

Edge Finding decomposition (single-machine)

- Channel to precedence variables (Carlier and Pinson, 1989), (Applegate and Cook, 1991)
  - adding i to Ω not last leads to overload $\implies$ i is last

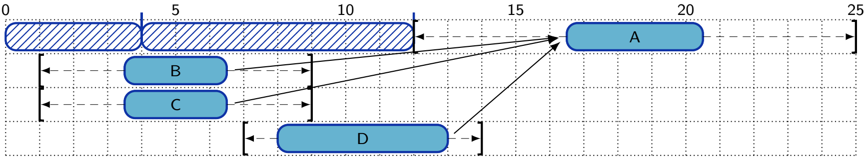
$$\forall \Omega \subseteq \mathcal{T} \quad s_\Omega = min(\{s_j \mid j \in \Omega\})$$
$$\forall \Omega \subseteq \mathcal{T} \quad e_\Omega = max(\{e_j \mid j \in \Omega\})$$
$$\forall \Omega \subseteq \mathcal{T} \quad s_\Omega + p_\Omega \leq e_\Omega$$
$$\forall \Omega \subseteq \mathcal{T},\ \forall i \notin \Omega \quad b_{\Omega i} \iff \bigwedge_{j \in \Omega} e_j \leq s_i$$
$$\forall \Omega \subseteq \mathcal{T},\ \forall i \notin \Omega \quad s_{\Omega \cup \{i\}} + p_i + p_\Omega > e_\Omega \implies b_{\Omega i}$$



Edge Finding decomposition (single-machine)

- Channel to precedence variables (Carlier and Pinson, 1989), (Applegate and Cook, 1991)
  - adding i to Ω not last leads to overload $\implies$ i is last

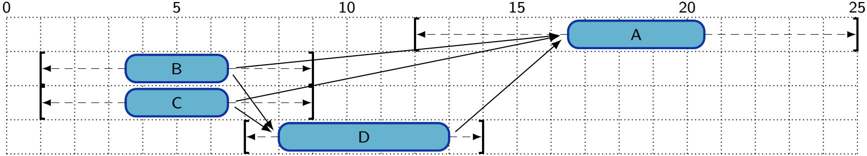
$$\forall \Omega \subseteq \mathcal{T} \qquad s_\Omega = min(\{s_j \mid j \in \Omega\})$$
$$\forall \Omega \subseteq \mathcal{T} \qquad e_\Omega = max(\{e_j \mid j \in \Omega\})$$
$$\forall \Omega \subseteq \mathcal{T} \qquad s_\Omega + p_\Omega \leq e_\Omega$$
$$\forall \Omega \subseteq \mathcal{T},\ \forall i \notin \Omega \qquad b_{\Omega i} \iff \bigwedge_{j \in \Omega} e_j \leq s_i$$
$$\forall \Omega \subseteq \mathcal{T},\ \forall i \notin \Omega \qquad s_{\Omega \cup \{i\}} + p_i + p_\Omega > e_\Omega \implies b_{\Omega i}$$



Edge Finding decomposition (single-machine)

- Channel to precedence variables (Carlier and Pinson, 1989), (Applegate and Cook, 1991)
  - adding i to Ω not last leads to overload $\implies$ i is last

$$
\begin{array}{lr}
\forall \Omega \subseteq \mathcal{T} & s_\Omega = min(\{s_j \mid j \in \Omega\}) \\
\forall \Omega \subseteq \mathcal{T} & e_\Omega = max(\{e_j \mid j \in \Omega\}) \\
\forall \Omega \subseteq \mathcal{T} & s_\Omega + p_\Omega \leq e_\Omega \\
\forall \Omega \subseteq \mathcal{T},\ \forall i \notin \Omega & b_{\Omega i} \iff \bigwedge_{j \in \Omega} e_j \leq s_i \\
\forall \Omega \subseteq \mathcal{T},\ \forall i \notin \Omega & s_{\Omega \cup \{i\}} + p_i + p_\Omega > e_\Omega \implies b_{\Omega i}
\end{array}
$$



Not first / Not last decomposition (single-machine)

- Reverse implication (Pinson, 1988)
  - adding i to Ω last leads to overload $\implies$ i is not last

$$\forall \Omega \subseteq \mathcal{T} \qquad s_\Omega = min(\{s_j \mid j \in \Omega\})$$

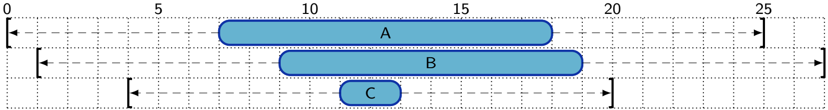
$$\forall \Omega \subseteq \mathcal{T} \qquad e_\Omega = max(\{e_j \mid j \in \Omega\})$$

$$\forall \Omega \subseteq \mathcal{T} \qquad s_\Omega + p_\Omega \leq e_\Omega$$

$$\forall \Omega \subseteq \mathcal{T},\ \forall i \notin \Omega \qquad b_{\Omega i} \iff \bigwedge_{j \in \Omega} e_j \leq s_i$$

$$\forall \Omega \subseteq \mathcal{T},\ \forall i \notin \Omega \qquad s_{\Omega \cup \{i\}} + p_i + p_\Omega > e_\Omega \implies b_{\Omega i}$$

$$\forall \Omega \subseteq \mathcal{T},\ \forall i \notin \Omega \qquad e_i - s_\Omega < p_\Omega + p_i \implies \neg b_{\Omega i}$$



Not first / Not last decomposition (single-machine)

- Reverse implication (Pinson, 1988)
  - adding i to Ω last leads to overload $\implies$ i is not last

$$\forall \Omega \subseteq \mathcal{T} \qquad s_\Omega = mjn(\{s_j \mid j \in \Omega\})$$

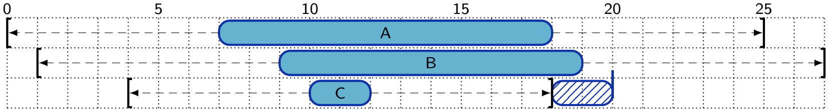
$$\forall \Omega \subseteq \mathcal{T} \qquad e_\Omega = max(\{e_j \mid j \in \Omega\})$$

$$\forall \Omega \subseteq \mathcal{T} \qquad s_\Omega + p_\Omega \leq e_\Omega$$

$$\forall \Omega \subseteq \mathcal{T},\ \forall i \notin \Omega \qquad b_{\Omega i} \iff \bigwedge_{j \in \Omega} e_j \leq s_i$$

$$\forall \Omega \subseteq \mathcal{T},\ \forall i \notin \Omega \qquad s_{\Omega \cup \{i\}} + p_i + p_\Omega > e_\Omega \implies b_{\Omega i}$$

$$\forall \Omega \subseteq \mathcal{T},\ \forall i \notin \Omega \qquad e_i - s_\Omega < p_\Omega + p_i \implies \neg b_{\Omega i}$$



Not first / Not last decomposition (single-machine)

- Reverse implication (Pinson, 1988)
  - adding i to Ω last leads to overload $\implies$ i is not last

$$\forall \Omega \subseteq \mathcal{T} \qquad s_\Omega = min(\{s_j \mid j \in \Omega\})$$

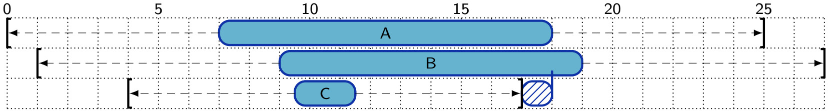
$$\forall \Omega \subseteq \mathcal{T} \qquad e_\Omega = max(\{e_j \mid j \in \Omega\})$$

$$\forall \Omega \subseteq \mathcal{T} \qquad s_\Omega + p_\Omega \leq e_\Omega$$

$$\forall \Omega \subseteq \mathcal{T},\ \forall i \notin \Omega \qquad b_{\Omega i} \iff \bigwedge_{j \in \Omega} e_j \leq s_i$$

$$\forall \Omega \subseteq \mathcal{T},\ \forall i \notin \Omega \qquad s_{\Omega \cup \{i\}} + p_i + p_\Omega > e_\Omega \implies b_{\Omega i}$$

$$\forall \Omega \subseteq \mathcal{T},\ \forall i \notin \Omega \qquad e_i - s_\Omega < p_\Omega + p_i \implies \neg b_{\Omega i}$$



Overload Checking algorithm (Vilím, Barták, and Čepek, 2004)

- Definitions

$$\begin{aligned} ect_\Omega &= \max\{r_{\Omega'} + p_{\Omega'} \mid \Omega' \subseteq \Omega\} \\ \mathcal{T}|_j &= \{i \in \mathcal{T} \mid d_i \leq d_j\} \end{aligned}$$

- Reformulation

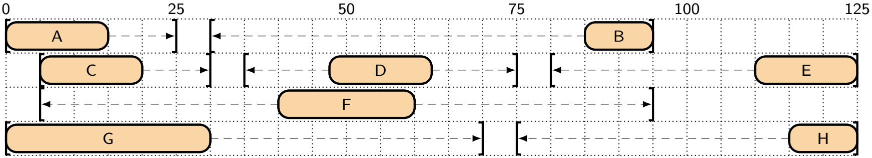
$$\forall \Omega \subseteq \mathcal{T}\ s_\Omega + p_\Omega \leq e_\Omega \iff \forall j \in \mathcal{T}\ ect_{\mathcal{T}|_j} \leq d_j$$

- May not seem like a big progress
  - Check $2^{|\mathcal{T}|}$ relations
  - Check $|\mathcal{T}|$ relations, each requiring to compute the max of $2^{|\mathcal{T}|}$ elements

Overload Checking – Dynamic Programming

$$ect_\Omega = \max\{ect_{\Omega_L} + p_{\Omega_R}, ect_{\Omega_R}\}$$

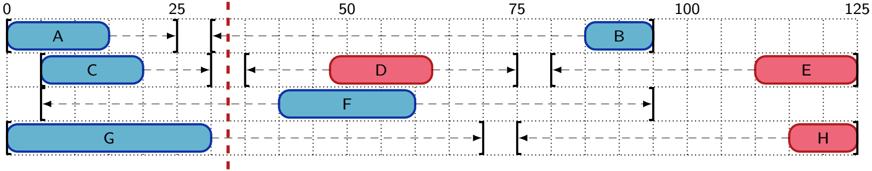
- With Ω_L, Ω_R two disjoint sets s.t. $\max\{r_i \mid i \in \Omega_L\} \leq \min\{r_i \mid i \in \Omega_R\}$



Overload Checking – Dynamic Programming

$$ect_\Omega = \max\{ect_{\Omega_L} + p_{\Omega_R}, ect_{\Omega_R}\}$$

- With Ω_L, Ω_R two disjoint sets s.t. $\max\{r_i \mid i \in \Omega_L\} \leq \min\{r_i \mid i \in \Omega_R\}$

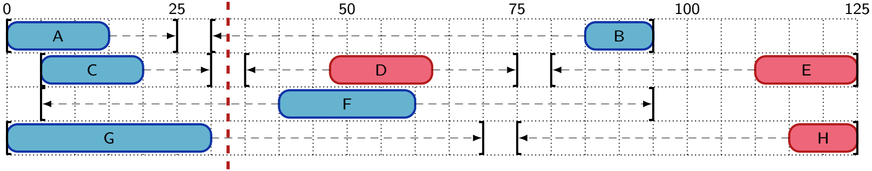


- Let $\Omega' \subseteq \Omega$ be such that $ect_\Omega = r_{\Omega'} + p_{\Omega'}$

Overload Checking – Dynamic Programming

$$ect_\Omega = \max\{ect_{\Omega_L} + p_{\Omega_R}, ect_{\Omega_R}\}$$

- With Ω_L, Ω_R two disjoint sets s.t. $\max\{r_i \mid i \in \Omega_L\} \leq \min\{r_i \mid i \in \Omega_R\}$

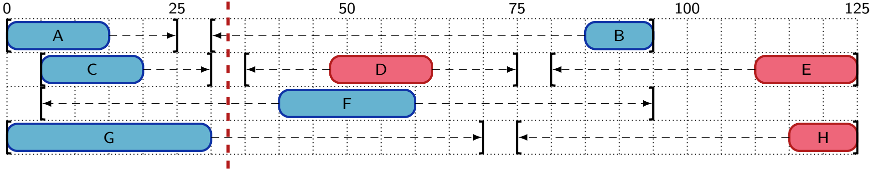


- Let $\Omega' \subseteq \Omega$ be such that $ect_\Omega = r_{\Omega'} + p_{\Omega'}$
  - Either $\Omega' \subseteq \Omega_R$ and therefore $ect_\Omega = ect_{\Omega'} = ect_{\Omega_R}$

Overload Checking – Dynamic Programming

$$ect_\Omega = \max\{ect_{\Omega_L} + p_{\Omega_R}, ect_{\Omega_R}\}$$

- With Ω_L, Ω_R two disjoint sets s.t. $\max\{r_i \mid i \in \Omega_L\} \leq \min\{r_i \mid i \in \Omega_R\}$

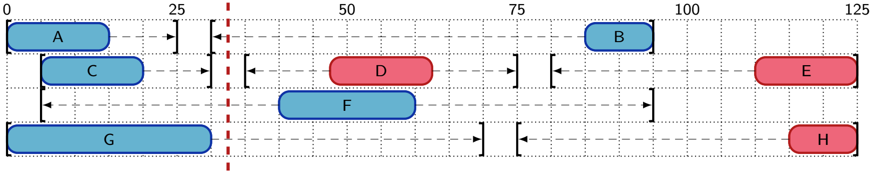


- Let $\Omega' \subseteq \Omega$ be such that $ect_\Omega = r_{\Omega'} + p_{\Omega'}$
  - Either $\Omega' \subseteq \Omega_R$ and therefore $ect_\Omega = ect_{\Omega'} = ect_{\Omega_R}$
  - Or $\Omega' \not\subseteq \Omega_R$ and $ect_\Omega = r_{\Omega'} + p_{\Omega'}$

Overload Checking – Dynamic Programming

$$ect_{\Omega} = \max\{ect_{\Omega_L} + p_{\Omega_R}, ect_{\Omega_R}\}$$

- With Ω_L, Ω_R two disjoint sets s.t. $\max\{r_i \mid i \in \Omega_L\} \leq \min\{r_i \mid i \in \Omega_R\}$

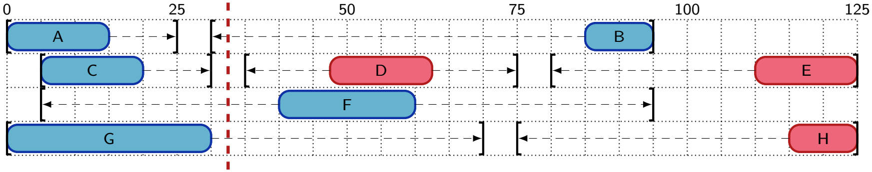


- Let $\Omega' \subseteq \Omega$ be such that $ect_{\Omega} = r_{\Omega'} + p_{\Omega'}$
  - Either $\Omega' \subseteq \Omega_R$ and therefore $ect_{\Omega} = ect_{\Omega'} = ect_{\Omega_R}$
  - Or $\Omega' \not\subseteq \Omega_R$ and $ect_{\Omega} = r_{\Omega' \cap \Omega_L} + p_{\Omega'}$

Overload Checking – Dynamic Programming

$$ect_\Omega = \max\{ect_{\Omega_L} + p_{\Omega_R}, ect_{\Omega_R}\}$$

- With Ω_L, Ω_R two disjoint sets s.t. $\max\{r_i \mid i \in \Omega_L\} \leq \min\{r_i \mid i \in \Omega_R\}$

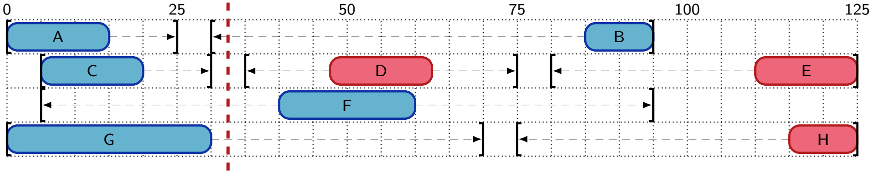


- Let $\Omega' \subseteq \Omega$ be such that $ect_\Omega = r_{\Omega'} + p_{\Omega'}$
  - Either $\Omega' \subseteq \Omega_R$ and therefore $ect_\Omega = ect_{\Omega'} = ect_{\Omega_R}$
  - Or $\Omega' \not\subseteq \Omega_R$ and $ect_\Omega = r_{\Omega' \cap \Omega_L} + p_{\Omega' \cap \Omega_L} + p_{\Omega' \cap \Omega_R}$

Overload Checking – Dynamic Programming

$$ect_\Omega = \max\{ect_{\Omega_L} + p_{\Omega_R}, ect_{\Omega_R}\}$$

- With Ω_L, Ω_R two disjoint sets s.t. $\max\{r_i \mid i \in \Omega_L\} \leq \min\{r_i \mid i \in \Omega_R\}$

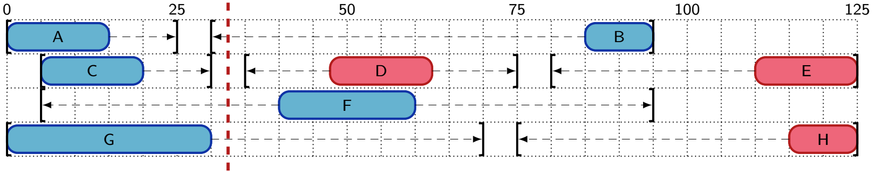


- Let $\Omega' \subseteq \Omega$ be such that $ect_\Omega = r_{\Omega'} + p_{\Omega'}$
  - Either $\Omega' \subseteq \Omega_R$ and therefore $ect_\Omega = ect_{\Omega'} = ect_{\Omega_R}$
  - Or $\Omega' \not\subseteq \Omega_R$ and $ect_\Omega = r_{\Omega' \cap \Omega_L} + p_{\Omega' \cap \Omega_L} + p_{\Omega_R}$

Overload Checking – Dynamic Programming

$$ect_{\Omega} = \max\{ect_{\Omega_L} + p_{\Omega_R}, ect_{\Omega_R}\}$$

- With Ω_L, Ω_R two disjoint sets s.t. $\max\{r_i \mid i \in \Omega_L\} \leq \min\{r_i \mid i \in \Omega_R\}$

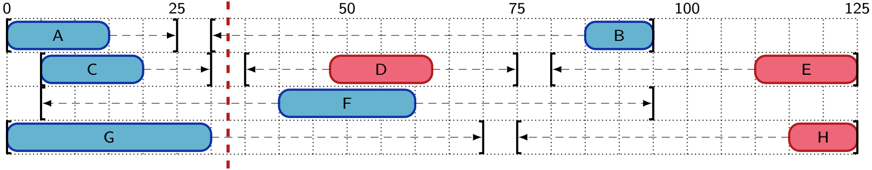


- Let $\Omega' \subseteq \Omega$ be such that $ect_{\Omega} = r_{\Omega'} + p_{\Omega'}$
  - Either $\Omega' \subseteq \Omega_R$ and therefore $ect_{\Omega} = ect_{\Omega'} = ect_{\Omega_R}$
  - Or $\Omega' \not\subseteq \Omega_R$ and $ect_{\Omega} = ect_{\Omega' \cap \Omega_L} + p_{\Omega_R}$

Overload Checking – Dynamic Programming

$$ect_{\Omega} = \max\{ect_{\Omega_L} + p_{\Omega_R}, ect_{\Omega_R}\}$$

- With Ω_L, Ω_R two disjoint sets s.t. $\max\{r_i \mid i \in \Omega_L\} \leq \min\{r_i \mid i \in \Omega_R\}$



- Let $\Omega' \subseteq \Omega$ be such that $ect_{\Omega} = r_{\Omega'} + p_{\Omega'}$
  - Either $\Omega' \subseteq \Omega_R$ and therefore $ect_{\Omega} = ect_{\Omega'} = ect_{\Omega_R}$
  - Or $\Omega' \not\subseteq \Omega_R$ and $ect_{\Omega} = ect_{\Omega_L} + p_{\Omega_R}$

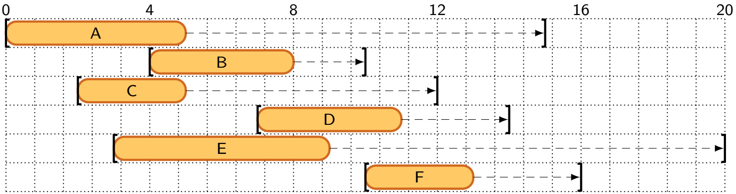
Overload Checking – Theta Tree

- Order the tasks by non-decreasing due date to compute $\mathcal{T}|_j$ for all $j \in \mathcal{T}$
- Order the tasks by non-decreasing release date to compute $ect_{\mathcal{T}|_j}$

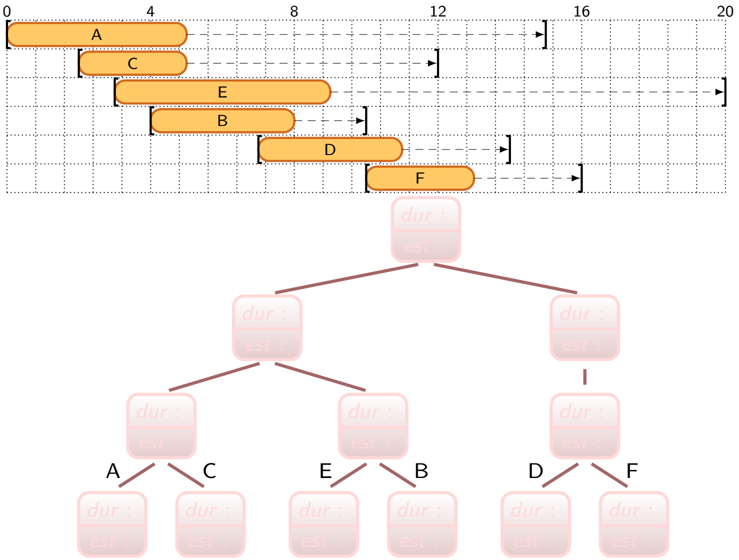
Solution

- Theta tree (Vilím, Barták, and Čepek, 2004)
  - Explore nested sets of tasks in any order (here non-decreasing due dates)
  - Incrementally compute a property (here $ect_{\mathcal{T}|_j}$) requiring another order

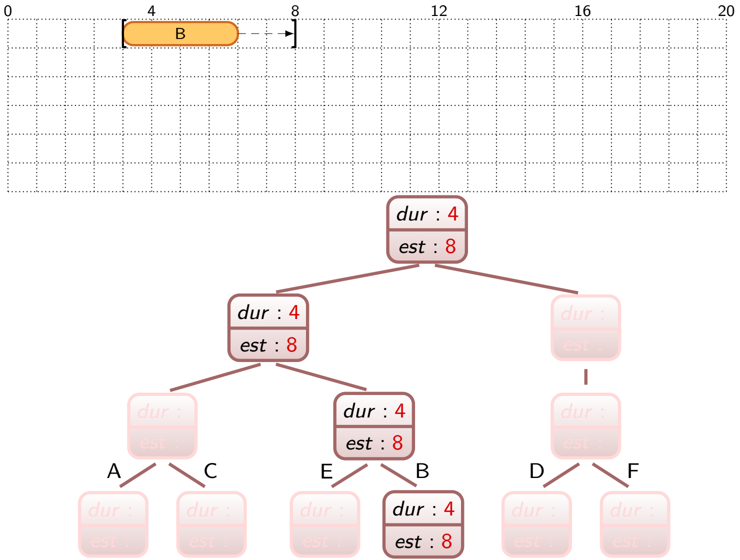
Theta Tree



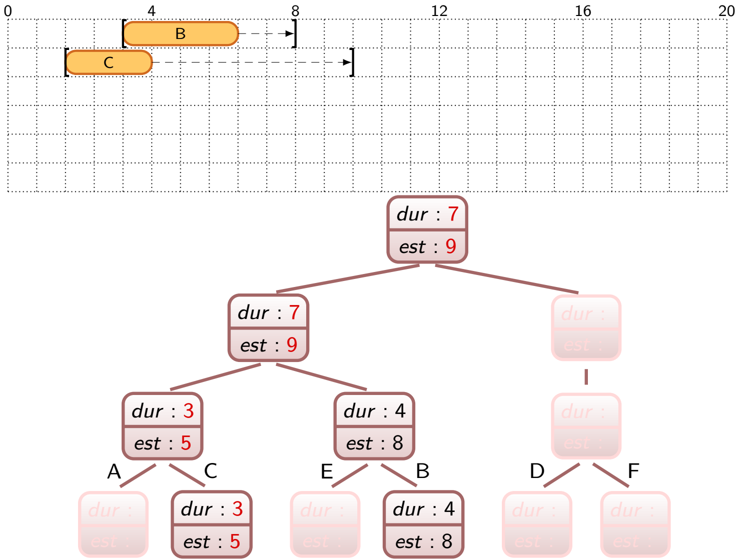
Theta Tree



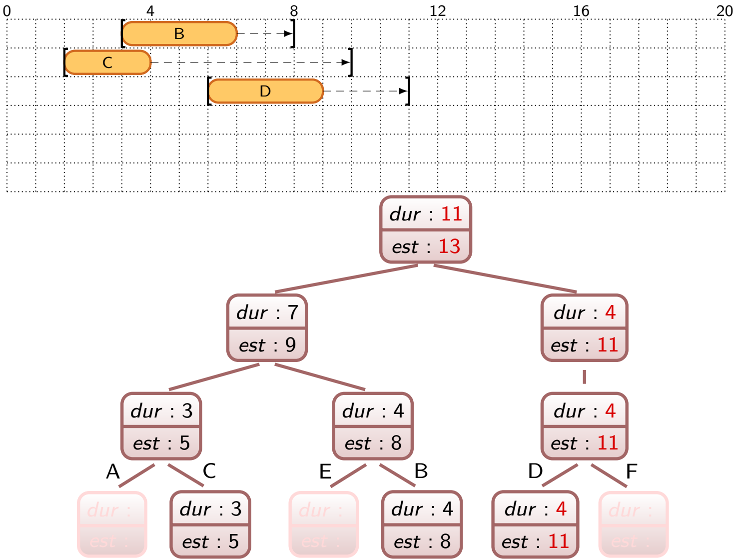
Theta Tree



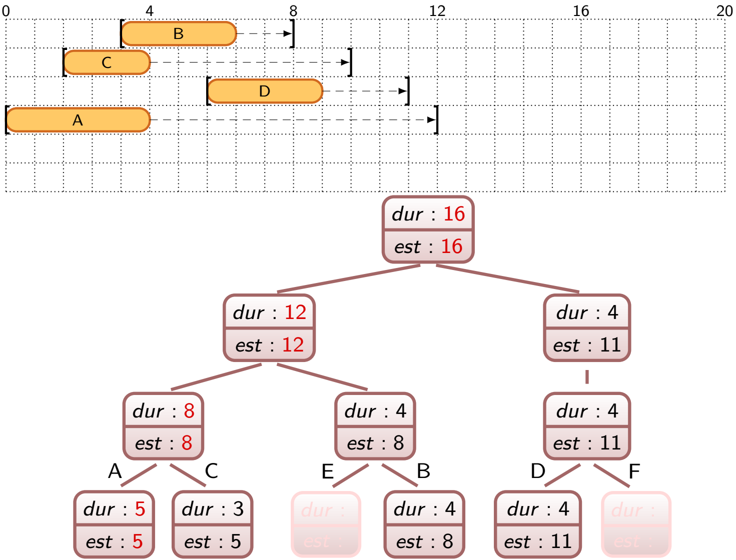
Theta Tree


0
4
8
12
16
20
B
C
dur : 7
est : 9
dur : 7
est : 9
dur : 3
est : 5
dur : 4
est : 8
A
C
E
B
D
F
dur : 3
est : 5
dur : 4
est : 8

Theta Tree

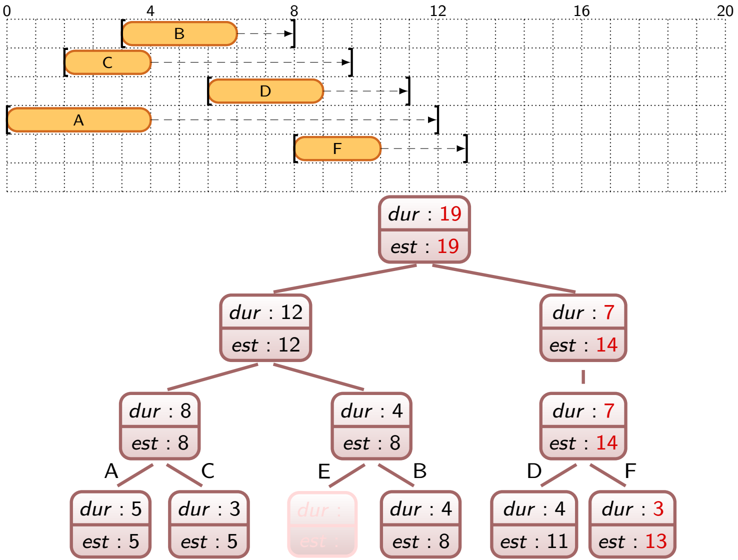

0
4
8
12
16
20
B
C
D
dur : 11
est : 13
dur : 7
est : 9
dur : 4
est : 11
dur : 3
est : 5
dur : 4
est : 8
dur : 4
est : 11
A
C
E
B
D
F
dur : 3
est : 5
dur : 4
est : 8
dur : 4
est : 11

Theta Tree



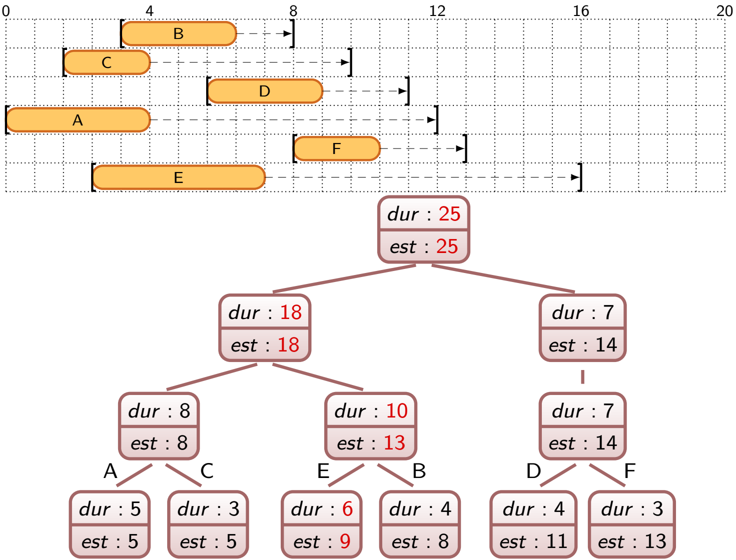
0
4
8
12
16
20
B
C
D
A
dur : 16
est : 16
dur : 12
est : 12
dur : 4
est : 11
dur : 8
est : 8
dur : 4
est : 8
dur : 4
est : 11
A
C
E
B
D
F
dur : 5
est : 5
dur : 3
est : 5
dur : 4
est : 8
dur : 4
est : 11

Theta Tree



0
4
8
12
16
20
B
C
D
A
F
dur : 19
est : 19
dur : 12
est : 12
dur : 7
est : 14
dur : 8
est : 8
dur : 4
est : 8
dur : 7
est : 14
A
C
E
B
D
F
dur : 5
est : 5
dur : 3
est : 5
dur :
est :
dur : 4
est : 8
dur : 4
est : 11
dur : 3
est : 13

Theta Tree


0
4
8
12
16
20
B
C
D
A
F
E
dur : 25
est : 25
dur : 18
est : 18
dur : 7
est : 14
dur : 8
est : 8
dur : 10
est : 13
dur : 7
est : 14
A
C
E
B
D
F
dur : 5
est : 5
dur : 3
est : 5
dur : 6
est : 9
dur : 4
est : 8
dur : 4
est : 11
dur : 3
est : 13

Edge finding

- $O(n \log n)$ algorithm based on Jackson Preemptive Schedule (Carlier and Pinson, 1994)

Edge finding

- $O(n \log n)$ algorithm based on Jackson Preemptive Schedule (Carlier and Pinson, 1994)
- $O(n^2)$ algorithm easier to implement / better in practice (Martin and Shmoys, 1996) and (Baptiste, Le Pape, and Nuijten, 2001)

Single-machine algorithms

Edge finding

- $O(n \log n)$ algorithm based on Jackson Preemptive Schedule (Carlier and Pinson, 1994)
- $O(n^2)$ algorithm easier to implement / better in practice (Martin and Shmoys, 1996) and (Baptiste, Le Pape, and Nuijten, 2001)
- Simpler algorithm in $O(n \log n)$ (Vilím, Barták, and Čepek, 2004)
  - Overload checking with theta tree, then mark the tasks as “maybe” one at a time

Single-machine algorithms

Edge finding

- $O(n \log n)$ algorithm based on Jackson Preemptive Schedule (Carlier and Pinson, 1994)
- $O(n^2)$ algorithm easier to implement / better in practice (Martin and Shmoys, 1996) and (Baptiste, Le Pape, and Nuijten, 2001)
- Simpler algorithm in $O(n \log n)$ (Vilím, Barták, and Čepek, 2004)
  - Overload checking with theta tree, then mark the tasks as “maybe” one at a time

Not first / Not last

- $O(n^2)$ algorithm (Le Pape and Baptiste, 1996)

Single-machine algorithms

Edge finding

- $O(n \log n)$ algorithm based on Jackson Preemptive Schedule (Carlier and Pinson, 1994)
- $O(n^2)$ algorithm easier to implement / better in practice (Martin and Shmoys, 1996) and (Baptiste, Le Pape, and Nuijten, 2001)
- Simpler algorithm in $O(n \log n)$ (Vilím, Barták, and Čepek, 2004)
  - Overload checking with theta tree, then mark the tasks as "maybe" one at a time

Not first / Not last

- $O(n^2)$ algorithm (Le Pape and Baptiste, 1996)
- Simpler algorithm in (Torres and Lopez, 2000)

Single-machine algorithms

Edge finding

- $O(n \log n)$ algorithm based on Jackson Preemptive Schedule (Carlier and Pinson, 1994)
- $O(n^2)$ algorithm easier to implement / better in practice (Martin and Shmoys, 1996) and (Baptiste, Le Pape, and Nuijten, 2001)
- Simpler algorithm in $O(n \log n)$ (Vilím, Barták, and Čepek, 2004)
  - Overload checking with theta tree, then mark the tasks as "maybe" one at a time

Not first / Not last

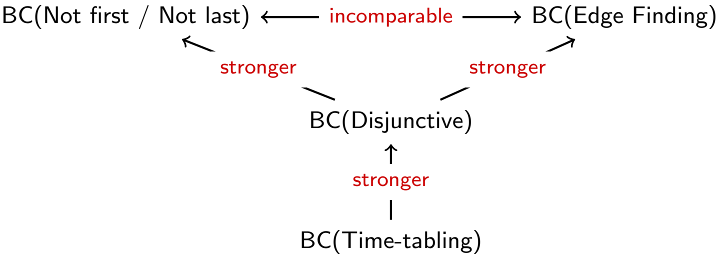
- $O(n^2)$ algorithm (Le Pape and Baptiste, 1996)
- Simpler algorithm in (Torres and Lopez, 2000)
- Theta-tree-based algorithm in $O(n \log n)$ (Vilím, 2004)

In the previous episode

- Time-tabling: usage profile $O(n)$ (Gay, Hartert, and Schaus, 2015)
- Disjunctive: either $i \prec j$ or $j \prec i$ $O(n^3)$ but incremental
- Edge finding, not-first, not-last: overload on Ω if we add i [not] firt/last $O(n \log n)$ (Vilím, Barták, and Čepek, 2004)

In the previous episode (single machine)

- Time-tabling: usage profile (good in the cumulative case) $O(n)$ (Gay, Hartert, and Schaus, 2015)
- Disjunctive: either $i \prec j$ or $j \prec i$ $O(n^3)$ but incremental
- Edge finding, not-first, not-last: overload on Ω if we add i [not] firt/last $O(n \log n)$ (Vilím, Barták, and Čepek, 2004)



Today

- Finish the review of resource propagation algorithms
  - Cumulative case
- Search
- Other types of resource

Cumulative Overload Checking

Notion of energy

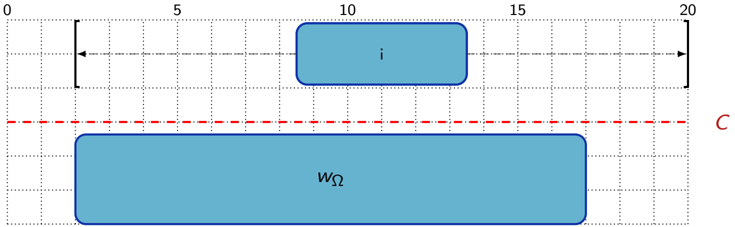
- $w_j = p_i c_i$
- $w_\Omega = \sum_{j \in \Omega} w_j$

- Overload checking is similar to the single-machine case

$$\forall \Omega \subseteq \mathcal{T} \quad w_\Omega \leq C(e_\Omega - s_\Omega)$$

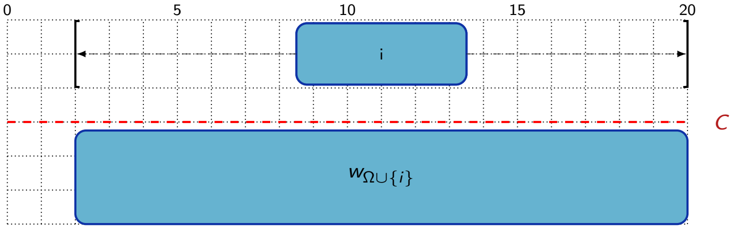
Cumulative Edge Finding (Nuijten and Aarts, 1994)

$$\forall \Omega \subseteq \mathcal{T} \qquad w_\Omega \leq C(e_\Omega - s_\Omega)$$



Cumulative Edge Finding (Nuijten and Aarts, 1994)

$$\forall \Omega \subseteq \mathcal{T} \qquad w_\Omega \leq C(e_\Omega - s_\Omega)$$

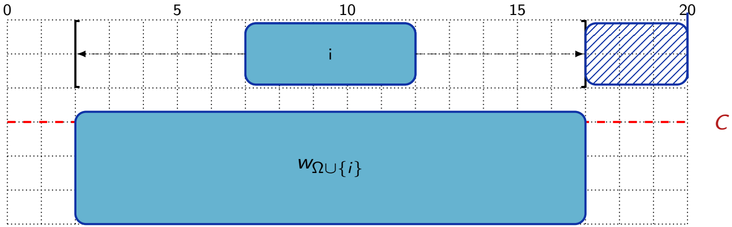


Cumulative Edge Finding (Nuijten and Aarts, 1994)

- If energy of Ω and i exceeds capacity when i is not last
  - then i has to be last

$$\forall \Omega \subseteq \mathcal{T} \qquad w_\Omega \leq C(e_\Omega - s_\Omega)$$

$$\forall \Omega \subseteq \mathcal{T},\ \forall i \notin \Omega \quad w_{\Omega \cup \{i\}} > C(e_\Omega - s_{\Omega \cup \{i\}}) \implies \Omega \prec i$$

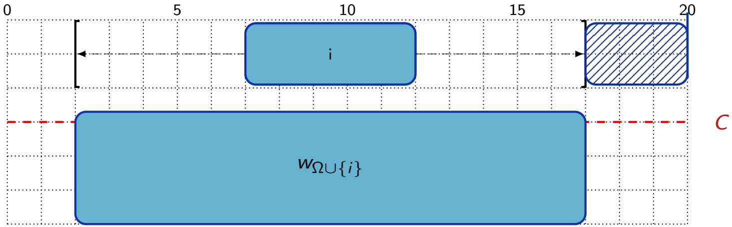


Cumulative Edge Finding (Nuijten and Aarts, 1994)

- If energy of Ω and i exceeds capacity when i is not last
  - then i has to be last but not necessarily follows every task in Ω!

$$\forall \Omega \subseteq \mathcal{T} \qquad w_\Omega \leq C(e_\Omega - s_\Omega)$$

$$\forall \Omega \subseteq \mathcal{T},\ \forall i \notin \Omega \quad w_{\Omega \cup \{i\}} > C(e_\Omega - s_{\Omega \cup \{i\}}) \not\Longrightarrow \Omega \prec i$$



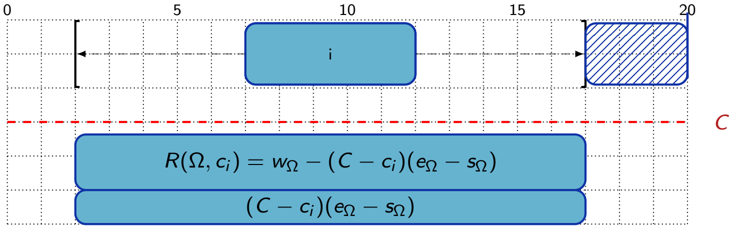
Cumulative Edge Finding (Nuijten and Aarts, 1994)

- If energy of Ω and i exceeds capacity when i is not last
  - then i has to be last but not necessarily follows every task in Ω!

$$\forall \Omega \subseteq \mathcal{T} \qquad w_\Omega \leq C(e_\Omega - s_\Omega)$$

$$\forall \Omega \subseteq \mathcal{T},\ \forall c \qquad R(\Omega, c) = w_\Omega - (C - c)(e_\Omega - s_\Omega)$$

$$\forall \Omega \subseteq \mathcal{T},\ \forall i \notin \Omega \qquad w_{\Omega \cup \{i\}} > C(e_\Omega - s_{\Omega \cup \{i\}}) \not\Longrightarrow \Omega \prec i$$



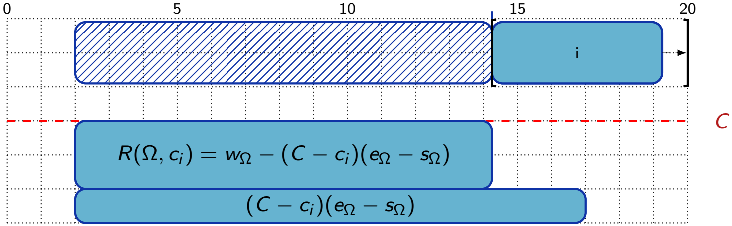
Cumulative Edge Finding (Nuijten and Aarts, 1994)

- If energy of Ω and i exceeds capacity when i is not last
  - then i has to be last but not necessarily follows every task in Ω!

$$\forall \Omega \subseteq \mathcal{T} \qquad w_\Omega \leq C(e_\Omega - s_\Omega)$$

$$\forall \Omega \subseteq \mathcal{T},\ \forall c \qquad R(\Omega, c) = w_\Omega - (C - c)(e_\Omega - s_\Omega)$$

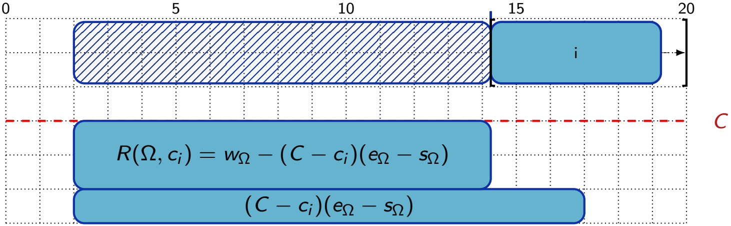
$$\forall \Omega \subseteq \mathcal{T},\ \forall i \notin \Omega,\ \forall \Omega' \in \Omega \qquad w_{\Omega \cup \{i\}} > C(e_\Omega - s_{\Omega \cup \{i\}}) \implies s_i \geq s_\Omega + \frac{1}{c_i} R(\Omega, c_i)$$



Cumulative Edge Finding (Nuijten and Aarts, 1994)

- If energy of Ω and i exceeds capacity when i is not last
  - then i has to be last but not necessarily follows every task in Ω!

$$\begin{aligned}
\forall \Omega \subseteq \mathcal{T} \quad & w_\Omega \leq C(e_\Omega - s_\Omega) \\
\forall \Omega \subseteq \mathcal{T},\ \forall c \quad & R(\Omega, c) = w_\Omega - (C - c)(e_\Omega - s_\Omega) \\
\forall \Omega \subseteq \mathcal{T},\ \forall i \notin \Omega,\ \forall \Omega' \in \Omega \quad & w_{\Omega \cup \{i\}} > C(e_\Omega - s_{\Omega \cup \{i\}}) \implies s_i \geq s_{\Omega'} + \frac{1}{c_i} R(\Omega', c_i)
\end{aligned}$$



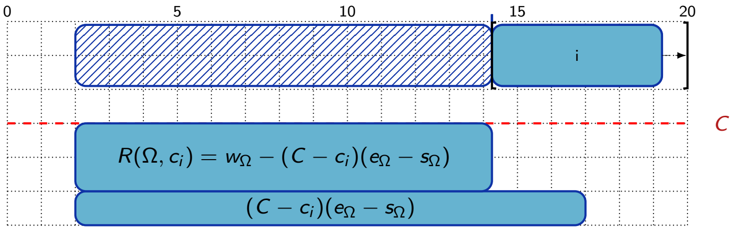
Cumulative Edge Finding (Nuijten and Aarts, 1994)

- If energy of Ω and i exceeds capacity when i is not last
  - then i has to be last but not necessarily follows every task in Ω!
- $O(n^2 k)$ (Nuijten and Aarts, 1994), $O(n^2)$ (Baptiste, Le Pape, and Nuijten, 2001), $O(kn \log n)$ (Vilím, 2009), $O(n^2)$ (Kameugne et al., 2011)

$$\forall \Omega \subseteq \mathcal{T} \quad w_\Omega \leq C(e_\Omega - s_\Omega)$$

$$\forall \Omega \subseteq \mathcal{T},\ \forall c \quad R(\Omega, c) = w_\Omega - (C - c)(e_\Omega - s_\Omega)$$

$$\forall \Omega \subseteq \mathcal{T},\ \forall i \notin \Omega,\ \forall \Omega' \in \Omega \quad w_{\Omega \cup \{i\}} > C(e_\Omega - s_{\Omega \cup \{i\}}) \implies s_i \geq s_{\Omega'} + \frac{1}{c_i} R(\Omega', c_i)$$

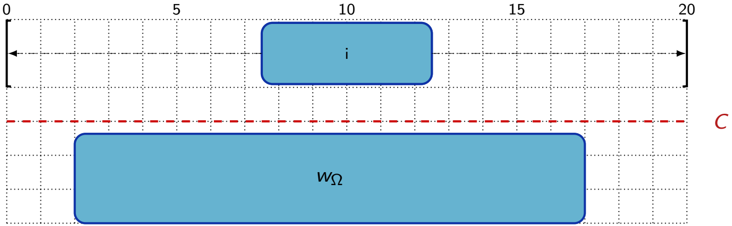


C. Extended Edge Finding (Nuijten and Aarts, 1994)

$$\forall \Omega \subseteq \mathcal{T} \qquad w_\Omega \leq C(e_\Omega - s_\Omega)$$

$$\forall \Omega \subseteq \mathcal{T},\ \forall c \qquad R(\Omega, c) = w_\Omega - (C - c)(e_\Omega - s_\Omega)$$

$$\forall \Omega \subseteq \mathcal{T},\ \forall i \notin \Omega,\ \forall \Omega' \in \Omega \qquad w_{\Omega \cup \{i\}} > C(e_\Omega - s_{\Omega \cup \{i\}}) \implies s_i \geq s_{\Omega'} + \frac{1}{c_i} R(\Omega', c_i)$$



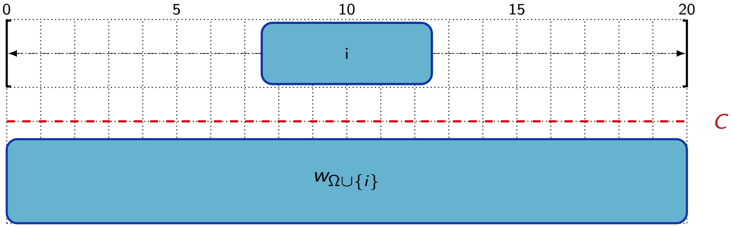
C. Extended Edge Finding (Nuijten and Aarts, 1994)

- Case where i may start before Ω but cannot be completed before Ω starts
  - Stronger constraint if we consider the overlap: $(e_i - s_\Omega)c_i$

$$\forall \Omega \subseteq \mathcal{T} \qquad w_\Omega \leq C(e_\Omega - s_\Omega)$$

$$\forall \Omega \subseteq \mathcal{T},\ \forall c \qquad R(\Omega, c) = w_\Omega - (C - c)(e_\Omega - s_\Omega)$$

$$\forall \Omega \subseteq \mathcal{T},\ \forall i \notin \Omega,\ \forall \Omega' \in \Omega \qquad w_{\Omega \cup \{i\}} > C(e_\Omega - s_{\Omega \cup \{i\}}) \implies s_i \geq s_{\Omega'} + \frac{1}{c_i} R(\Omega', c_i)$$



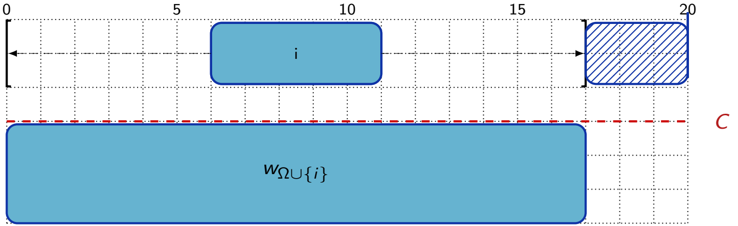
C. Extended Edge Finding (Nuijten and Aarts, 1994)

- Case where i may start before Ω but cannot be completed before Ω starts
  - Stronger constraint if we consider the overlap: $(e_i - s_\Omega)c_i$

$$\forall \Omega \subseteq \mathcal{T} \qquad w_\Omega \leq C(e_\Omega - s_\Omega)$$

$$\forall \Omega \subseteq \mathcal{T},\ \forall c \qquad R(\Omega, c) = w_\Omega - (C - c)(e_\Omega - s_\Omega)$$

$$\forall \Omega \subseteq \mathcal{T},\ \forall i \notin \Omega,\ \forall \Omega' \in \Omega \qquad w_{\Omega \cup \{i\}} > C(e_\Omega - s_{\Omega \cup \{i\}}) \implies s_i \geq s_{\Omega'} + \frac{1}{c_i} R(\Omega', c_i)$$



C. Extended Edge Finding (Nuijten and Aarts, 1994)

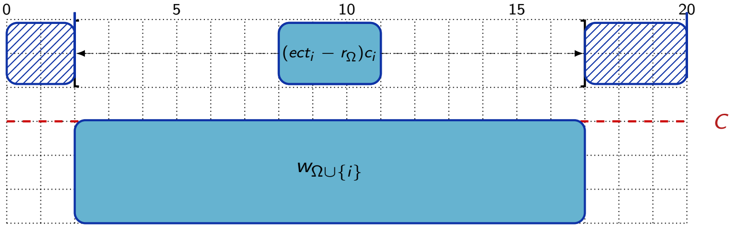
- Case where i may start before Ω but cannot be completed before Ω starts
  - Stronger constraint if we consider the overlap: $(e_i - s_\Omega)c_i$

$$\forall \Omega \subseteq \mathcal{T} \quad w_\Omega \leq C(e_\Omega - s_\Omega)$$

$$\forall \Omega \subseteq \mathcal{T},\ \forall c \quad R(\Omega, c) = w_\Omega - (C - c)(e_\Omega - s_\Omega)$$

$$\forall \Omega \subseteq \mathcal{T},\ \forall i \notin \Omega,\ \forall \Omega' \in \Omega \quad w_{\Omega \cup \{i\}} > C(e_\Omega - s_{\Omega \cup \{i\}}) \implies s_i \geq s_{\Omega'} + \frac{1}{c_i} R(\Omega', c_i)$$

$$\forall \Omega \subseteq \mathcal{T},\ \forall i \notin \Omega,\ \forall \Omega' \in \Omega \quad w_\Omega + (e_i - s_\Omega)c_i > C(e_\Omega - s_\Omega) \implies s_i \geq s_{\Omega'} + \frac{1}{c_i} R(\Omega', c_i)$$



C. Extended Edge Finding (Nuijten and Aarts, 1994)

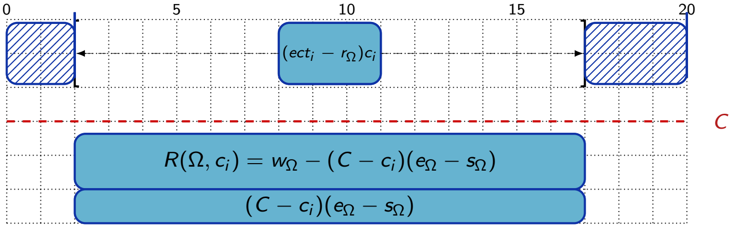
- Case where i may start before Ω but cannot be completed before Ω starts
  - Stronger constraint if we consider the overlap: $(e_i - s_\Omega)c_i$

$$\forall \Omega \subseteq \mathcal{T} \qquad w_\Omega \leq C(e_\Omega - s_\Omega)$$

$$\forall \Omega \subseteq \mathcal{T},\ \forall c \qquad R(\Omega, c) = w_\Omega - (C - c)(e_\Omega - s_\Omega)$$

$$\forall \Omega \subseteq \mathcal{T},\ \forall i \notin \Omega,\ \forall \Omega' \in \Omega \qquad w_{\Omega \cup \{i\}} > C(e_\Omega - s_{\Omega \cup \{i\}}) \implies s_i \geq s_{\Omega'} + \frac{1}{c_i} R(\Omega', c_i)$$

$$\forall \Omega \subseteq \mathcal{T},\ \forall i \notin \Omega,\ \forall \Omega' \in \Omega \qquad w_\Omega + (e_i - s_\Omega)c_i > C(e_\Omega - s_\Omega) \implies s_i \geq s_{\Omega'} + \frac{1}{c_i} R(\Omega', c_i)$$



C. Extended Edge Finding (Nuijten and Aarts, 1994)

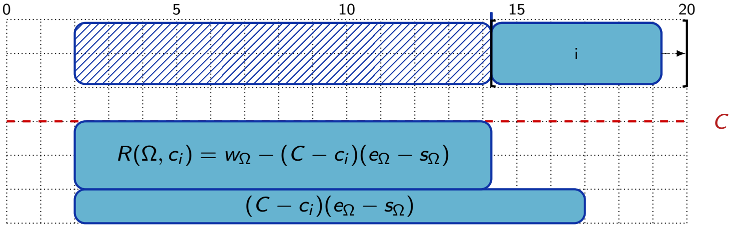
- Case where i may start before Ω but cannot be completed before Ω starts
  - Stronger constraint if we consider the overlap: $(e_i - s_\Omega)c_i$

$$\forall \Omega \subseteq \mathcal{T} \quad w_\Omega \leq C(e_\Omega - s_\Omega)$$

$$\forall \Omega \subseteq \mathcal{T},\ \forall c \quad R(\Omega, c) = w_\Omega - (C - c)(e_\Omega - s_\Omega)$$

$$\forall \Omega \subseteq \mathcal{T},\ \forall i \notin \Omega,\ \forall \Omega' \in \Omega \quad w_{\Omega \cup \{i\}} > C(e_\Omega - s_{\Omega \cup \{i\}}) \implies s_i \geq s_{\Omega'} + \frac{1}{c_i} R(\Omega', c_i)$$

$$\forall \Omega \subseteq \mathcal{T},\ \forall i \notin \Omega,\ \forall \Omega' \in \Omega \quad w_\Omega + (e_i - s_\Omega)c_i > C(e_\Omega - s_\Omega) \implies s_i \geq s_{\Omega'} + \frac{1}{c_i} R(\Omega', c_i)$$



C. Extended Edge Finding (Nuijten and Aarts, 1994)

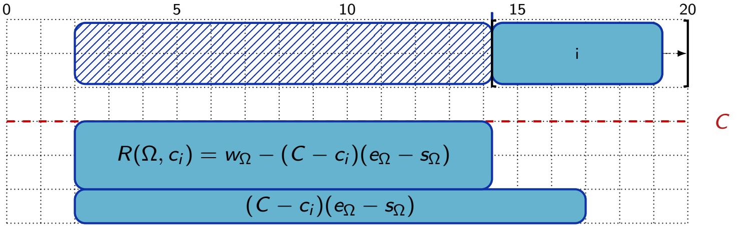
- Case where i may start before Ω but cannot be completed before Ω starts
  - Stronger constraint if we consider the overlap: $(e_i - s_\Omega)c_i$
- $O(n^3)$ (Nuijten and Aarts, 1994), $O(n^2)$ (Mercier and Hentenryck, 2008), $O(kn \log n)$ (Ouellet and Quimper, 2013)

$$\forall \Omega \subseteq \mathcal{T} \qquad w_\Omega \leq C(e_\Omega - s_\Omega)$$

$$\forall \Omega \subseteq \mathcal{T},\ \forall c \qquad R(\Omega, c) = w_\Omega - (C - c)(e_\Omega - s_\Omega)$$

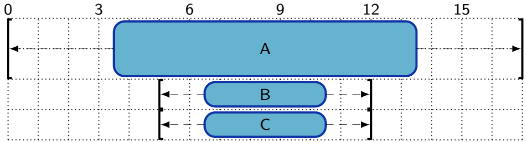
$$\forall \Omega \subseteq \mathcal{T},\ \forall i \notin \Omega,\ \forall \Omega' \in \Omega \qquad w_{\Omega \cup \{i\}} > C(e_\Omega - s_{\Omega \cup \{i\}}) \implies s_i \geq s_{\Omega'} + \frac{1}{c_i} R(\Omega', c_i)$$

$$\forall \Omega \subseteq \mathcal{T},\ \forall i \notin \Omega,\ \forall \Omega' \in \Omega \qquad w_\Omega + (e_i - s_\Omega)c_i > C(e_\Omega - s_\Omega) \implies s_i \geq s_{\Omega'} + \frac{1}{c_i} R(\Omega', c_i)$$



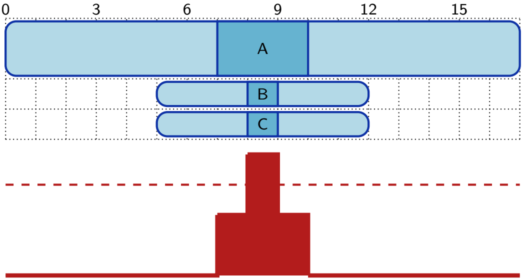
Time-tabling Edge Finding (Vilím, 2011)

- Time-tabling and (Extended) Edge finding are incomparable when $C > 1$



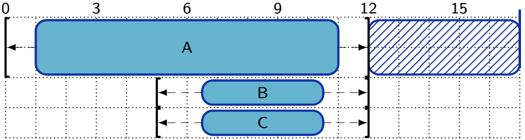
Time-tabling Edge Finding (Vilím, 2011)

- Time-tabling and (Extended) Edge finding are incomparable when $C > 1$



Time-tabling Edge Finding (Vilím, 2011)

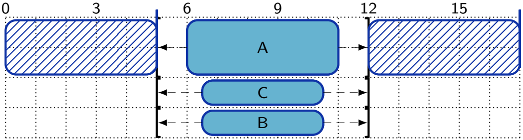
- Time-tabling and (Extended) Edge finding are incomparable when $C > 1$



$$20 + 4 + 4 \leq 3 \times 12 = 36$$

Time-tabling Edge Finding (Vilím, 2011)

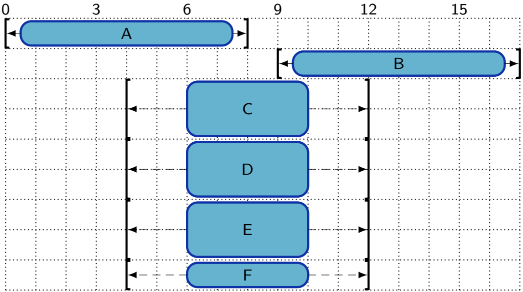
- Time-tabling and (Extended) Edge finding are incomparable when $C > 1$



$$10 + 4 + 4 \leq 3 \times 7 = 21$$

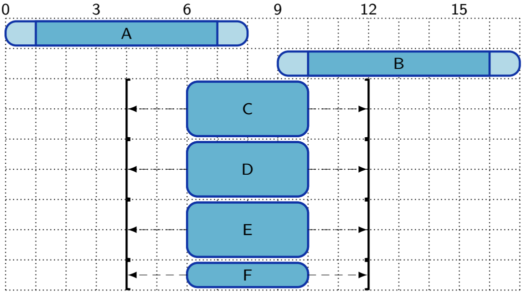
Time-tabling Edge Finding (Vilím, 2011)

- Time-tabling and (Extended) Edge finding are incomparable when $C > 1$
  - Combine them! $3 \times 8 + 4 + 5 > 4 \times 8$



Time-tabling Edge Finding (Vilím, 2011)

- Time-tabling and (Extended) Edge finding are incomparable when $C > 1$
  - Combine them! $3 \times 8 + 4 + 5 > 4 \times 8$



Time-tabling Edge Finding decomposition

- Channel Time-tabling and (Extended) Edge finding decompositions

$$
\begin{array}{rc}
\forall \Omega \subseteq \mathcal{T} & w_\Omega \leq C(e_\Omega - s_\Omega) \\
\forall \Omega \subseteq \mathcal{T},\ \forall c & R(\Omega, c) = w_\Omega - (C - c)(e_\Omega - s_\Omega) \\
\forall \Omega \subseteq \mathcal{T},\ \forall i \notin \Omega,\ \forall \Omega' \in \Omega & w_{\Omega \cup \{i\}} > C(e_\Omega - s_{\Omega \cup \{i\}}) \implies s_i \geq s_{\Omega'} + \frac{1}{c_i} R(\Omega', c_i) \\
\forall \Omega \subseteq \mathcal{T},\ \forall i \notin \Omega,\ \forall \Omega' \in \Omega & w_\Omega + (e_i - s_\Omega) c_i > C(e_\Omega - s_\Omega) \implies s_i \geq s_{\Omega'} + \frac{1}{c_i} R(\Omega', c_i)
\end{array}
$$

Time-tabling Edge Finding decomposition

- Channel Time-tabling and (Extended) Edge finding decompositions

$$
\begin{array}{rl}
\forall \Omega \subseteq \mathcal{T} & w_\Omega \leq C(e_\Omega - s_\Omega) - \sum_{t=s_\Omega}^{e_\Omega} \sum_{i \notin \Omega} c_i a_i^t \\
\forall \Omega \subseteq \mathcal{T},\ \forall c & R(\Omega, c) = w_\Omega - (C - c)(e_\Omega - s_\Omega) \\
\forall \Omega \subseteq \mathcal{T},\ \forall i \notin \Omega,\ \forall \Omega' \in \Omega & w_{\Omega \cup \{i\}} > C(e_\Omega - s_{\Omega \cup \{i\}}) \implies s_i \geq s_{\Omega'} + \frac{1}{c_i} R(\Omega', c_i) \\
\forall \Omega \subseteq \mathcal{T},\ \forall i \notin \Omega,\ \forall \Omega' \in \Omega & w_\Omega + (e_i - s_\Omega) c_i > C(e_\Omega - s_\Omega) \implies s_i \geq s_{\Omega'} + \frac{1}{c_i} R(\Omega', c_i) \\
\forall i \forall t & a_i^t \iff s_i \leq t \wedge t < e_i
\end{array}
$$

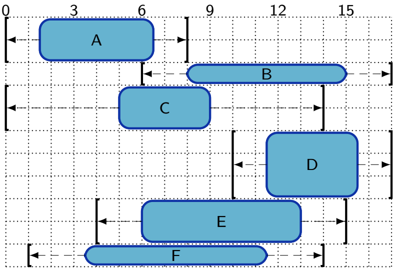
Time-tabling Edge Finding decomposition

- Channel Time-tabling and (Extended) Edge finding decompositions
  - Algorithm in $O(n^2)$ for Edge finding + Time-tabling (Vilím, 2011)
  - Algorithm in $O(kn\log n)$ for Edge finding + Extended Edge finding + Time-tabling (Ouellet and Quimper, 2013)

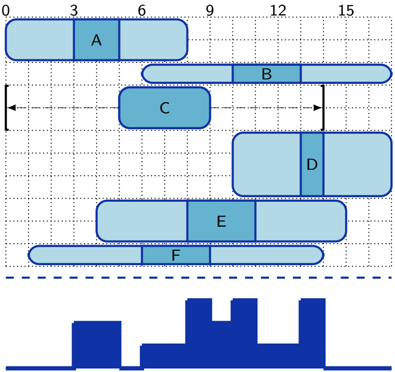
$$\begin{aligned}
\forall \Omega \subseteq \mathcal{T} &\quad w_\Omega \leq C(e_\Omega - s_\Omega) - \textstyle\sum_{t=s_\Omega}^{e_\Omega} \sum_{i \notin \Omega} c_i a_i^t \\
\forall \Omega \subseteq \mathcal{T},\ \forall c &\quad R(\Omega, c) = w_\Omega - (C - c)(e_\Omega - s_\Omega) \\
\forall \Omega \subseteq \mathcal{T},\ \forall i \notin \Omega,\ \forall \Omega' \in \Omega &\quad w_{\Omega \cup \{i\}} > C(e_\Omega - s_{\Omega \cup \{i\}}) \implies s_i \geq s_{\Omega'} + \tfrac{1}{c_i} R(\Omega', c_i) \\
\forall \Omega \subseteq \mathcal{T},\ \forall i \notin \Omega,\ \forall \Omega' \in \Omega &\quad w_\Omega + (e_i - s_\Omega)c_i > C(e_\Omega - s_\Omega) \implies s_i \geq s_{\Omega'} + \tfrac{1}{c_i} R(\Omega', c_i) \\
\forall i \forall t &\quad a_i^t \iff s_i \leq t \wedge t < e_i
\end{aligned}$$

Time-tabling Edge Finding algorithm

Algorithm of (Ouellet and Quimper, 2013)



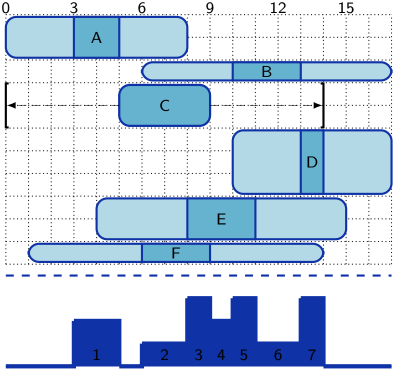
Algorithm of (Ouellet and Quimper, 2013)

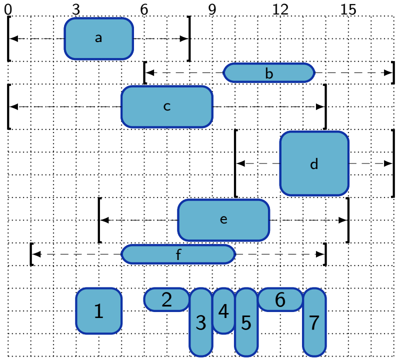


Time-tabling Edge Finding algorithm

Algorithm of (Ouellet and Quimper, 2013)

- A fixed task for each interval of the profile, remove the compulsory part from the task
- Apply any extended edge finding algorithm

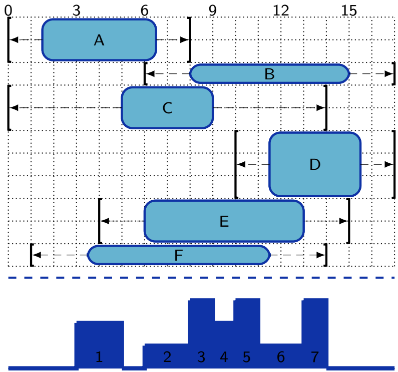


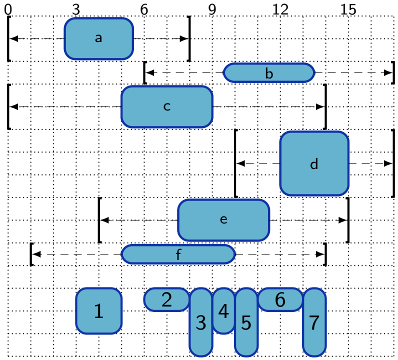


Time-tabling Edge Finding algorithm

Algorithm of (Ouellet and Quimper, 2013)

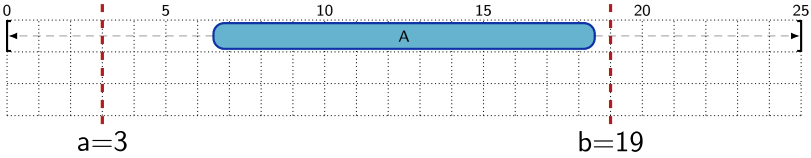
- A fixed task for each interval of the profile, remove the compulsory part from the task
- Apply any extended edge finding algorithm





Energetic reasoning (Lopez, 1991)

- Energy of a set of tasks $\Longrightarrow$ Energy of all tasks over a given interval

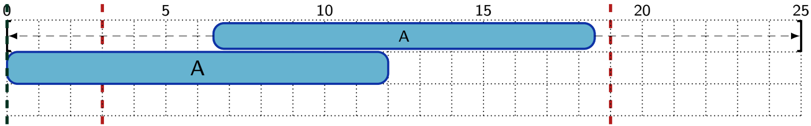


Notion of energy (over an interval)

- $p_j^+(a) = \max(0, p_j - \max(0, a - s_j))$
- $p_j^-(b) = \max(0, p_j - \max(0, e_j - b))$
- $w_j(a, b) = c_j \min(b - a, p_j^+(a), p_j^-(b))$
- $W(a, b) = \sum_{j \in \mathcal{T}} w_j(a, b)$

Energetic reasoning (Lopez, 1991)

- Energy of a set of tasks $\Longrightarrow$ Energy of all tasks over a given interval



$r_A = 0$ a=3 b=19

$\longleftarrow 3 \longrightarrow$

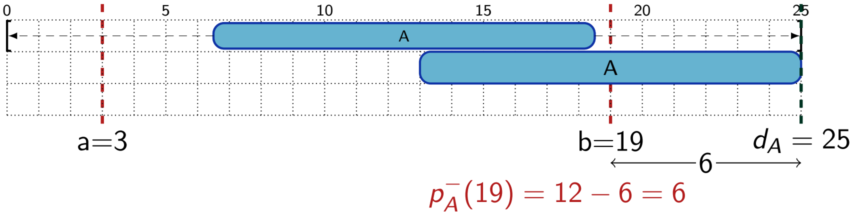
$p_A^+(3) = 12 - 3 = 9$

Notion of energy (over an interval)

- $p_j^+(a) = \max(0, p_j - \max(0, a - s_j))$
- $p_j^-(b) = \max(0, p_j - \max(0, e_j - b))$
- $w_j(a,b) = c_j \min(b - a, p_j^+(a), p_j^-(b))$
- $W(a,b) = \sum_{j \in \mathcal{T}} w_j(a,b)$

Energetic reasoning (Lopez, 1991)

- Energy of a set of tasks $\Longrightarrow$ Energy of all tasks over a given interval

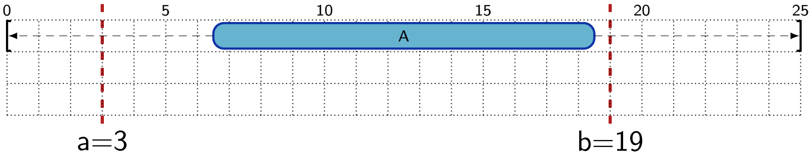


Notion of energy (over an interval)

- $p_j^+(a) = \max(0, p_j - \max(0, a - s_j))$
- $p_j^-(b) = \max(0, p_j - \max(0, e_j - b))$
- $w_j(a,b) = c_j \min(b - a, p_j^+(a), p_j^-(b))$
- $W(a,b) = \sum_{j \in \mathcal{T}} w_j(a,b)$

Energetic reasoning (Lopez, 1991)

- Energy of a set of tasks $\Longrightarrow$ Energy of all tasks over a given interval



Notion of energy (over an interval)

- $p_j^+(a) = \max(0, p_j - \max(0, a - s_j))$
- $p_j^-(b) = \max(0, p_j - \max(0, e_j - b))$
- $w_j(a, b) = c_j \min(b - a, p_j^+(a), p_j^-(b))$
- $W(a, b) = \sum_{j \in \mathcal{T}} w_j(a, b)$

Energetic reasoning decomposition

$$\forall 0 \leq a < b < h \quad W(a, b) \leq C(b - a) \quad \text{(ER)}$$

Energetic reasoning decomposition

$$\forall 0 \leq a < b < h \quad W(a,b) \leq C(b-a) \quad \text{(ER)}$$

Relevant intervals (Baptiste, 1998)

$$\begin{aligned} O_1(i) &= \{r_i, lst_i, ect_i\} \\ O_2(i) &= \{d_i, lst_i, ect_i\} \\ O(i,t) &= \{lst_i - t \mid i \in \mathcal{T}\} \end{aligned}$$

- (ER) holds iff it holds for every $i \neq j$ and every $a < b$ s.t.
  - $a, b \in O_1(i) \times O_2(j)$
  - $a, b \in O_1(i) \times O(j, a)$
  - $a, b \in O_2(i) \times O(j, b)$

- $O(n^2)$ relevant intervals, 15 for each pair i, j (Baptiste, Le Pape, and Nuijten, 2001)
  - Algorithm to check them all in $O(n^2)$ (Baptiste, Le Pape, and Nuijten, 2001)

Energetic reasoning decomposition

$$\forall 0 \leq a < b < h \quad W(a,b) \leq C(b-a) \quad \text{(ER)}$$

Relevant intervals (Baptiste, 1998)

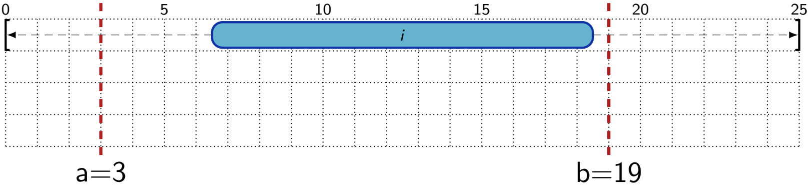
$$O_1(i) = \{r_i, lst_i, ect_i\}$$
$$O_2(i) = \{d_i, lst_i, ect_i\}$$
$$O(i,t) = \{lst_i - t \mid i \in \mathcal{T}\}$$

- (ER) holds iff it holds for every $i \neq j$ and every $a < b$ s.t.
  - $a, b \in O_1(i) \times O_2(j)$
  - $a, b \in O_1(i) \times O(j, a)$
  - $a, b \in O_2(i) \times O(j, b)$

- $O(n^2)$ relevant intervals, 15 for each pair i, j (Baptiste, Le Pape, and Nuijten, 2001)
  - Algorithm to check them all in $O(n^2)$ (Baptiste, Le Pape, and Nuijten, 2001)
- Possible with only 2 intervals for each pair (Derrien and Petit, 2014)

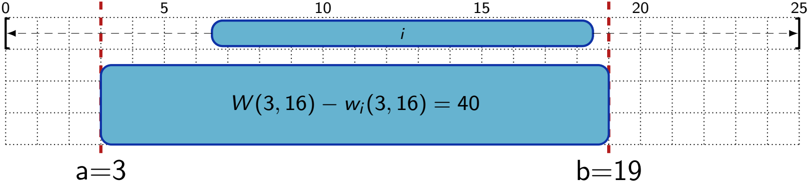
Energetic reasoning decomposition

$$\forall 0 \leq a < b < h \quad W(a,b) \leq C(b-a)$$



Energetic reasoning decomposition

$$\forall 0 \leq a < b < h \quad W(a,b) \leq C(b-a)$$

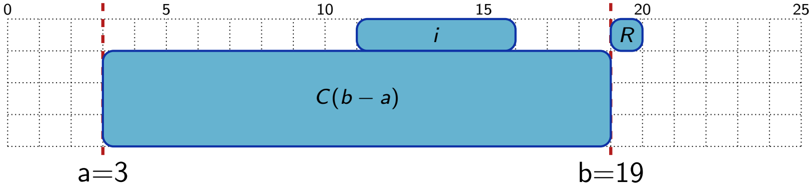


- $C = 3$, maximum energy on $[3,19) = 48$ $(W(3,16) = 46,\ w_i(3,16) = 6)$

Energetic reasoning decomposition

$$\forall 0 \leq a < b < h \quad W(a,b) \leq C(b-a)$$

$$\underbrace{W(a,b) + p_i^+(a) - w_i(a,b) - C(b-a)}_{R \text{ (at least after } b)} > 0 \implies e_i \geq b + \frac{R}{c_i}$$



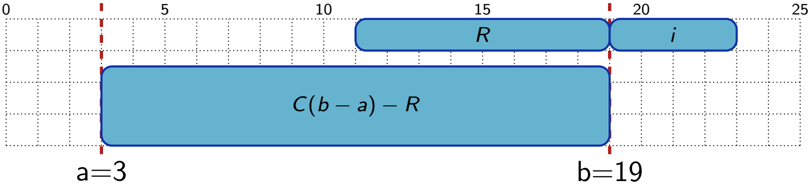
- $C = 3$, maximum energy on $[3, 19) = 48$ $(W(3,16) = 46,\ w_i(3,16) = 6)$

Energetic reasoning decomposition

$$\forall 0 \leq a < b < h \quad W(a,b) \leq C(b-a)$$

$$\underbrace{W(a,b) + p_i^+(a) - w_i(a,b) - C(b-a)}_{R \text{ (at least after } b)} > 0 \implies e_i \geq b + \frac{R}{c_i}$$

$$\min(b-a, p_i^+(a)) - \underbrace{w_i(a,b) + C(b-a) - W(a,b)}_{R \text{ (at most before } b)} > 0 \implies s_i \geq b - \frac{R}{c_i}$$



- $C = 3$, maximum energy on $[3, 19) = 48$ $(W(3,16) = 46, \; w_i(3,16) = 6)$

LAAS CNRS

Energetic reasoning algorithm

- $O(n^3)$ (Baptiste, Le Pape, and Nuijten, 2001)

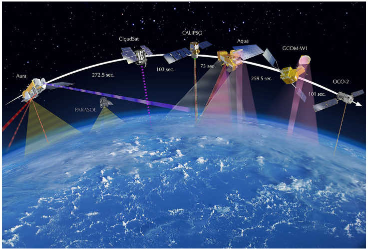
Energetic reasoning algorithm

- $O(n^3)$ (Baptiste, Le Pape, and Nuijten, 2001)
- $O(n^2 \log n)$ (Bonifas, 2014; Tesch, 2016)
  - Not complete, but at least one bound adjustment (hence $O(kn^2 \log n)$ where k is the number of tasks requiring a bound adjustment)

Part II: Search

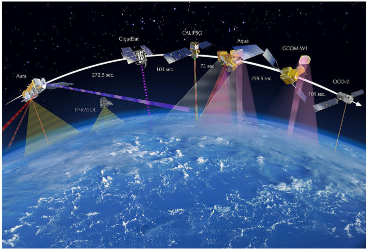
The importance of search: a short story

- Jobs: Files to transfer
- Resources:
  - Download channels: at most that many simultaneous downloads
  - Memory banks: cannot download two files stored on the same memory bank simultaneously
- Download as much data as possible within a given time window



The importance of search: a short story

- Jobs: Files to transfer
- Resources:
  - Download channels: at most that many simultaneous downloads
    - Cumulative resource shared by every task
  - Memory banks: cannot download two files stored on the same memory bank simultaneously
    - Tasks partitioned in as many unary resources as memory banks (m)
- Download as much data as possible within a given time window
  - Minimize makespan

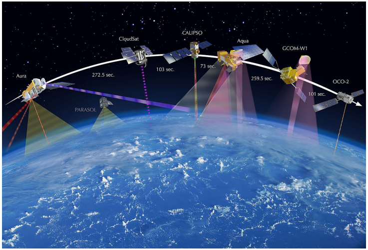


The importance of search: a short story

- Alas, our method was hardly better than a very basic greedy algorithm...

Greedy algorithm

- **Repeat:**
  - Choose the largest task a from the resource with highest demand
  - Schedule a as soon as possible



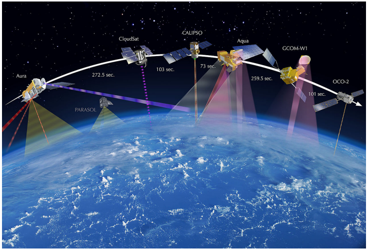
The importance of search: a short story

- Alas, our method was hardly better than a very basic greedy algorithm...

Greedy algorithm

- **Repeat:**
  - Choose the largest task a from the resource with highest demand
  - Schedule a as soon as possible

- Approximation ratio: $2 - \frac{2}{m+1}$ (Hebrard et al., 2016)



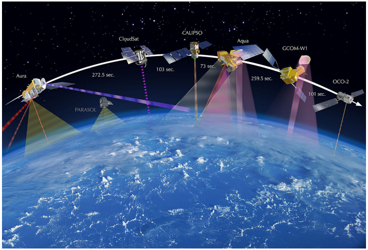
The importance of search: a short story

- Alas, our method was hardly better than a very basic greedy algorithm...

Greedy algorithm

- **Repeat:**
  - Choose the largest task a from the resource with highest demand
  - Schedule a as soon as possible

- Approximation ratio: $2 - \frac{2}{m+1}$ (Hebrard et al., 2016)
- Approximation ratio: $1 + \rho\frac{m-1}{n}$ where ρ is the ratio between largest and smallest task size



The importance of search: a short story

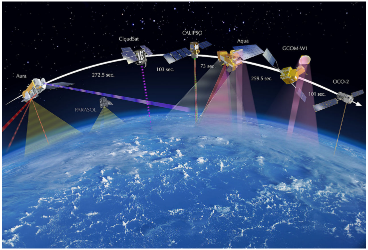
- Alas, our method was hardly better than a very basic greedy algorithm…

Greedy algorithm

- **Repeat:**
  - Choose the largest task a from the resource with highest demand
  - Schedule a as soon as possible

- Approximation ratio: $2 - \frac{2}{m+1}$ (Hebrard et al., 2016)
- Approximation ratio: $1 + \rho\frac{m-1}{n}$ where ρ is the ratio between largest and smallest task size

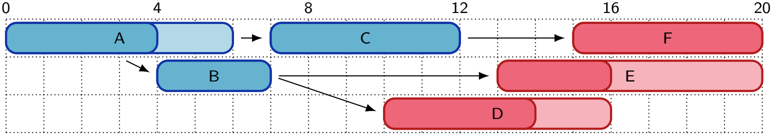
Not all resource scheduling are hard!

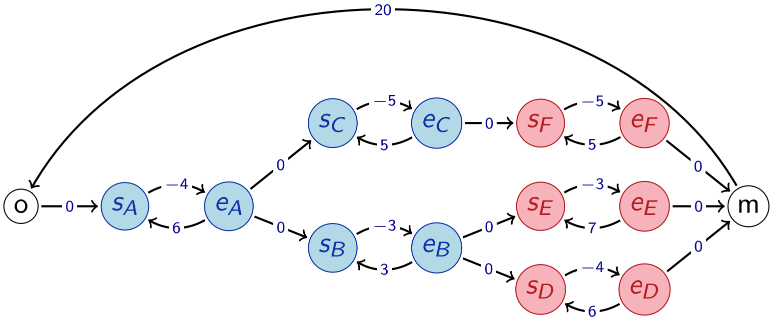


Search strategy

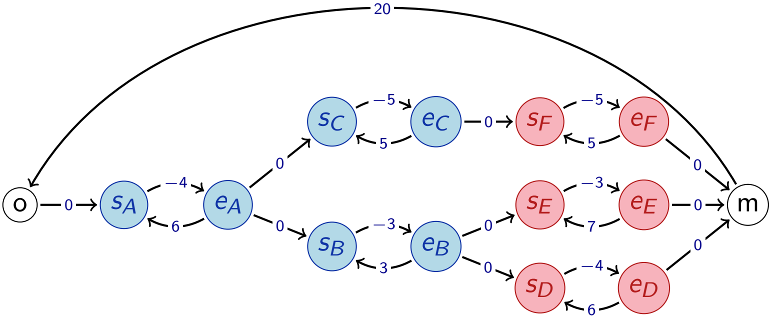
- Default CP strategy is a very bad idea:
  - Choose task i minimizing $d_i - r_i - p_i$ (and/or most constrained by resources and precedences)
  - Branch on $s_i = r_i$ or $s_i > r_i$: create "holes", dependent on the precision
- "Schedule or postpone" $\simeq$ branch on the task to start at the first available slot
  - Circumvent the precision problem, usually efficient for makespan minimization
  - Non trivial to implement in a classical CP solver
  - Might be incomplete for some objectives or constraints
- Sophisticated techniques in the latest CP solver (CP Optimizer) (Vilím, Laborie, and Shaw, 2015)
  - Alternate between Large Neighborhood Search and "Failure Directed Search"

Precedence graph / Difference system

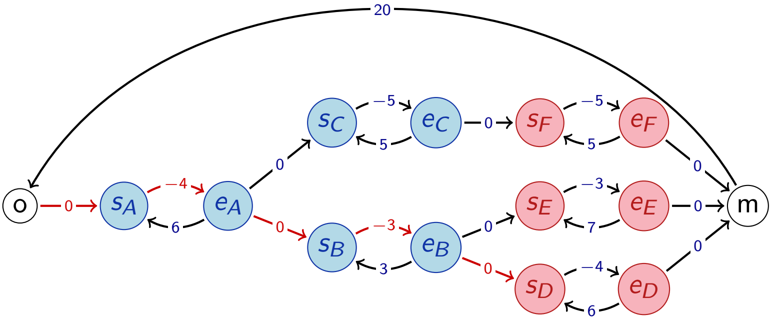




Precedence graph as "domain"

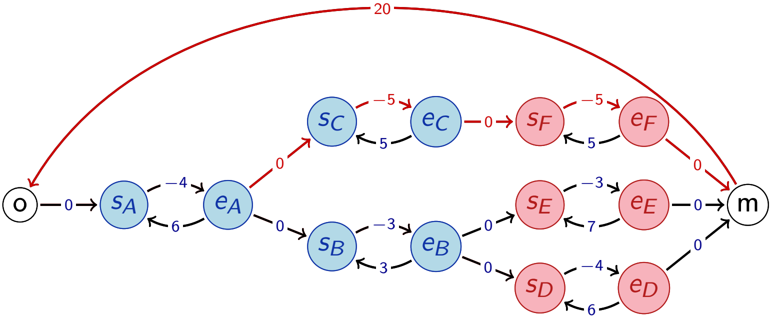


Precedence graph as "domain"



- Lower bound of node x is $-p_{0,x}$ where $p_{0,x}$ is the shortest path from 0 to x

Precedence graph as "domain"



- Lower bound of node x is $-p_{0,x}$ where $p_{0,x}$ is the shortest path from 0 to x
- Upper bound of node x is $p_{x,0}$

Precedence graph as "domain"

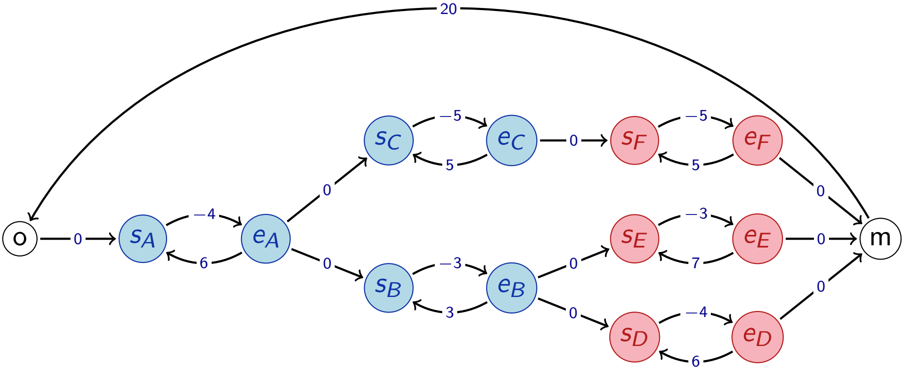
- Convenient way to implement the domain / solution space
  - Fewer concepts (nodes, arcs vs. min/max duration p_i, release and due dates r_i, d_i, precedences)
  - Clean propagation
    - Lower, upper bounds and negative cycles: [Bellman–Ford]
    - Transitive closure on precedences: [Floyd–Warshall]

Precedence graph as "domain"

- Convenient way to implement the domain / solution space
  - Fewer concepts (nodes, arcs vs. min/max duration p_i, release and due dates r_i, d_i, precedences)
  - Clean propagation
    - Lower, upper bounds and negative cycles: [Bellman–Ford]
    - Transitive closure on precedences: [Floyd–Warshall]
- Simulated by precedence constraints propagation and the "orchestra's conductor"

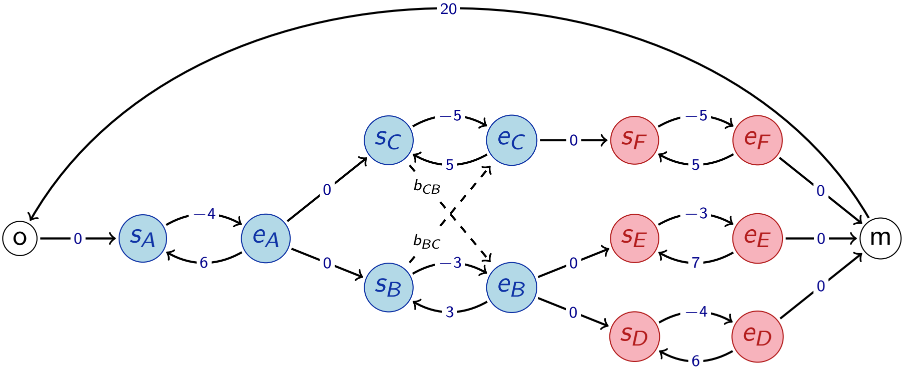
Search on the precedence graph

- Variable b_{ij} standing for $i \prec j$ for each pair of tasks i, j sharing a resource



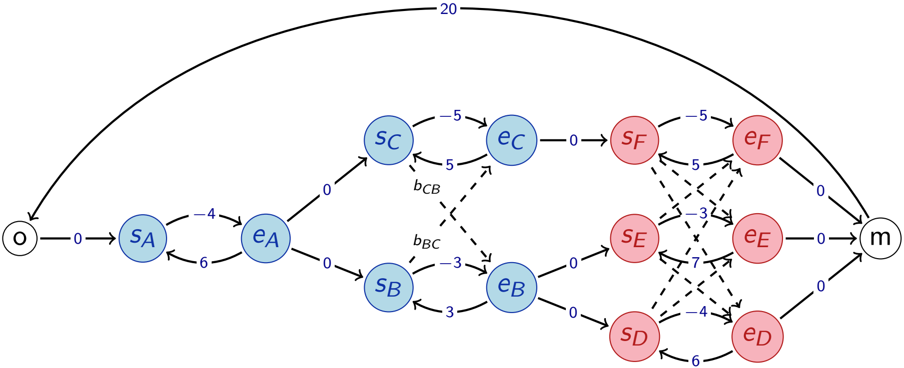
Search on the precedence graph

- Variable b_{ij} standing for $i \prec j$ for each pair of tasks i, j sharing a resource



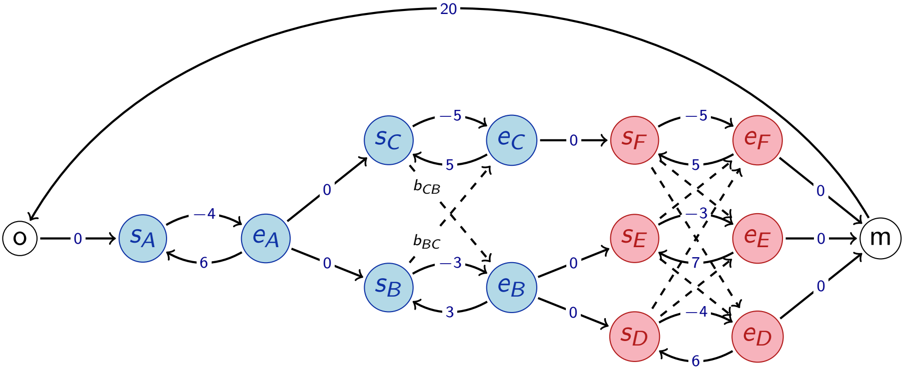
Search on the precedence graph

- Variable b_{ij} standing for $i \prec j$ for each pair of tasks i, j sharing a resource



Search on the precedence graph

- Variable b_{ij} standing for $i \prec j$ for each pair of tasks i, j sharing a resource
  - Disjunctive constraints $\implies$ there is a solution iff there is no negative cycle



Conflict directed scheduling (Grimes and Hebrard, 2015)

Conflict weighting

- Only disjunctive constraints, one Boolean variable for each
- Weighted Degree: choose the tasks involved in the most conflicts
- Branch on the Boolean variables (post precedence one way)

IBM CP Optimizer

- Every algorithm seen here (use Cplex's linear relaxation?)
- Alternate large neighborhood search and Failure Directed Search
- Specialized strategies and auto-tuning

Conflict directed scheduling (Grimes and Hebrard, 2015)

Conflict weighting

- Only disjunctive constraints, one Boolean variable for each
- Weighted Degree: choose the tasks involved in the most conflicts
- Branch on the Boolean variables (post precedence one way)

IBM CP Optimizer

- Every algorithm seen here (use Cplex's linear relaxation?)
- Alternate large neighborhood search and Failure Directed Search
- Specialized strategies and auto-tuning

	Objective	Proof	Objective	Proof
C_{max}	0.50%	**75%**	**0.03%**	56%

Conflict directed scheduling (Grimes and Hebrard, 2015)

Conflict weighting

- Only disjunctive constraints, one Boolean variable for each
- Weighted Degree: choose the tasks involved in the most conflicts
- Branch on the Boolean variables (post precedence one way)

IBM CP Optimizer

- Every algorithm seen here (use Cplex's linear relaxation?)
- Alternate large neighborhood search and Failure Directed Search
- Specialized strategies and auto-tuning

	Objective	Proof	Objective	Proof
C_{max}	0.50%	**75%**	**0.03%**	56%
setup	**0.00%**	**52%**	0.27%	4%

Conflict directed scheduling (Grimes and Hebrard, 2015)

Conflict weighting

- Only disjunctive constraints, one Boolean variable for each
- Weighted Degree: choose the tasks involved in the most conflicts
- Branch on the Boolean variables (post precedence one way)

IBM CP Optimizer

- Every algorithm seen here (use Cplex's linear relaxation?)
- Alternate large neighborhood search and Failure Directed Search
- Specialized strategies and auto-tuning

	Conflict weighting Objective	Conflict weighting Proof	IBM CP Optimizer Objective	IBM CP Optimizer Proof
C_{max}	0.50%	**75%**	**0.03%**	56%
setup	**0.00%**	**52%**	0.27%	4%
lags	**0.39%**	**64%**	0.54%	22%

Conflict directed scheduling (Grimes and Hebrard, 2015)

Conflict weighting

- Only disjunctive constraints, one Boolean variable for each
- Weighted Degree: choose the tasks involved in the most conflicts
- Branch on the Boolean variables (post precedence one way)

IBM CP Optimizer

- Every algorithm seen here (use Cplex's linear relaxation?)
- Alternate large neighborhood search and Failure Directed Search
- Specialized strategies and auto-tuning

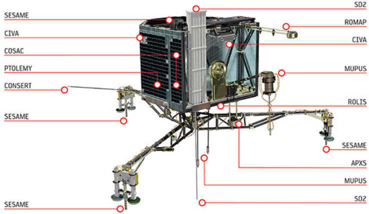
	Conflict weighting: Objective	Conflict weighting: Proof	IBM CP Optimizer: Objective	IBM CP Optimizer: Proof
C_{max}	0.50%	**75%**	**0.03%**	56%
setup	**0.00%**	**52%**	0.27%	4%
lags	**0.39%**	**64%**	0.54%	22%
$T_{\sum}$	**1.59%**	**73%**	2.52%	33%

Part III: Other types of resource

Planning the mission of Philae on the comet 67P

(Simonin et al., 2015)

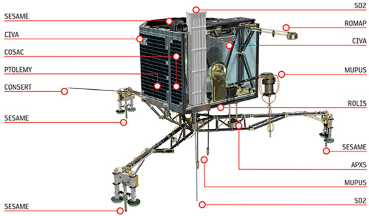
- Jobs: Scientific experiments
- Resources:
  - Batteries: threshold on the instant energy consumption
  - Memory: experiments produce data and transfers are possible only when Rosetta is visible



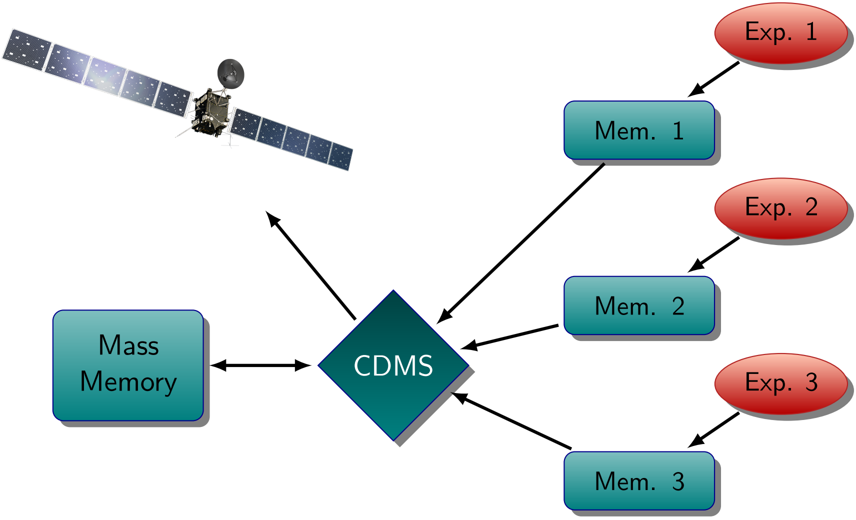
Planning the mission of Philae on the comet 67P

(Simonin et al., 2015)

- Jobs: Scientific experiments
- Resources:
  - Batteries: threshold on the instant energy consumption
    - ★ Nested cumulative constraints
  - Memory: experiments produce data and transfers are possible only when Rosetta is visible
    - ★ Memory / transfer channel resources (?)

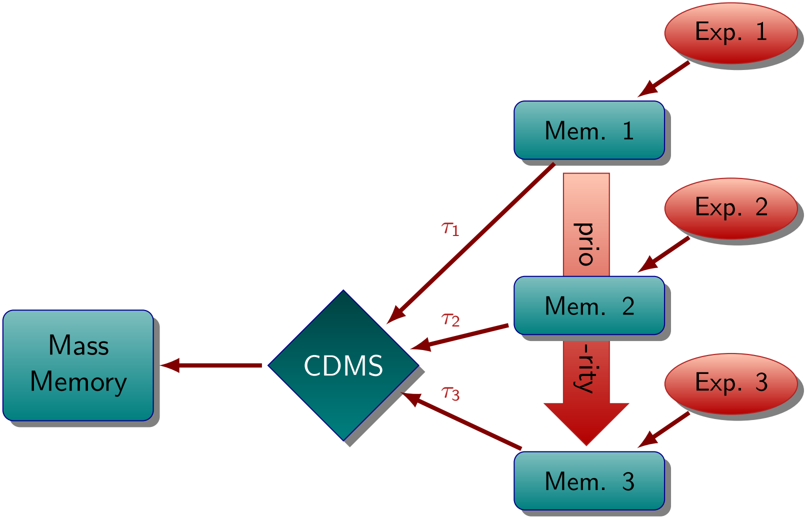


Command and Data Management Subsystem



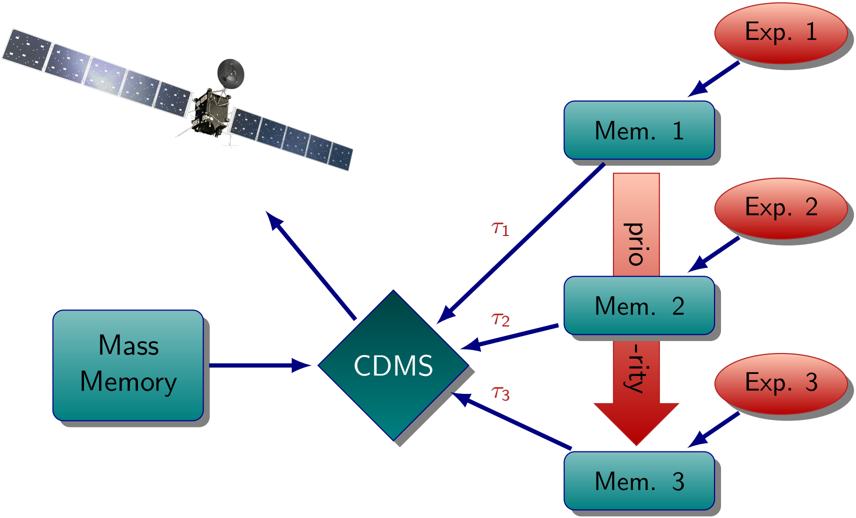
Command and Data Management Subsystem

(non-visibility)



Command and Data Management Subsystem

(visibility)



Simulating the Transfers

Original implementation

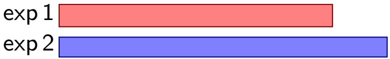
`IlcReservoir` constraints and optional transfer tasks

Simulating the Transfers

Original implementation

`IlcReservoir` constraints and optional transfer tasks

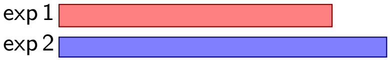
- Two experiments producing data
  - Exp.2 has higher priority
  - production rate $<$ transfer rate



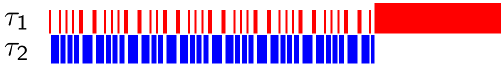
Original implementation

`IlcReservoir` constraints and optional transfer tasks

- Two experiments producing data
  - Exp.2 has higher priority
  - production rate < transfer rate



- Transfers
  - Switch back and forth from Exp.2 to Exp.1

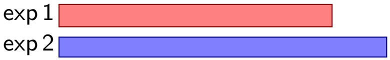


Simulating the Transfers

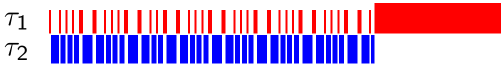
Original implementation

`IlcReservoir` constraints and optional transfer tasks

- Two experiments producing data
  - Exp.2 has higher priority
  - production rate < transfer rate

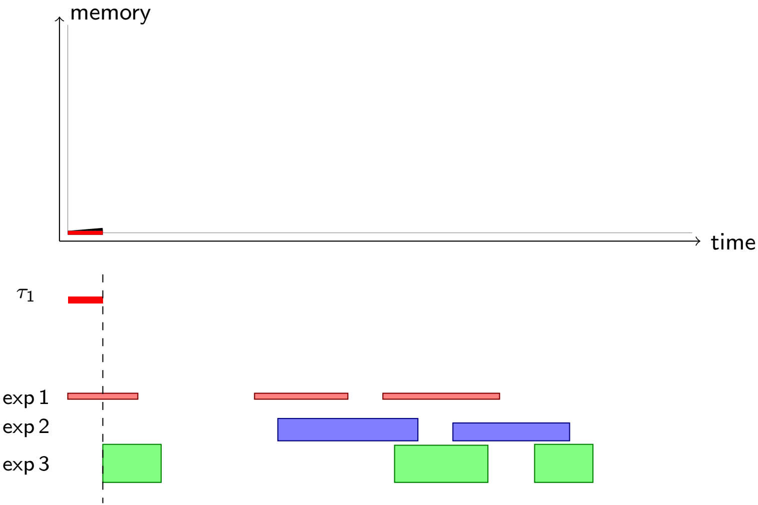


- Transfers
  - Switch back and forth from Exp.2 to Exp.1

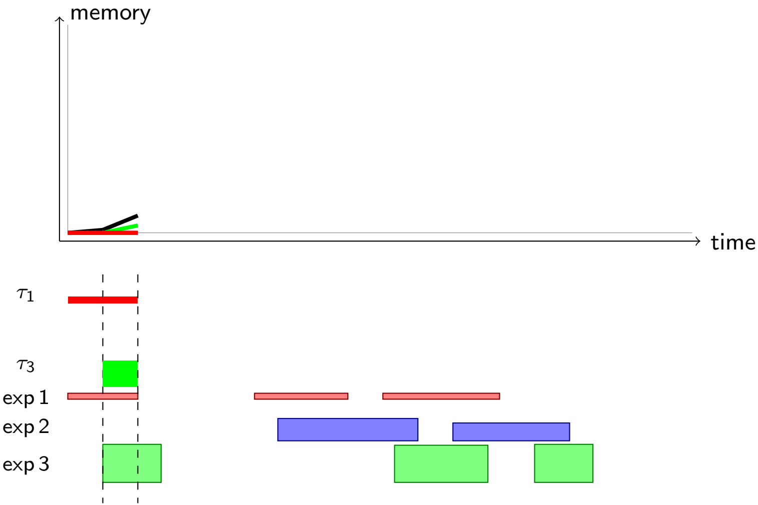


- Modeled as bandwidth sharing
  - Depends on priority, production and transfer rates

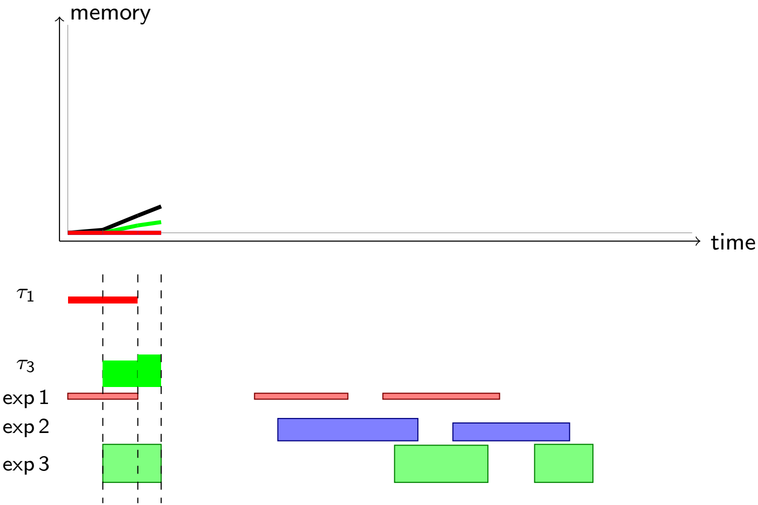
Checking the Constraint



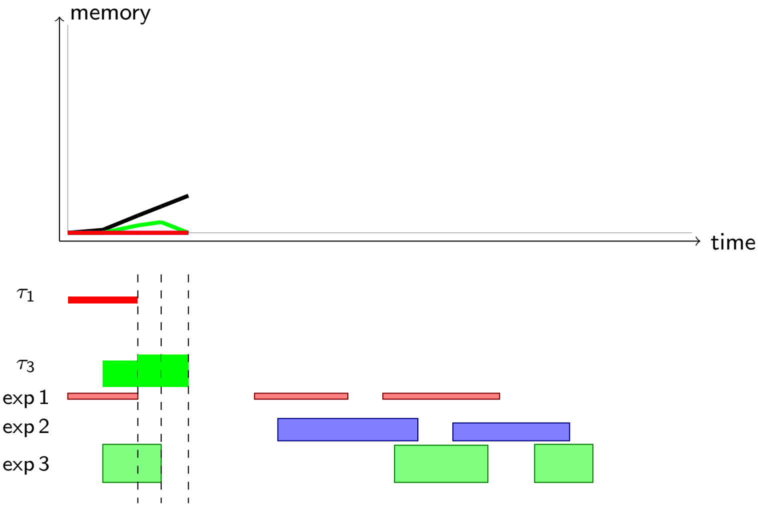
Checking the Constraint



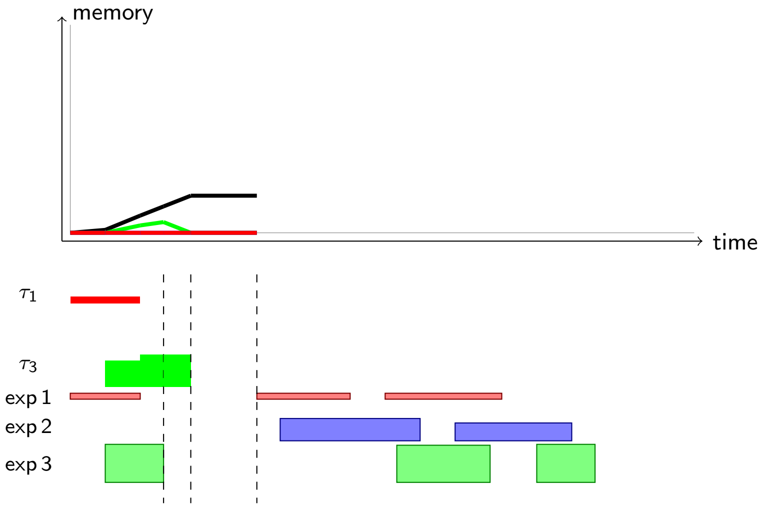
Checking the Constraint



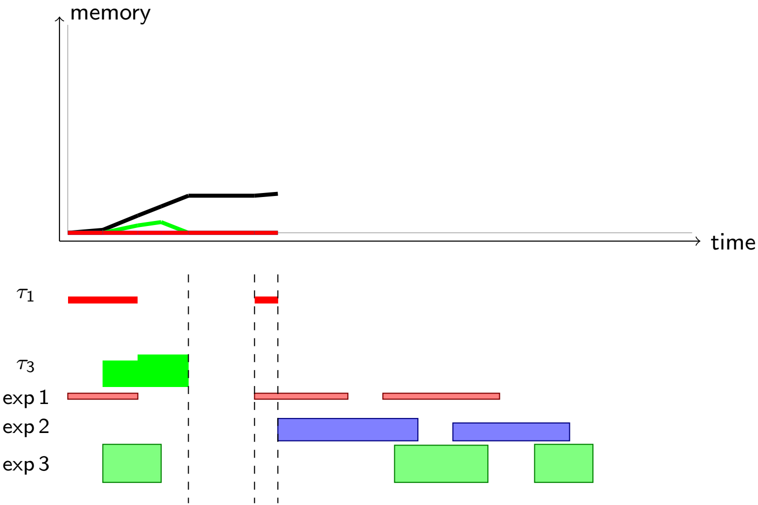
Checking the Constraint



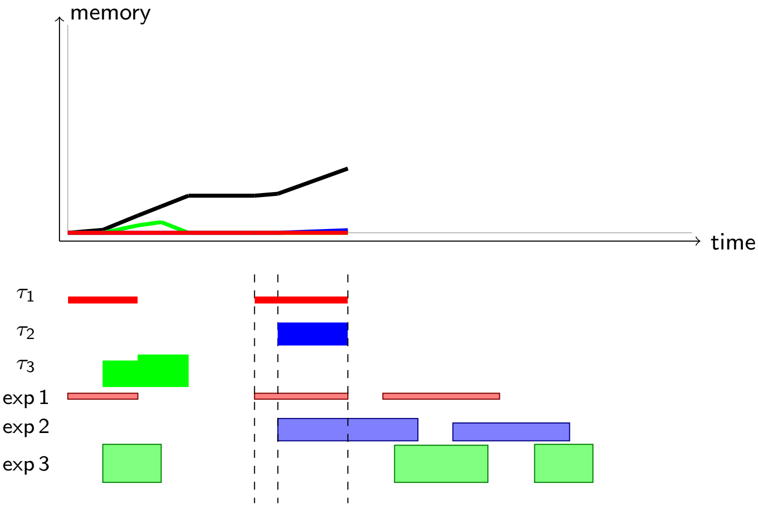
Checking the Constraint



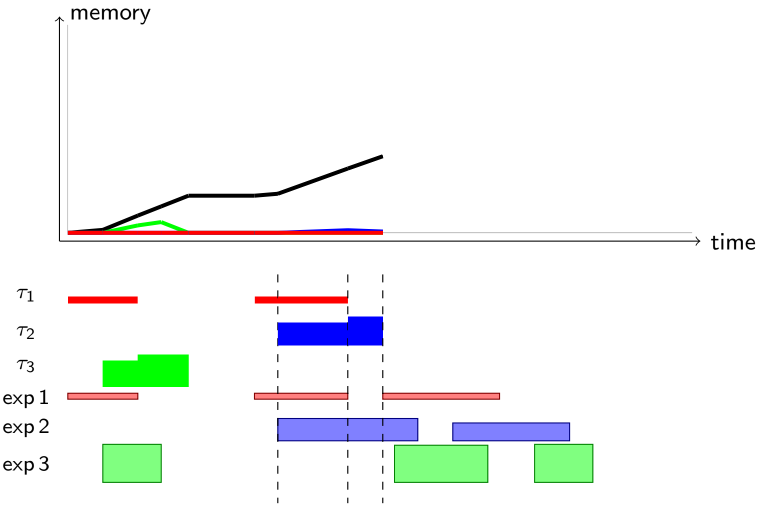
Checking the Constraint



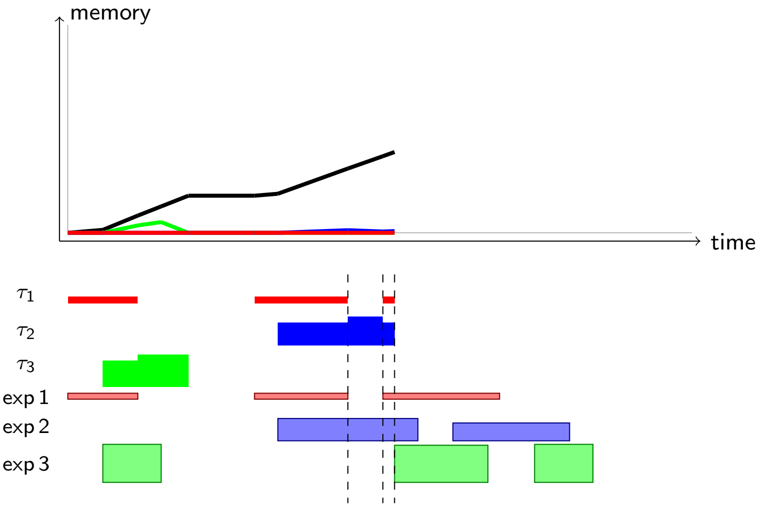
Checking the Constraint



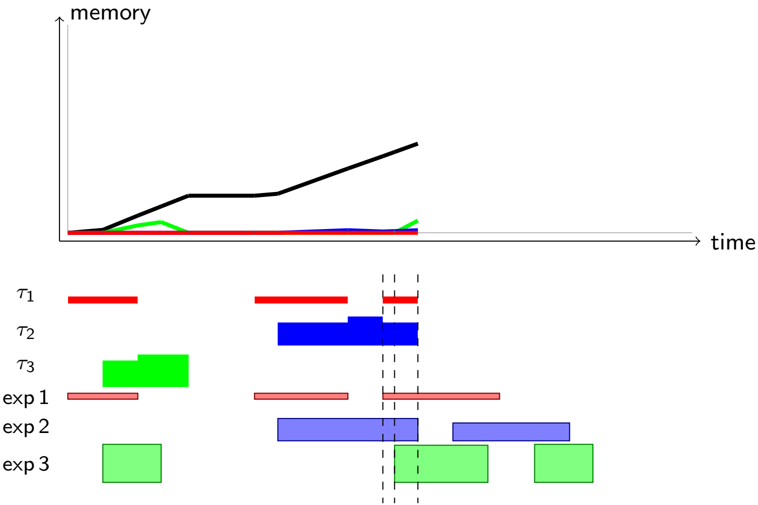
Checking the Constraint



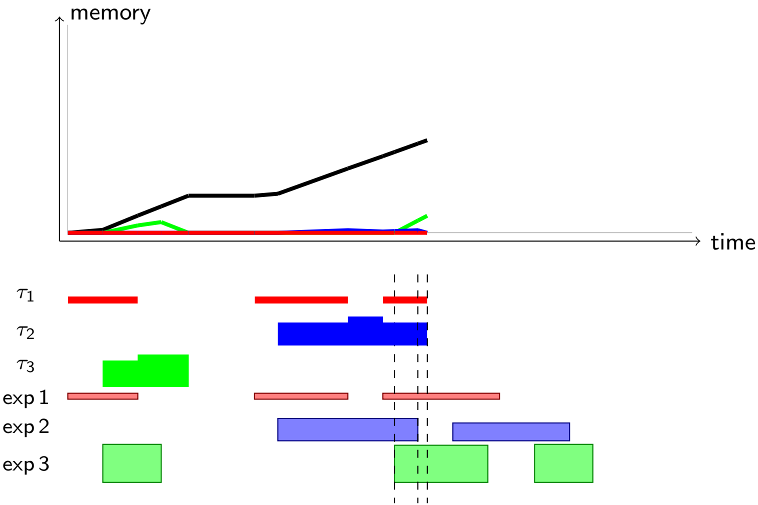
Checking the Constraint



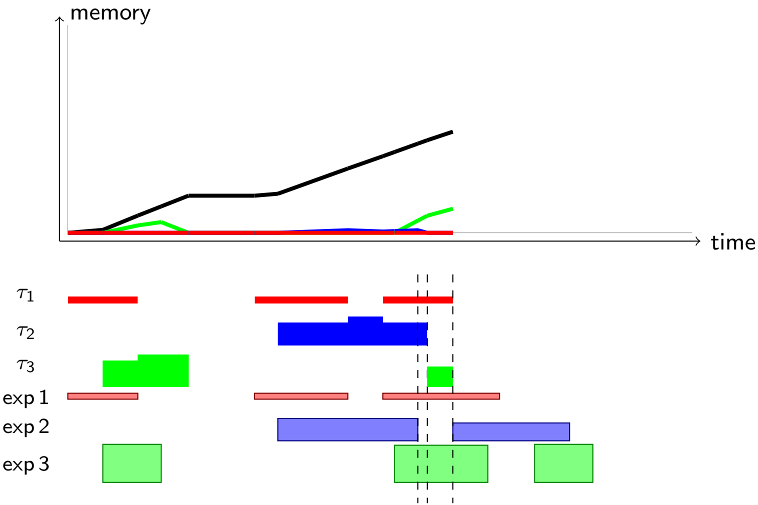
Checking the Constraint



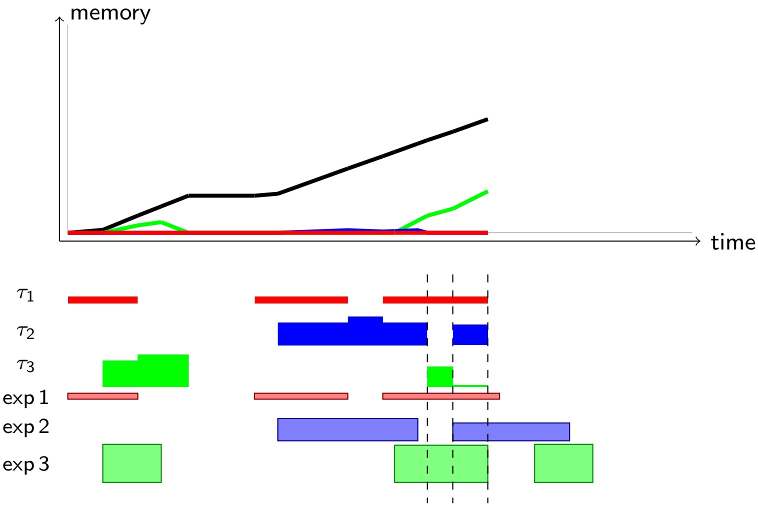
Checking the Constraint



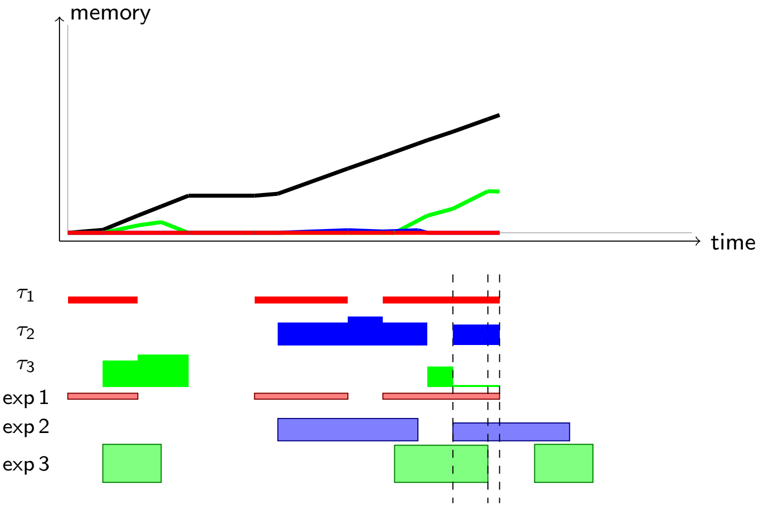
Checking the Constraint



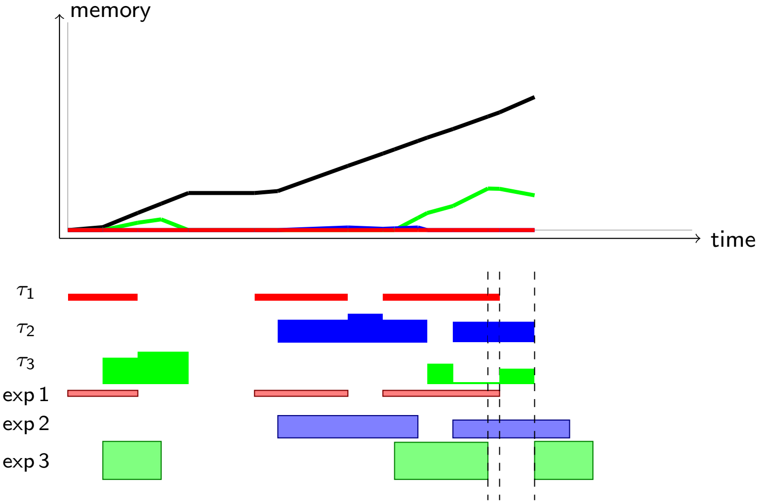
Checking the Constraint



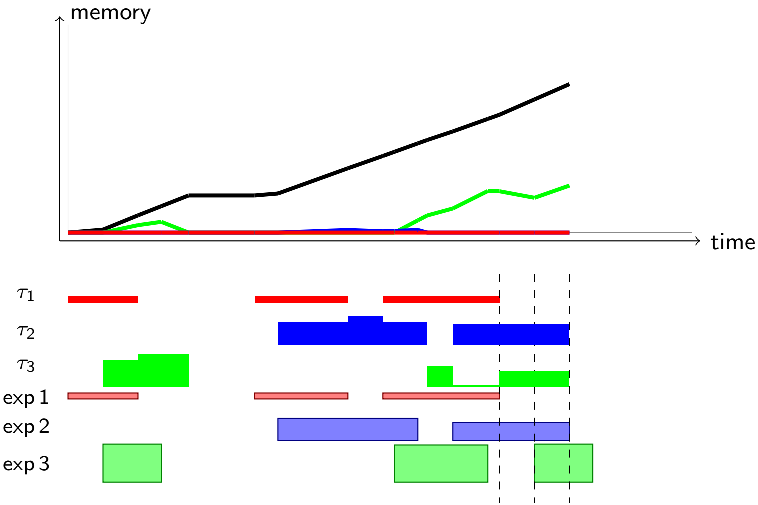
Checking the Constraint



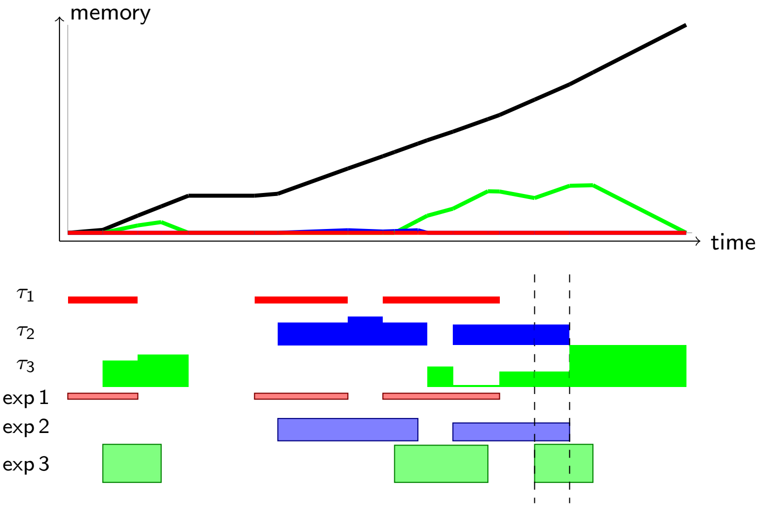
Checking the Constraint



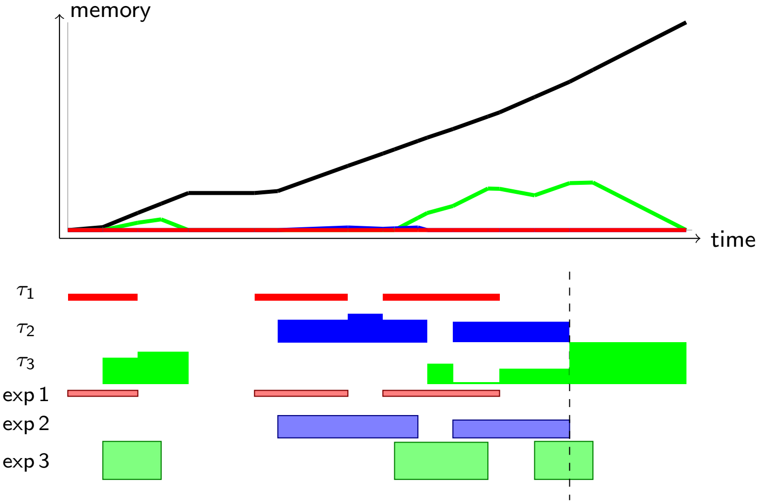
Checking the Constraint



Checking the Constraint



Checking the Constraint



Sweep algorithm (Beldiceanu and Carlsson, 2001)

- The constraint can be checked in $O(n \log n)$

Sweep algorithm (Beldiceanu and Carlsson, 2001)

- The constraint can be checked in $O(n \log n)$
- Several experiments active simultaneously:
  - Error is less than or equal to $1 + \frac{\tau^{max}}{\tau^{min}} \simeq 3$ blocks

Sweep algorithm (Beldiceanu and Carlsson, 2001)

- The constraint can be checked in $O(n \log n)$
- Several experiments active simultaneously:
  - Error is less than or equal to $1 + \frac{\tau^{max}}{\tau^{min}} \simeq 3$ blocks
- Faster and more accurate than the reservoir/transfer tasks model

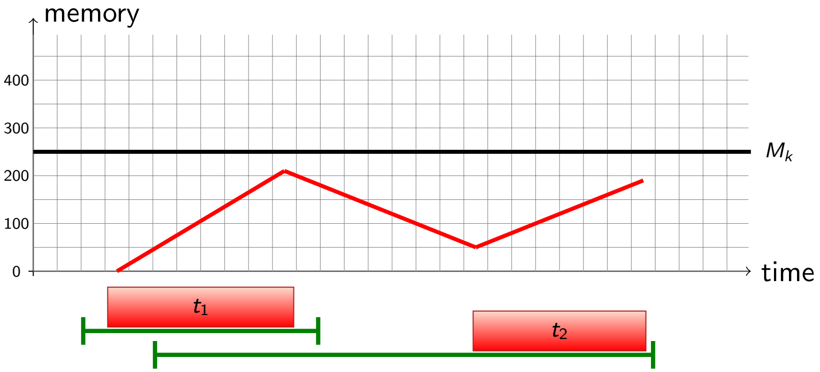
Sweep algorithm (Beldiceanu and Carlsson, 2001)

- The constraint can be checked in $O(n \log n)$
- Several experiments active simultaneously:
  - Error is less than or equal to $1 + \frac{\tau^{max}}{\tau^{min}} \simeq 3$ blocks
- Faster and more accurate than the reservoir/transfer tasks model
- Bound adjustments? Two principles:
  - Producing too much data too quickly can lead to data loss
  - Filling up the mass memory while not in visibility can lead to data loss

Propagation: production/transfer rate

For a set of tasks

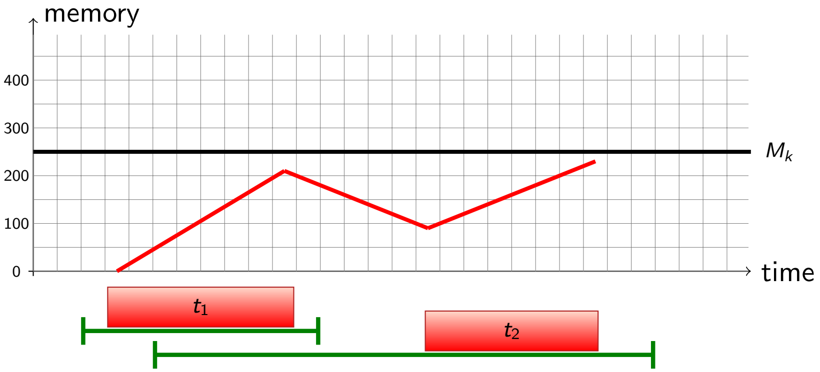
- lower bound on how much time the CDMS needs to transfer without data loss



Propagation: production/transfer rate

For a set of tasks

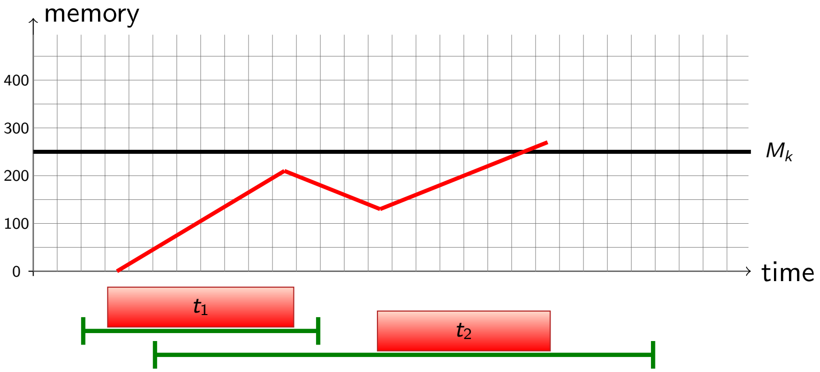
- lower bound on how much time the CDMS needs to transfer without data loss



Propagation: production/transfer rate

For a set of tasks

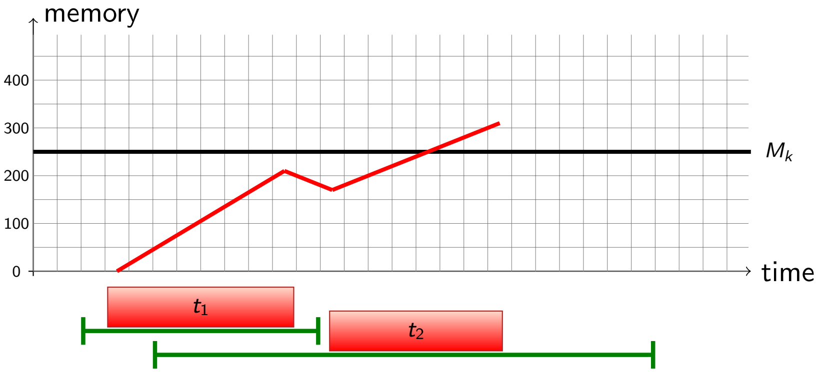
- lower bound on how much time the CDMS needs to transfer without data loss



Propagation: production/transfer rate

For a set of tasks

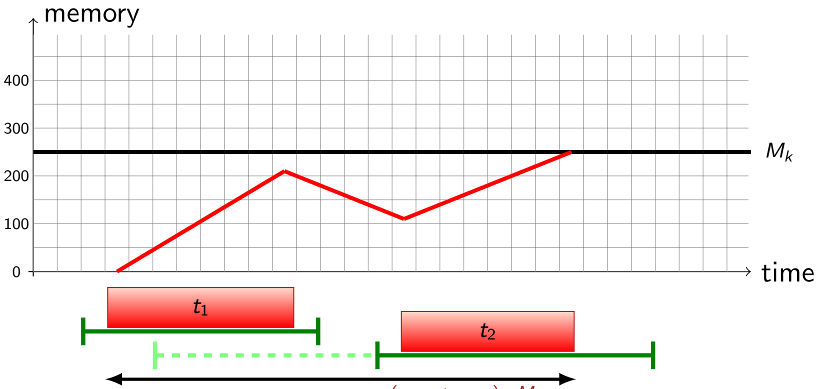
- lower bound on how much time the CDMS needs to transfer without data loss



Propagation: production/transfer rate

For a set of tasks

- lower bound on how much time the CDMS needs to transfer without data loss



$$end(t_2) - start(t_1) \geq \frac{(\pi_1 p_1 + \pi_2 p_2) - M_k}{\tau}$$

Propagation: production/transfer rate

More generally, we consider a set of tasks Ω of a given experiment

Propagation: production/transfer rate

More generally, we consider a set of tasks Ω of a given experiment

- Let $[a, b]$ be a time interval necessarily contained in Ω's transfer period

Propagation: production/transfer rate

More generally, we consider a set of tasks Ω of a given experiment

- Let $[a, b]$ be a time interval necessarily contained in Ω's transfer period
- We can take into account the tasks of higher priority producing during $[a, b]$

Propagation: production/transfer rate

More generally, we consider a set of tasks Ω of a given experiment

- Let $[a, b]$ be a time interval necessarily contained in Ω's transfer period
- We can take into account the tasks of higher priority producing during $[a, b]$

Filtering rule

- Minimum amount of higher priority data to transfer on $[a, b]$ over transfer rate:
  - Lower bound on the time dedicated to higher priority experiment: $T_k(a, b)$

Propagation: production/transfer rate

More generally, we consider a set of tasks Ω of a given experiment

- Let $[a, b]$ be a time interval necessarily contained in Ω's transfer period
- We can take into account the tasks of higher priority producing during $[a, b]$

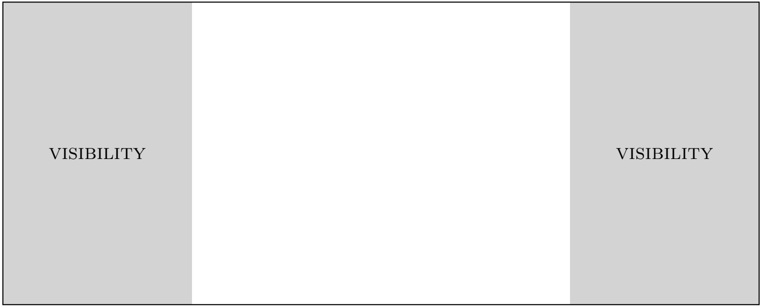
Filtering rule

- Minimum amount of higher priority data to transfer on $[a, b]$ over transfer rate:
  - Lower bound on the time dedicated to higher priority experiment: $T_k(a, b)$
- Induced constraint:

$$Makespan(\Omega) \geq \frac{(\sum_{t_{ki} \in \Omega} \pi_i p_i) - M_k}{\tau} + T_k(a, b)$$

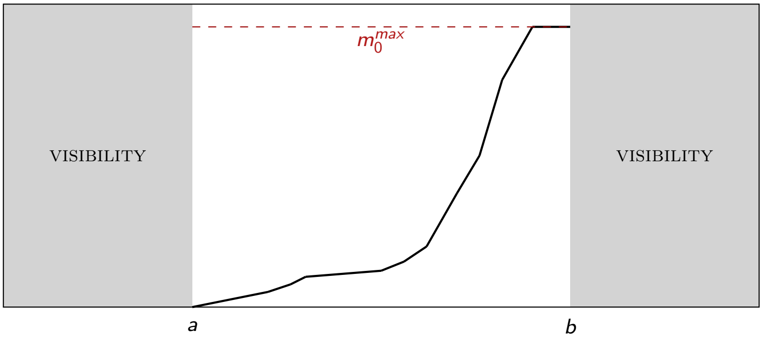
Propagation: visibility

- When the mass memory is full, no more data is transferred to it



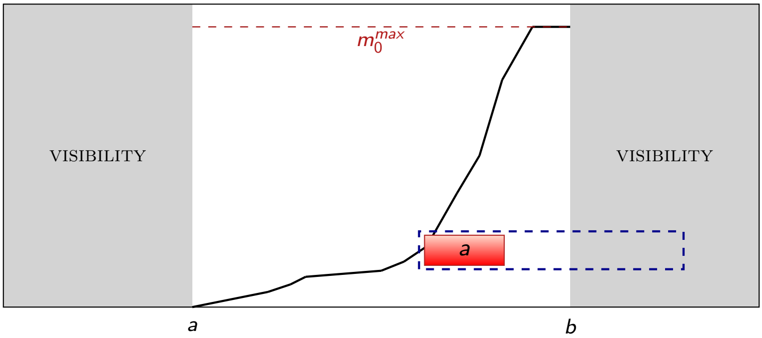
Propagation: visibility

- When the mass memory is full, no more data is transferred to it
- Minimal usage and peak m_0^{max}



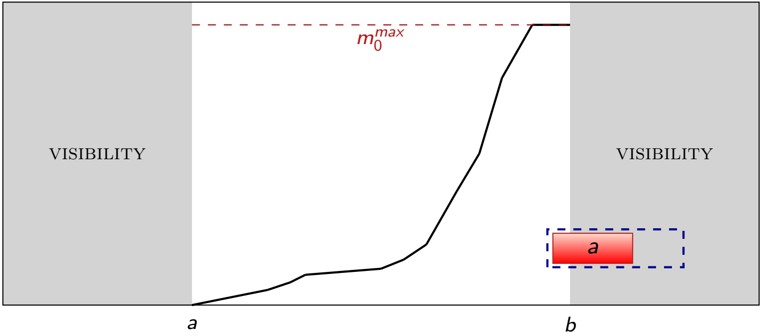
Propagation: visibility

- When the mass memory is full, no more data is transferred to it
- Minimal usage and peak m_0^{max}
- If the production exceeds $M_k + M_0 - m_0^{max}$ in $[a, b]$ then data will be lost



Propagation: visibility

- When the mass memory is full, no more data is transferred to it
- Minimal usage and peak m_0^{max}
- If the production exceeds $M_k + M_0 - m_0^{max}$ in $[a, b]$ then data will be lost
  - Filtering: bound start time w.r.t. this quantity of data and production rate



Conclusions

Conclusions

- Extremely successful application of constraint progamming

Conclusions

- Extremely successful application of constraint progamming
- Still a lot to achieve (better algorithms, different resources, objectives, search,...)

Conclusions

- Extremely successful application of constraint progamming
- Still a lot to achieve (better algorithms, different resources, objectives, search,...)
- Hybrid approaches (CP & MIP, CP & SAT,...)

Bibliography I

Applegate, David and William Cook (1991). “A Computational Study of the Job-Shop Scheduling Problem”. In: ORSA Journal on Computing 3.2, pp. 149–156.

Baker, Kenneth R. (1974).

Baptiste, Philippe (1998). “A Theoretical and Experimental Study of Resource Constraint Propagation”. PhD thesis. Université de Compiègne.

Baptiste, Philippe, Claude Le Pape, and Wim Nuijten (1998). Satisfiability Tests and Time-Bound Adjustments for Cumulative Scheduling Problems. Research Report 98-97, Université de Technologie de Compiègne.

— (2001). Constraint-Based Scheduling. Kluwer Academic Publishers.

Beldiceanu, Nicolas and Mats Carlsson (2001). “Sweep as a Generic Pruning Technique Applied to the Non-overlapping Rectangles Constraint”. In: Proceedings of the seventh International Conference on Principles and Practice of Constraint Programming – CP, pp. 377–391.

Bonifas, Nicolas (2014). “Fast propagation for the Energy Reasoning)”. In: Doctoral program of the 20th International Conference on Principles and Practice of Constraint Programming – CP.

Carlier, Jacques (1982). “The one-machine sequencing problem”. In: European Journal of Operational Research 11.1, pp. 42–47.

Carlier, Jacques and Eric Pinson (1989). “An Algorithm for Solving the Job-Shop Problem”. In: Management Science 35.2, pp. 164–176.

Bibliography II

Carlier, Jacques and Eric Pinson (1994). "Adjustment of heads and tails for the job-shop problem". In: European Journal of Operational Research 78.2, pp. 146–161.

Caseau, Yves and François Laburthe (1996). "Cumulative Scheduling with Task Intervals". In: Proceedings of the Joint International Conference and Symposium on Logic Programming, pp. 363–377.

Derrien, Alban and Thierry Petit (2014). "A New Characterization of Relevant Intervals for Energetic Reasoning". In: Proceedings of the 20th International Conference on Principles and Practice of Constraint Programming – CP, pp. 289–297.

Federgruen, A. and H. Groenevelt (1986). "Preemptive Scheduling of Uniform Machines by Ordinary Network Flow Techniques". In: Management Science 32.3, pp. 341–349.

Fox, B. R. (1990). "Non-chronological scheduling". In: Proceedings of AI, Simulation and Planning in High Autonomy Systems, pp. 72–77.

Gay, Steven, Renaud Hartert, and Pierre Schaus (2015). "Simple and Scalable Time-Table Filtering for the Cumulative Constraint". In: Proceedings of the 21st International Principles and Practice of Constraint Programming – CP, pp. 149–157.

Grimes, Diarmuid and Emmanuel Hebrard (2015). "Solving Variants of the Job Shop Scheduling Problem Through Conflict-Directed Search". In: INFORMS Journal on Computing 27.2, pp. 268–284.

Hebrard, Emmanuel et al. (2016). "Approximation of the Parallel Machine Scheduling Problem with Additional Unit Resources". In: Discrete Applied Mathematics 215, pp. 126–135.

Jackson, J.R. (1955). Scheduling a production line to minimize maximum tardiness. Research Report 43, Management Science Research Project. University of California, Los Angeles.

Bibliography III

Kameugne, Roger et al. (2011). “A Quadratic Edge-Finding Filtering Algorithm for Cumulative Resource Constraints”. In: Proceedings of the Seventeenth International Conference on Principles and Practice of Constraint Programming – CP, pp. 478–492.

Lahrichi, A. (1982). “Scheduling: the Notions of Hump, Compulsory Parts and their Use in Cumulative Problems”. In: C.R. Acad. Sc. Pairs 294, pp. 209–211.

Le Pape, Claude (1988). “Des systm̀es d’ordonnancement flexibles et opportunistes”. PhD thesis. Université Paris XI.

Le Pape, Claude and Philippe Baptiste (1996). “Constraint Propagation Techniques for Disjunctive Scheduling: The Preemptive Case”. In: Proceedings of the Twelfth European Conference on Artificial Intelligence – ECAI.

Lopez, Pierre (1991). “Approche énergétique pour l’ordonnancement de tâches sous contraintes de temps et de ressources”. PhD thesis. Université Paul Sabatier.

Martin, Paul and David B. Shmoys (1996). “A New Approach to Computing Optimal Schedules for the Job-Shop Scheduling Problem”. In: Proceedings of the Fifth International Conference on Integer Programming and Combinatorial Optimization – IPCO, pp. 389–403.

Mercier, Luc and Pascal Van Hentenryck (2008). “Edge Finding for Cumulative Scheduling”. In: INFORMS Journal on Computing 20.1, pp. 143–153.

Nuijten, Wim and Emile Aarts (1994). “Constraint Satisfaction for Multiple Capacitated Job Shop Scheduling”. In: Proceedings of the Tenth European Conference on Artificial Intelligence – ECAI, pp. 635–639.

Ouellet, Pierre and Claude-Guy Quimper (2013). “Time-Table Extended-Edge-Finding for the Cumulative Constraint”. In: Proceedings of the Nineteenth International Conference on Principles and Practice of Constraint Programming – CP, pp. 562–577.

Bibliography IV

Pinson, Eric (1988). "Le problème de job-shop". PhD thesis. Université Paris VI.

Roy, B. and M. A. Sussman (1964). Les problèmes d'ordonnancement avec contraintes disjonctive. Tech. rep. Note DS No.9 bis. SEMA, Paris.

Simonin, Gilles et al. (2015). "Scheduling Scientific Experiments for Comet Exploration". In: Constraints 20.1, pp. 77–99.

Tesch, Alexander (2016). "A Nearly Exact Propagation Algorithm for Energetic Reasoning in $\mathcal{O}(n^2 \log n)$". In: Proceedings of the 22nd International Conference on Principles and Practice of Constraint Programming – CP, pp. 493–519.

Torres, Philippe and Pierre Lopez (2000). "On Not-First/Not-Last conditions in disjunctive scheduling". In: European Journal of Operational Research 127.2, pp. 332–343.

Vilím, Petr (2004). "$\mathcal{O}(nlogn)$ Filtering Algorithms for Unary Resource Constraint". In: Proceedings of the First International Conference on Integration of AI and OR Techniques in Constraint Programming – CPAIOR pp. 335–347.

— (2009). "Max Energy Filtering Algorithm for Discrete Cumulative Resources". In: Proceedings of the Sixth International Conference on Integration of AI and OR Techniques in Constraint Programming – CPAIO pp. 294–308.

— (2011). "Timetable Edge Finding Filtering Algorithm for Discrete Cumulative Resources". In: Proceedings of the Eighth International Conference on Integration of AI and OR Techniques in Constraint Programming – CPAI

Vilím, Petr, Roman Barták, and Ondřej Čepek (2004). "Unary Resource Constraint with Optional Activities". In: Proceedings of the Tenth International Conference on Principles and Practice of Constraint Programming – CP, pp. 62–76.

Bibliography V

Vilím, Petr (2007). "Global Constraints in Scheduling". PhD thesis. Charles University in Prague, Faculty of Mathematics, Physics, Department of Theoretical Computer Science, and Mathematical Logic. URL: http://vilim.eu/petr/disertace.pdf.

Vilím, Petr, Philippe Laborie, and Paul Shaw (2015). "Failure-Directed Search for Constraint-Based Scheduling". In: Proceedings of the 12th International Conference on the Integration of AI and OR Techniques in Constraint Programming – CPA pp. 437–453.