Resource Constraints in Scheduling

Emmanuel Hebrard
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)
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
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
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 $i \in T$ 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 $i \in T$ such that:
  - Precedence constraints are satisfied
  - Resource constraints are satisfied
- Objectives:
  - Minimize makespan ($\max\{e_i \mid i \in T\}$)
  - Minimize total weighted tardiness ($\sum_{i \in T} w_i \max(0, e_i - \delta_i)$)
  - ...
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
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 $i < j$
  - Constraint predicate:
    \[ e_i \leq s_j \]
  - Algorithmic rules:
    \[ ect_i > r_j \implies r_j = ect_i \]
    \[ lst_j < d_i \implies d_i = lst_j \]
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
**Constraint** $C$ over variables $\mathcal{X}$

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

**Example:** $i < j$

- Predicate: $e_i \leq s_j$

---

Result of propagation algorithm is entailed by "bound consistency on $e_i \leq s_j$"

Its complexity is not..."
**Bound consistency**

**Constraint \( C \) over variables \( \mathcal{X} \)**
- \( C \): predicate defining a relation in \( \mathbb{N}^{\lvert \mathcal{X} \rvert} \)

**Bound support \( \sigma \) 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: \( \langle e_i : 7, s_j : 6 \rangle \)
**Constraint \( C \) over variables \( \mathcal{X} \)**

- \( C \): predicate defining a relation in \( \mathbb{N}^{\mid \mathcal{X} \mid} \)

**Bound support \( \sigma \) of \( x, t \) for \( C \) over \( \mathcal{X} \)**

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

- Result of propagation algorithm is entailed by “bound consistency on \( e_i \leq s_j \)”
- Its complexity is not

**Example:** \( i < j \)

- Predicate: \( e_i \leq s_j \)

![Diagram showing example]

- 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

**Ex: \( i < j \)**

- Predicate: \( e_i \leq s_j \)

[Diagram showing example with intervals for \( i \) and \( j \)]

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

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

A model of \texttt{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
  \[ \sum_{s_i \leq t \leq e_i} c_i \leq C \ \forall t \]
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
  \[ s_i + p_i \leq e_i \quad \forall i \]
- **Non preemption**: cannot be interrupted
  \[ s_i + p_i \geq e_i \quad \forall i \]
- **Bounds**: release and due dates
  \[ r_i \leq s_i \leq e_i \leq d_i \quad \forall i \]
- **Resource capacity**: additive, upper bounded resource usage
  \[ \sum_{s_i \leq t \leq e_i} c_i \leq C \quad \forall t \]

- **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
  \[ s_i + p_i \leq e_i \quad \forall i \]
- **Non preemption**: cannot be interrupted
  \[ s_i + p_i \geq e_i \quad \forall i \]
- **Bounds**: release and due dates
  \[ r_i \leq s_i \leq e_i \leq d_i \quad \forall i \]
- **Resource capacity**: additive, upper bounded resource usage
  \[ \sum_{s_i \leq t \leq e_i} c_i \leq C \quad \forall t \]

- **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`

\[
\begin{align*}
    s_i + p_i & \geq e_i \quad \forall i \\
    s_i + p_i & \leq e_i \quad \forall i \\
    r_i & \leq s_i \leq e_i \leq d_i \quad \forall i \\
    \sum_{s_i \leq t \leq e_i} c_i & \leq C \quad \forall t
\end{align*}
\]

- 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`

\[
\begin{align*}
    s_i + p_i & \geq e_i \quad \forall i \\
    s_i + p_i & \leq e_i \quad \forall i \\
    r_i & \leq s_i \leq e_i \leq d_i \quad \forall i \\
    \sum_{s_i \leq t \leq e_i} c_i & \leq C \quad \forall t
\end{align*}
\]

- 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 \textsc{CumulativeResource}

\[ \begin{align*}
    s_i + p_i & \geq e_i \quad \forall i \\
    s_i + p_i & \leq e_i \quad \forall i \\
    r_i & \leq s_i \leq e_i \leq d_i \quad \forall i \\
    \sum_{s_i \leq t \leq e_i} c_i & \leq C \quad \forall t
\end{align*} \]

- The problem \textsc{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 \textsc{CumulativeResource} is NP-hard (global support)
  - Enforcing bound consistency on the model above is polynomial (local supports)
Notion of “compulsory part” (Lahrichi, 1982)

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

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

1. $\forall i \quad s_i + p_i = e_i$  
   processing time & non-preemption

2. $\forall t \quad \sum_{i \in T} c_i a_i^t \leq C$  
   resource capacity

3. $\forall i \forall t \quad a_i^t \iff s_i \leq t \land t < e_i$  
   channeling
**Time-tabling decomposition**

- Boolean variables $a^t_i$ standing for: “task $i$ is in process at time $t$”
- Enforce bounds consistency on:

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

  \[
  \forall t \quad \sum_{i \in T} c_i a^t_i \leq C \quad \text{resource capacity} \quad (2)
  \]

  \[
  \forall i \forall t \quad a^t_i \iff s_i \leq t \land t < e_i \quad \text{channeling} \quad (3)
  \]

  - $s_A \leq 1 \land 1 < e_A \Rightarrow a^1_A = 1$
  - $\Rightarrow a^1_B = 0 \Rightarrow e_B \leq 1 \lor 1 < s_B$

  $A$ must be in process at $t = 1$

  $B$ 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:**

\[
\begin{align*}
\forall i \quad s_i + p_i &= e_i & \text{processing time & non-preemption} \\
\forall t \quad \sum_{i \in T} c_i a_i^t &\leq C & \text{resource capacity} \\
\forall i \forall t \quad a_i^t &\iff s_i \leq t \land t < e_i & \text{channeling}
\end{align*}
\]

- $s_A \leq 1 \land 1 < e_A \implies a_A^1 = 1$
- $\implies a_B^1 = 0 \implies e_B \leq 1 \lor 1 < s_B$
- $\implies 1 < s_B$

*A must be in process at $t = 1*

*B cannot be in process at $t = 1*

*B cannot end earlier than $t = 2$ so it must start at $t \geq 2*
Time-tabling decomposition

- Boolean variables \( a^t_i \) standing for: “task \( i \) is in process at time \( t \)”
- Enforce bounds consistency on:

\[
\forall i \quad s_i + p_i = e_i \quad \text{processing time & non-preemption} \tag{1}
\]
\[
\forall t \quad \sum_{i \in T} c_i a^t_i \leq C \quad \text{resource capacity} \tag{2}
\]
\[
\forall i \forall t \quad a^t_i \iff s_i \leq t \land t < e_i \quad \text{channeling} \tag{3}
\]

- \( s_A \leq 1 \land 1 < e_A \Rightarrow a^1_A = 1 \)

- \( \Rightarrow a^1_B = 0 \Rightarrow e_B \leq 1 \lor 1 < s_B \)

- \( \Rightarrow 1 < s_B \)
Time-tabling decomposition

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

\[
\forall i : s_i + p_i = e_i \quad \text{processing time & non-preemption} \\
\forall t : \sum_{i \in T} c_i a^t_i \leq C \quad \text{resource capacity} \\
\forall i \forall t : a^t_i \iff s_i \leq t \land t < e_i \quad \text{channeling}
\]

- $s_A \leq 1 \land 1 < e_A \Rightarrow a^1_A = 1$
- $\Rightarrow a^1_B = 0 \Rightarrow e_B \leq 1 \lor 1 < s_B$
- $\Rightarrow 1 < s_B$
Time-tabling decomposition

- Boolean variables $a^t_i$ standing for: “task $i$ is in process at time $t$”
- Enforce bounds consistency on:

\[
\begin{align*}
\forall i & \quad s_i + p_i = e_i & \text{processing time & non-preemption} \\
\forall t & \quad \sum_{i \in T} c_i a^t_i \leq C & \text{resource capacity} \\
\forall i \forall t & \quad a^t_i \iff s_i \leq t \land t < e_i & \text{channeling}
\end{align*}
\]

- $s_A \leq 1 \land 1 < e_A \Rightarrow a_A^1 = 1$
- $\Rightarrow a_B^1 = 0 \Rightarrow e_B \leq 1 \lor 1 < s_B$
- $\Rightarrow 1 < s_B$

$A$ must be in process at $t = 1$
$B$ cannot be in process at $t = 1$
$B$ cannot end earlier than $t = 2$ so it must start at $t \geq 2$
Time-tabling algorithm: satisfiability check

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

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

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

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

A task $B$ counted in the profile
• Assume $C = 3$ and consider the profile interval $[5, 10) : 2$
• A task $B$ counted in the profile
• $[a, b)$ overlaps with $[r_i, \min(\text{lst}_i, \text{ect}_i)) \implies r_i = \min(b, \text{lst}_i)$
Interval \([a, b) : h, \text{ Task } i\)
- overloaded iff \(c_i + h > C\)
- relevant iff
  - \(r_i < b\) and
  - \(\min(ect_i, lst_i) > a\)
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
Interval \([a, b) : h, \text{ Task } i\)

- overloaded iff \(c_i + h > C\)
- relevant iff
  - \(r_i < b\) and
  - \(\min(\text{ect}_i, \text{lst}_i) > a\)

**Algorithm** (Ouellet and Quimper, 2013)

- compute profile
- order chunks by decreasing \(h\)
- for \(i \in T\) by increasing \(c_i\):
  - add \(i\)'s over. intervals to \(S\)
Interval $[a, b) : h, Task i$

- overloaded iff $c_i + h > C$
- relevant iff
  - $r_i < b$ and
  - $\min(\text{ect}_i, \text{lst}_i) > a$

**Algorithm (Ouellet and Quimper, 2013)**

- compute profile
- order chunks by decreasing $h$
- for $i \in T$ 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
    - \(\min(\text{ect}_i, \text{lst}_i) > a\)

**Algorithm** (Ouellet and Quimper, 2013)

- compute profile
- order chunks by decreasing \(h\)
- for \(i \in T\) 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
    - \(\min(ect_i, \text{lst}_i) > a\)

---

**Algorithm (Ouellet and Quimper, 2013)**

- compute profile
- order chunks by decreasing \(h\)
- for \(i \in T\) 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
Interval \([a, b) : h, \text{ Task } i\)
- overloaded iff \(c_i + h > C\)
- relevant iff
  - \(r_i < b\) and
  - \(\min(\text{ect}_i, \text{lst}_i) > a\)

**Algorithm** *(Ouellet and Quimper, 2013)*

- compute profile
- order chunks by decreasing \(h\)
- for \(i \in T\) 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
    - $\min(\text{ect}_i, \text{lst}_i) > a$

**Algorithm (Ouellet and Quimper, 2013)**

- compute profile
- order chunks by decreasing $h$
- for $i \in T$ 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
    - \(\min(\text{ect}_i, \text{lst}_i) > a\)

**Algorithm** *(Ouellet and Quimper, 2013)*

- compute profile
- order chunks by decreasing \(h\)
- for \(i \in T\) 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
    - \(\min(\text{ect}_i, \text{lst}_i) > a\)

**Algorithm (Ouellet and Quimper, 2013)**

- compute profile
- order chunks by decreasing \(h\)
- for \(i \in T\) 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
Interval $[a, b) : h$, Task $i$
- overloaded iff $c_i + h > C$
- relevant iff
  - $r_i < b$ and
  - $\min(\text{ect}_i, \text{lst}_i) > a$

**Algorithm** (Ouellet and Quimper, 2013)
- compute profile
- order chunks by decreasing $h$
- for $i \in T$ 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 \(i\) overloads iff \(c_i + h > C\)
- Relevant iff
  - \(r_i < b\) and
  - \(\min(\text{ect}_i, \text{lst}_i) > a\)

**Algorithm (Ouellet and Quimper, 2013)**

- Compute profile
- Order chunks by decreasing \(h\)
- For \(i \in T\) 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)
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 \((\text{Roy and Sussman, 1964})\) central to \((\text{Carlier and Pinson, 1989})\)’s method
  - If \(i\) and \(j\) require the same exclusive resource

\[
\begin{array}{c}
0 & 2 & 4 & 6 & 8 & 10 \\
\hline
\text{i} & & & & & \\
\text{j} & & & & & \\
\end{array}
\]
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 \( i \preceq j \)

![Diagram showing disjunctive reasoning with tasks i and j requiring the same resource at different time slots.](image-url)
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)

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

\[ b_{ij} \neq b_{ji} \]
Disjunctive decomposition (single-machine)

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

\[ b_{ij} \neq b_{ji} \]
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)

∀ \( i < j \in T \), \( b_{ij} \iff e_i \leq s_j \)

\( b_{ij} \neq b_{ji} \)

- **Completeness** of disjunctive propagation: deciding only \( b_{ij} \) variables is sufficient
∀i < j ∈ ℰ, \[ b_{ij} \] \implies i < j \quad \text{(post } i < j \text{'s propagator)}

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

\text{ect}_j > \text{lst}_i \implies [b_{ij}]

\text{ect}_i > \text{lst}_j \implies \neg [b_{ij}]
∀i < j ∈ T, [b_{ij}] \implies i < j \quad \text{(post } i < j \text{'s propagator)}

\neg[b_{ij}] \implies j < i \quad \text{(post } j < 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 $e_i > t$
∀i < j ∈ ℰ,  
\[ b_{ij} \implies i < j \]  
(post \( i \prec j \)'s propagator)

\[ \neg b_{ij} \implies j < i \]  
(post \( j \prec i \)'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 \( e_i > t \)
  - \( \exists j \text{ s.t. } a_j^t = 1 \)
\( \forall i < j \in \mathcal{T}, \quad [b_{ij}] \implies i < j \quad \text{(post } i < j\text{'s propagator)} \)
\( \neg[b_{ij}] \implies j < i \quad \text{(post } j < i\text{'s propagator)} \)

\( \text{ect}_j > \text{lst}_i \implies [b_{ij}] \)
\( \text{ect}_i > \text{lst}_j \implies \neg[b_{ij}] \)

**Strictly stronger than Time-tabling**

- Suppose \( a_i^t = 0 \) and \( e_i > t \)
  - \( \exists j \text{ s.t. } a_j^t = 1 \) hence \( \text{lst}_j \leq t < \text{ect}_j \)
  - Therefore, \( \text{lst}_j < \text{ect}_i \) so \( b_{ij} \) must be false
\( \forall i < j \in \mathcal{T}, \quad [b_{ij}] \implies i < j \)  
(post \( i < j \)'s propagator)

\( \neg [b_{ij}] \implies j < i \)  
(post \( j < i \)'s propagator)

\( \text{ect}_j > \text{lst}_i \implies [b_{ij}] \)

\( \text{ect}_i > \text{lst}_j \implies \neg [b_{ij}] \)

**Strictly stronger than Time-tabling**

- Suppose \( a_i^t = 0 \) and \( e_i > t \)
  - \( \exists j \text{ s.t. } a_j^t = 1 \text{ hence } \text{lst}_j \leq t < \text{ect}_j \)
  - Therefore, \( \text{lst}_j < \text{ect}_i \) so \( b_{ij} \) must be false
  - and thus \( j < i \) is posted
Generalisation to cumulative resources

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

- Either $i \prec j$, $j \prec i$ or $i \not\prec j \land j \not\prec i$
  - Easy change: $b_{ii} \neq b_{ji}$ becomes $\neg (b_{ij} \land b_{ji})$, 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
- Partially Elastic relaxation
- Edge finding
- Energetic reasoning
A lot of different forms and flavours

- Preemptive relaxation
- Fully Elastic relaxation
- Partially Elastic relaxation
- Edge finding
- Energetic reasoning
Energetic reasoning

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

- Preemptive relaxation
- Fully Elastic relaxation
- Partially Elastic relaxation
- Edge finding
- Energetic reasoning
Energetic reasoning

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

- A lot of different forms and flavours
  - Preemptive relaxation
  - Fully Elastic relaxation
  - Partially 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)

\[ \forall i \forall t \notin [r_i, d_i] \quad a_i^t = 0 \quad \text{bounds} \]
\[ \forall t \quad \sum_i c_i a_i^t \leq C \quad \text{resource capacity} \]
\[ \forall i \quad \sum_t a_i^t = p_i \quad \text{processing times} \]

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 $[0, c_i]$

\[\forall i \forall t \not\in [r_i, d_i) \quad u_i^t = 0 \quad \text{bounds}\]
\[\forall t \quad \sum_i u_i^t \leq C \quad \text{resource capacity}\]
\[\forall i \quad \sum_t u_i^t = p_i c_i \quad \text{energy}\]

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

\[ \forall i \forall t \not\in [r_i, d_i) \quad u_i^t = 0 \]  
\[ \forall t \quad \sum_i u_i^t \leq C \]  
\[ \forall i \quad \sum_t u_i^t = p_i c_i \]  
\[ \forall i \forall t \in [r_i, d_i) \quad \sum_{x < t} u_i^t \leq c_i(t - r_i) \]  
\[ \forall i \forall t \in [r_i, d_i) \quad \sum_{x \geq t} u_i^t \leq c_i(d_i - t) \]

Properties of the partially elastic relaxation

- Equivalent to the \texttt{SUBSETSUM} bound (Perregaard95)
- Equivalent to the preemptive energetic reasoning (LopezEtAl92; Lopez, 1991)
- Algorithm in \( O(n^2 \log n) \) (Baptiste, 1998)
The relaxation gives a *satisfiability test* but not a *propagator*.

**Decomposition (Carlier, 1982):**

\[
\forall \Omega \subseteq T \quad s_\Omega = \min(\{s_j | j \in \Omega\}) \\
\forall \Omega \subseteq T \quad e_\Omega = \max(\{e_j | j \in \Omega\}) \\
\forall \Omega \subseteq T \quad w_\Omega \leq C(e_\Omega - s_\Omega)
\]

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.
Channel to precedence variables (Carlier and Pinson, 1989), (Applegate and Cook, 1991)

- adding \( i \) to \( \Omega \) not last leads to overload \( \implies \) \( i \) is last

\[
\forall \Omega \subseteq \mathcal{T} \quad s_\Omega = \text{mjn}(\{s_j \mid j \in \Omega\})
\]
\[
\forall \Omega \subseteq \mathcal{T} \quad e_\Omega = \text{max}(\{e_j \mid j \in \Omega\})
\]
\[
\forall \Omega \subseteq \mathcal{T} \quad s_\Omega + p_\Omega \leq e_\Omega
\]
\[
\forall \Omega \subseteq \mathcal{T}, \forall i \notin \Omega \quad b_{\Omega i} \iff \bigwedge_{j \in \Omega} e_j \leq s_i
\]
\[
\forall \Omega \subseteq \mathcal{T}, \forall i \notin \Omega \quad s_{\Omega \cup \{i\}} + p_i + p_\Omega > e_\Omega \implies b_{\Omega i}
\]
Edge Finding decomposition (single-machine)

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

\[
\forall \Omega \subseteq \mathcal{T} \quad s_\Omega = m\text{jn}(\{s_j \mid j \in \Omega\})
\]
\[
\forall \Omega \subseteq \mathcal{T} \quad e_\Omega = m\text{ax}(\{e_j \mid j \in \Omega\})
\]
\[
\forall \Omega \subseteq \mathcal{T} \quad s_\Omega + p_\Omega \leq e_\Omega
\]
\[
\forall \Omega \subseteq \mathcal{T}, \, \forall i \notin \Omega \quad b_{\Omega i} \iff \bigwedge_{j \in \Omega} e_j \leq s_i
\]
\[
\forall \Omega \subseteq \mathcal{T}, \, \forall i \notin \Omega \quad s_{\Omega \cup \{i\}} + p_i + p_\Omega > e_\Omega \implies b_{\Omega i}
\]
Edge Finding decomposition (single-machine)

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

\[
\begin{align*}
\forall \Omega \subseteq T & \quad s_\Omega = m\text{j}n(\{s_j \mid j \in \Omega\}) \\
\forall \Omega \subseteq T & \quad e_\Omega = \max(\{e_j \mid j \in \Omega\}) \\
\forall \Omega \subseteq T & \quad s_\Omega + p_\Omega \leq e_\Omega \\
\forall \Omega \subseteq T, \forall i \notin \Omega & \quad b_{\Omega i} \iff \bigwedge_{j \in \Omega} e_j \leq s_i \\
\forall \Omega \subseteq T, \forall i \notin \Omega & \quad s_{\Omega \cup \{i\}} + p_i + p_\Omega > e_\Omega \implies b_{\Omega i}
\end{align*}
\]
Not first / Not last decomposition
(single-machine)

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

\[
\begin{align*}
\forall \Omega \subseteq \mathcal{T} & \quad s_\Omega = \text{mjn}(\{s_j \mid j \in \Omega\}) \\
\forall \Omega \subseteq \mathcal{T} & \quad e_\Omega = \max(\{e_j \mid j \in \Omega\}) \\
\forall \Omega \subseteq \mathcal{T} & \quad s_\Omega + p_\Omega \leq e_\Omega \\
\forall \Omega \subseteq \mathcal{T}, \ \forall i \notin \Omega & \quad b_{\Omega i} \iff \bigwedge_{j \in \Omega} e_j \leq s_i \\
\forall \Omega \subseteq \mathcal{T}, \ \forall i \notin \Omega & \quad s_{\Omega \cup \{i\}} + p_i + p_\Omega > e_\Omega \implies b_{\Omega i} \\
\forall \Omega \subseteq \mathcal{T}, \ \forall i \notin \Omega & \quad e_i - s_\Omega < p_\Omega + p_i \implies \neg b_{\Omega i}
\end{align*}
\]
Not first / Not last decomposition (single-machine)

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

\[\forall \Omega \subseteq \mathcal{T} \quad s_{\Omega} = mjn(\{s_j \mid j \in \Omega\})\]
\[\forall \Omega \subseteq \mathcal{T} \quad e_{\Omega} = \max(\{e_j \mid j \in \Omega\})\]
\[\forall \Omega \subseteq \mathcal{T} \quad s_{\Omega} + p_{\Omega} \leq e_{\Omega}\]
\[\forall \Omega \subseteq \mathcal{T}, \forall i \notin \Omega \quad b_{\Omega i} \iff \bigwedge_{j \in \Omega} e_j \leq s_i\]
\[\forall \Omega \subseteq \mathcal{T}, \forall i \notin \Omega \quad s_{\Omega \cup \{i\}} + p_i + p_{\Omega} > e_{\Omega} \implies b_{\Omega i}\]
\[\forall \Omega \subseteq \mathcal{T}, \forall i \notin \Omega \quad e_i - s_{\Omega} < p_{\Omega} + p_i \implies \neg b_{\Omega i}\]
Not first / Not last decomposition (single-machine)

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

\[
\begin{align*}
\forall \Omega \subseteq \mathcal{T} & \quad s_\Omega = \operatorname{mjn}(\{s_j \mid j \in \Omega\}) \\
\forall \Omega \subseteq \mathcal{T} & \quad e_\Omega = \max(\{e_j \mid j \in \Omega\}) \\
\forall \Omega \subseteq \mathcal{T} & \quad s_\Omega + p_\Omega \leq e_\Omega \\
\forall \Omega \subseteq \mathcal{T}, \forall i \notin \Omega & \quad b_{\Omega i} \iff \bigwedge_{j \in \Omega} e_j \leq s_i \\
\forall \Omega \subseteq \mathcal{T}, \forall i \notin \Omega & \quad s_{\Omega \cup \{i\}} + p_i + p_\Omega > e_\Omega \implies b_{\Omega i} \\
\forall \Omega \subseteq \mathcal{T}, \forall i \notin \Omega & \quad e_i - s_\Omega < p_\Omega + p_i \implies \neg b_{\Omega i}
\end{align*}
\]
Overload Checking algorithm (Vilím, Barták, and Čepek, 2004)

Definitions

\[ ect_{\Omega} = \max\{r_{\Omega'} + p_{\Omega'} \mid \Omega' \subseteq \Omega\} \]

\[ T|_j = \{i \in T \mid d_i \leq d_j\} \]

Reformulation

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

May not seem like a big progress

- Check \(2^{|T|}\) relations
- Check \(|T|\) relations, each requiring to compute the max of \(2^{|T|}\) elements
Overload Checking – Dynamic Programming

\[ \text{ect}_\Omega = \max\{\text{ect}_{\Omega_L} + p_{\Omega_R}, \text{ect}_{\Omega_R}\} \]

With \( \Omega_L, \Omega_R \) two disjoint sets s.t. \( \max\{r_i \mid i \in \Omega_L\} \leq \min\{r_i \mid i \in \Omega_R\} \)

---

Energetic reasoning
Overload Checking – Dynamic Programming

\[ \text{ect}_\Omega = \max\{\text{ect}_{\Omega_L} + p_{\Omega_R}, \text{ect}_{\Omega_R}\} \]

- With \( \Omega_L, \Omega_R \) 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 \( \text{ect}_\Omega = r_{\Omega'} + p_{\Omega'} \)
Overload Checking – Dynamic Programming

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

With \( \Omega_L, \, \Omega_R \) 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 \( \Omega_L, \Omega_R \) 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 \( \Omega_L, \Omega_R \) 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 \( \Omega' \nsubseteq \Omega_R \) and \( ect_\Omega = r_{\Omega' \cap \Omega_L} + p_{\Omega'} \)

Energetic reasoning
Overload Checking – Dynamic Programming

\[ \text{ect}_\Omega = \max\{ \text{ect}_{\Omega_L} + p_{\Omega_R}, \text{ect}_{\Omega_R} \} \]

- With \( \Omega_L, \Omega_R \) 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 \( \text{ect}_\Omega = r_{\Omega'} + p_{\Omega'} \)

- Either \( \Omega' \subseteq \Omega_R \) and therefore \( \text{ect}_\Omega = \text{ect}_{\Omega'} = \text{ect}_{\Omega_R} \)
- Or \( \Omega' \not\subseteq \Omega_R \) and \( \text{ect}_\Omega = r_{\Omega' \cap \Omega_L} + p_{\Omega' \cap \Omega_L} + p_{\Omega' \cap \Omega_R} \)
Overload Checking – Dynamic Programming

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

- With \( \Omega_L, \Omega_R \) 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

\[ \text{ect}_\Omega = \max\{ \text{ect}_{\Omega_L} + p_{\Omega_R}, \text{ect}_{\Omega_R} \} \]

- With \( \Omega_L, \Omega_R \) 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 \( \text{ect}_\Omega = r_{\Omega'} + p_{\Omega'} \)

- Either \( \Omega' \subseteq \Omega_R \) and therefore \( \text{ect}_\Omega = \text{ect}_{\Omega'} = \text{ect}_{\Omega_R} \)
- Or \( \Omega' \not\subseteq \Omega_R \) and \( \text{ect}_\Omega = \text{ect}_{\Omega' \cap \Omega_L} + p_{\Omega_R} \)
Overload Checking – Dynamic Programming

$$\text{ect}_\Omega = \max\{\text{ect}_{\Omega_L} + p_{\Omega_R}, \text{ect}_{\Omega_R}\}$$

With $\Omega_L, \Omega_R$ 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 $\text{ect}_\Omega = r_{\Omega'} + p_{\Omega'}$

- Either $\Omega' \subseteq \Omega_R$ and therefore $\text{ect}_\Omega = \text{ect}_{\Omega'} = \text{ect}_{\Omega_R}$
- Or $\Omega' \nsubseteq \Omega_R$ and $\text{ect}_\Omega = \text{ect}_{\Omega_L} + p_{\Omega_R}$
Overload Checking – Theta Tree

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

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 $ect_{T|_j}$) requiring another order
Theta Tree

![Theta Tree Diagram]

Energetic reasoning
Theta Tree

```
Θ Tree
```

dur: 7
est: 9

```
A
```
dur: 3
est: 5

```
B
```
dur: 4
est: 8

```
C
```
dur: 3
est: 5

```
E
```
dur: 4
est: 8

```
B
```
dur: 4
est: 8

```
F
```
dur: 4
est: 8

```
D
```
dur: 4
est: 8

```
```
```
Energetic reasoning
```

```
LAAS-CNRS
Laboratoire d'analyse et d'architecture des systèmes du CNRS
```

32 / 67
Theta Tree

```
0  4  8  12  16  20
A  B  C  D  E  F
```

```
B
C
D
```

```
dur: 11
est: 13
```

```
dur: 7
est: 9
```

```
dur: 3
est: 5
```

```
dur: 4
est: 8
```

```
dur: 4
est: 11
```

```
dur: 4
est: 11
```

```
A
C
E
B
D
F
```

```
dur: 11
est: 13
```

```
dur: 7
est: 9
```

```
dur: 3
est: 5
```

```
dur: 4
est: 8
```

```
dur: 4
est: 11
```

```
dur: 4
est: 11
```

```
A
C
E
B
D
F
```

```
dur: 3
est: 5
```

```
dur: 4
est: 8
```

```
dur: 4
est: 11
```

```
dur: 4
est: 11
```

```
A
C
E
B
D
F
```

```
dur: 3
est: 5
```

```
dur: 4
est: 8
```

```
dur: 4
est: 11
```

```
dur: 4
est: 11
```

```
Energetic reasoning
```

```
dur: 11
est: 13
```

```
dur: 7
est: 9
```

```
dur: 3
est: 5
```

```
dur: 4
est: 8
```

```
dur: 4
est: 11
```

```
dur: 4
est: 11
```

```
A
C
E
B
D
F
```

```
dur: 3
est: 5
```

```
dur: 4
est: 8
```

```
dur: 4
est: 11
```

```
dur: 4
est: 11
```

```
Energetic reasoning
```

```
dur: 11
est: 13
```

```
dur: 7
est: 9
```

```
dur: 3
est: 5
```

```
dur: 4
est: 8
```

```
dur: 4
est: 11
```

```
dur: 4
est: 11
```

```
A
C
E
B
D
F
```

```
dur: 3
est: 5
```

```
dur: 4
est: 8
```

```
dur: 4
est: 11
```

```
dur: 4
est: 11
```

```
Energetic reasoning
```

```
dur: 11
est: 13
```

```
dur: 7
est: 9
```

```
dur: 3
est: 5
```

```
dur: 4
est: 8
```

```
dur: 4
est: 11
```

```
dur: 4
est: 11
```

```
A
C
E
B
D
F
```

```
dur: 3
est: 5
```

```
dur: 4
est: 8
```

```
dur: 4
est: 11
```

```
dur: 4
est: 11
```

```
Energetic reasoning
```

```
dur: 11
est: 13
```

```
dur: 7
est: 9
```

```
dur: 3
est: 5
```

```
dur: 4
est: 8
```

```
dur: 4
est: 11
```

```
dur: 4
est: 11
```

```
A
C
E
B
D
F
```

```
dur: 3
est: 5
```

```
dur: 4
est: 8
```

```
dur: 4
est: 11
```

```
dur: 4
est: 11
```

```
Energetic reasoning
```

```
dur: 11
est: 13
```

```
dur: 7
est: 9
```

```
dur: 3
est: 5
```

```
dur: 4
est: 8
```

```
dur: 4
est: 11
```

```
dur: 4
est: 11
```

```
A
C
E
B
D
F
```

```
dur: 3
est: 5
```

```
dur: 4
est: 8
```

```
dur: 4
est: 11
```

```
dur: 4
est: 11
```

```
Energetic reasoning
```
Theta Tree

Energetic reasoning
Theta Tree

Energetic reasoning
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

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

- Theta-tree-based algorithm in $O(n \log n)$ (Vilím, 2004)

- Energetic reasoning
Single-machine algorithms

**Edge finding**

- $O(n \log n)$ algorithm based on Jackson Preemptive Schedule \cite{carlier1994}
- $O(n^2)$ algorithm easier to implement / better in practice \cite{martin1996} and \cite{baptiste2001}
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 \cite{CarlierPinson1994}
- $O(n^2)$ algorithm easier to implement / better in practice \cite{MartinShmoys1996, BaptisteLePapeNuijten2001}
- Simpler algorithm in $O(n \log n)$ \cite{VilimBartakCepk2004}
  - Overload checking with theta tree, then mark the tasks as “maybe” one at a time

## Not first / Not last
- $O(n^2)$ algorithm \cite{LePapeBaptiste1996}
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 $\Omega$ if we add $i$ [not] first/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 $\Omega$ if we add $i$ [not] first/last $O(n \log n)$ (Vilím, Barták, and Čepek, 2004)

BC(Not first / Not last) $\leftrightarrow$ incomparable $\rightarrow$ BC(Edge Finding)

BC(Disjunctive) $\leftrightarrow$ stronger

BC(Time-tabling) $\leftrightarrow$ stronger
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

\[ \forall \Omega \subseteq T \quad w_\Omega \leq C (e_\Omega - s_\Omega) \]
Cumulative Edge Finding *(Nuijten and Aarts, 1994)*

\[
\forall \Omega \subseteq T \quad w_\Omega \leq C(e_\Omega - s_\Omega)
\]
Cumulative Edge Finding \((\text{Nuijten and Aarts, 1994})\)

\[ \forall \Omega \subseteq T \quad w_\Omega \leq C(e_\Omega - s_\Omega) \]

\[ w_\Omega \cup \{i\} \]
Cumulative Edge Finding (Nuijten and Aarts, 1994)

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

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

\[ \forall \Omega \subseteq T, \forall i \notin \Omega \quad w_{\Omega \cup \{i\}} > C(e_{\Omega} - s_{\Omega \cup \{i\}}) \implies \Omega \prec i \]
Cumulative Edge Finding (Nuijten and Aarts, 1994)

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

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

$$\forall \Omega \subseteq T, \forall i \notin \Omega \quad w_{\Omega \cup \{i\}} > C(e_{\Omega} - s_{\Omega \cup \{i\}}) \quad \Rightarrow \quad \Omega \prec i$$
Cumulative Edge Finding (Nuijten and Aarts, 1994)

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

\[
\forall \Omega \subseteq \mathcal{T} \quad w_\Omega \leq C(e_\Omega - s_\Omega)
\]
\[
\forall \Omega \subseteq \mathcal{T}, \forall c \quad R(\Omega, c) = w_\Omega - (C - c)(e_\Omega - s_\Omega)
\]
\[
\forall \Omega \subseteq \mathcal{T}, \forall i \notin \Omega \quad w_{\Omega \cup \{i\}} > C(e_\Omega - s_{\Omega \cup \{i\}}) \quad \Rightarrow \quad \Omega \prec i
\]
Cumulative Edge Finding (Nuijten and Aarts, 1994)

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

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

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

$$
\forall \Omega \subseteq \mathcal{T}, \forall i \notin \Omega, \forall \Omega' \in \Omega \\
w_{\Omega \cup \{i\}} > C(e_\Omega - s_{\Omega \cup \{i\}}) \implies s_i \geq s_\Omega + \frac{1}{c_i} R(\Omega, c_i)
$$
Cumulative Edge Finding (Nuijten and Aarts, 1994)

If energy of $\Omega$ and $i$ exceeds capacity when $i$ is not last

- then $i$ has to be last but not necessarily follows every task in $\Omega$!

\[
\forall \Omega \subseteq \mathcal{T} \\
\forall \Omega \subseteq \mathcal{T}, \forall c \\
\forall \Omega \subseteq \mathcal{T}, \forall i \notin \Omega, \forall \Omega' \in \Omega \\
w_{\Omega} = C(e_{\Omega} - s_{\Omega}) \\
R(\Omega, c) = w_{\Omega} - (C - c)(e_{\Omega} - s_{\Omega}) \\
w_{\Omega \cup \{i\}} > C(e_{\Omega} - s_{\Omega \cup \{i\}}) \implies s_i \geq s_{\Omega'} + \frac{1}{c_i} R(\Omega', c_i)
\]
Cumulative Edge Finding \textit{(Nuijten and Aarts, 1994)}

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

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

\[ R(\Omega, c) = w_{\Omega} - (C - c)(e_\Omega - s_\Omega) \]

\[ \forall \Omega \subseteq T, \forall i \notin \Omega, \forall \Omega' \in \Omega \quad w_{\Omega \cup \{i\}} > C(e_\Omega - s_{\Omega \cup \{i\}}) \implies s_i \geq s_{\Omega'} + \frac{1}{c_i} R(\Omega', c_i) \]
C. Extended Edge Finding (Nuijten and Aarts, 1994)

\[ \forall \Omega \subseteq T, \forall i \notin \Omega, \forall \Omega' \in \Omega \]

\[ w_\Omega \leq C(e_\Omega - s_\Omega) \]

\[ R(\Omega, c) = w_\Omega - (C - c)(e_\Omega - s_\Omega) \]

\[ w_{\Omega \cup \{i\}} > C(e_\Omega - s_{\Omega \cup \{i\}}) \implies s_i \geq s_{\Omega'} + \frac{1}{c_i} R(\Omega', c_i) \]
C. Extended Edge Finding (Nuijten and Aarts, 1994)

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

\[
\forall \Omega \subseteq T, \forall c \quad w_\Omega \leq C(e_\Omega - s_\Omega)
\]
\[
R(\Omega, c) = w_\Omega - (C - c)(e_\Omega - s_\Omega)
\]
\[
\forall \Omega \subseteq T, \forall i \notin \Omega, \forall \Omega' \in \Omega \quad w_{\Omega \cup \{i\}} > C(e_\Omega - s_{\Omega \cup \{i\}}) \implies s_i \geq s_{\Omega'} + \frac{1}{c_i}R(\Omega', c_i)
\]
C. Extended Edge Finding \textbf{(Nuijten and Aarts, 1994)}

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

\[\forall \Omega \subseteq \mathcal{T}, \forall c\]
\[w_\Omega \leq C(e_\Omega - s_\Omega)\]
\[R(\Omega, c) = w_\Omega - (C - c)(e_\Omega - s_\Omega)\]
\[w_{\Omega \cup \{i\}} > C(e_\Omega - s_{\Omega \cup \{i\}}) \implies s_i \geq s_{\Omega'} + \frac{1}{c_i}R(\Omega', c_i)\]
C. Extended Edge Finding (Nuijten and Aarts, 1994)

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

\[
\forall \Omega \subseteq T, \forall i \not\in \Omega, \forall \Omega' \in \Omega
\]

\[
\forall \Omega \subseteq T, \forall \Omega' \in \Omega
\]

\[
w_\Omega \leq C(e_\Omega - s_\Omega)
\]

\[
R(\Omega, c) = w_\Omega - (C - c)(e_\Omega - s_\Omega)
\]

\[
w_{\Omega \cup \{i\}} > C(e_\Omega - s_{\Omega \cup \{i\}}) \implies s_i \geq s_{\Omega'} + \frac{1}{c_i} R(\Omega', c_i)
\]

\[
w_{\Omega} + (e_i - s_\Omega)c_i > C(e_\Omega - s_\Omega) \implies s_i \geq s_{\Omega'} + \frac{1}{c_i} R(\Omega', c_i)
\]
C. Extended Edge Finding \textit{(Nuijten and Aarts, 1994)}

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

\[
\forall \Omega \subseteq T, \forall c \quad w_\Omega \leq C(e_\Omega - s_\Omega) \\
R(\Omega, c) = w_\Omega - (C - c)(e_\Omega - s_\Omega) \\
w_{\Omega \cup \{i\}} > C(e_\Omega - s_{\Omega \cup \{i\}}) \implies s_i \geq s_{\Omega'} + \frac{1}{c_i}R(\Omega', c_i) \\
w_\Omega + (e_i - s_\Omega)c_i > C(e_\Omega - s_\Omega) \implies s_i \geq s_{\Omega'} + \frac{1}{c_i}R(\Omega', c_i)
\]

\[
R(\Omega, c_i) = w_\Omega - (C - c_i)(e_\Omega - s_\Omega) \\
(C - c_i)(e_\Omega - s_\Omega)
\]
C. Extended Edge Finding \cite{NuijtenAarts1994}

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

\[
\begin{align*}
\forall \Omega \subseteq T, \forall c & \quad w_\Omega \leq C(e_\Omega - s_\Omega) \\
\forall \Omega \subseteq T, \forall i \notin \Omega, \forall \Omega' \in \Omega & \quad R(\Omega, c) = w_\Omega - (C - c)(e_\Omega - s_\Omega) \\
\forall \Omega \subseteq T, \forall i \notin \Omega, \forall \Omega' \in \Omega & \quad w_{\Omega \cup \{i\}} > C(e_{\Omega \cup \{i\}} - s_{\Omega \cup \{i\}}) \implies s_i \geq s_{\Omega'} + \frac{1}{c_i} R(\Omega', c_i) \\
\forall \Omega \subseteq T, \forall i \notin \Omega, \forall \Omega' \in \Omega & \quad w_\Omega + (e_i - s_\Omega)c_i > C(e_\Omega - s_\Omega) \implies s_i \geq s_{\Omega'} + \frac{1}{c_i} R(\Omega', c_i)
\end{align*}
\]
C. Extended Edge Finding (Nuijten and Aarts, 1994)

- Case where \( i \) may start before \( \Omega \) but cannot be completed before \( \Omega \) 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)

\[
\forall \Omega \subseteq T \quad w_\Omega \leq C(e_\Omega - s_\Omega)
\]

\[
R(\Omega, c) = w_\Omega - (C - c)(e_\Omega - s_\Omega)
\]

\[
\forall \Omega \subseteq T, \forall \Omega', \forall i \notin \Omega, \Omega' \in \Omega
\]

\[
w_{\Omega \cup \{i\}} > C(e_\Omega - s_{\Omega \cup \{i\}}} \implies s_i \geq s_\Omega' + \frac{1}{c_i} R(\Omega', c_i)
\]

\[
\forall \Omega \subseteq T, \forall i \notin \Omega, \forall \Omega' \in \Omega
\]

\[
w_\Omega + (e_i - s_\Omega) c_i > C(e_\Omega - s_\Omega) \implies s_i \geq s_\Omega' + \frac{1}{c_i} R(\Omega', c_i)
\]
Time-tabling and (Extended) Edge finding are incomparable when $C > 1$
Time-tabling and (Extended) Edge finding are incomparable when \( C > 1 \)
Time-tabling and (Extended) Edge finding are incomparable when $C > 1$

$$20 + 4 + 4 \leq 3 \times 12 = 36$$
Time-tabling and (Extended) Edge finding are incomparable when \( C > 1 \)

\[
10 + 4 + 4 \leq 3 \times 7 = 21
\]
Time-tabling and (Extended) Edge finding are incomparable when $C > 1$

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

- Combine them! $3 \times 8 + 4 + 5 > 4 \times 8$
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)

\[
\forall \Omega \subseteq T \quad w_\Omega \leq C(e_\Omega - s_\Omega)
\]

\[
R(\Omega, c) = w_\Omega - (C - c)(e_\Omega - s_\Omega)
\]

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

\[
\forall \Omega \subseteq T, \forall i \not\in \Omega, \forall \Omega' \in \Omega \quad w_\Omega + (e_i - s_\Omega)c_i > C(e_\Omega - s_\Omega) \implies s_i \geq s_{\Omega'} + \frac{1}{c_i} R(\Omega', c_i)
\]
Time-tabling Edge Finding decomposition

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

\[
\begin{align*}
\forall \Omega \subseteq T, \quad w_{\Omega} &\leq C(e_{\Omega} - s_{\Omega}) - \sum_{t=s_{\Omega}}^{e_{\Omega}} \sum_{i \not\in \Omega} c_i a_i^t \\
R(\Omega, c) &= w_{\Omega} - (C - c)(e_{\Omega} - s_{\Omega}) \\
\forall \Omega \subseteq T, \forall i \not\in \Omega, \forall \Omega' \in \Omega \\
\forall i \forall t \\
\forall \Omega' \in \Omega, w_{\Omega \cup \{i\}} > C(e_{\Omega} - s_{\Omega \cup \{i\}}) \implies s_i \geq s_{\Omega'} + \frac{1}{c_i} R(\Omega', c_i) \\
\forall \Omega \subseteq T, \forall i \not\in \Omega, \forall \Omega' \in \Omega \\
\forall i \forall t \\
\forall \Omega' \in \Omega, w_{\Omega} + (e_{i} - s_{\Omega})c_i > C(e_{\Omega} - s_{\Omega}) \implies s_i \geq s_{\Omega'} + \frac{1}{c_i} R(\Omega', c_i) \\
a_i^t \iff s_i \leq t \land t < e_i
\end{align*}
\]
Channel Time-tableing and (Extended) Edge finding decompositions

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

\[
\forall \Omega \subseteq T, \forall c \\
\forall \Omega \subseteq T, \forall i \notin \Omega, \forall \Omega' \in \Omega \\
\forall \Omega \subseteq T, \forall i \notin \Omega, \forall \Omega' \in \Omega \\
\forall i \forall t
\]

\[
\begin{align*}
\forall \Omega \subseteq T & \quad w_\Omega \leq C(e_\Omega - s_\Omega) - \sum_{t=s_\Omega}^{e_\Omega} \sum_{i \notin \Omega} c_i a_i^t \\
R(\Omega, c) &= w_\Omega - (C - c)(e_\Omega - s_\Omega) \\
w_{\Omega \cup \{i\}} > C(e_\Omega - s_{\Omega \cup \{i\}}) & \implies s_i \geq s_{\Omega'} + \frac{1}{c_i} R(\Omega', c_i) \\
w_\Omega + (e_i - s_\Omega)c_i > C(e_\Omega - s_\Omega) & \implies s_i \geq s_{\Omega'} + \frac{1}{c_i} R(\Omega', c_i) \\
\end{align*}
\]

\[
a_i^t \iff s_i \leq t \land t < e_i
\]
Time-tabling Edge Finding algorithm

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
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 \(\Rightarrow\) Energy of all tasks over a given interval

\[ E_{\text{set}} = \sum_{j \in \mathcal{T}} \text{energy}(j) \]

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 $\implies$ Energy of all tasks over a given interval

\[
\begin{align*}
E_j(a) &= \text{max}(0, p_j - \text{max}(0, a - s_j)) \\
p_j^-(b) &= \text{max}(0, p_j - \text{max}(0, e_j - b)) \\
w_j(a, b) &= c_j \min(b - a, p_j^+(a), p_j^-(b)) \\
W(a, b) &= \sum_{j \in T} w_j(a, b)
\end{align*}
\]
Energetic reasoning (Lopez, 1991)

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

\[ E_{\text{set}} = \sum_{j \in \mathcal{T}} E_j \]

**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 $\implies$ Energy of all tasks over a given interval

\[
\begin{align*}
0 & \quad 5 & \quad 10 & \quad 15 & \quad 20 & \quad 25 \\
A & \\
\hline
a=3 & \quad b=19
\end{align*}
\]

Notion of energy (over an interval)

- \( p_j^+(a) = \max(0, p_j - \max(0, a - s_j)) \)
- \( p_j^-(b) = \max(0, p_j - \max(0, e_j - b)) \)
- \( w_j(a, b) = c_j \min(b - a, p_j^+(a), p_j^-(b)) \)
- \( W(a, b) = \sum_{j \in T} w_j(a, b) \)
∀0 ≤ a < b < h \ W(a, b) ≤ C(b − a) \ (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{align*}
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 T \}
\end{align*}

(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)
∀0 ≤ a < b < h \ W(a, b) ≤ C(b − a) \ (ER)

Relevant intervals (Baptiste, 1998)

\[ O_1(i) = \{r_i, lst_i, ect_i\} \]
\[ O_2(i) = \{d_i, lst_i, ect_i\} \]
\[ O(i, t) = \{lst_i − t \mid i \in T\} \]

(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

\[ \forall 0 \leq a < b < h \quad W(a, b) \leq C(b - a) \]
∀0 ≤ a < b < h \ W(a, b) \leq C(b - a)

\[ W(3, 16) - w_i(3, 16) = 40 \]

- \( C = 3 \), maximum energy on \([3, 19]\) = 48 (\( W(3, 16) = 46, \ w_i(3, 16) = 6 \))
Energetic reasoning decomposition

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

\[
W(a, b) + p_i^+(a) - w_i(a, b) - C(b - a) > 0 \quad \Rightarrow \quad e_i \geq b + \frac{R}{c_i}
\]

\( R \) (at least after \( b \))

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

\[0 \leq a < b < h \quad W(a, b) \leq C(b - a)\]

\[W(a, b) + p_i^+(a) - w_i(a, b) - C(b - a) > 0 \implies e_i \geq b + \frac{R}{c_i}\]

\[\text{R (at least after } b)\]

\[\min(b - a, p_i^+(a)) - w_i(a, b) + C(b - a) - W(a, b) > 0 \implies s_i \geq b - \frac{R}{c_i}\]

\[\text{R (at most before } b)\]

\[a = 3 \quad \quad b = 19\]

\[C = 3, \text{ maximum energy on } [3, 19] = 48 \quad (W(3, 16) = 46, \quad w_i(3, 16) = 6)\]
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
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)
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 \( \rho \) 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 $\rho$ is the ratio between largest and smallest task size

Not all resource scheduling are hard!
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](image)
Precedence graph as “domain”

The image shows a precedence graph with nodes labeled O, SA, eA, SC, eC, SF, eF, SE, eE, SD, eD, and m. The graph depicts the flow of information or tasks, with directed edges indicating the sequence or precedence of these nodes. The edge labels indicate the time or step required to transition from one node to another.
Precedence graph as “domain”

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

Search on the precedence graph

- Variable $b_{ij}$ standing for $i < j$ for each pair of tasks $i, j$ sharing a resource
**Precedence search**

**Search on the precedence graph**

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

Search on the precedence graph

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

Search on the precedence graph

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

Precedence Graph
**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)

<table>
<thead>
<tr>
<th>Objective Proof</th>
<th>Objective Proof</th>
</tr>
</thead>
<tbody>
<tr>
<td>( C_{\text{max}} )</td>
<td>0.50%</td>
</tr>
<tr>
<td>Setup</td>
<td>0.00%</td>
</tr>
<tr>
<td>Lags</td>
<td>0.39%</td>
</tr>
<tr>
<td>( T_{\sum} )</td>
<td>1.59%</td>
</tr>
</tbody>
</table>

### 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

<table>
<thead>
<tr>
<th>Objective</th>
<th>Proof</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_{\text{max}}$</td>
<td>0.50%</td>
</tr>
<tr>
<td>Objective</td>
<td>Proof</td>
</tr>
<tr>
<td>0.03%</td>
<td>56%</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Objective</th>
<th>Proof</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_{max}$</td>
<td>0.50%</td>
</tr>
<tr>
<td>setup</td>
<td>0.00%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Objective</th>
<th>Proof</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.03%</td>
<td>56%</td>
</tr>
<tr>
<td>0.27%</td>
<td>4%</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th></th>
<th>Objective</th>
<th>Proof</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_{max}$</td>
<td>0.50%</td>
<td>75%</td>
</tr>
<tr>
<td>setup</td>
<td>0.00%</td>
<td>52%</td>
</tr>
<tr>
<td>lags</td>
<td>0.39%</td>
<td>64%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Objective</th>
<th>Proof</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.03%</td>
<td>56%</td>
</tr>
<tr>
<td></td>
<td>0.27%</td>
<td>4%</td>
</tr>
<tr>
<td></td>
<td>0.54%</td>
<td>22%</td>
</tr>
</tbody>
</table>

Precedence Graph
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

<table>
<thead>
<tr>
<th></th>
<th>Objective</th>
<th>Proof</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_{\text{max}}$</td>
<td>0.50%</td>
<td>75%</td>
</tr>
<tr>
<td>setup</td>
<td>0.00%</td>
<td>52%</td>
</tr>
<tr>
<td>lags</td>
<td>0.39%</td>
<td>64%</td>
</tr>
<tr>
<td>$T_{\sum}$</td>
<td>1.59%</td>
<td>73%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Objective</th>
<th>Proof</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_{\text{max}}$</td>
<td>0.03%</td>
<td>56%</td>
</tr>
<tr>
<td>setup</td>
<td>0.27%</td>
<td>4%</td>
</tr>
<tr>
<td>lags</td>
<td>0.54%</td>
<td>22%</td>
</tr>
<tr>
<td>$T_{\sum}$</td>
<td>2.52%</td>
<td>33%</td>
</tr>
</tbody>
</table>
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

[Diagram showing CDMS connected to Mass Memory and Exp. 1, Mem. 1, Exp. 2, Mem. 2, Exp. 3, Mem. 3.]
Command and Data Management Subsystem

CDMS

Mass Memory

Mem. 1

Exp. 1

Mem. 2

Exp. 2

Mem. 3

Exp. 3

τ₁

τ₂

τ₃

Priority

Other types of resource
Command and Data Management Subsystem

Mass Memory

CDMS

Exp. 1
Mem. 1

Exp. 2
Mem. 2

Exp. 3
Mem. 3

\( \tau_1 \)

\( \tau_2 \)

\( \tau_3 \)

prio

visibility

Other types of resource
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

![Diagram showing exp1 and exp2 with production rate and transfer rate]

Other types of resource
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
Simulating the Transfers

**Original implementation**

IlcReservoir constraints and optional transfer tasks

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

- 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

Memory

Time

\( \tau_1 \)

\( \text{exp 1} \)

\( \text{exp 2} \)

\( \text{exp 3} \)

Other types of resource
Checking the Constraint

- exp 1
- exp 2
- exp 3

- $\tau_1$
- $\tau_3$

Other types of resource
Checking the Constraint

memory

\( \tau_1 \)
\( \tau_3 \)
exp 1
exp 2
exp 3

\( \text{time} \)
Checking the Constraint

![Time vs. Memory Graph]

- $\tau_1$
- $\tau_3$
- exp 1
- exp 2
- exp 3
Checking the Constraint

memory

time

$\tau_1$

$\tau_3$

exp 1

exp 2

exp 3

Other types of resource
Checking the Constraint

memory

time

\( \tau_1 \)

\( \tau_3 \)

exp 1

exp 2

exp 3

Other types of resource
Checking the Constraint

memory vs. time

\( \tau_1 \)  
\( \tau_2 \)  
\( \tau_3 \)  
exp 1  
exp 2  
exp 3
Checking the Constraint

\[ \tau_1, \tau_2, \tau_3 \]

memory vs. time

exp 1, exp 2, exp 3
Checking the Constraint

memory

time

$\tau_1$

$\tau_2$

$\tau_3$

exp 1

exp 2

exp 3

Other types of resource
Checking the Constraint

memory

time

\[ \tau_1 \]
\[ \tau_2 \]
\[ \tau_3 \]

exp 1
exp 2
exp 3
Checking the Constraint

-exp 1
-exp 2
-exp 3

\(\tau_1\)
\(\tau_2\)
\(\tau_3\)

memory

\(\rightarrow\) time
Checking the Constraint
Checking the Constraint

memory

\(\tau_1\)
\(\tau_2\)
\(\tau_3\)
exp 1
exp 2
exp 3

Other types of resource
Checking the Constraint

memory

time

$\tau_1$

$\tau_2$

$\tau_3$

exp 1

exp 2

exp 3

Other types of resource
Checking the Constraint

\[ \text{memory} \]

\[ \text{time} \]

\[ \tau_1 \]
\[ \tau_2 \]
\[ \tau_3 \]

exp 1
exp 2
exp 3

Other types of resource
Checking the Constraint

memory


time

\( \tau_1 \)
\( \tau_2 \)
\( \tau_3 \)

exp 1
exp 2
exp 3

Other types of resource
Checking the Constraint

Other types of resource
Checking the Constraint

memory

\[ \tau_1 \]

\[ \tau_2 \]

\[ \tau_3 \]

exp 1

exp 2

exp 3

Other types of resource
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 $1 + \frac{\tau_{\text{max}}}{\tau_{\text{min}}} \simeq 3$ 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 $1 + \frac{\tau_{\text{max}}}{\tau_{\text{min}}} \simeq 3$ blocks
- Faster and more accurate than the reservoir/transfer tasks model
The constraint can be checked in $O(n \log n)$

Several experiments active simultaneously:

- Error is less than or equal to $1 + \frac{\tau_{\text{max}}}{\tau_{\text{min}}} \approx 3$ 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.

![Diagram showing memory over time with time intervals](image)

- Other types of resource
Propagation: production/transfer rate

For a set of tasks

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

\[ t_1 < t_2 \]
Propagation: production/transfer rate

For a set of tasks

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

\[ t_1 \leq t_2 - (t_2 - t_1) \geq (\pi_1 p_1 + \pi_2 p_2) - M_k \tau \]

Other types of resource
Propagation: production/transfer rate

For a set of tasks

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

\[ \text{time} \]

\[ \text{memory} \]

\[ M_k \]

\[ t_1 \]

\[ t_2 \]
Propagation: production/transfer rate

For a set of tasks

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

\[
\text{end}(t_2) - \text{start}(t_1) \geq \frac{(\pi_1 p_1 + \pi_2 p_2) - M_k}{\tau}
\]
Propagation: production/transfer rate

More generally, we consider a set of tasks $\Omega$ of a given experiment
More generally, we consider a set of tasks $\Omega$ of a given experiment

Let $[a, b]$ be a time interval \textit{necessarily} contained in $\Omega$’s transfer period
Propagation: production/transfer rate

More generally, we consider a set of tasks $\Omega$ of a given experiment

- Let $[a, b]$ be a time interval necessarily contained in $\Omega$’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 $\Omega$ of a given experiment

- Let $[a, b]$ be a time interval necessarily contained in $\Omega$’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 $\Omega$ of a given experiment

- Let $[a, b]$ be a time interval necessarily contained in $\Omega$’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:
  \[
  \text{Makespan}(\Omega) \geq \frac{(\sum_{t_{ki} \in \Omega} \pi_i p_i) - M_k}{\tau} + T_k(a, b)
  \]
Propagation: visibility

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

\[
\text{Minimal usage and peak: } m_{\text{max}}
\]

\[
\text{If the production exceeds } M_{\text{k}} + M_{0} - m_{\text{max}}_{0} \text{ in } [a, b] \text{ then data will be lost}
\]
Propagation: visibility

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

![Graph showing the visibility of data with a threshold at $m_0^{\text{max}}$ between points $a$ and $b$.]
Propagation: visibility

- When the mass memory is full, no more data is transferred to it.
- Minimal usage and peak $m_{0}^{\text{max}}$.
- If the production exceeds $M_{k} + M_{0} - m_{0}^{\text{max}}$ 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 $m_0^{\text{max}}$
- If the production exceeds $M_k + M_0 - m_0^{\text{max}}$ in $[a, b]$ then data will be lost
  - Filtering: bound start time w.r.t. this quantity of data and production rate
Conclusions

Extremely successful application of constraint programming

Still a lot to achieve (better algorithms, different resources, objectives, search, ...)

Hybrid approaches (CP & MIP, CP & SAT, ...)

Other types of resource
Conclusions

- Extremely successful application of constraint programming
Conclusions

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

Other types of resource
Conclusions

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


