Constraint-based Scheduling: Propagation

Hana Rudová*, Rong Qu’

*Fakulty of Informatics, Masaryk University
Brno, Czech Republic
http://www.fi.muni.cz/~hanka

’School of Computer Science
University of Nottingham, UK
http://www.cs.nott.ac.uk/~rxq
Outline

1. Unary resources
2. Cumulative resources
3. Alternative resources
4. Summary
Notation

- \( \text{est}(A) \) earliest start time of activity \( A \)
- \( \text{ect}(A) \) earliest completion time of activity \( A \)
- \( \text{lst}(A) \) latest start time of activity \( A \)
- \( \text{lct}(A) \) latest completion time of activity \( A \)

- \( \Omega \) is the set of activities
- \( p(\Omega) = \sum_{A \in \Omega} p(A) \)
- \( \text{est}(\Omega) = \min\{\text{est}(A) \mid A \in \Omega\} \)
- \( \text{lct}(\Omega) = \max\{\text{lct}(A) \mid A \in \Omega\} \)
Edge finding: example

What happens if activity A is not processed first?

![Diagram showing activities A, B, and C with constraints and their start and end times.]

- A: Start 4, End 16
- B: Start 6, End 16
- C: Start 7, End 15
What happens if activity A is not processed first?

Not enough time for A, B, and C and thus A must be first!
Edge finding: example with filtering rules

- \( p(\Omega \cup \{A\}) > lct(\Omega \cup \{A\}) - est(\Omega) \Rightarrow A << \Omega \)
Edge finding: example with filtering rules

- \( p(\Omega \cup \{A\}) > lct(\Omega \cup \{A\}) - est(\Omega) \Rightarrow A << \Omega \)

- \( A << \Omega \Rightarrow end(A) \leq \min\{lct(\Omega') - p(\Omega') | \Omega' \subseteq \Omega\} \)

Diagram:
- \( A(2) \rightarrow 7 \rightarrow A(16) \)
- \( B(4) \rightarrow 6 \rightarrow 16 \)
- \( C(5) \rightarrow 7 \rightarrow 15 \)
Edge finding: all filtering rules

- **Edge-finding rules**
  - \( p(\Omega \cup \{A\}) > lct(\Omega \cup \{A\}) - est(\Omega) \)
    \[ \Rightarrow A << \Omega \]
  - \( A << \Omega \Rightarrow \)
    \[ end(A) \leq \min\{lct(\Omega') - p(\Omega') \mid \Omega' \subseteq \Omega\} \]

- **Edge-finding (symmetrical) rules**
  - \( p(\Omega \cup \{A\}) > lct(\Omega) - est(\Omega \cup \{A\}) \)
    \[ \Rightarrow \Omega << A \]
  - \( \Omega << A \Rightarrow \)
    \[ start(A) \geq \max\{est(\Omega') + p(\Omega') \mid \Omega' \subseteq \Omega\} \]
Edge finding: all filtering rules

**Edge-finding rules**
- $p(\Omega \cup \{A\}) > lct(\Omega \cup \{A\}) - est(\Omega)$
  - $\Rightarrow A << \Omega$
- $A << \Omega \Rightarrow$
  - $end(A) \leq \min\{lct(\Omega') - p(\Omega') | \Omega' \subseteq \Omega\}$

**Edge-finding (symmetrical) rules**
- $p(\Omega \cup \{A\}) > lct(\Omega) - est(\Omega \cup \{A\})$
  - $\Rightarrow \Omega << A$
- $\Omega << A \Rightarrow$
  - $start(A) \geq \max\{est(\Omega') + p(\Omega') | \Omega' \subseteq \Omega\}$

**In practice:**
- there are $n \cdot 2^n$ pairs $(A, \Omega)$ to consider (too many!)
- instead of $\Omega$ use so called **task intervals** $[A, B]$
  - $\{C | est(A) \leq est(C) \land lct(C) \leq lct(B)\}$
  - time complexity $O(n^3)$, frequently used incremental algorithm
- there are also $O(n^2)$ and $O(n \log n)$ algorithms
What happens if activity A is processed first?

- Not-first/not-last: example

- Not enough time for B and C and thus A cannot be first!
Not-first/not-last: example

- What happens if activity A is processed first?

- Not enough time for B and C and thus A cannot be first!
Not-first/not-last: example with filtering rules

- \( p(\Omega \cup \{A\}) > lct(\Omega) - est(A) \Rightarrow \neg A << \Omega \)
Not-first/not-last: example with filtering rules

- \( p(\Omega \cup \{A\}) > lct(\Omega) - est(A) \Rightarrow \neg A << \Omega \)

- \( \neg A << \Omega \Rightarrow start(A) \geq \min\{ect(B) | B \in \Omega\} \)
Not-first/not-last: all filtering rules

- **Not-first rules:**
  - $p(\Omega \cup \{A\}) > lct(\Omega) - est(A)$
    \[ \Rightarrow \neg A \ll \Omega \]
  - $\neg A \ll \Omega$
    \[ \Rightarrow \text{start}(A) \geq \min\{\text{ect}(B) | B \in \Omega\} \]

- **Not-last (symmetrical) rules:**
  - $p(\Omega \cup \{A\}) > lct(A) - est(\Omega)$
    \[ \Rightarrow \neg \Omega \ll A \]
  - $\neg \Omega \ll A \Rightarrow$
    \[ \text{end}(A) \leq \max\{\text{lst}(B) | B \in \Omega\} \]
Not-first/not-last: all filtering rules

- **Not-first rules:**
  - \( p(\Omega \cup \{A\}) > lct(\Omega) - est(A) \)
    \( \Rightarrow \neg A << \Omega \)
  - \( \neg A << \Omega \)
    \( \Rightarrow \text{start}(A) \geq \min\{\text{ect}(B) | B \in \Omega\} \)

- **Not-last (symmetrical) rules:**
  - \( p(\Omega \cup \{A\}) > lct(A) - est(\Omega) \)
    \( \Rightarrow \neg \Omega << A \)
  - \( \neg \Omega << A \Rightarrow \text{end}(A) \leq \max\{\text{lst}(B) | B \in \Omega\} \)

- **In practice:**
  - can be implemented with
    time complexity \( O(n^2) \) and space complexity \( O(n) \)
Cumulative resources

- Each activity uses some capacity of the resource $\text{cap}(A)$
- Activities can be processed in parallel, if a resource capacity is not exceeded
- Resource capacity may vary in time
  - modeled via fix capacity over time and fixed activities consuming the resource until the requested capacity level is reached

![Diagram showing capacity profile over time with fixed activities and used capacity]
Aggregated demands

- Where is enough capacity for processing the activity?
Aggregated demands

- Where is enough capacity for processing the activity?

- How aggregated demand is constructed?
Discrete time is expected

How to ensure that capacity is not exceed at any time point?

\[
\forall t \sum_{\text{start}(A_i) \leq t \leq \text{end}(A_i)} \text{cap}(A_i) \leq \text{MaxCapacity}
\]
Timetable constraint

- Discrete time is expected
- How to ensure that capacity is not exceed at any time point?

\[
\forall t \sum_{\text{start}(A_i) \leq t \leq \text{end}(A_i)} \text{cap}(A_i) \leq \text{MaxCapacity}
\]

- **Timetable** for activity A is a set of Boolean domain variables \(X(A, t)\) indicating whether A is processed in time t

\[
\forall t \sum_{A_i} X(A_i, t) \times \text{cap}(A_i) \leq \text{MaxCapacity}
\]

\[
\forall t, i \ \text{start}(A_i) \leq t < \text{end}(A_i) \iff X(A_i, t)
\]
**Initial situation**

- **est(A)**
- **ect(A)**
- **lst(A)**
- **lct(A)**

The diagram shows a time interval divided into segments. The initial situation is represented by the shaded area, indicating a specific time period for the event A. The constraints are visualized with the boundaries and labels corresponding to the estimated start (est(A)), estimated completion (ect(A)), latest start (lst(A)), and latest completion (lct(A)) times. The shaded area indicates the feasible time slots for the event A, with the constraints ensuring that the event fits within the allowed time frame.
Timetable constraint: filtering example

Initial situation

Some positions forbidden due to capacity
Timetable constraint: filtering example

Initial situation

Some positions forbidden due to capacity

New situation
Timetable constraint: filtering rules

How to do filtering through the constraint

$$\forall t, i \hspace{1mm} \text{start}(A_i) \leq t < \text{end}(A_i) \iff X(A_i, t)$$

Problem: t serves as an index and as a variable

- start(A) $\geq \min\{t \mid 1 \in X(A,t)\}$
- end(A) $\leq 1 + \max\{t \mid 1 \in X(A,t)\}$
Timetable constraint: filtering rules

\[ \forall t, i \quad \text{start}(A_i) \leq t < \text{end}(A_i) \iff X(A_i, t) \ ? \]

Problem: \( t \) serves as an index and as a variable

- \( \text{start}(A) \geq \min\{t \mid 1 \in X(A,t)\} \)
- \( \text{end}(A) \leq 1 + \max\{t \mid 1 \in X(A,t)\} \)
- \( X(A,t) = 0 \land t < \text{ect}(A) \Rightarrow \text{start}(A) > t \)
- \( X(A,t) = 0 \land \text{lst}(A) \leq t \Rightarrow \text{end}(A) \leq t \)
Timetable constraint: filtering rules

How to do filtering through the constraint

$$\forall t, i \ start(A_i) \leq t < end(A_i) \Leftrightarrow X(A_i, t)$$

Problem: t serves as an index and as a variable

- start(A) $\geq \min \{t \mid 1 \in X(A,t)\}$
- end(A) $\leq 1 + \max \{t \mid 1 \in X(A,t)\}$

- $X(A,t) = 0 \land t < ect(A) \Rightarrow start(A) > t$
- $X(A,t) = 0 \land lst(A) \leq t \Rightarrow end(A) \leq t$
- $lst(A) \leq t \land t < ect(A) \Rightarrow X(A,t) = 1$
Alternative resources

How to model alternative resources for a given activity?

Use a **duplicate activity** for each resource

- duplicate activity participates in a respective resource constraint but does not restrict other activities there
  - "failure" means removing the resource from the domain of variable resource(A)
  - deleting the resource from the domain of variable resource(A) means "deleting" the respective duplicate activity
How to model alternative resources for a given activity?

Use a **duplicate activity** for each resource

- duplicate activity participates in a respective resource constraint but does not restrict other activities there
  - "failure" means removing the resource from the domain of variable resource(A)
  - deleting the resource from the domain of variable resource(A) means "deleting" the respective duplicate activity
- original activity participates in precedence constraints (e.g., within a job)
- restricted times of duplicate activities are propagated to the original activity and vice versa
Let $A_u$ be the duplicate activity of $A$ allocated to resource $u \in \text{resource}(A)$

- $u \in \text{resource}(A) \Rightarrow \text{start}(A) \leq \text{start}(A_u)$
- $u \in \text{resource}(A) \Rightarrow \text{end}(A_u) \leq \text{end}(A)$
- $\text{start}(A) \geq \min\{\text{start}(A_u): u \in \text{resource}(A)\}$
- $\text{end}(A) \leq \max\{\text{end}(A_u): u \in \text{resource}(A)\}$
- failure related to $A_u \Rightarrow \text{resource}(A) \setminus \{u\}$
Summary and extensions

**Disjunctive constraint**
- unary resources, non-preemptive activities
- extensions: preemptive activities, cumulative resources

**Edge finding**
- unary resources, non-preemptive activities
- extensions: preemptive activities, cumulative resources

**Not-first/not-last**
- unary resources, non-preemptive activities
- extensions: cumulative resources

**Timetable constraint**
- cumulative resource, non-preemptive activities
- extensions: preemptive activities