Planning & Scheduling

Roman Barták
Department of Theoretical Computer Science and Mathematical Logic

Constraint-based Scheduling

Scheduling model

- **Scheduling problem** is static so it can be directly **encoded as a CSP**.
- Constraint technology is used for **full scheduling**.

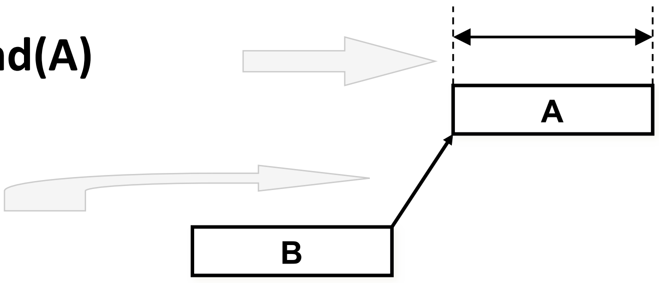
CSP model:

- **variables**
  - position of activity A in time and space
  - time allocation: **start(A), [p(A), end(A)]**
  - resource allocation: **resource(A)**
- **Domain**
  - **release times** and **deadlines** for the time variables
  - **alternative resources** for the resource variables
- **constraints**
  - sequencing and resource capacities

Scheduling model: constraints

- **time relations**
  - **start(A)+p(A)=end(A)**
  - sequencing
    - B<<A

    ⇘**end(B)≤start(A)**



⇘**resource capacity constraints**

- unary resource (activities cannot overlap)
  - A<<B ∨ B<<A

  ⇘ **end(A)≤start(B) ∨ end(B)≤start(A)**

Scheduling model: resources

Resources are used in slightly different meanings in planning and scheduling!

- **scheduling**
  - resource
    = a **machine** (space) for processing the activity
- **planning**
  - resource
    = consumed/produced **material** by the activity
  - resource in the scheduling sense is often handled via logical precondition (e.g. hand is free)

Resource types

- **Unary resource**
  - at most one activity can be processed at any time
- **Cumulative resource**
  - several activities can be processed in parallel if resource capacity is not exceeded.
- **Producible/consumable resource**
  - activity consumes/produces some quantity of the resource
  - minimal capacity is requested (consumption) and maximal capacity cannot be exceeded (production)

Unary resources

Activities cannot overlap in time!

- at any time at most one activity can be processed, hence these resources are called **unary**

We assume that activities are **not interruptible** (non-preemptible).

- non-interruptible (non-preemptible) activity occupies the resource from its start till its completion

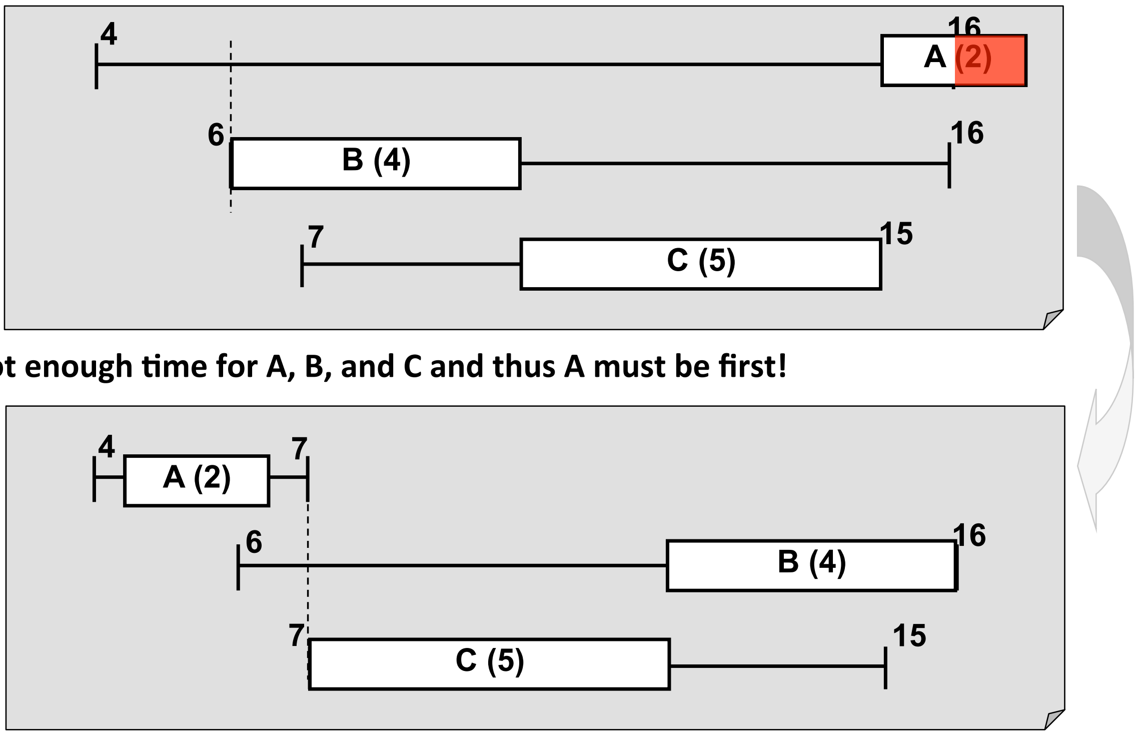
A simple mode is based on disjunctive resources:

- **A « B ∨ B « A**

  end(A) ≤ start(B) ∨ end(B) ≤ start(A)
- hence these resources ale also called **disjunctive**

Edge-finding

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 rules

Inference rules:

- **$p(\Omega \cup \{A\}) > lct(\Omega \cup \{A\}) - est(\Omega) \Rightarrow A \ll \Omega$**
- **$p(\Omega \cup \{A\}) > lct(\Omega) - est(\Omega \cup \{A\}) \Rightarrow \Omega \ll A$**
- **$A \ll \Omega \Rightarrow end(A) \leq \min\{ lct(\Omega') - p(\Omega') \mid \Omega' \subseteq \Omega \}$**
- **$\Omega \ll A \Rightarrow start(A) \geq \max\{ est(\Omega') + p(\Omega') \mid \Omega' \subseteq \Omega \}$**

In practice:

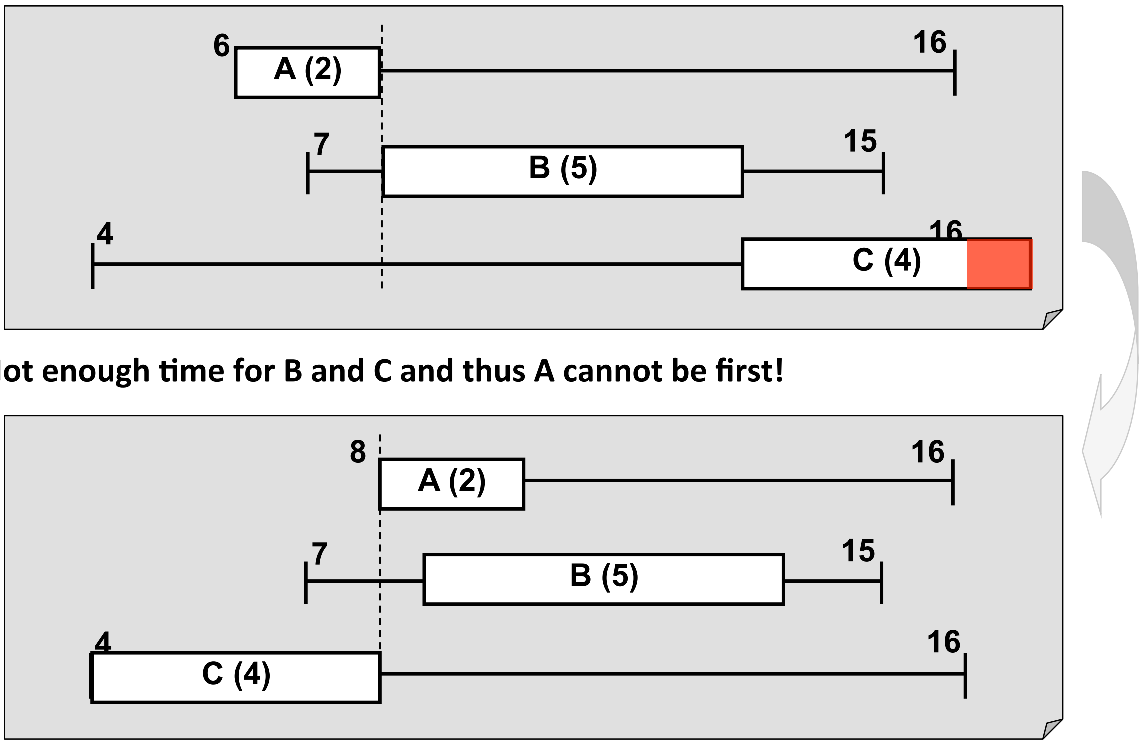
- we need to explore $n.2^n$ pairs (A,**Ω**) (that is too many!)
- instead of all Ω we can use **task intervals** [X,Y]
  $\{C \mid est(X) \leq est(C) \wedge lct(C) \leq lct(Y)\}$
  - ➯time complexity is $O(n^3)$; this is frequently used incremental approach

➯there are also algorithms with time complexity $O(n^2)$ and $O(n.\log n)$

Torres & Lopez (2000)

Not-first/not-last

What happens if activity A is processed first?



Not enough time for B and C and thus A cannot be first!

Torres & Lopez (2000)

Not-first/not-last rules

Not-first inference rules:

$$\min(start(A)) + p(\Omega) + p(A) > \max(end(\Omega)) \Rightarrow \neg A \ll \Omega$$

$$\neg A \ll \Omega \Rightarrow start(A) \geq \min\{ end(B) \mid B \in \Omega \}$$

Not-last (symmetrical) inference rules:

$$\min(start(\Omega)) + p(\Omega) + p(A) > \max(end(A)) \Rightarrow \neg \Omega \ll A$$

$$\neg \Omega \ll A \Rightarrow end(A) \leq \max\{ start(B) \mid B \in \Omega \}$$

In practice:

– it is possible to use selected sets Ω only

– time complexity $O(n^2)$

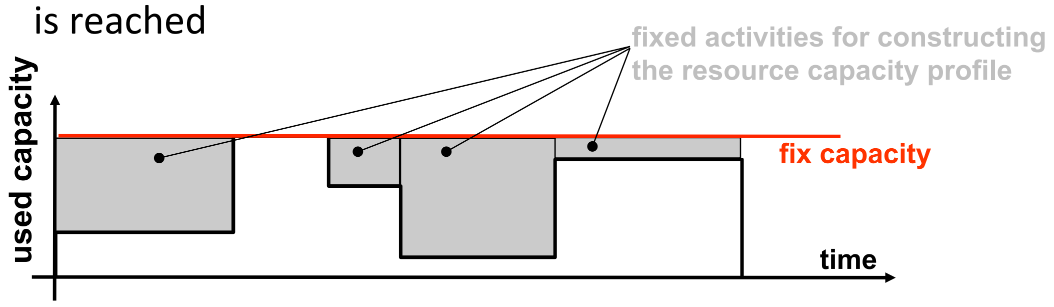
Cumulative resources

Each **activity uses some capacity** of the resource – **cap(A)**.

Activities can be **processed in parallel** if a resource capacity is not exceeded.

Resource capacity **may vary in time**

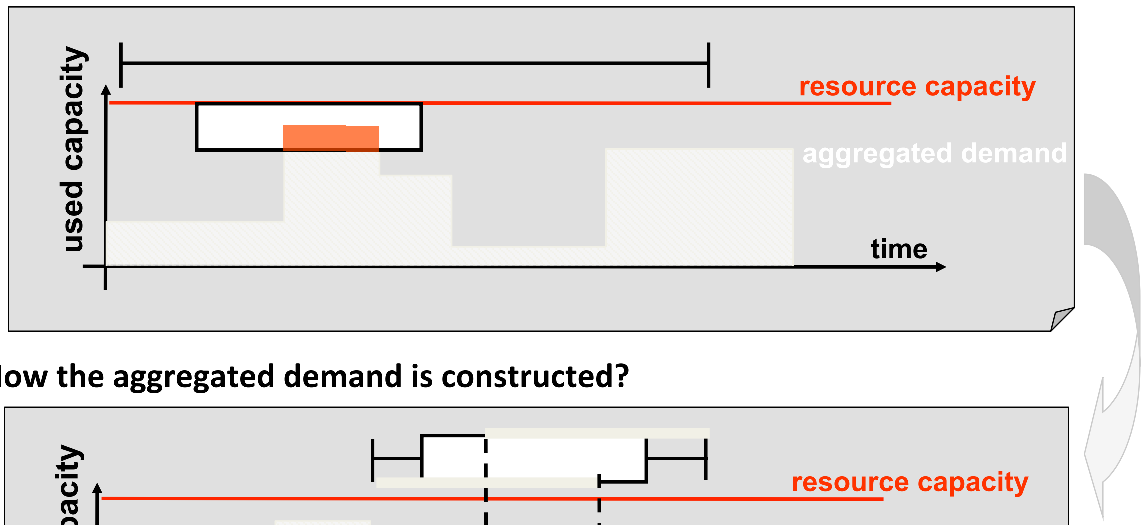
– modeled via fix capacity over time and fixed activities consuming the resource until the requested capacity level is reached


fixed activities for constructing the resource capacity profile
fix capacity
used capacity
time

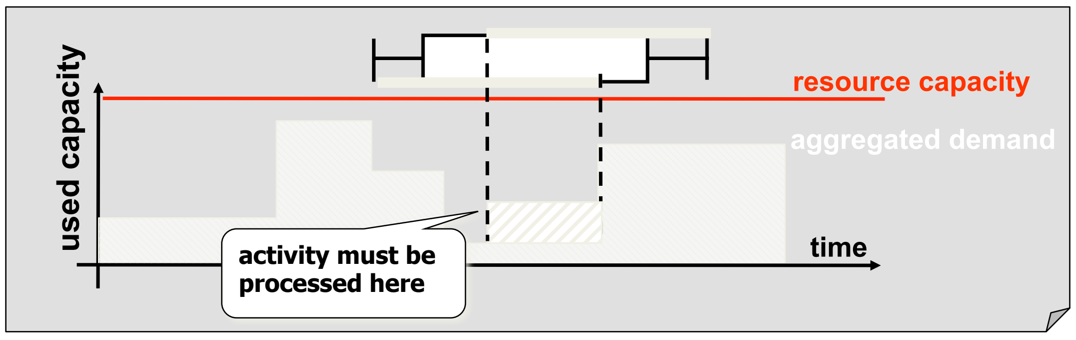
Baptiste et al. (2001)

Aggregated demands

Where is enough capacity for processing the activity?



How the aggregated demand is constructed?



Baptiste et al. (2001)

Timetable constraint

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

$$\forall t \sum_{start(A_i) \le t \le end(A_i)} cap(A_i) \le MaxCapacity$$

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

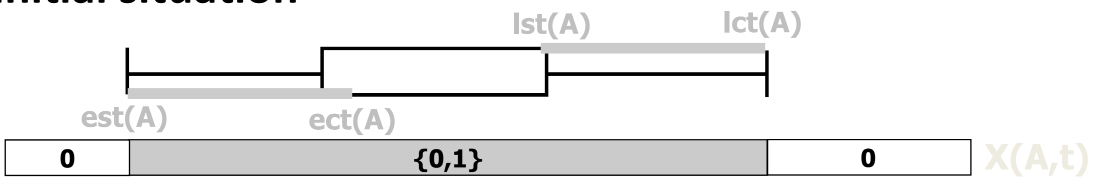
$$\forall t \sum_{A_i} X(A_i,t) \cdot cap(A_i) \le MaxCapacity$$

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

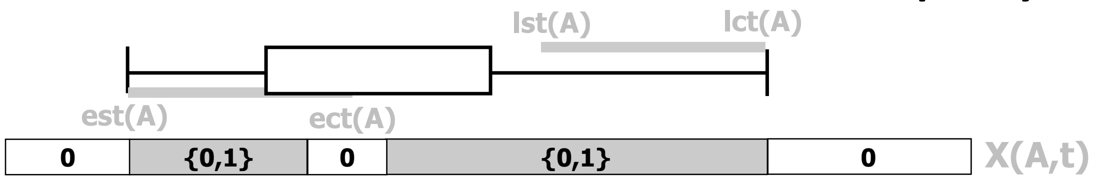
* discrete time is expected

Timetable constraint: example

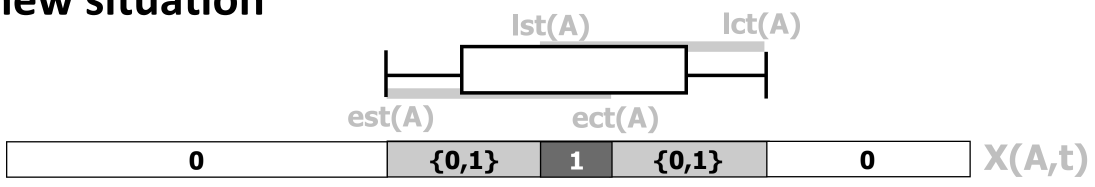
initial situation



some times are forbidden due to exceeded capacity



new situation



Baptiste et al. (2001)

Timetable constraint: rules

How to implement domain filtering?

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

Problem:

t is both an index and variable

start(A) ≥ min{t : ub(X(A,t))=1}

end(A) ≤ 1+max{t : ub(X(A,t))=1}

X(A,t)=0 ∧ t<ect(A) ⇒ start(A)>t

X(A,t)=0 ∧ lst(A)≤t ⇒ end(A)≤t

(lst(A)≤t ∧ t<ect(A) ⇒ 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 res(A)
    - deleting the resource from the domain of variable res(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.

Alternative resources: rules

Let A_u be a duplicate of activity A allocated to resource $u \in res(A)$.

$u \in res(A) \Rightarrow start(A) \leq start(A_u)$

$u \in res(A) \Rightarrow end(A_u) \leq end(A)$

$start(A) \geq \min\{start(A_u) : u \in res(A)\}$

$end(A) \leq \max\{end(A_u) : u \in res(A)\}$

empty time windows for $A_u \Rightarrow res(A) \setminus \{u\}$

In practice, this inference is identical to constructive disjunction between the alternative resources.

Relative ordering

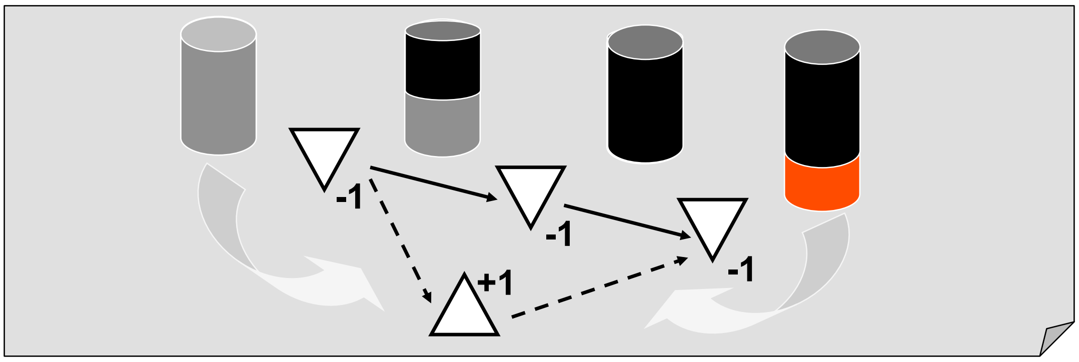
When time is relative (ordering of activities)

then edge-finding and aggregated demand deduce nothing

We can still use information about ordering of activities and resource production/consumption!

Example:

Reservoir: activities consume and supply items



Resource profiles

Activity A „produces" **prod(A)** quantity:

- positive number means **production**
- negative number means **consumption**

- **Optimistic resource profile (orp)**
  - maximal possible level of the resource when A is processed
  - activities known to be before A are assumed together with the production activities that can be before A

$$orp(A) = InitLevel + prod(A) + \sum_{B<<A} prod(B) + \sum_{B??A\ \&\ prod(B)>0} prod(B)$$

- **Pessimistic resource profile (prp)**
  - minimal possible level of the resource when A is processed
  - activities known to be before A are assumed together with the consumption activities that can be before A

$$prp(A) = InitLevel + prod(A) + \sum_{B<<A} prod(B) + \sum_{B??A\ \&\ prod(B)<0} prod(B)$$

***B??A means that order of A and B is unknown yet**

orp inference

$$orp(A) < MinLevel \Rightarrow fail$$

- "despite the fact that all production is planned before A, the minimal required level in the resource is not reached"

$$orp(\mathrm{A}) - prod(\mathrm{B}) - \sum_{\mathrm{B<<C}\ \&\ C\mathrm{??A}\ \&\ prod(\mathrm{C})>0} prod(\mathrm{C}) < MinLevel \Rightarrow \mathrm{B<<A},$$

for any B such that B??A and *prod*(B)>0

- "if production in B is planned after A and the minimal required level in the resource is not reached then B must be before A"

Cesta & Stella (1997)

prp inference

$prp(\mathrm{A}) > MaxLevel \Rightarrow fail$

- "despite the fact that all consumption is planned before A, the maximal required level (resource capacity) in the resource is exceeded"

$prp(\mathrm{A}) - prod(\mathrm{B}) - \sum_{\mathrm{B} << \mathrm{C}\ \&\ \mathrm{C}??\mathrm{A}\ \&\ prod(\mathrm{C})<0} prod(\mathrm{C}) > MaxLevel$
$\Rightarrow \mathrm{B} << \mathrm{A},$

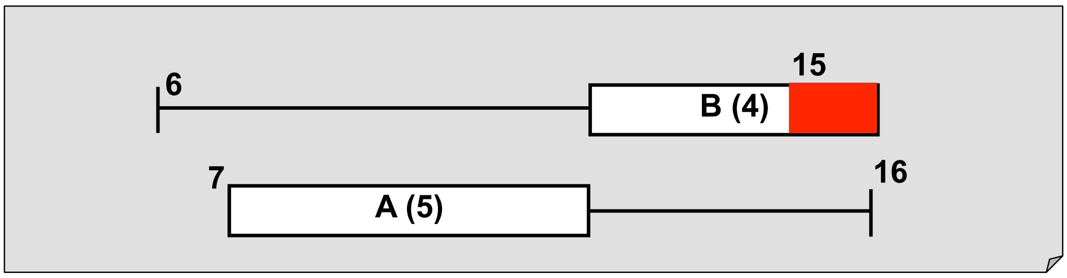
for any B such that B??A and *prod*(B)<0

- "if consumption in B is planned after A and the maximal required level in the resource is exceeded then B must be before A"

Vilím (2002)

Detectable precedence relations

What happens if A is processed before B?



Restricted time windows can be used to infer inevitable precedence relations.

est(A)+p(A)+p(B)>lct(B) ⇒ B«A

Laborie (2003)

Energy precedence

We can "energy" of activities processed before A to infer a more precise value of est(A)

start(A) ≥ max{ est(Ω') + ⌈e(Ω')/cap⌉ | Ω'⊆{C : C«A} }

For unary resources:

start(A) ≥ max{ est(Ω') + p(Ω') | Ω'⊆{C : C«A} }

it is enough to explore sets **Ω(X,A) = {Y | Y « A ∧ est(X) ≤ est(Y)}**

start(A) ≥ max{ est(Ω(X,A)) + p(Ω(X,A)) | X « A }

```
dur ← 0
end ← est(A)
for each Y∈{ X | X « A } in the non-increasing order of est(Y) do
    dur ← dur + p(Y)
    end ← max(end, est(Y) + dur)
end for
est(A) ← end
```

Optimization in CSP

The objective function in a CSP is usually encoded using an equality constraint and a new variable:

v = obj(Xs)

Example: **makespan = max{end(A_i)}**

– We can infer better bounds for variable v using the current domains of variables Xs:

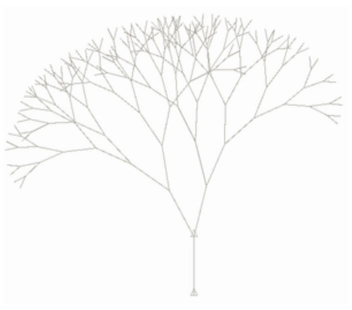
makespan_{min} = max{ect(A_i)}

– We can infer better bounds for variables Xs using the current bounds for variable v:

end(A_i) ≤ makespan_{max}

– For more complex objective functions we can use problem relaxation to compute the bounds.

Branching schemes



Branching = resolving disjunctions

Traditional scheduling approaches:

- take **critical decisions first**
  - resolve bottlenecks …
  - defines the shape of the search tree
  - recall the **first-fail** principle
- prefer an **alternative leaving more flexibility**
  - defines order of branches to be explored
  - recall the **succeed-first** principle

How to describe criticality and flexibility formally?

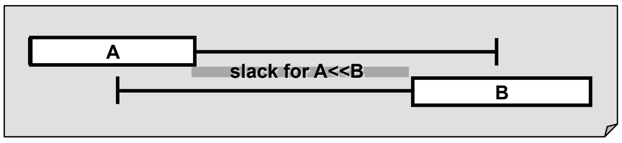
Smith and Cheng (1993)

Slack

Slack is a formal description of flexibility

- Slack for **a given order of two activities**

  „free time for shifting the activities“



  slack(A<<B) = max(end(B))-min(start(A))-p({A,B})

- Slack for **two activities**

  slack({A,B}) = max{slack(A<<B),slack(B<<A)}

- Slack for **a group of activities**

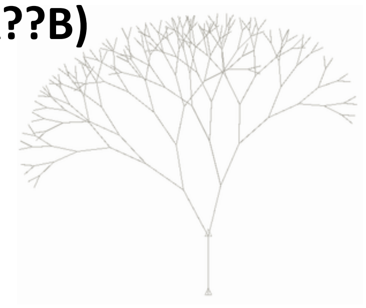
  slack(Ω) = max(end(Ω)) - min(start(Ω)) - p(Ω)

Smith and Cheng (1993)

Order branching

A<<B ∨ ¬A<<B

- **Which activities should be ordered first?**
  - the most critical pair (first-fail)
  - the pair with **the minimal slack({A,B})**
- **What order should be selected?**
  - the most flexible order (succeed-first)
  - the order with **the maximal slack(A??B)**
- O(n^2) choice points

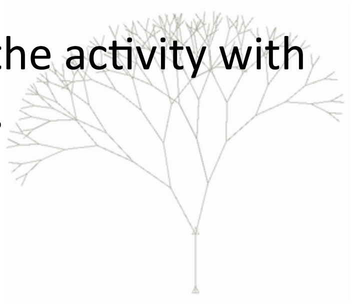


Baptiste et al. (1995)

First/last branching

(A<<Ω ∨ ¬A<<Ω) or (Ω<<A ∨ ¬ Ω<<A)

- **Should we look for first or last activity?**
  - select a **smaller set** among possible first or possible last activities (first-fail)
- **What activity should be selected?**
  - If first activity is being selected then the activity with the **smallest min(start(A))** is preferred.
  - If last activity is being selected then the activity with the **largest max(end(A))** is preferred.
- O(n) choice points



Resource slack is defined as a slack of the set of activities processed by the resource.

How to use a resource slack?

- choosing a resource on which **the activities will be ordered** first
  - resource with a minimal slack (**bottleneck**) preferred
- choosing a resource on which the **activity will be allocated**
  - resource with a maximal slack (**flexibility**) preferred

© 2014 Roman Barták
Department of Theoretical Computer Science and Mathematical Logic
bartak@ktiml.mff.cuni.cz