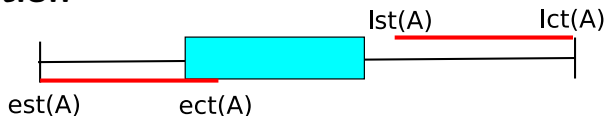


Constraint-based Scheduling: Propagation

① Unary resources

② Cumulative resources

Notation



- $start(A)$ domain variable for starting time of activity A
- $end(A)$ domain variable for completion time of activity A
- $p(A)$ domain variable for processing time of activity A

- $est(A)$ earliest start time of activity A
- $ect(A)$ earliest completion time of activity A
- $lst(A)$ latest start time of activity A
- $lct(A)$ latest completion time of activity A

- Ω is the set of activities
- $p(\Omega) = \sum_{A \in \Omega} p(A)$
- $est(\Omega) = \min\{est(A) \mid A \in \Omega\}$
- $lct(\Omega) = \max\{lct(A) \mid A \in \Omega\}$

Unary Resource for Unit Time Activities: all_different

$U = \{X_2, X_4, X_5\}$, $dom(U) = \{2, 3, 4\}$:
 $\{2,3,4\}$ impossible for $X_1, X_3, X_6 \Rightarrow$
 X_1 in 5..6, $X_3 = 5$, X_6 in $\{1\} \setminus / (5..6)$

Consistency:

$\forall \{X_1, \dots, X_k\} \subset V : card\{D_1 \cup \dots \cup D_k\} \geq k$

teacher	min	max
X1	3	6
X2	3	4
X3	2	5
X4	2	4
X5	3	4
X6	1	6

Inference rule

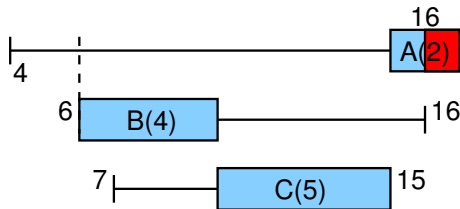
- $U = \{X_1, \dots, X_k\}$, $dom(U) = \{D_1 \cup \dots \cup D_k\}$
- $card(U) = card(dom(U)) \Rightarrow \forall v \in dom(U), \forall X \in (V - U), X \neq v$
- values in $dom(U)$ unavailable for other variables

Complexity

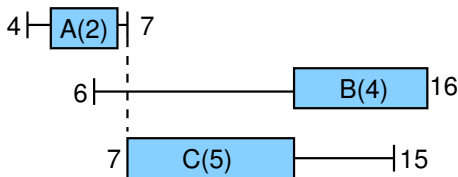
- $O(2^n)$: search through all subsets of the set of n variables
- $O(n \log n)$: changes of bounds propagated only (1998)

Edge finding: example

- What happens if activity A is not processed first?

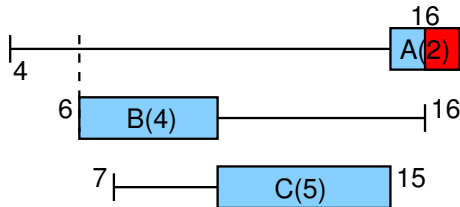


- Not enough time for A, B, and C and thus A must be first!

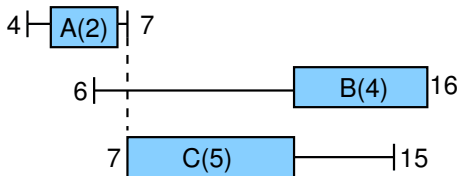


Edge finding: example with filtering rules

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



- $A \ll \Omega \Rightarrow end(A) \leq \min\{lct(\Omega') - p(\Omega') \mid \Omega' \subseteq \Omega\}$



Edge finding: all filtering rules

- **Edge-finding rules**

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

- **Edge-finding (symmetrical) rules**

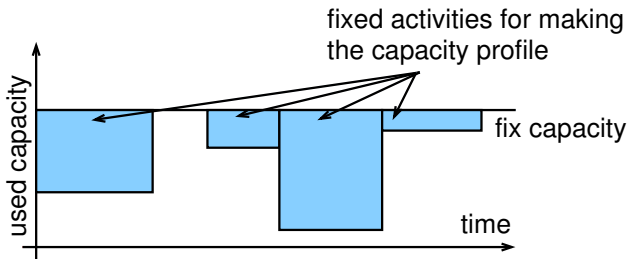
- similar rules for $\Omega \ll A$ and $start(A)$

- In practice:

- there are $n \cdot 2^n$ pairs (A, Ω) to consider (too many!)
- instead of Ω use so called **task intervals** $[A, B]$
 $\{C \mid est(A) \leq est(C) \wedge lct(C) \leq lct(B)\}$
time complexity $O(n^3)$, frequently used incremental algorithm
- there are also $O(n^2)$ and $O(n \log n)$ algorithms

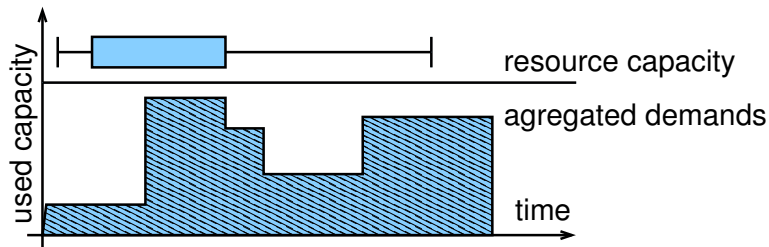
Cumulative resources

- 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

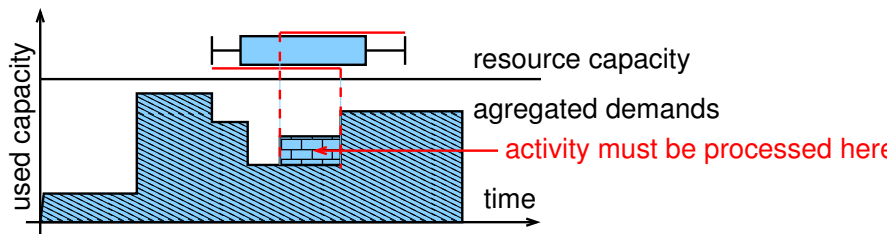


Aggregated demands

- Where is enough capacity for processing the activity?



- How aggregated demand is constructed?



Timetable constraint

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

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

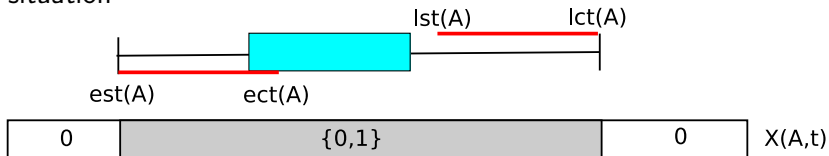
- **Timetable** for activity A is a set of Boolean domain variables $X(A, t)$ indicating whether A is processed in time t

$$\forall t \sum_{A_i} X(A_i, t) * cap(A_i) \leq MaxCapacity$$

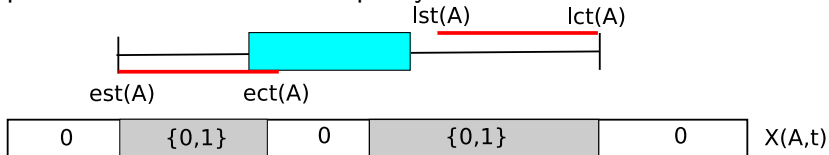
$$\forall t, i \ start(A_i) \leq t < end(A_i) \Leftrightarrow X(A_i, t)$$

Timetable constraint: filtering example

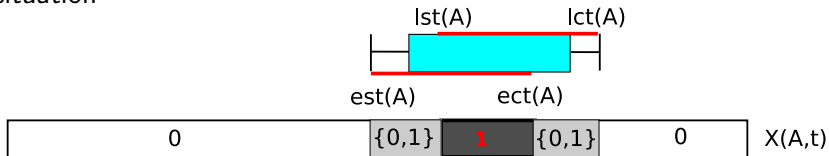
Initial situation



Some positions forbidden due to capacity



New situation



Constraint-based Scheduling: Search Strategies

3 Search

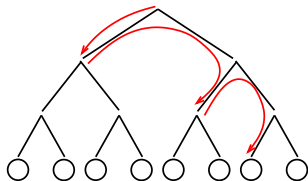
4 Search Strategies for Scheduling

Search (revision)

Constraint propagation techniques are (usually) incomplete
⇒ search algorithm is needed to solve the "rest"

Labeling

- depth-first search (DFS/BT)
 - assign value to variable
 - propagate = make the problem locally consistent
 - backtrack in case of failure



- $X \text{ in } 1..5 \quad \equiv \quad X=1 \vee X=2 \vee X=3 \vee X=4 \vee X=5$

Generally search algorithm solves remaining disjunctions

- $X=1 \vee X \neq 1$ standard assignment
- $X < 3 \vee X \geq 3$ domain splitting
- $X < Y \vee X \geq Y$ variable ordering (scheduling: tasks ordering)

Search: variable ordering

Which variable should be assigned first?

First-fail principle

- prefer variable with the hardest assignment
- for example variable with the smallest domain:
domain can be emptied easily
- or variable with the most constraints:
assignment of other variables constrain and
make the domain smaller easily

Variable ordering defines **shape of the search tree**

- selection of variable with small domain size:
small branching on this level more options left for later
- selection of variable with large domain size:
large branching on this level less options left for later

Search: value ordering

What value should be chosen first?

Succeed-first principle

- prefers values with probably belongs to the solution
- for example the values with most supports in neighbouring variables
- this heuristic is usually problem specific

Value ordering defines the **order how the branches are explored**

Branching schemes in scheduling

Branching=resolving disjunctions

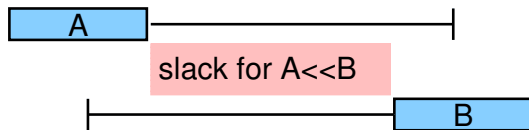
Traditional scheduling approaches

- take the **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 **succeed-first** principle

How to describe criticality and flexibility formally?

Slack

- **Slack** is a formal description of flexibility
- Slack for a **given order of two activities**
"free time for shifting the activities"



$$\text{slack}(A \ll B) = \max(\text{end}(B)) - \min(\text{start}(A)) - p(A) - p(B)$$

- Slack for **two activities** (without any ordering)
 $\text{slack}(\{A, B\}) = \max(\text{slack}(A \ll B), \text{slack}(B \ll A))$
- Slack for a **group of activities**
 $\text{slack}(\Omega) = \max(\text{end}(\Omega)) - \min(\text{start}(\Omega)) - p(\Omega)$

Order branching

$$A \ll B \quad \vee \quad \neg A \ll B$$

What activities A, B should be ordered first?

- the most critical pair (first-fail)
- **the pair with the minimal slack**($\{A, B\}$)

What order of activities A and B should be selected?

- the most flexible order
- the order with **with the maximal slack**($A??B$)

$O(n^2)$ choice points

First/last branching

$$(A \ll \Omega \vee \neg A \ll \Omega) \quad \vee \quad (\Omega \ll A \vee \neg \Omega \ll A)$$

Should we look for the first or last activity?

- look to the set of possible candidates for first activity and to the set of possible candidates for last activities
- select a **smaller set** from these (first-fail)
 - smaller number of candidates means that it is harder to find a suitable candidate

What activity should be selected?

- if first activity is being selected then the activity with the **smallest** $\min(\text{start}(A))$ is preferred
- if last activity is being selected then the activity with the **largest** $\max(\text{end}(A))$ is preferred

$O(n)$ choice points

Resource slack

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