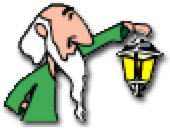


Metode şi Algoritmi de Planificare (MAP)

2009-2010

Curs 2

Introducere în problematica planificării

Introduction to scheduling

- Scheduling problem definition
- Classification of scheduling problems
- Complexity of scheduling problem
- DAG Scheduling

General Scheduling Problem (1/3)

- *Resource Constrained Project Scheduling Problem* (RCPSP)
- We have:
  - Tasks $j = 1, ..., n$ with processing times p_j
  - Resources $k = 1, ..., r$
  - A constant amount of R_k units of resource k is available at any time.
  - During processing, task j occupies r_{jk} units of resource k.
  - n and r are finite.
  - Precedence constrains $i \rightarrow j$ between some activities i, j with the meaning that activity j cannot start before i is finished.

General Scheduling Problem (2/3)

- The objective is to determine starting times *Sj* for all tasks *j* in such a way that:
  - at each time *t* the total demand for resource *k* is not greater than the availability R_k
  - the given precedence constraints are fulfilled, i. e.

$$S_i + p_i \leq S_j \text{ if } i \rightarrow j$$

  - some objective function $f(C_1, ..., C_n)$ is minimized where

$$C_j = S_j + p_j$$

  is the completion time of activity *j*.
- The fact that activities *j* start at time S_j and finish at time $S_j + p_j$ implies that the activities *j* are not preempted. We may relax this condition by allowing **preemption** (activity splitting).

General Scheduling Problem (3/3)

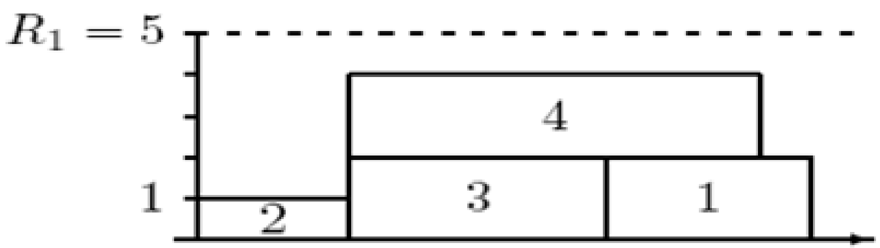
- If preemption is not allowed the vector $S = (S_j)$ defines a schedule.
- S is called feasible if all resource and precedence constraints are fulfilled.
- One has to find a feasible schedule which minimizes the objective function $f(C_1, ..., C_n)$.
- In project planning $f(C_1, ..., C_n)$ is often replaced by the makespan C_{max} which is the maximum of all C_j values.

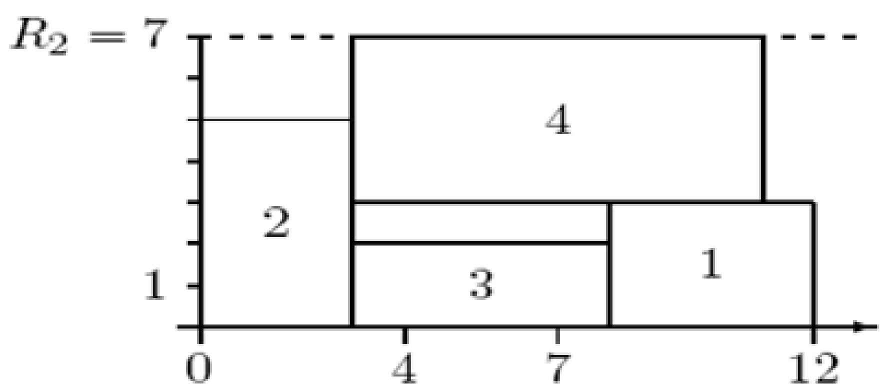
- The constraints $S_i + p_i \leq S_j$ may be replaced by $S_i + d_{ij} \leq S_j$, were d_{ij} represents the deadline for task j on resource i.

Scheduling example

- Let's consider:
  - $n = 4$, $r = 2$ ($R_1 = 5$, $R_2 = 7$)
  - Precedence constrain $2 \rightarrow 3$, and:

j	1	2	3	4
p_j	4	3	5	8
r_{j1}	2	1	2	2
r_{j2}	3	5	2	4





Example of Scheduling Problems

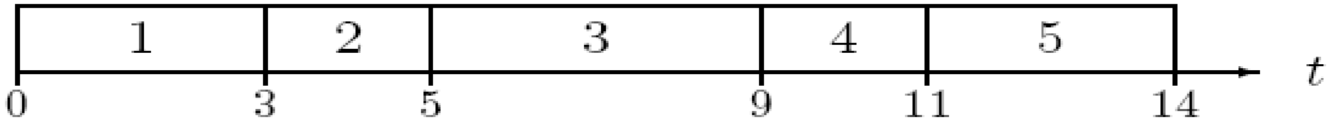
- Task scheduling for computer machine
- Production scheduling
- Robotic cell scheduling
- Computer processor scheduling
- Timetabling
- Personnel scheduling
- Railway scheduling
- Air traffic control

Machine Scheduling Problems

- Here we will consider
  - single machine problems,
  - parallel machine problems, and
  - shop scheduling problems

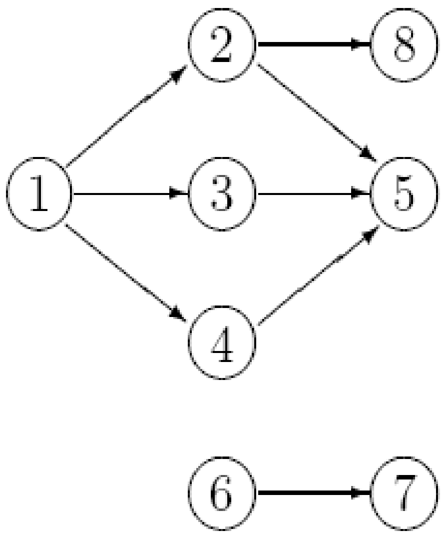
Single machine problem

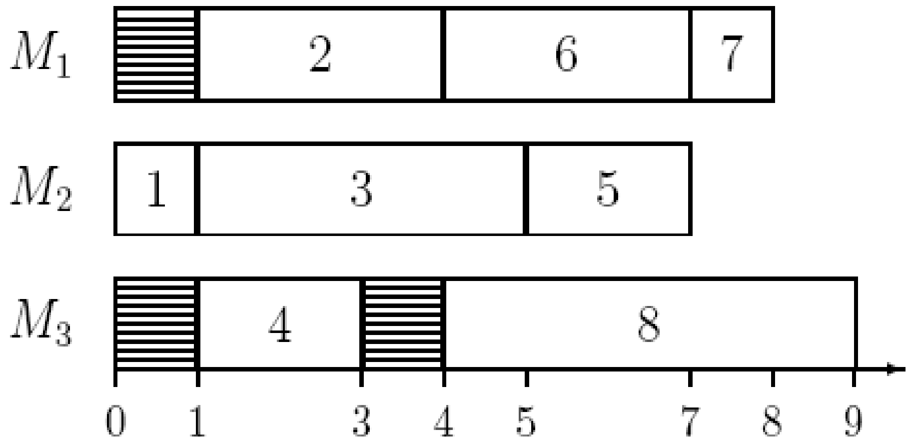
- We have n tesks to be processed on a single machine ($r = 1$).
- Additionally precedence constraints between the tasks may be given.
- This problem can be modeled with $r = 1$, $R_1 = 1$, and $r_{j1} = 1$ for all tasks j.



Parallel Machine Problem

- We have tasks *j* as before and *m* **identical machines** $M_1, \ldots, M_m$. The processing time for *j* is the same on each machine.
- One has to assign the tasks to the machines and to schedule them on the assigned machines.
- This problem correspond with $r = 1$, $R_1 = m$, and $r_{j1} = 1$ for all tasks *j*.





Parallel Machine Problem

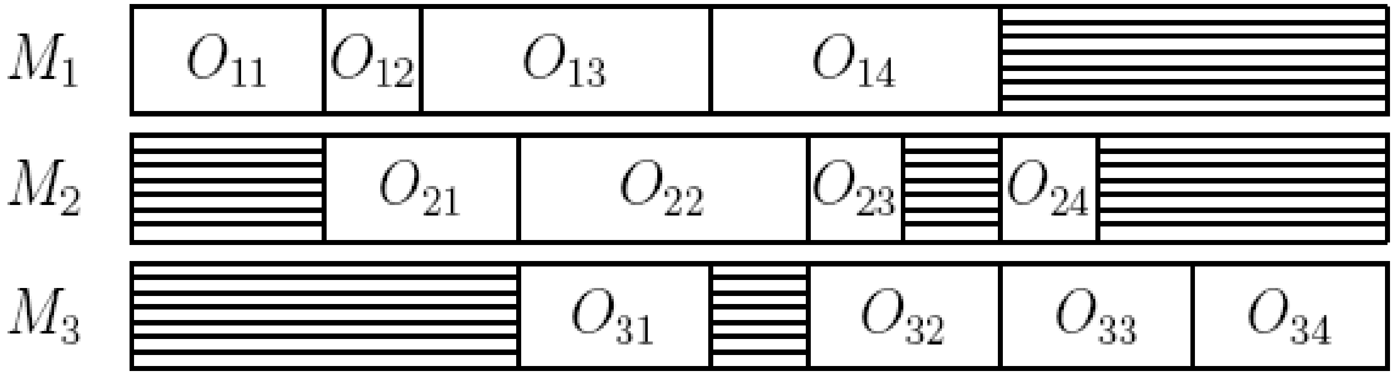
- For **unrelated machines** the processing time p_{jk} depends on the machine M_k on which j is processed.
- The machines are called **uniform** if $p_{jk} = p_j/r_k$.
- In a problem with **multi-purpose machines** a set of machines μ_j is associated with each task *j* indicating that *j* can be processed on one machine in μ_j only.

Shop Scheduling Problem

- In a **general shop scheduling problem** we have m machines $M_1, \ldots, M_m$ and n tasks.
- Job j consists of $n(j)$ operations $O_{1j}, O_{2j}, \ldots, O_{n(j)j}$ where O_{ij} must be processed for p_{ij} time units on a dedicated machine $\mu_{ij} \in \{M_1, \ldots, M_m\}$.
- Two operations of the same job cannot be processed at the same time. Precedence constraints are given between the operations.
- To model the general shop scheduling problem
  - $r = n + m$ resources $k = 1, \ldots, n + m$ with $R_k = 1$ for all k. While resources $k = 1, \ldots, m$ correspond to the machines, resources $m + j$ ($j = 1, \ldots, n$) are needed to model that different operations of the same job cannot be scheduled at the same time.
  - $n(1) + n(2) + \ldots + n(n)$ activities O_{ij} where operation O_{ij} needs one unit of "machine resource" μ_{ij} and one unit of the "job resource" $m + j$.

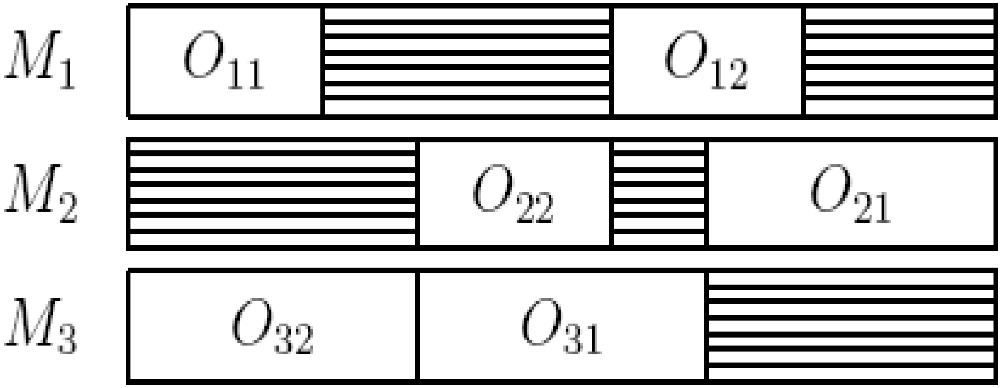
Shop Scheduling Problem

- A **job-shop problem** is a general shop scheduling problem with chain precedence constraints of the form:
  - $O_{1j} \rightarrow O_{2j} \rightarrow \ldots \rightarrow O_{n(j)j}$.
- A **flow-shop problem** is a special job-shop problem with
  - $n(j) = m$ operations for $j = 1, \ldots, n$ and
  - $\mu_{ij} = M_i$ for $i = 1, \ldots, m$ and $j = 1, \ldots, n$.
- In a **permutation flow-shop problem** the jobs have to be processed in the same order on all machines.



Shop Scheduling Problem

- An **open-shop problem** is like a flow-shop problem but without precedence constraints between the operations.



Classification of Scheduling Problems

- Classes of scheduling problems can be specified in terms of the three-field classification $\alpha \mid \beta \mid \gamma$ where:
  - α specifies the **machine environment**
  - β specifies the **task characteristics**, and
  - γ describes the **objective function(s).**

α – Machine Environment

- To describe the machine environment the following symbols are used:
  - 1: single machine
  - P: parallel identical machines
  - Q: uniform machines
  - R: unrelated machines
  - MPM: multipurpose machines
  - J: job-shop
  - F: flow-shop
  - O: open-shop
- The above symbols are used if the number of machines is part of the input. If the number of machines is fixed to m we write P_m, Q_m, R_m, MPM_m, J_m, F_m, O_m.

β – Task Characteristics

- **pmtn:** preemption
- **r_j**: release times
- **d_j**: deadlines
- **$p_j = 1$ or $p_j = p$ or $p_j \in \{1,2\}$**: restricted processing times
- **Prec**: arbitrary precedence constraints
- **intree (outtree)**: intree (or outtree) precedences
- **chains**: chain precedences
- **series-parallel**: a series-parallel precedence graph

γ – Objective Functions

- Two types of objective functions are most common:
  - **bottleneck objective functions**

$$\max \{f_j(C_j) \mid j=1, ..., n\}$$

  - **sum objective functions**

$$\Sigma f_j(C_j) = f_1(C_1) + f_2(C_2) + ... + f_n(C_n).$$

- $\mathbf{C_{max}}$ and $\mathbf{L_{max}}$ symbolize the bottleneck objective functions with $f_j(C_j) = C_j$ (makespan) and $f_j(C_j) = C_j - d_j$ (maximum lateness), respectively.

γ – Objective Functions

- Common sum objective functions are:
  - ΣC_j (mean flow-time) and $\Sigma \omega_j C_j$ (weighted flow-time)
  - ΣU_j (number of late jobs) and $\Sigma \omega_j U_j$ (weighted number of late jobs) where $U_j = 1$ if $C_j > d_j$ and $U_j = 0$ otherwise.
  - ΣT_j (sum of tardiness) and $\Sigma \omega_j T_j$ (weighted sum of tardiness) where the tardiness of job j is given by $T_j = \max\{0, C_j - d_j\}$.

Scheduling problem - Examples

- $1 \mid prec; p_j = 1 \mid \Sigma\omega_j C_j$
- $P_2 \mid \mid C_{max}$
- $P \mid p_j = 1; r_j \mid \Sigma\omega_j U_j$
- $R_2 \mid pmtn; intree \mid C_{max}$
- $J_3 \mid \mid C_{max}$
- $F \mid p_{ij} = 1; outtree; r_j \mid \Sigma C_j$
- $O_m \mid p_j = 1 \mid \Sigma T_j$

- ☺ **The scheduling zoo:**
  - A searchable bibliography on scheduling
  - Peter Brucker and Sigrid Knust
  - http://www.lix.polytechnique.fr/~durr/query/

Complexity Theory

- Polynomial algorithms
- Classes *P* and *NP*
- *NP*-complete and *NP*-hard problems

Polynomial algorithms

- A problem is called polynomially solvable if it can be solved by a polynomial algorithm.

- **Example:** $1 \mid \mid \Sigma \omega_j C_j$ can be solved by scheduling the jobs in an ordering of non-increasing ω_j/p_j – values. Complexity: O($n$log$n$)

- If we replace the binary encoding by an unary encoding we get the concept of a pseudo-polynomial algorithm.

- **Example:** An algorithm for a scheduling problem with computational effort $O(\Sigma p_j)$ is pseudo-polynomial.

Classes *P* and *NP*

- A problem is called a **decision problem** if the output range is {yes, no}.
- We may associate with each scheduling problem a decision problem by defining a threshold k for the objective function f. The decision problem is: Does a feasible schedule S exist satisfying $f(S) \leq k$?
- *P* is the class of decision problems which are polynomially solvable.
- *NP* is the class of decision problems with the property that for each “yes”-answer a certificate exists which can be used to verify the “yes”-answer in polynomial time.
- Decision versions of scheduling problems belong to *NP* (a “yes”-answer is certified by a feasible schedule *S* with $f(S) \leq k$).
- $P \subseteq NP$ holds. It is open whether $P = NP$.

NP- complete and *NP*- hard problems

- For two decision problems P and Q, we say that **P reduces to Q** (denoted by P α Q) if there exists a polynomial-time computable function g that transforms inputs for P into inputs for Q such that x is a “yes”-input for P if and only if g(x) is a “yes”-input for Q.
- **Properties of polynomial reductions:**
  - Let P, Q be decision problems. If P α Q then Q $\in$ *P* implies P $\in$ *P* (and, equivalently, P $\notin$ *P* implies Q $\notin$ *P)* .
  - Let P, Q, R be decision problems. If P α Q and Q α R, then P α R.
  - A decision problem Q is called ***NP* - complete** if Q $\in$ *NP* and, for all other decision problems P $\in$ *NP,* we have P α Q.
- If any single *NP*-complete decision problem Q could be solved in polynomial time then we would have *P* = *NP*.

NP- complete and *NP*- hard problems

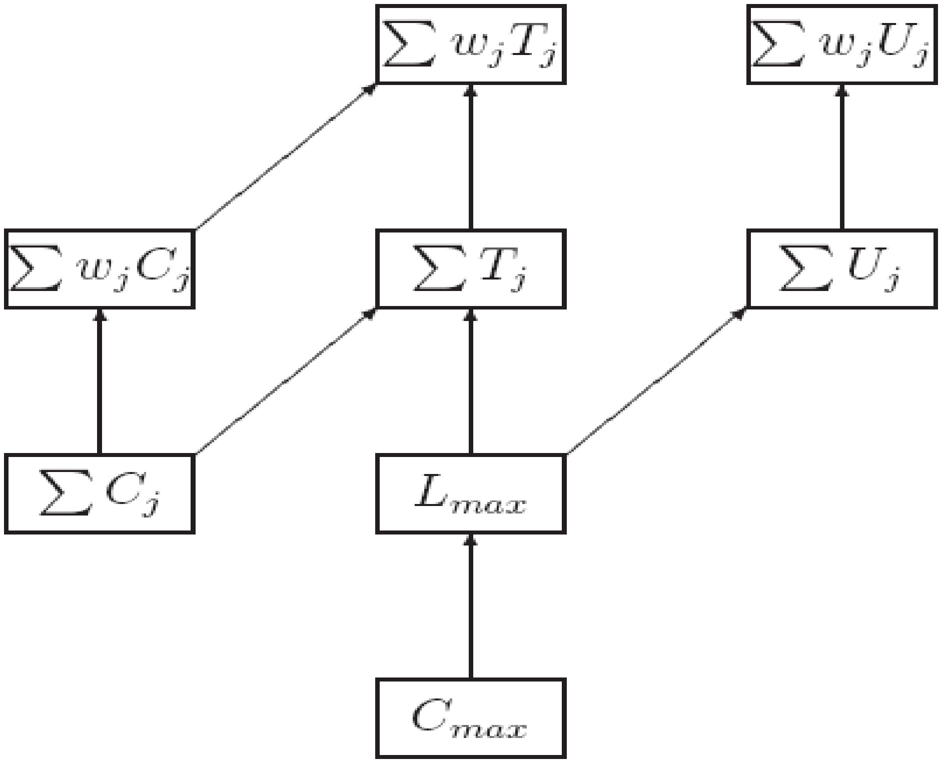
- To prove that a decision problem P is *NP* - complete it is sufficient to prove the following two properties:
  - P $\in$ *NP*, and
  - there exist an *NP*- complete problem Q with Q α P.

- An optimization problem is ***NP*- hard** if its decision version is *NP*- complete.

- Cook [1971] has shown that the satisfiability problem from Boolean logic is *NP*- complete. Using this result he used reduction to prove that other combinatorial problems are *NP*- complete as well.

Complexity of machine scheduling problems

- http://www.mathematik.uni-osnabrueck.de/research/OR/class/
- Elementary reductions



Polynomially solvable single machine problems

$1 \mid prec; r_j \mid C_{\max}$	$O(n^2)$
$1 \mid prec; r_j; p_j = p \mid L_{\max}$	$O(n^3 \log\log n)$
$1 \mid prec \mid f_{\max}$	$O(n^2)$
$1 \mid prec; r_j; p_j = 1 \mid f_{\max}$	$O(n^2)$
$1 \mid prec; r_j; pmtn \mid f_{\max}$	$O(n^2)$
$1 \mid r_j; pmtn \mid \sum C_j$	$O(n \log n)$
$1 \mid prec; r_j; p_j = p \mid \sum C_j$	$O(n^2)$
$1 \mid prec; r_j; p_j = p, pmtn \mid \sum C_j$	$O(n^2)$
$1 \mid r_j; p_j = p \mid \sum w_j C_j$	$O(n^7)$
$1 \mid sp\text{-graph} \mid \sum w_j C_j$	$O(n \log n)$
$1 \mid\mid \sum U_j$	$O(n \log n)$
$1 \mid r_j; pmtn \mid \sum U_j$	$O(n^5)$
	$O(n^4)$
$1 \mid r_j; p_j = p \mid \sum w_j U_j$	$O(n^7)$
$1 \mid r_j; p_j = p; pmtn \mid \sum w_j U_j$	$O(n^{10})$
$1 \mid r_j; p_j = p \mid \sum T_j$	$O(n^7)$
$1 \mid r_j; p_j = 1 \mid \sum f_j$	$O(n^3)$
$1 \mid r_j; p; pmtn \mid \sum T_j$	$O(n^2)$

NP - hard single machine scheduling problems

$*1 \mid r_j \mid L_{\max}$

$*1 \mid r_j \mid \sum C_j$

$*1 \mid prec \mid \sum C_j$

$*1 \mid chains; r_j; pmtn \mid \sum C_j$

$*1 \mid prec; p_j = 1 \mid \sum w_j C_j$

$*1 \mid chains; r_j; p_j = 1 \mid \sum w_j C_j$

$*1 \mid r_j; pmtn \mid \sum w_j C_j$

$*1 \mid chains; p_j = 1 \mid \sum U_j$

$1 \parallel \sum w_j U_j$

$1 \parallel \sum T_j$

$*1 \mid chains; p_j = 1 \mid \sum T_j$

$*1 \parallel \sum w_j T_j$

Minimal and maximal open problems

- **minimal open:**

$$1|pmtn; p_i = p; r_i| \sum w_i C_i$$
$$1|p_i = p; r_i| \sum w_i T_i$$

- **maximal open:**

$$1|p_i = p; r_i| \sum w_i T_i$$
$$1|pmtn; p_i = p; r_i| \sum w_i T_i$$

Polynomially solvable parallel machine problems without preemption

$Q \mid p_i = 1 \mid f_{\max}$	$O(n^2)$
$P \mid p_i = p; outtree; r_i \mid C_{\max}$	$O(n)$
$P \mid p_i = p; tree \mid C_{max}$	$O(n)$
$P2 \mid p_i = p; prec \mid C_{max}$	$O(n^{\log^7})$
$Q \mid p_i = 1; r_i \mid C_{max}$	$O(n \log n)$
$Q \mid p_i = p; r_i \mid C_{\max}$	$O(n \log n)$

Polynomially solvable parallel machine problems without preemption

$P \mid p_i = p; intree \mid L_{max}$	$O(n)$
$P \mid p_i = p; r_i \mid L_{max}$	$O(n^3 \log\log n)$
$P2 \mid p_i = 1; prec; r_i \mid L_{max}$	$O(n^3 \log n)$
$P \mid p_i = 1; outtree; r_i \mid \sum C_i$	$O(n^2)$
$P \mid p_i = p; outtree \mid \sum C_i$	$O(n \log n)$
$Pm \mid p_i = p; tree \mid \sum C_i$	$O(n^m)$
$P \mid p_i = p; r_i \mid \sum C_i$	$O(n \log n)$
$P2 \mid p_i = p; prec \mid \sum C_i$	$O(n^{\log^7})$
$P2 \mid p_i = 1; prec; r_i \mid \sum C_i$	$O(n^9)$

NP- hard parallel machine problems without preemption

$P2 \,||\, C_{max}$

* $P \,||\, C_{max}$
* $P \mid p_i = 1; intree; r_i \mid C_{max}$
* $P \mid p_i = 1; prec \mid C_{max}$
* $P2 \mid chains \mid C_{max}$
* $Q \mid p_i = 1; chains \mid C_{max}$
* $P \mid p_i = 1; outtree \mid L_{max}$
* $P \mid p_i = 1; intree; r_i \mid \sum C_i$
* $P \mid p_i = 1; prec \mid \sum C_i$
* $P2 \mid chains \mid \sum C_i$
* $P2 \mid r_i \mid \sum C_i$

$P2 \,||\, \sum w_i C_i$

* $P \,||\, \sum w_i C_i$
* $P2 \mid p_i = 1; chains \mid \sum w_i C_i$
* $P2 \mid p_i = 1; chains \mid \sum U_i$
* $P2 \mid p_i = 1; chains \mid \sum T_i$

Description of task dependencies

- A task graph is a directed acyclic graph G(V, E, c, τ) where:
  - V is a set of nodes (tasks);
  - E is a set of directed edge (dependencies);
  - c is a function that associates a weight $c(u)$ to each node; represent the execution time of the task T_u, which is represented by the node u in V;
  - τ is a function that associates a weight to a directed edge; if u and v are two nodes in V then τ denotes the inter-tasks communication time between T_u and T_v.
- $st(u)$ is *start time* and $ft(u)$ is *finish time*

$$Makespan = max\{ft(u)\}$$

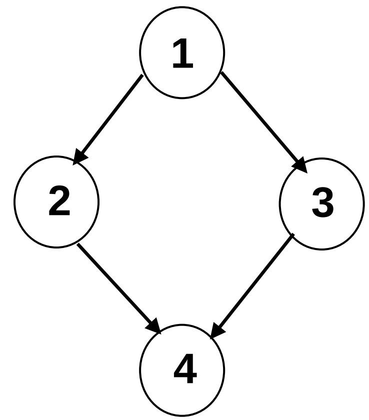
DAG – Directed Acyclic Graph

1: a = 2

2: u = a + 2

3: v = a * 7

4: x = u + v



Scheduling examples:

P1	P2
1	
	2
	3
4	

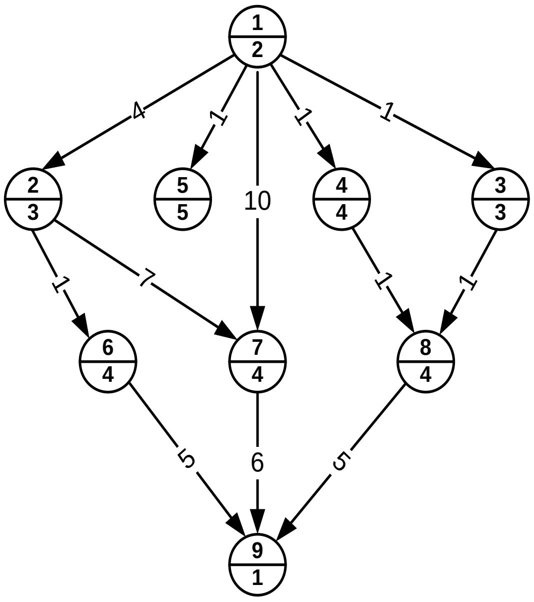
time

P1	P2
1	
2	3
	4

time

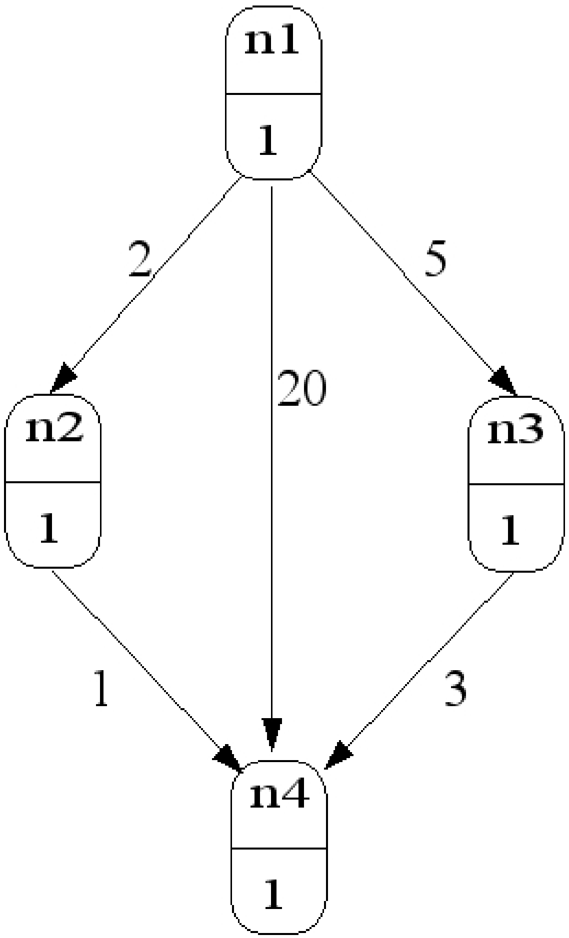
Example of DAG Task

- Assigning priority:
  - *tlevel* (top-level) - for a node *u* is the weight of the longest path from the source node to *u.*
  - *blevel* (bottom-level) - for a node *u* is the weight of the longest path from *u* to an exit node.
- Critical path (*CP*) - the longest path in a graph.
- *ALAP* (As Late As Possible): *ALAP*(*u*) =*CP*−*blevel*(*u*)



Node	tlevel	blevel	ALAP
1	0	23	0
2	6	15	8
3	3	14	9
4	3	15	8
5	3	5	18
6	10	10	13
7	12	11	12
8	8	10	13
9	22	1	22

Task Dependencies Model and DAG Scheduling



$$ALAP(u) = CP - blevel(u)$$

Node	b-level	ALAP
1	22	0
2	3	19
3	5	17
4	1	21

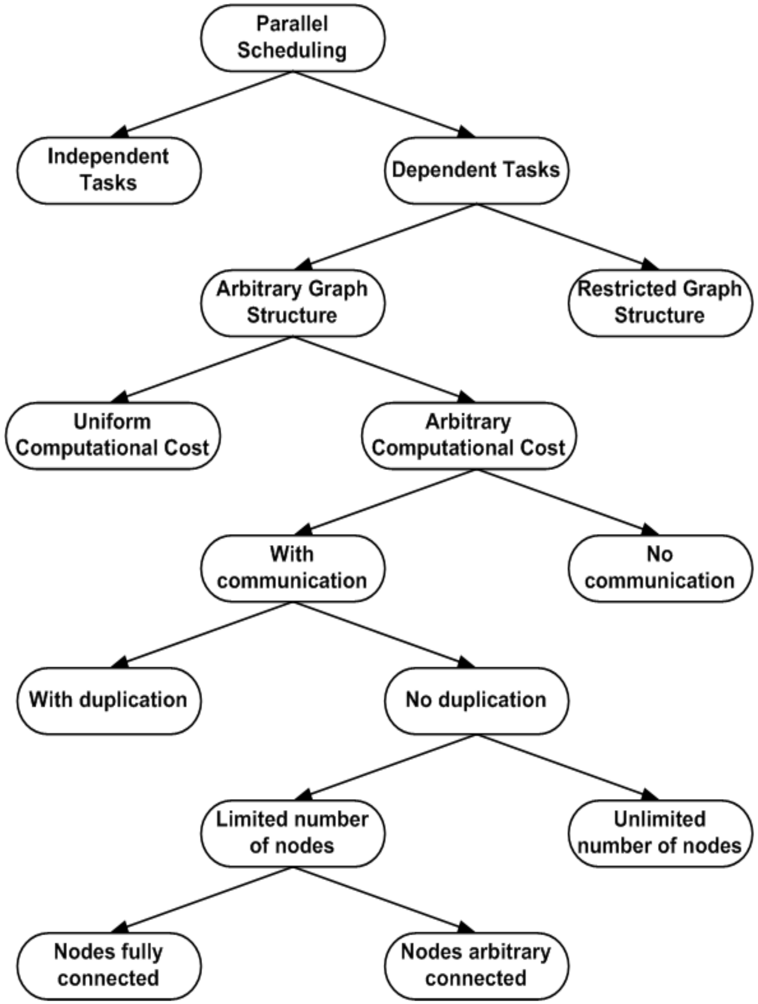
t-level

```
Create TList, a list of nodes in topological order.
foreach node n of TList do
  max = 0
  foreach parent p of n do
    if (tlevel(p) + $\tau_p$ + $c_{p,n}$) > max then
      max = tlevel(p) + $\tau_p$ + $c_{p,n}$
    endif
  endfor
  tlevel(n) = max
endfor
```

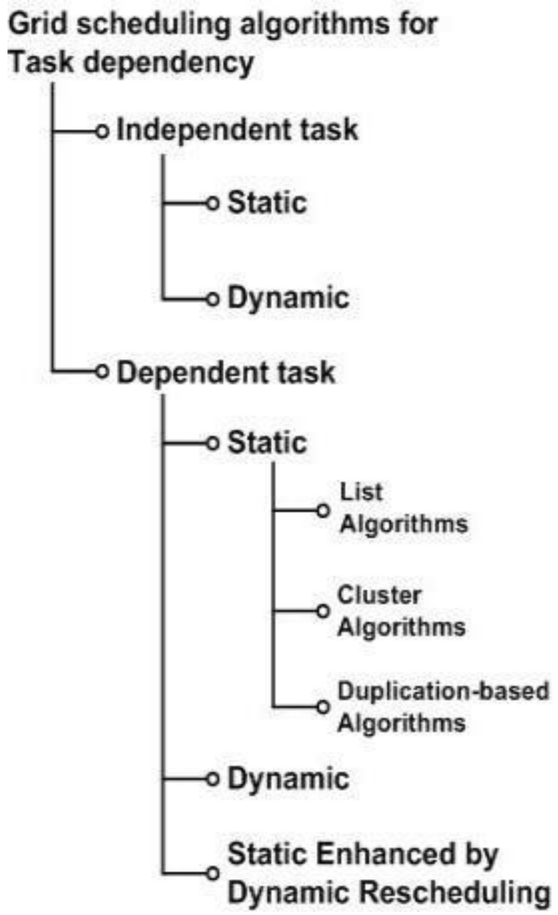
b-level

```
Create RTList, a list of nodes in reversed topological order.
foreach node n of RTList do
  max = 0
  foreach child c of n do
    if ($c_{n,c}$ + blevel(c)) > max then
      max = $c_{n,c}$ + blevel(c)
    endif
  endfor
  blevel(n) = $\tau_n$ + max
endfor
```

Algorithms for DAG Scheduling



A taxonomy of the DAG scheduling problem



Taxonomy of task dependency scheduling algorithms in Grid environments

How to Live with *NP* - hard Scheduling Problems

- Small sized problems can be solved by
  - Mixed integer linear programming
  - Dynamic programming
  - Branch and bound methods
- To solve problems of larger size one has to apply
  - Approximation algorithms
  - Heuristics

Other Types of Scheduling Problems

- Due-date scheduling
- Batching problems
- Multiprocessor task scheduling
- Cyclic scheduling
- Scheduling with controllable data
- Shop problems with buffers
- Inverse scheduling
- No-idle time scheduling
- Multi-criteria scheduling
- Scheduling with no-available constraints
- Scheduling problems are also discussed in connection with other areas:
  - Scheduling and transportation
  - Scheduling and game theory
  - Scheduling and location problems
  - Scheduling and supply chains

Exam's quizzes

- **1.** Descrieţi pe scurt clasificarea $\alpha \mid \beta \mid \gamma$ pentru problema planificării activităţilor.
- **2.** Care sunt principalele funcţii obiectiv folosite în optimizarea planificării activităţilor?
- **3.** Descrieţi modelul de DAG pentru reprezentare a activităţilor.
- **4.** Definiţi *t-level*, *b-level* şi *CP*. Descrieţi pe scurt o modalitate de calcul pentru aceste mărimi.
- **5.** Comentaţi afirmaţia: "*Problema planificării face parte din clasa de probleme NP-Complete*".