CS528

Task Scheduling (Part I)

A Sahu

Dept of CSE, IIT Guwahati

Outline

- Scheduling Concepts
- Independent Tasks, Dependent Tasks

Scheduling Problems

Ref: “Scheduling Algorithm” Book by P. Brucker

Google “Scheduling Algorithm Brucker pdf” to get a PDF copy of the Book
Soft copy will be uploaded to MS Team

Common Terminology

- Given N Tasks need to execute, Goal: time, power, energy, ….
  - Example 10 tasks: A, B, C, D, E, F, G, H, I, J
- **Allocation:** how many compute unit ? Of what type? Tells abut the number : Example 3 processor
- **Binding :** Where to execute
  - Which task on which processor
  - Example : {A, C, F}, {D, E, H, I}, {B, J, G}
- **Scheduling:** When to execute
  - At what time the task execute on binded processor : Gant chart

Scheduling Problems

- Find time slots in which activities (or jobs) should be processed under given constraints.
- Constraints
  - Resource constraints
  - Precedence constraints between activities.
- A quite general scheduling problem is
  - **Resource Constrained Project Scheduling Problem (RCPSP)**

Resource Constraints Project Scheduling Problem

- We have
  - Activities $j = 1, \ldots, n$ with processing times p_j.
  - Resources $k = 1, \ldots, r$. A constant amount of R_k units of resource k is available at any time.
  - During processing, activity j occupies r_{jk} units of resource k for $k = 1, \ldots, r$.
  - Precedence constrains $i \rightarrow j$ between some activities i, j with the meaning that activity j cannot start before i is finished..

RCPSP

- Objective : Determine starting times S_j for all activities j in such a way that
  - at each time t the total demand for resource k is not greater than the availability R_k for $k = 1, \ldots, r$,
  - the given precedence constraints are fulfilled, i. e. $S_i + p_i \leq S_j$ if $i \rightarrow j$,

RCPSP

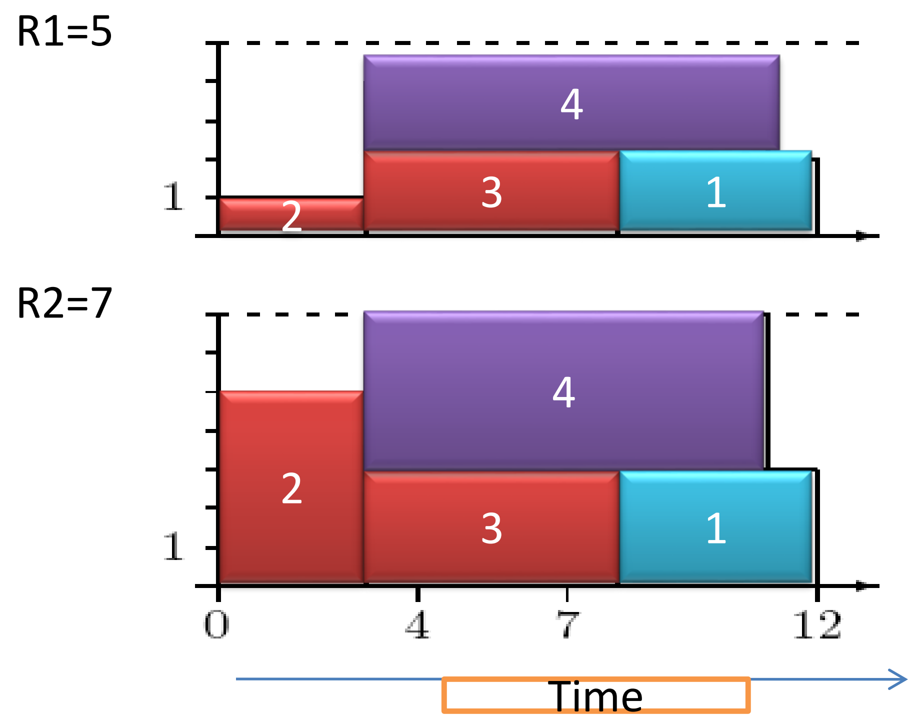
- Some objective function $f(C_1, \ldots, C_n)$ is minimized where $\mathbf{C_j = S_j + p_j}$ is the completion time of activity j.
- The fact that activities j start at time S_j and finish at time $S_j + p_j$ implies that the activities j are not preempted.
- We may relax this condition by allowing **preemption** (activity splitting).

RCPSP: An Example

- Consider a project with n = 4 activities, r = 2
- resources with capacities $R_1 = 5$ and $R_2 = 7$,
- A precedence relation $2 \rightarrow 3$ and the following data:

i	**1**	**2**	**3**	**4**
p_i	4	3	5	8
r_{i1}	2	1	2	2
r_{i2}	3	5	3	4

$2 \rightarrow 3$



A corresponding schedule with minimal makespan

Applications of Scheduling

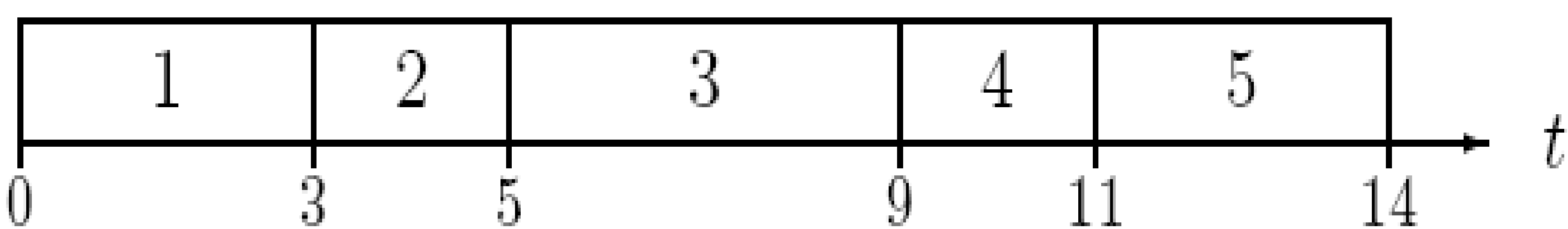
- Production scheduling
- Robotic cell scheduling
- **Computer Processor scheduling**
- Timetabling
- Personnel scheduling
- Railway sc
- Air traffic control, Etc.

Machine Scheduling Problems and their Classification

- Most machine scheduling problems are special cases of the RCPSP.
  - Single machine problems,
    - **Online Problem: FCFS, SJF, SRF, RR…**
  - Parallel machine problems, and
  - Shop scheduling problems, etc.

Single machine problems

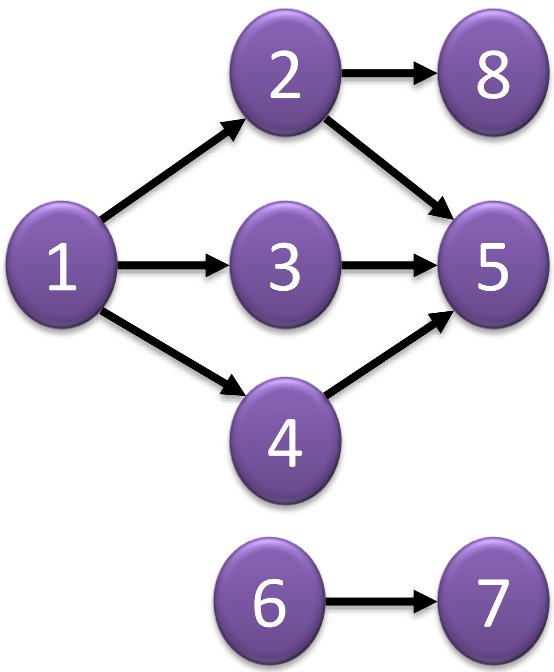
- We have n jobs j =1, ... , n to be processed on a single machine. Additionally precedence constraints between the jobs may be given.
- This problem can be modeled by an RCPSP with $r = 1$, $R_1 = 1$, and $r_{j1} = 1$ for all jobs j.

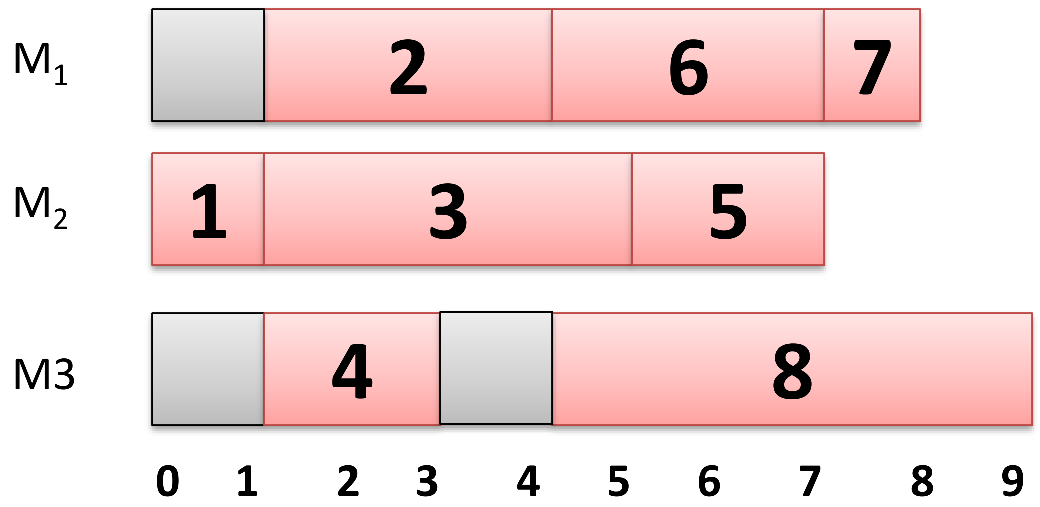


Parallel Machine Problems

- **P:** We have jobs 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 jobs to the machines and to schedule them on the assigned machines.
- This problem corresponds to an RCPSP with $r = 1$, $R_1 = m$, and $r_{j1} = 1$ for all jobs j.

Parallel Machine Problems





Parallel Machine Problems

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

Parallel Machines

Ti	P1	P2	P3	P4
T1	10	10	10	10
T2	12	12	12	12
T3	16	16	16	16
T4	20	20	20	20

P: Identical

Ti	P1	P2	P3	P4
T1	10	15	20	25
T2	12	18	24	30
T3	16	24	32	40
T4	20	30	40	50

Q: Uniform : with speed difference (S_1=1, S_2=2/3, S_3=1/2, S_4=2/5

Ti	P1	P2	P3	P4
T1	10	8	12	2
T2	12	28	25	13
T3	16	4	32	14
T4	20	38	42	22

R: Unrelated : heterogeneous

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 **job characteristics**, and
- γ describes the **objective function(s).**

Machine Environment : α

Symbol	Meaning
1	Single Machine
P	Parallel Identical Machine
Q	Uniform Machine
R	Unrelated Machine
MPM	*Multipurpose Machine*
J	*Job Shop*
F	*Flow Shop*

If the number of machines is fixed to m we write

Pm, Qm, Rm, MPMm, Jm, Fm, Om.

Job Characteristics : β

Symbol	meaning
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) precedence
chains	chain precedence
series-parallel	*a series-parallel precedence graph*

Objective Functions : γ

Two types of objective functions are most common:

- **bottleneck objective functions** $\max \{f_j(C_j) \mid j = 1, \ldots, n\}$, and
- **sum objective functions** $\sum f_j(C_j) = f_1(C_1) + f_2(C_2) + \ldots \ldots + f_n(C_n)$.

C_j is completion time of task j

Objective Functions : γ

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

- Common sum objective functions are:
  - $\Sigma\ \mathbf{C_j}$ (mean flow-time)
  - $\Sigma\ \omega_j\ \mathbf{C_j}$ (weighted flow-time)

Objective Functions : γ

- Σ $\mathbf{U_j}$ (number of late jobs) and Σ ω_j $\mathbf{U_j}$ (weighted number of late jobs) where $U_j = 1$ if $C_j > d_j$ and $U_j = 0$ otherwise.

- Σ $\mathbf{T_j}$ (sum of tardiness) and Σ ω_j $\mathbf{T_j}$ (weighted sum of tardiness/lateness) where the tardiness of job j is given by

$$T_j = \max \{ 0, C_j - d_j \}.$$

Examples of Scheduling Problem

- $1 \mid prec; p_j = 1 \mid \Sigma\, \omega_j C_j$
- $P2 \mid \mid C_{max}$
- $P \mid p_j = 1; r_j \mid \Sigma\, \omega_j U_j$
- $R2 \mid chains; pmtn \mid C_{max}$
- $R \mid n = 3 \mid C_{max}$
- $P \mid p_{ij} = 1; outtree; r_j \mid \Sigma\, C_j$
- $Q \mid p_j = 1 \mid \Sigma\, T_j$

Polynomial algorithms

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

Example

1 | | $\Sigma\ \omega_j C_j$ can be solved by

Scheduling the jobs in an ordering of non-increasing ω_j/p_j - values.

Complexity: O(n log n)

Polynomial algorithms for $1 \mid\mid \Sigma C_j$

Example

$1 \mid\mid \Sigma C_j$ can be solved by

Scheduling the jobs in an ordering of non-increasing $1/p_j$ - values. == > SJF

$C_i = Q_i + P_i$: Waiting time + Processing time

(SJF is optimal)

Complexity: O(n log n)