CS528
Task Scheduling (LS)

A Sahu
Dept of CSE, IIT Guwahati
Outline

• List Scheduling Concepts and Proof
• Task with Hard Deadlines
• Real Time Tasks
• Energy Efficiency
• Energy Efficient Scheduling
• Reliability Aware Scheduling
\[ P_m \mid p_j \mid C_{\text{max}} \]

**Minimum makespan scheduling**

- \( P_m \mid p_j \mid C_{\text{max}} \) in NPC
- Given processing times for \( n \) jobs, \( p_1, p_2, \ldots, p_n \), and an integer \( m \)
- Find an assignment of the jobs to \( m \) identical machines
- So that the completion time, also called the makespan, is minimized.
Linear Programming Solution to Scheduling Problem

\[ x_{ij} = \{0, 1\} \quad \text{whether job } j \text{ is scheduled in machine } i \]

\[ \min \quad T \]

\[ \sum_{i=1}^{m} x_{ij} = 1 \quad \text{for each job } j \quad \text{Each job is scheduled in one machine.} \]

\[ \sum_{j=1}^{n} x_{ij} \cdot p_{ij} \leq T \quad \text{for each machine } i \quad \text{Each machine can finish its jobs by time } T \]

\[ 0 \leq x_{ij} \quad \text{for each job } j, \text{ machine } i \]

A Sahu
Minimum makespan scheduling: Arbitrary List

• List Scheduling : Approximation
• Algorithm
  – 1. Order the jobs arbitrarily.
  – 2. Schedule jobs on machines in this order, scheduling the next job on the machine that has been assigned the least amount of work so far.
• Above algorithm achieves an approximation guarantee of 2
Load Balancing: List Scheduling

A B C D E
F G
H I J

Machine 1
Machine 2
Machine 3

Time

A Sahu
Load Balancing: List Scheduling

Machine 1
Machine 2
Machine 3

Time

A Sahu
Load Balancing: List Scheduling

A Sahu
Load Balancing: List Scheduling

Machine 1

Machine 2

Machine 3

Time

A Sahu
Load Balancing: List Scheduling

- Machine 1: A
- Machine 2: B, D
- Machine 3: C

Time

A Sahu
Load Balancing: List Scheduling
Load Balancing: List Scheduling

Machine 1
A E Machine 2
B D
C F
Load Balancing: List Scheduling
Load Balancing: List Scheduling

A Sahu
Load Balancing: List Scheduling

A Sahu
Load Balancing: List Scheduling
Load Balancing: List Scheduling

Optimal Schedule

List schedule
LS is 2 APPRX
**Algorithm:** List scheduling

**Basic idea:** In a list of jobs, schedule the next one as soon as a machine is free.

Good or bad?

A Sahu
List Scheduling is “2-approximation” (Graham, 1966)

Algorithm: List scheduling
Basic idea: In a list of jobs,
schedule the next one as soon as a machine is free

S A

job f finishes last, at time A

cmpare to time OPT of best schedule: how?

A Sahu
List Scheduling is “2-approximation”

job f finishes last, at time A

c ompare to time OPT of best schedule: how?

(1) job f must be scheduled in the best schedule at some time:
   \[ f \leq \text{OPT}. \quad \Rightarrow \quad A - S \leq \text{OPT}. \]

(2) up to time S, all machines were busy all the time, and OPT cannot beat that, and job f was not yet included:
   \[ S < \text{OPT}. \]

(3) both together:
   \[ A = A - S + S < 2 \text{OPT}. \]

   “2-approximation” (Graham, 1966)