Coping with NP-Complete Problems

- A combinatorial optimization problem P is either a minimization or a maximization problem and consists of the following three parts
  - A set D_P of instances
  - For each instance I a finite set $S_P(I)$ of candidate solutions for I
  - A function m_P that assigns to each instance I and each candidate solution s ϵ $S_P(I)$ a positive rational number $m_P(I, s)$ called the solution value for s.

- If a combinatorial optimization problem is NP-Complete,

  it might take an absurdly long time (e.g., 300 centuries) to find the optimal solution for the problem.

- Probably cannot wait 300 centuries to find the solution

- However, the problem does not go away. One still has to find a solution!

Approximation Algorithms

- If the optimal solution is unattainable then it is reasonable to sacrifice optimality and settle for a “good” (close to optimal) feasible solution that can be computed efficiently.

- We would like to sacrifice as little optimality as possible, while gaining as much as possible in efficiency.

- Trading-off optimality in favor of tractability is the paradigm of approximation algorithms

- Dorit. S. Hochbaum

Approximation Algorithms for NP-Hard Problems

Approximation Algorithms

ε-approximation algorithm

Let Π be an optimization (minimization or maximization) problem and A an algorithm which, given an instance I of Π, returns a feasible (but not necessarily optimal) solution denoted by APP(I). Let the optimal solution be denoted by OPT(I). The algorithm A is called an ε-approximation algorithm for Π for some $\varepsilon \geq 0$ if and only if

$$| \mathrm{APP}(I) - \mathrm{OPT}(I) | / \mathrm{OPT}(I) \leq \varepsilon \text{ for all instances } I$$

Yet Another Job Scheduling Problem

- $P_1, \ldots, P_m$: A set of m independent processors with similar (dissimilar) performance characteristics
- $T_1, \ldots, T_n$: A set of n independent tasks with no ordering relationship between them
- If the processors are dissimilar (heterogeneous computing environment), the execution time of a task on different processors are different.
- t_{ij} = Execution time of task T_i on Processor P_j
- If the processors are similar (homogeneous computing environment), the execution time of a task on different processors are same.

Yet Another Job Scheduling Problem

- Total execution time used by Processor P_j is the sum of the execution times of the tasks assigned to this processor
- Makespan of an assignment is maximum of the total execution times of individual processors
- Objective: Find the assignment that minimizes the makespan
- Question: How difficult is it to find the schedule with the minimum makespan?

Algorithm for the Job Scheduling Problem

(Heterogeneous Environment)

Step1: for j:=1 to m do

$F_j := 0;$

Step2: for i:=1 to n do

begin

Step 2.1 Find k where k is the smallest integer j for which $F_j + t_{i,j}$ is minimum $(1 \le j \le m)$

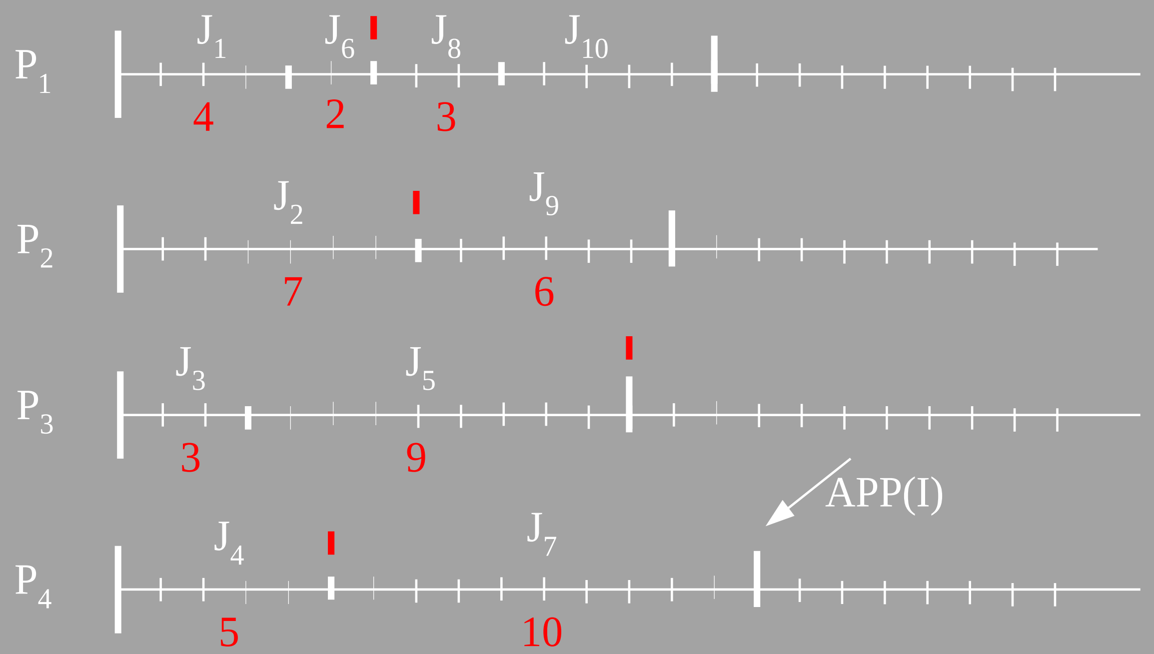
Step 2.2 Assign task T_i to processor P_k

Step 2.3 Update $F_k \Leftarrow F_k + t_{i,j}$

end

Jobs $\Rightarrow \{J_1, J_2, J_3, J_4, J_5, J_6, J_7, J_8, J_9, J_{10}\}$

Execution Time $\Rightarrow \{4, 7, 3, 5, 9, 2, 10, 3, 6, 8/5\}$



Case 1 $t_{10} + F_1 \geq F_4 \;\; \Rightarrow \;\; APP(I) = F_1 + t_{10}$

Case 2 $t_{10} + F_1 < F_4 \;\; \Rightarrow \;\; APP(I) = F_4$

Case 1: $App(I) \le F_1 + t_{10}$

$$App(I) \le F_1 + t_{10}$$
$$App(I) \le F_2 + t_{10}$$
$$App(I) \le F_3 + t_{10}$$
$$App(I) \le F_4 + t_{10}$$

$$App(I) \le F_j + t_n$$
$$(1 \le j \le m)$$

$$m\,App(I) \le F_1 + F_2 + \dots + F_m + mt_n$$

$$= \sum_{i=1}^{n-1} t_i + mt_n$$

$$= \sum_{i=1}^{n} t_i + (m-1)t_n$$

or $$App(I) \le \frac{1}{m}\sum_{i=1}^{n} t_i + \frac{m-1}{m} t_n$$

$$\le OPT(I) + \frac{m-1}{m} OPT(I)$$

$$App(I) \le (2 - \frac{1}{m}) OPT(I)$$

as $$OPT(I) \ge \frac{1}{m}\sum_{i=1}^{n} t_i \quad .....(1)$$

and $$OPT(I) \ge \max\{t_i \mid 1 \le i \le m\}$$

Case 2 $App(I) = F_4$

F_4 was the finish time at the end of scheduling jobs J_1,.., J_7. Let the finish time on the processors at this time be $F_1', F_2', ..., F_m'$ (before scheduling job J_7).

$$App(I) \le F_1' + t_7$$
$$App(I) \le F_2' + t_7$$
$$App(I) \le F_3' + t_7$$
$$App(I) \le F_4' + t_7$$

$$App(I) \le F_j' + t_k$$
$$(1 \le j \le m)$$

$$m\,App(I) \le F_1{}' + F_2{}' + \ldots + F_m{}' + mt_k$$

$$= \sum_{i=1}^{k-1} t_i + mt_k = \sum_{i=1}^{k} t_i + (m-1)t_k$$

$$\le \sum_{i=1}^{n} t_i + (m-1)t_k$$

or

$$App(I) \le \frac{1}{m}\sum_{i=1}^{n} t_i + \frac{m-1}{m} t_k$$

$$App(I) \le OPT(I) + \frac{m-1}{m} OPT(I)$$

$$= (2 - \frac{1}{m}) OPT(I)$$