Solvers: Minimize Maximum Stretch time during Process Scheduling

Md Maruf Hossain
Kyle Tibbetts
Anirudh Narayana

I. PROBLEM

Input: n tasks of processing time p_i to schedule on one processor and release time r_i.
Output: For each task, a start time $\sigma_i \geq 0$. Task i can not start before it is released: $\sigma_i \geq r_i$. Task i completes at $C_i = \sigma_i + p_i$. No two tasks can be excuting at the same time: $\forall i, j[\sigma_i; C_i[\cap[\sigma_j; C_j[= \emptyset$

Metric: The flow time of a task is the time that elapsed between its release and completion: $F_i = C_i - r_i$. The stretch of a task is its Flow time normalized by its processing time: $S_i = \frac{F_i}{p_i}$. Minimize maximum stretch: $S_{max} = max_i S_i$

II. IMPORTANCE OF THE PROBLEM

This problem performs the task of taking into consideration flow time, and processing time, which will account for time taken of a process and the resulting schedule designed will be more efficient. The following are applications of this problem:

- Scheduling transport of goods in large industries.
- Minimizing total cost of a telecommunication networks by optimizing topology. (An expanded case of this problem).
- Production Planning.

III. SUBSETS AND PARTITIONING

The subsets of the problem is the permutation of the process.

$$S = P! \qquad \text{Where } P = \{p_1, \ldots, p_n\}$$

We can partition the set S into n subsets where,

$$S = S_1 \cup S_2 \cup \cdots \cup S_n \qquad \text{Where } S_i = \{p_{i1}, \ldots, p_{in}\} \text{ and } S_i \subset S | S_{p_1} = p_i$$

IV. PARTITION ORDER

In this problem, release time and length of the process is the key point to minimize the max stretch of the scheduling. If we choose a partition where the release time of the first process is smaller compare to others then it can converge to the optimal solution faster.

explore S_i branch before S_j if $r_{i1} \leq r_{j1}$

V. BOUND AND RELAXATIONS

In this problem, we have n different sets that can generate n different max stretch for the scheduling. We are going to maintain one global variable S_{gmin} for all and one local variable $S_l max$ for each branch to maintain the bound. Here S_{gmin} always holds the global minimum max stretch and local S_{lmax} only holds the maximum stretch of a particular branch. We are going to once initialize the S_{gmin}

$$S_{gmin} = \frac{\sum_{i=0}^{n} p_i - min_{i \in \{1,\ldots,n\}} r_i}{min_{i \in \{1,\ldots,n\}} p_i}$$

On the other hand S_{lmax} will be initialized with 0 for every branch and in every depth of the tree it will update to maintain the max stretch of that branch. A branch will be pruned if $S_{lmax} \geq S_{gmin}$. If a branch reach at the leaf then it will update $S_{gmin} = min\{S_{gmin}, S_{lmax}\}$.

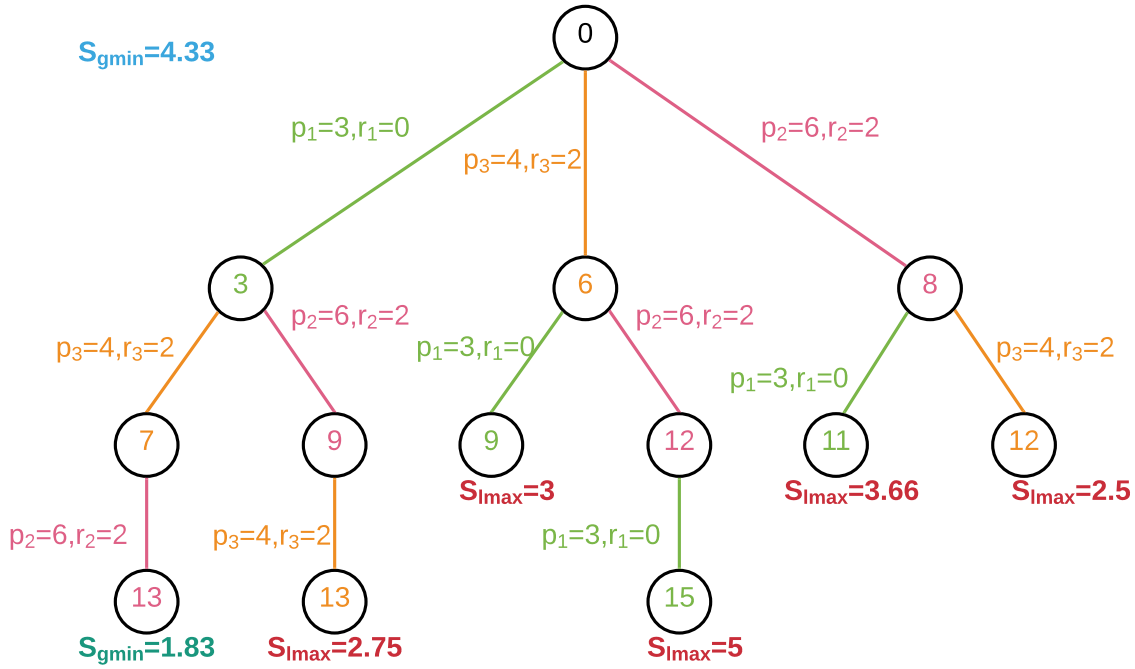

Fig. 1. Consider 3 process $p_1 = 3, p_2 = 6, p_3 = 4$ with release time $r_1 = 0, r_2 = 2, r_3 = 2$ need to schedule in a system where no two job can schedule at a time.

VI. Traversal Order and Optimal Solution

Solution always follow the depth first search(DFS) traversal mechanism. But in every depth, if the immediate next branches has the processes with the same release time then it always traverse the branch first that contain smaller process time.

VII. Weak Dominance Properties

If two process has the same release time then always schedule the job that has smaller process time. If $r_i = r_j = t_r$ and $p_i < p_j$ then if we schedule p_i first at start time t_σ,

$$\begin{aligned} Stretch_{i1} &= \frac{t_\sigma + p_i - t_r}{p_i} \\ Stretch_{j1} &= \frac{t_\sigma + p_i + p_j - t_r}{p_j} \end{aligned} \tag{1}$$

Now if schedule p_j before p_i

$$\begin{aligned} Stretch_{j2} &= \frac{t_\sigma + p_j - t_r}{p_j} \\ Stretch_{i2} &= \frac{t_\sigma + p_i + p_j - t_r}{p_i} \end{aligned} \tag{2}$$

Here, $Stretch_{j2} < Stretch_{j1}$ but $Stretch_{i2}$ is greater than both $Stretch_{i1}$ and $Stretch_{j1}$. So, Schedule the smaller job with same release time can give better result.