CENG567: Homework #3

Yiğit Sever

December 1, 2020

1 Job Scheduling

First we will state the variables we will use. A job $n_i \in n$ is split into preprocessing $p_i \in p$ and indexing $f_i \in f$ per the question text. With the given question text, we will ignore any kind of utilisation or efficiency concerns. In other words, we will not care about the number of PCs we employ nor the time they will stay idle. So, every indexing job $f_i \in f$ will be run on a separate PC.

Since we cannot pipeline the preprocessing jobs $p_i \in p$, any kind of real choice we have will be done on the $f_i \in f$. The trivial case is when the preprocessing jobs are the "bottleneck" of the system, where;

$$\forall p_i \in p > \forall f_i \in f$$

The completion time in this case is minimized by sorting the $f_i \in f$ and placing the *shortest* f_i last.

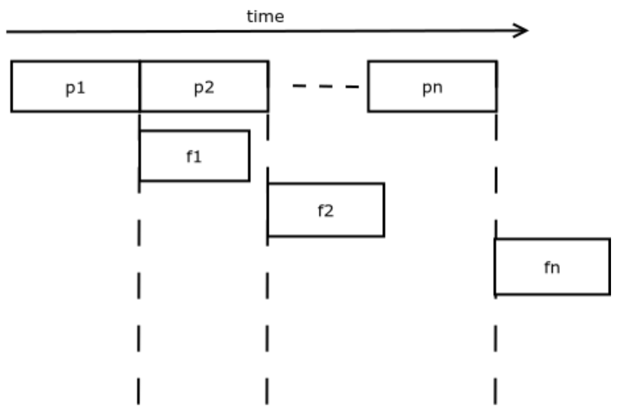
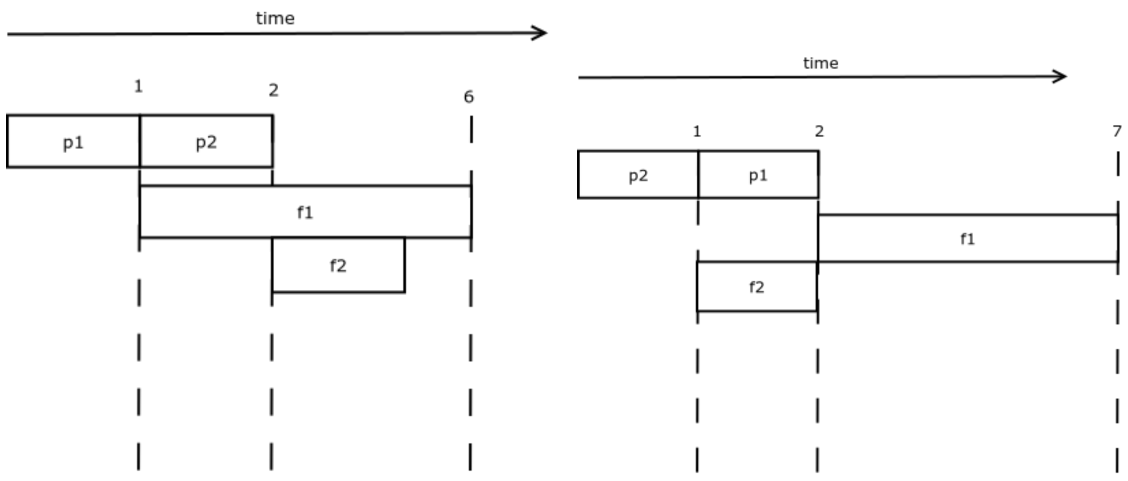


Figure 1: The trivial case where preprocessing jobs are all longer than the indexing jobs

This approach extends to the case where the preprocessing time can be shorter than the indexing time as well. In the example given in Figure 2, two preprocessing jobs p_1 and p_2 both take 1 unit of time. Whereas, the indexing jobs tied to the preprocessing jobs f_1 and f_2 take 5 and 1 units of time, respectively. By scheduling the preprocessing task that is tied to the longest indexing job p_1 first, we can finish the whole computation in 6 time units which takes 7 time units in the other case.

Formally, the algorithm we propose simply sorts the $f_i \in f$ in $\mathcal{O}(n \log(n))$ time and schedules the jobs with respect to $f_i \in f$ (so the $p_i \in p$ tied to $f_i \in f$) from longest to shortest.



(a) The longest task is scheduled first

(b) The longest task is scheduled last

Figure 2: The worked example of the cases