IN4301 Advanced Algorithm - Homework Assignment 1

> Lu Dai (4506677) L.Dai-1@student.tudelft.nl

Question 1

1.

Considering the given algorithm, the main idea is sorting the jobs by their release time in increasing order and taking this sorted order as the schedule. The schedule guarantees that each job can be processed as soon as possible once it is released. That is to say, each job reaches its smallest completion time and also the minimum flow time. In addition, interchanging the position of jobs with the same release times will make no difference to the maximum flow time among them. Thus, we do not need to do further changes for the schedule.
Above all, the first step can be considered as sorting problem which reaches the time complexity O(nlog(n)) (taking the fastest sorting algorithms) while the second step is to process the jobs following schedule which requires O(n).
Thus, the time complexity of the algorithms can be considered as O(nlog(n)).

2.

Following the algorithm given, the schedule is (2,2), (3,2), (1,2), (1,3), (1,3) and the maximum flow time is 7. In total, there are A_5^5 possible schedules. As there are only tow release times 2 and 3 and the minimum processing time is 1. Thus, in any schedule, there exists no spare time for the machine once starting processing. The whole time for processing all the jobs is 8. Considering the machine starts processing with a job released either at 2 or at 3 and ends processing with a job either at 2 or at 3. There are only four possible situations in the schedule with which the last job in the schedule guarantees one of the maximum flow time:

- The first job in the schedule is released at 2 and the last job in the schedule is released at 2, then all the jobs are finished at 10 and the maximum flow time is 8;
- The first job is released at 2 and the last job is released at 3, then all the jobs are finished at 10 and the maximum flow time is 7;
- The first job is released at 3 and the last job is released at 2, then all the jobs are finished at 11 and the maximum flow time is 9;
- The first job is released at 3 and the last job is released at 3, then all the jobs are finished at 11 and the maximum flow time is 8;

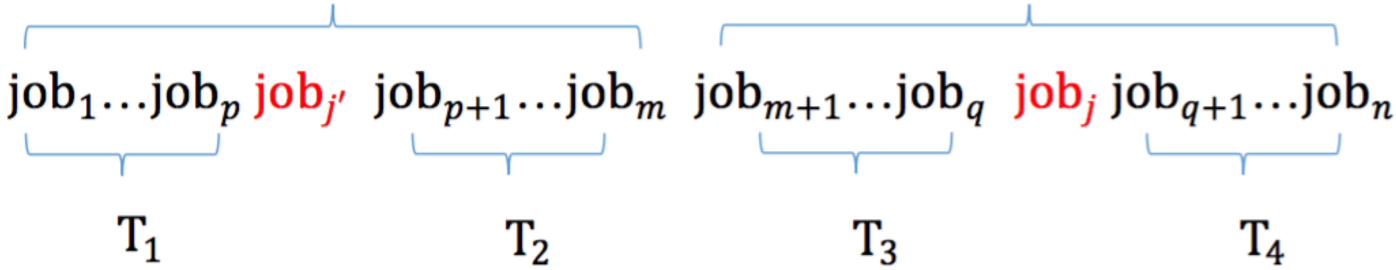
Thus, the algorithm given reaches the optimal solution.

3.

According to question, we know that there are only two different release times r < r', suppose the following:

> m jobs with release time r: job_1 ... job_m
> n-m jobs with release time r': job_{m+1} ... job_n
> There exists job j (released at r) scheduled after job j' (release at r'), which is out of the order of their release time, shown in the picture below.
> t_i denotes the processing time of job_i, c_i denotes the completion time of job_i, f_i denotes the flow

> time of job_i



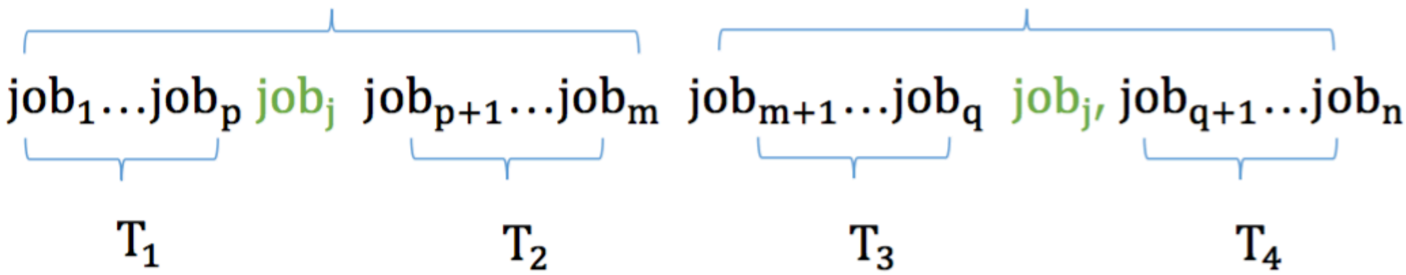
$T_1 = \sum_{i=1}^{p} t_i \quad T_2 = \sum_{i=p+1}^{m} t_i \quad T_3 = \sum_{i=m+1}^{q} t_i \quad T_4 = \sum_{i=q+1}^{n} t_i$

As it is easy to know, there are only two positions which can possibly hit the maximum flow times, they are either the last job in the schedule which is released at r or at r':

(1 ⩽ j ⩽ m; m+1 ⩽ j' ⩽ n)

- if r' <= r + T_1 (machine has no spare time):
  - the last job released at r in the schedule (job_j in this case):
    $f_j = T_1 + t_{j'} + T_2 + T_3 + t_j$
  - the last job released at r in the schedule (job_n in this case):
    f_n = r + $T_1 + t_{j'} + T_2 + T_3 + t_j + T_4$ - r'
- if r' > r + T_1 (machine has spare time):
  - the last job released at r in the schedule (job_j in this case):
    $f_j = T_1$ + [r' - (r + T_1)] + $t_{j'} + T_2 + T_3 + t_j$
  - the last job released at r' (job_n in this case):
    f_n = r + T_1 + [r' - (r + T_1)] + $t_{j'} + T_2 + T_3 + t_j + T_4$ - r'

After interchanging the position of job_j and $job_{j'}$, we got the new schedule, shown below:



Re-calculate the maximum flow time following the situations described above:

(1 ⩽ j ⩽ m; m+1 ⩽ j' ⩽ n)

- if r' <= r + $T_1 + t_j + T_2$ (machine has no spare time):
  - the last job released at r in the schedule (job_m in this case):
    $f_m = T_1 + t_j + T_2$
  - the last job released at r in the schedule (job_n in this case):
    f_n = r + $T_1 + t_j + T_2 + T_3 + t_{j'} + T_4$ - r'
- if r' > r + $T_1 + t_j + T_2$ (machine has spare time):
  - the last job released at r in the schedule (job_j in this case):
    $f_m = T_1$ + [r' - (r + $T_1 + t_j + T_2$)] + $t_j + T_2$
  - the last job released at r' (job_n in this case):
    f_n = r + T_1 + [r' - (r + $T_1 + t_j + T_2$)] + $t_{j'} + T_2 + T_3 + t_j + T_4$ - r'

Above, no matter on which position the maximum flow time happens, the new schedule always holds the smaller (or the same)maximum flow time than the previous schedule in every situations discussed.

4.

Finding a schedule that minimizes $max_j \{f_j\}$ on 2 or more machines is NP-hard because it is reducible to Load Balancing problem. As Load Balancing is NP-Hard, thus this problem is NP-hard.
The problem can be simplified to Load Balancing problem as follows:

- Classify jobs with the **same release time** into the same group and treat each group as a sub-problem of the original problem.
- Sort the sub-problems in an increasing manner by release time.
- Sovle each sub-problems in the above order. For each sub-problem, due to the same release, finding the the maximum flow time equals finding the maximum makespan which is actully load balancing problem.

The above is applicapable because it follows the optimal algorithms mentioned in previous questions. That is, the jobs released first get processed first.

Question 2

1.

As $t_n > \frac{1}{3} T^*$, if one machine has three jobs, then $3 * t_n > T^*$.
In this case, we can't reach the optimal schedule. Thus, at most 2 jobs can be scheduled to each machine.

2.

As proved in the previous question that at most 2 jobs per machine. Thus 2m should satifies all the jobs, that is,

$$2m \geq n$$

3.

(a) As $n \leq m$, We already know the fact:

$$max_j\{t_j\} \leq T^*$$

$$T^* \leq T \leq \frac{3}{2} T^*$$

if , each machine will receive at most one job. The maximum makespan equals the longgest processing time which also equals OPT, which holds an optimal schedule:

$$T^* \leq T = max_j\{t_j\} \leq T^*$$

$$T = T^*$$

(b) As $m < n < 2m$, this means at most 2 jobs for each machine. Assume maximum makespan is achieved on machine i:

$$T = t_i + t_j$$

where $1 \leq i \leq m, m+1 \leq j \leq 2m, t_i \geq t_j$.
Machine starts receiving the second job from job_{m+1} where job_{m+1} will be assigned to machine m and job_{m+2} will assigned to machine m-1 according to the greedy algorithm. Thus, the algorithm will have the optimal schedule as:

$$\begin{aligned} T_1 &= t_1 + t_{2m} \\ T_2 &= t_2 + t_{2m-1} \\ &\ldots \\ T_{m-1} &= t_{m-1} + t_{m+2} \\ T_m &= t_m + t_{m+1} \end{aligned}$$

where

$$t_{2m} \leq t_{2m-1} \leq \ldots \leq t_{m+2} \leq t_{m+1} \leq t_m \leq t_{m-1} \leq \ldots \leq t_2 \leq t_1$$

Thus, the maximum makespan will be

$$T = max\{T_1, T_2, \ldots, T_{m-1}, T_m\}$$
$$T = max\{max\{T_1, max\{T_2, \ldots, max\{T_{m-1}, T_m\}\}\}\}$$

Starting from $max\{T_{m-1}, T_m\}$, there are only four possible combinations which are:

$$T_{m-1} = t_{m-1} + t_{m+2}, T_m = t_m + t_{m+1}$$
$$T'_{m-1} = t_{m-1} + t_{m+1}, T'_m = t_m + t_{m+2}$$

as $t_{m+2} \leq t_{m+1} \leq t_m \leq t_{m-1}$, we found that

$$T_{m-1} \leq T'_{m-1}, T'_m \leq T'_{m-1}, T_m \leq T'_{m-1}$$

which means T'_{m-1} is the biggest among them, thus it can be easily observed:

$$max\{T_m, T_{m-1}\} \leq max\{T'_m, T'_{m-1}\}$$

same applies for,

$$max\{T_m, T_{m-2}\} \leq max\{T'_m, T'_{m-2}\}$$
$$max\{T_{m-1}, T_{m-2}\} \leq max\{T'_{m-1}, T'_{m-2}\}$$

thus, we can have

$$max\{max\{T_m, T_{m-1}\}, T_{m-2}\} \leq max\{max\{T'_m, T'_{m-1}\}, T'_{m-2}\}$$

$$\vdots$$

$$max\{max\{T_1, max\{T_2, \ldots, max\{T_{m-1}, T_m\}\}\}\} \leq max\{max\{T'_1, max\{T'_2, \ldots, max\{T'_{m-1}, T'_m\}\}\}\}.$$

Above all, the schedule given by the Sorted-Balance algorithm holds an optimal solution.