Scheduling jobs on one machine

Table of Contents

This task is about finding an optimal schedule for n jobs on a single machine, if the job i can be processed in t_i time, but for the t seconds waiting before processing the job a penalty of $f_i(t)$ has to be paid.

Thus the task asks to find such an permutation of the jobs, so that the total penalty is minimal. If we denote by π the permutation of the jobs (π_1 is the first processed item, π_2 the second, etc.), then the total penalty is equal to:

$$F(\pi) = f_{\pi_1}(0) + f_{\pi_2}(t_{\pi_1}) + f_{\pi_3}(t_{\pi_1} + t_{\pi_2}) + \cdots + f_{\pi_n}\left(\sum_{i=1}^{n-1}$$

Solutions for special cases

Linear penalty functions

First we will solve the problem in the case that all penalty functions $f_i(t)$ are linear, i.e. they have the form $f_i(t) = c_i \cdot t$, where c_i is a non-negative number. Note that these functions don't have a constant term.

Otherwise we can sum up all constant term, and resolve the problem without them.

Let us fixate some permutation π, and take an index $i = 1 \dots n-1$. Let the permutation π' be equal to the permutation π with the elements i and $i+1$ switched. Let's see how much the penalty changed.

$$F(\pi') - F(\pi) =$$

It is easy to see that the changes only occur in the i-th and $(i+1)$-th summands:

$$= c_{\pi'_i} \cdot \sum_{k=1}^{i-1} t_{\pi'_k} + c_{\pi'_{i+1}} \cdot \sum_{k=1}^{i} t_{\pi'_k} - c_{\pi_i} \cdot \sum_{k=1}^{i-1} t_{\pi_k} - c_{\pi_{i+1}} \cdot \sum_{k=1}^{i} t_{\pi_k}$$

$$= c_{\pi_{i+1}} \cdot \sum_{k=1}^{i-1} t_{\pi'_k} + c_{\pi_i} \cdot \sum_{k=1}^{i} t_{\pi'_k} - c_{\pi_i} \cdot \sum_{k=1}^{i-1} t_{\pi_k} - c_{\pi_{i+1}} \cdot \sum_{k=1}^{i} t_{\pi_k}$$

$$= c_{\pi_i} \cdot t_{\pi_{i+1}} - c_{\pi_{i+1}} \cdot t_{\pi_i}$$

It is easy to see, that if the schedule π is optimal, than any change in it leads to an increased penalty (or to the identical penalty), therefore for the optimal schedule we can write down the following condition:

$$c \cdot t_{\pi_{i+1}} - c_{\pi_{i+1}} \cdot t_{\pi_i} \geq 0 \quad \forall i = 1 \dots n-1$$

And after rearranging we get:

$$\frac{c_{\pi_i}}{t_{\pi_i}} \geq \frac{c_{\pi_{i+1}}}{t_{\pi_{i+1}}} \quad \forall i = 1 \dots n-1$$

Thus we obtain the **optimal schedule** by simply **sorting** the jobs by the fraction $\frac{c_i}{t_i}$ in non-ascending order.

It should be noted, that we constructed this algorithm by the so-called **permutation method**: we tried to swap two adjacent elements, calculated how much the penalty changed, and then derived the algorithm for finding the optimal method.

Exponential penalty function

Let the penalty function look like this:

$$f_i(t) = c_i \cdot e^{\alpha \cdot t},$$

where all numbers c_i are non-negative and the constant α is positive.

By applying the permutation method, it is easy to determine that the jobs must be sorted in non-ascending order of the value:

$$v_i = \frac{1 - e^{\alpha \cdot t_i}}{c_i}$$

Identical monotone penalty function

In this case we consider the case that all $f_i(t)$ are equal, and this function is monotone increasing.

It is obvious that in this case the optimal permutation is to arrange the jobs by non-ascending processing time t_i.

The Livshits-Kladov theorem

The Livshits-Kladov theorem establishes, that the permutation method is only applicable for the above mentioned three cases, i.e.:

- Linear case: $f_i(t) = c_i(t) + d_i$, where c_i are non-negative constants,
- Exponential case: $f_i(t) = c_i \cdot e_{\alpha \cdot t} + d_i$, where c_i and α are positive constants,
- Identical case: $f_i(t) = \phi(t)$, where ϕ is a monotone increasing function.

In all other cases the method cannot be applied.

The theorem is proven under the assumption that the penalty functions are sufficiently smooth (the third derivatives exists).

In all three case we apply the permutation method, through which the desired optimal schedule can be found by sorting, hence in $O(n \log n)$ time.