WaitNoMore

Isaac Gutt

April 9, 2022

1 The Algorithm

As the notation above explains, the algorithm used for this problem is to go through every job i, each time multiplying w$_i$ with W$_i$, and add it to a variable x. Once all the jobs are passed through, x should return the minimum weighted waiting time for all the jobs.

Because of the multiplication factor, the ordering in which the jobs are done will have a big difference on the waiting time. My greedy intuition is that the multiplication factors, which are the weight and the growing waiting time, should try to be kept as small as possible from the beginning.

Because both weights and duration can have big impact on waiting times, we can't completely prioritize one over the other. Therefore, my greedy intuition was to sort the jobs based on duration/weight ratios, sorting by lowest to highest ratio. Typically, this will have the bigger weights going first, since the weights are the denominators, and bigger durations going last because they are the numerators.

Sorting the jobs in this way and running them straight through the summation will be my greedy solution.

2 Proving Correctness

Let x be the solution produces by my greedy algorithm, and X^* be some optimal solution. If X = X^* then the greedy algorithm is optimal.

Theorem 1. *$X = X^*$, the greedy algorithm is optimal.*

Proof by Induction:
Base Case: intialize H = [0], at that point $H^* = 0$ also.
Inductive Step: assume that after job$_i$ has been added to H, there exists an optimal order of jobs H^* such that $H^*[0..j] = H[0..j]$

Suppose the greedy algorithm adds job k as H[j+1] (the next job to finish), we know that $k \leq H^*[j+1]$ because $H^*[j+1] \geq r + H^*[j] = r + H[j]$. Greedy chose k as the lowest possible waiting time for the next job.

Consider H^*x as the optimal solution after we added k (greedy's choice) to it's ordering as $H^*x[j+1] = k$. This is still feasible, because
$H^*x[j+1] = k \leq r + H^*x[j]$.
Since H^* and H^*x were equal at j, and k is the lowest duration to add now, H^*x is better or worse at this point.

Now, we can move H^*x back into H^* and not change the optimal solution, by doing the same job H^* did with [j+1] as $H^*x[j+2]$. By induction, we can continue doing this the rest of the algorithm and end up with the same optimal result.

Proof. X = X^*, the greedy algorithm is optimal. $\square$

3 Running Time

My algorithm begins with an O(n) run filling a list with the jobs, I created a job class that compares the jobs and weights to order the list with the greedy solution. Then there is a Collections.Sort() call which is O(n log n), and then another O(n) run on the sorted list to calculate the waiting time. Since the dominant factor is O(n log n), the algorithm's run time is O(n log n).