MSSC 6000 - Midterm Exam

Henri Medeiros Dos Reis

March 30, 2023

(a) In the rec function is where the branching procedure begins by generating a list of possible next jobs that can be scheduled.

This is done by iterating over all possible jobs in the `all_jobs` list and checking whether a job has not already been scheduled and whether it can be completed before its deadline.

The resulting list of possible next jobs is stored in the `next_jobs` variable.

If there are no possible next jobs, the function returns the current best score and solution seen so far. Otherwise, the function computes an upper bound for the partial solution represented by the current schedule S using the `S.upper_bound()` function.

If the upper bound is not greater than the current best score, the function prunes this branch by returning the current best score and solution. Otherwise, for all possible next jobs, the function creates a new copy of the schedule by appending the index of the next job to the current schedule and begins the recursive exploring of each branch by calling the rec function.

(b) A possible upper bound for this problem is to use relaxation and not worry about one of the constraints. I chose to not worry about the deadline, and treat the rest of the problem as a knapsack problem. Where the profit of that job is the value of the item, the duration of the job is equivalent to the weight, and the time I have left to complete the jobs is equivalent to the capacity left in the knapsack.

So to get the partial solution, the upper bound, we first look at the time we have already used. Then remove all jobs that the deadline has already passed.

Next, we know that the fractional knapsack problem has an optimal greedy solution, so we can use that to find the upper bound.

However, it is possible to add back a little bit of the deadline while we are trying to find the solution for the fractional problem, and make the upper bound a little bit better. In order to do this, when we are choosing a job, before adding it to the partial solution, make sure that it can be done before the deadline.

(c) Only code

(d) You did not start the branching process already having an upper bound at the "root". So from the start having an upper bound computed as I did in the `upper_bound` method, it could possibly cut down a bunch of branches from the start.