IE552 Homework Assignment 2 Deadline: January 2, 2023

One of the most common types of problems in combinatorial optimization is that of creating a schedule. We often consider the problem of scheduling jobs on machines. Let's consider one of the simplest possible versions of this problem.

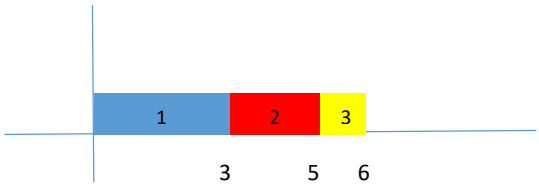
There are n jobs to be scheduled on a single machine, where the machine can process at most one job at a time, and must complete the job without a break once it has begun processing. Each job j has processing time p_j and a specific due date d_j. Its process cannot start before a release date r_j. If the completion time of job j is C_j, we define lateness $L_j = C_j - d_j$ and the aim is to minimize maximum lateness $L_{max} = max_j\ L_j$

Unfortunately this problem is NP-hard. A commonly used algorithm for this problem is Earliest Due Date (EDD) rule. This rules say at any moment that the machine is idle, start processing next available job with the earliest due date. (A job j is available at time t if its release date $r_j \leq t$.)

Example:

Jobs	Release date	Processing time	Due Date
1	0	3	5
2	1	2	3
3	3	1	6

Solution with EDD: At $t = 0$ only job 1 is available so it is scheduled first. Then at time $t = 3$ both jobs 2 and 3 are avaiable and job 2 has the earlier due date so job 2 is scheduled and then job 3.



Jobs	C_j	L_j
1	3	3-5=-2
2	5	5-3=2
3	6	6-6=0

So this schedule has $L_{max} = 2$ (Can you think of a better solution?)

For this assignment you are expected to do the following using any programming language

1) Create random instances for the problem with different sizes
2) Implement the EDD rule and solve these instances
3) Design and implement an improvement heuristic

Please provide a report including the following

1) How you generate random instances
2) Description of your improvement heuristic (e.g. definition of neighborhood and pseducode)
3) Examples where the improvement heuristic actually improves the solution and does not improve the solution
4) Appendix which has the code