Melis Tuncer **2019402156**

IE 203 SINGLE MACHINE SCHEDULING PROBLEM REPORT

Problem : There 5 jobs to be done with a single machine. Each job has a process time, the duration of the job; and a due time, the time unit that the job should be done before.

The objective is the following: Schedule all the jobs without conflicts (at each time unit, only one of the jobs can be in progress) with a minimum number of tardy jobs.

JOBS	1	2	3	4	5
PROCESS TIME	2	2	1	5	4
DUE TIME	3	3	1	9	6

The process times and due times of the jobs

Given Informations:

Jobs = [1, 2, 3, 4, 5]
Process Times (pj) = {1:2, 2:2, 3:1, 4:5, 5:4}
Due Times (dj) = {1:3, 2:3, 3:1, 4:9, 5:6}
Max_t = sum of the process times = 2+2+1+5+4 = 14
Possible Starting Times = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
Job-Starting Time Pairs = all combinations of the form (j, t) for j in *Jobs* for t in *Possible Starting Times*

LP Variables:

X_{jt} = The binary variable which takes the value;

$$\begin{cases} 1, \text{ if the job j starts at time t,} \\ 0, \text{ otherwise} \end{cases}$$

Y_{jt} = The binary variable which takes the value;

$$\begin{cases} 1, \text{ if the job j starting at time t is tardy,} \\ 0, \text{ otherwise} \end{cases}$$

FIRST FORMULATION

Objective Function: $\boxed{\text{Min } \sum (x_{jt} . y_{jt})}$ for ∀ (j,t) ∈ *Job-Starting Time Pairs*

Subject To:

1) $\sum x_{jt} = 1$ for ∀ j ∈ *Jobs* , t from 0 to (Max_t – pj[j]+1)

(All the jobs should be started only once.)

2) $t + p[j] - d[j] <= M.y_{jt}$ for ∀ (j,t) ∈ *Job-Starting Time Pairs*

(If the job is tardy, the left-hand side will be positive. Hence, the corresponding y value will be one indicating that the job is tardy. If the job is done before due time, the left-hand side will be negative. When this is the case, objective function will force the y value to be 0 indicating that the job is in fact not tardy. M will take the value of 13 since the maximum value of left-hand side is 13.)

3) $y_{jt} <= x_{jt}$ for ∀ (j,t) ∈ *Job-Starting Time Pairs*

(If x_{jt} is zero, meaning that the job did not start at time t, the corresponding y value will also be zero. If x_{jt} is one, the corresponding y value can be zero or one, meaning that the job may be tardy or not.)

4) $x_{jt} + x_{j't'} <= 1$ for ∀ (j,t), (j',t') overlapping

(In an overlapping situation, sum of the x-values of the job-time pairs should be smaller than or equal to one, providing maximum one of them to be one.)

- I have merged the constraints 2) and 3) into an if statement in a for loop for the code as:

```
for (j,t) in pairs:

    if (t + pj[j]) > dj[j]:
        y[j][t] = 1
    else:
        y[j][t] = 0
```

- I have written the following code to ensure the constraints 1) and 4)

```
for j in jobs:
    prob += (lpSum(x[j][t] for t in range(max_t-pj[j]+1))==1)


for j1 in jobs:
    for t1 in start_times:
        for j2 in jobs:
            for t2 in start_times:
                if j1!=j2 and (t1<=t2) and (t2+1<= pj[j1]+t1):
                    prob += (lpSum(x[j1][t1]+x[j2][t2]) <= 1)
```

After running the code, I am returned with the following result:

Optimal Value = 2

$X_{1,12} = 1$, $X_{2,1} = 1$, $X_{3,0} = 1$, $X_{4,3} = 1$, $X_{5,8} = 1$, $X_{jt} = 0$ for others

Time periods:

T[0,1] : Job 3
T[1,3] : Job 2
T[3, 8] : Job 4
T[8, 12] : Job 5
T[12, 14] : Job 1

Number of tardy jobs = 2 (job 5 and job 1)

SECOND FORMULATION

In the alternative formulation, the objective function and the constraints are the same as the first formulation, except the 4th constraint.

Objective Function: $\boxed{\text{Min } \sum (x_{jt} . y_{jt})}$ for ∀ (j,t) ∈ *Job-Starting Time Pairs*

Subject To:

1) $\sum x_{jt} = 1$ for ∀ j ∈ *Jobs* , t from 0 to (Max_t – pj[j]+1)

2) $t + p[j] - d[j] <= M.y_{jt}$ for ∀ (j,t) ∈ *Job-Starting Time Pairs*

3) $y_{jt} <= x_{jt}$ for ∀ (j,t) ∈ *Job-Starting Time Pairs*

4) $\sum \sum x_{jt} <= 1$ s from (max{0, t-pj[j]} to t , j ∈ Jobs

(At ant time t, the sum of the active jobs should be smaller than or equal to one, meaning that only one job can be in progress at that time.)

- I have written the following code to ensure constraint 4) :

```
for t in start_times:
    prob += (lpSum(lpSum(x[j][s] for j in jobs for s in
range(max(0, t-pj[j]),t)))<=1)
```

With the alternative formulation, I got the following results:

Optimal Value = 2

$X_{1,8} = 1$, $X_{2,1} = 1$, $X_{3,0} = 1$, $X_{4,3} = 1$, $X_{5,10} = 1$, $X_{jt} = 0$ for others

Time periods:

T[0,1] : Job 3
T[1,3] : Job 2
T[3, 8] : Job 4
T[8, 10] : Job 1
T[10, 14] : Job 5

Number of tardy jobs = 2 (job 5 and job 1)