Problem Definition

A factory needs to schedule three jobs on a single machine. Each job has a processing time and must be completed within a given time window. The objective is to minimize the total weighted completion time.

Given Data:

- **Jobs**: J_1, J_2, J_3
- **Processing Times**: $p_1 = 3, p_2 = 2, p_3 = 4$ (in hours)
- **Deadlines**: $d_1 = 8, d_2 = 6, d_3 = 10$ (must be completed before these times)
- **Weights**: $w_1 = 2, w_2 = 1, w_3 = 3$ (higher weight means higher priority)
- **Machine Availability**: Starts at time 0

Mathematical Formulation

Decision Variables

- S_j: Start time of job j (continuous)
- C_j: Completion time of job j (continuous), where $C_j = S_j + p_j$
- x_{ij}: Binary variable, 1 if job i is scheduled before job j, 0 otherwise.

Constraints

1. **No Overlapping**: Each job must be scheduled before or after the others:
$S_i + p_i \leq S_j + M(1 - x_{ij}), \quad \forall i \neq j$

$S_j + p_j \leq S_i + Mx_{ij}, \quad \forall i \neq j$

(Where M is a large number, ensuring that one of these inequalities holds.)

2. **Deadline Constraints**:
$C_j \leq d_j, \quad \forall j$

3. **Objective Function** (Minimize weighted completion time):
$\min \sum w_j C_j$

Brief Explanation of the Code

This Python code uses **Gurobi** to solve a **single-machine scheduling MILP problem**. The goal is to minimize the **weighted completion time** of jobs.

Libraries Used

- `gurobipy`: A powerful optimization library used to model and solve linear and integer programming problems.

Key Functions and Their Purpose

1. **Model("Integer_Scheduling")**
   - Creates an optimization model named "Integer_Scheduling".
2. **addVars(jobs, vtype=GRB.CONTINUOUS, name="Start")**
   - Defines **continuous variables** for job start times.
3. **addVars(jobs, vtype=GRB.CONTINUOUS, name="Completion")**
   - Defines **continuous variables** for job completion times.
4. **addVars(jobs, jobs, vtype=GRB.BINARY, name="Order")**

- Defines **binary variables** to determine job order.

5. **addConstr()**
   - Adds constraints to the model, ensuring:
     - Jobs follow the correct order.
     - Completion times are within deadlines.
     - No two jobs overlap.
6. **setObjective(sum(weight[j] * C[j] for j in jobs), GRB.MINIMIZE)**
   - Defines the **objective function**, minimizing the **total weighted completion time**.
7. **model.optimize()**
   - Solves the MILP problem using Gurobi's solver.
8. **model.status == GRB.OPTIMAL**
   - Checks if the solution is optimal and prints the **best job schedule.**

Results:

Gurobi Optimizer version 12.0.1 build v12.0.1rc0 (mac64[arm] - Darwin 24.3.0 24D70)

CPU model: Apple M3

Thread count: 8 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 18 rows, 15 columns and 45 nonzeros

Model fingerprint: 0xff729457

Variable types: 6 continuous, 9 integer (9 binary)

Coefficient statistics:

Matrix range [1e+00, 1e+02]

Objective range [1e+00, 3e+00]

Bounds range [1e+00, 1e+00]

RHS range [2e+00, 1e+02]

Presolve removed 14 rows and 11 columns

Presolve time: 0.00s

Presolved: 4 rows, 4 columns, 12 nonzeros

Variable types: 3 continuous, 1 integer (1 binary)

Found heuristic solution: objective 39.0000000

Root relaxation: objective 3.800000e+01, 0 iterations, 0.00 seconds (0.00 work units)

Nodes		Current Node		Objective Bounds		Work
Expl	Unexpl	Obj	Depth	IntInf	Incumbent	BestBd	Gap	It/Node	Time
* 0	0		0		38.0000000	38.00000	0.00%	-	0s

Explored 1 nodes (0 simplex iterations) in 0.01 seconds (0.00 work units)

Thread count was 8 (of 8 available processors)

Solution count 2: 38 39

Optimal solution found (tolerance 1.00e-04)

Best objective 3.800000000000e+01, best bound 3.800000000000e+01, gap 0.0000%

Optimal Schedule:

Job 1: Start at 0.0, Complete at 3.0

Job 2: Start at 3.0, Complete at 5.0

Job 3: Start at 5.0, Complete at 9.0

Total Weighted Completion Time: 38.0