Introduction

There are 20 jobs provided, each with its own processing time, due date, and weight and the goal is to schedule these jobs while minimizing some objective function. There are six given objective functions: tardiness, weighted tardiness, number of tardy jobs, weighted number of tardy jobs, maximum tardiness, and weighted maximum tardiness. However, minimizing one objective does not necessarily mean that the other five would be minimized to the optimum as well. So, it is imperative to formulate a model that can reasonably minimize all six objectives in a reasonable time frame (CPU time).

Approaching the Problem

There are multiple ways to approach the problem, but most of them may not very efficient and may take a lot of time to solve. Since the summation of the processing times of the 20 jobs is 2335, it may not be very feasible to use a time index in the decision variables, as it would make the problem extremely degenerate and take a lot more time. For more context, if a decision variable, x, has indices (j, t) corresponding to job j and time t, then there would be $20 \times 2335 = 46{,}700$ decision variables and most of them might be zero. Not only that, but any constraint indexed by t will be calculated 2335 times, leading to more slack or excess variables which further increases the degeneracy.

Thus, it is best to use decision variables indexed by something that has a smaller set. For that, two different formulations will be used. First, the parameters will be (consistently) defined as follows:

- p_j: Processing time of job j.
- d_j: Due date of job j.
- w_j: Weight of job j.

To solve each of the models using each formulation and under different objective functions, a Python library, `Pyomo`, in conjunction with a solver, Gurobi, and its Python API, `gurobipy`. More details are in Appendix A.

Furthermore, the first formulation is as follows:

Formulation One

Sets and Indices:

- $J = \{1, 2, \dots 20\}$: Set of jobs.
- $j \in J$: Job index j.
- $t \in J$: Position t of a job

Decision Variables:

- C_{jt}: Completion time of job j at position t.
- C_j: Completion time of job j.
- $x_{jt} = \begin{cases} 1 & if\ job\ j\ is\ in\ position\ t \\ 0 & otherwise \end{cases}$

Objective Function:

$$\min z = f\left(C_1, \dots C_{|J|}\right)$$

Constraints:

- $\sum_{t=1}^{|J|} x_{jt} = 1 \ \forall\, j \in J$
- $\sum_{j=1}^{|J|} x_{jt} = 1 \ \forall\, t \in J$
- $C_{jt} \geq \sum_{i \in J} C_{i,t-1} + p_j x_{jt} - M\left(1 - x_{jt}\right) \forall\, j, t \in J$
- $C_j = \sum_{t=1}^{|J|} C_{jt} \ \forall\, j \in J$

Boundaries:

- $C_{jt} \geq 0 \ \forall\, j, t \in J$
- $C_j \geq 0 \ \forall\, j \in J$
- $x_{jt} \in \{0,1\} \ \forall\, j, t \in J$

Using this formulation, there are only 400 binary variables and almost 420 continuous decision variables, meaning that it can be solved in a reasonable time. This formulation, however, also results in a very degenerate solution matrix, with only 20 binary variables attaining a value of 1, and only 40 of the continuous variables attaining a value greater than zero.

For the second formulation, the model is as follows:

Formulation Two

Sets and Indices:

- $J = \{1,2, \dots 20\}$: Set of Jobs.
- $j \in J$: Job index j.
- $i \in J$: Job index i.

Decision Variables:

- S_j: Start time of job j.
- C_j: Completion time of job j.
- $y_{ij} = \begin{cases} 1 & if\ job\ i\ is\ scheduled\ before\ job\ j \\ 0 & otherwise \end{cases}$

Objective Function:

$$\min z = f\left(C_1, \dots C_{|J|}\right)$$

Constraints:

- $y_{ij} + y_{ji} = 1 \; \forall \{i, j \in J |\ i \neq j\}$
- $C_j = S_j + p_j \; \forall\, j \in J$
- $S_i \geq C_j - M y_{ij} \; \forall\, i, j \in J$
- $S_j \geq C_i - M y_{ji} \; \forall\, i, j \in J$

Boundaries:

- $S_j, C_j \geq 0 \; \forall\, j \in J$
- $y_{ij} \in \{0,1\}$

This formulation is less degenerate than the previous model, with almost half of the binary variables attaining a value of one, and almost all of the continuous variables attaining a value greater than zero. This formulation also has only 40 binary variables and 40 continuous variables, which is less than the previous formulation.

The problem with both of the formulations above is that big M is used alongside the binary variables, making the relaxed counterpart of each formulation weak and disallow the solvers from quickly verifying the optimality of a proposed solution.

Extra Variables and Objective Formulation

For the solver to be able to calculate the objective functions accurately, extra variables may be needed:

- T_j: Tardiness of job j.
- $U_j = \begin{cases} 1 & if\ job\ j\ is\ tardy \\ 0 & otherwise \end{cases}$
- M_T: Maximum tardiness for the proposed schedule.
- M_T^W: Weighted maximum tardiness for the proposed schedule.

Relevant Constraints:

- $T_j \geq C_j - d_j \; \forall \; j \in J$
- $M \times U_j \geq T_j \; \forall \; j \in J$
- $M_T \geq T_j \; \forall \; j \in J$
- $M_T^W \geq w_j T_j \; \forall \; j \in J$

Boundaries:

- $T_j \geq 0 \; \forall \; j \in J$
- $U_j \in \{0,1\} \; \forall \; j \in J$
- $M_T, M_T^W \geq 0$

Objective Functions:

- $z_{Tardiness} = \sum_{j \in J} T_j$
- $z_{Weighted\ Tardiness} = \sum_{j \in J} w_j T_j$
- $z_{Tardy\ Jobs} = \sum_{j \in J} U_j$
- $z_{Weighted\ Tardy\ Jobs} = \sum_{j \in J} w_j U_j$
- $z_{Maximum\ Tardiness} = M_T$
- $z_{Weighted\ Maximum\ Tardiness} = M_T^W$

Results

The results of each formulation under each objective function is as follows:

Formulation 1:

Target Objective \ Other Statistics	Tardiness	W-Tardiness	Num of Tardies	W-Num of Tardies	Max_Tardiness	W-Max_Tardiness	Time Taken (Secs)	Node Count
Tardiness	**3,338**	17,531	7	42	1,521	10,647	76	336,652
Weighted Tardiness	4,235	**12,485**	10	45	1,317	3,258	52	303,142
Tardy Jobs	6,328	34,946	**6**	34	1,688	7,812	37	271,568
Weighted Tardy Jobs	5,195	14,355	6	**17**	1,120	3,904	40	272,409
Max Tardiness	5,276	28,597	12	69	**795**	7,950	186	17,320
Weighted Max Tardiness	6,205	18,819	10	42	1,154	**2,505**	57	189,434

Formulation 2:

Target Objective \ Other Statistics	Tardiness	W-Tardiness	Num of Tardies	W-Num of Tardies	Max_Tardiness	W-Max_Tardiness	Time Taken (Secs)	Node Count
Tardiness	**3,397**	13,166	7	33	1,086	3,480	509	740,916
Weighted Tardiness	4,147	**12,279**	8	31	1,317	3,258	47	571,171
Tardy Jobs	4,675	19,761	**5**	19	1,521	10,647	34	230,694
Weighted Tardy Jobs	4,374	13,530	5	**15**	1,416	5,664	67	627,714
Max Tardiness	6,744	37,863	13	68	**795**	7,950	31	431,813
Weighted Max Tardiness	6,051	22,959	13	63	1,154	**2,505**	34	346,724

In addition, this is the result of using EDD:

	Tardiness	W-Tardiness	Num of Tardies	W-Num of Tardies	Max_Tardiness	W-Max_Tardiness	Time Taken
EDD Formulation	4,455	25,949	12	63	795	7,950	Almost Instant

It seems that formulation two is better overall and even faster, on average – after discarding the 509 second model as an outlier.

Even though the EDD formulation provides, overall, very acceptable results, the number of tardy jobs is a bit too much; more than half of the jobs are tardy, which is unacceptable. Despite being almost instant to solve, the tradeoff is not worth it.

As for which schedule to choose from: it will be the schedule corresponding to formulation two and weighted tardiness. The reason is: not all jobs are the same, so unweighted objectives lose a lot of information. (Weighted) Maximum tardiness also delivers one part of the picture. What if there are many jobs that are close to the maximum tardiness value and are also a high priority? Next, weighted tardy jobs does not regard the degree of tardiness. Hence, weighted tardiness is the best objective to choose. The other objectives also have acceptable values. Its schedule is as follows:

$$8, 2, 13, 16, 10, 5, 9, 18, 17, 19, 4, 12, 20, 3, 7, 1, 15, 11, 14, 6$$

Relaxed Formulations

For each of the two formulations above, the relaxed formulations give objective function values of zero across the board, no matter what was being minimized. This is likely due to the usage of the Big M, which allows the binary variables to attain miniscule values to satisfy the constraints but without increasing the objective function values. Since the binary variables are the indicators for what the schedule actually is, these relaxed formulations amount to nothing, analytically.

As such, in order to utilize the easiness of relaxed formulations and their solving speed, it would be best to formulate a unimodular model. One such model could be in the form of the shortest path formulation. However, after testing it, the unimodularity would not hold. More details in Appendix B.

Conclusion

Concludingly, the positional assignment formulation and the completion time formulation proved the best two formulations to test, with the latter being the better one. The weighted tardiness objective function was the best objective to use, as it provided most sense and gave overall better results. The EDD formulation was good, except for the number of tardy jobs, which was too much to accept.

References

Zörnig, P. (1991). Degeneracy Problems in Mathematical Optimization. In P. Zörnig (Ed.), *Degeneracy Graphs and Simplex Cycling* (pp. 3-14). Springer. https://doi.org/10.1007/978-3-642-45702-9_2

Appendix A

The `pyomo` library works by creating a model through its own syntax, then the user is able to use a myriad of different types of solvers. Some solvers are specifically used for different types of problems. For this problem, Gurobi has been used, which is specialized in mixed integer programming models and quadratic NLP's.

In the code, there are four formulations and the EDD solution, each of the four formulations is for the Fx formulations present in the lecture notes: `F1_Formulation.py` corresponds to F1 formulation, etc.

There is also a main.py and functions.py. The latter file is used to define frequently used functions and the callback class to be used for the Gurobi solver.

In the `F`x`_Formulation.py` files, the constraints and objective functions are defined first as Python methods, then the model components (parameters, variables, objectives) are defined next. Next, the Gurobi solver is defined, and the callback function is established. Before solving each model, a copy of the initial model, m, is made then is pushed to the Gurobi solver to solve. This enables the user to guess more accurately each objective function took to solve, instead of continuing off of the previous solution.

The callback function's job is to check after every node check whether the current objective or the best bound has improved. If neither has improved for more than 10 iterations and 30 seconds (both parameters can be modified), then the callback function terminates the solver, and the current solution is regarded as the optimal solution.

Finally, after the model has been solved, the solution is printed out in an Excel file and placed in a folder according to the respective objective function. Then, the outputs are summarized in a "`Output Summary.xlsx`" file. Note that some (inactive) objective functions were "corrected" after solving since their respective constraints were not binding.

Appendix B

The shortest path formulation is as follows:

Sets and Indices:

- $J = \{1, 2, \dots 20\}$: Set of jobs.
- $T = \{1, 2, \dots 2335\}$: Set of possible times. ($T_{max} = \sum_{j \in J} p_j = 2335$)
- $j \in J$: Job index j.
- $t \in T$: Time t.

Decision Variables:

- C_j: Completion time of job j.
- $x_{j,t,t+p_j} = \begin{cases} 1 & if\ job\ j\ starts\ at\ time\ t\ and\ finishes\ at\ t + p_j \\ 0 & otherwise \end{cases}$

Objective Function:

$$\min z = f\left(C_1, \dots C_{|J|}\right)$$

Constraints:

- $\sum_{j=1}^{|J|} x_{j,0,p_j} = 1$
- $\sum_{j=1}^{|J|} x_{j,t,t+p_j} - \sum_{j=1}^{|J|} x_{j,t-p_j,t} = 0\ \forall \{t \in T - \{0, T_{\max}\} | \text{Index sets are valid}\}$
- $\sum_{j=1}^{|J|} x_{j,T_{max}-p_j,T_{max}} = 1$
- $\sum_{t=1}^{T_{max}} x_{j,t,t+p_j} = 1$

Boundaries:

- $x_{j,t,t+p_j} \in \{0,1\}\ \forall\ j \in J, t \in T$, relaxed to: $0 \leq x_{j,t,t+p_j} \leq 1$.

However, even after removing the extra variables and the additional constraints and changing the formulas of the objective functions to make it akin to a SPT, the unimodularity would not hold and the $x_{j,t,t'}$ variables would attain fractional values, making it improbable to form a schedule from such relaxed formulation.

Running this formulation without the relaxation would take hours upon hours, making it also impractical.