Let $J = \{1,2,3,\dots,N\} \subset \mathbb{Z}^+$, $S = \{1,2,3,\dots,K\} \subset \mathbb{Z}^+$ and $M = \{1,2,3,\dots,L\} \subset \mathbb{Z}^+$.

Also let $J_{NL} \subset J$ so that J_{NL} is the set of jobs that must be complete before a due time (in shifts, e.g., must be complete in 4 shifts) $d_j > 0$ for all $j \in J_{NL}$.

Let $x_{s,j,m} \in \{0,1\}$ be the decision variables that represent if job-j on machine-m is assigned to the shift-s or not, $\forall j \in J$, $\forall s \in S$ and $\forall m \in M$.

Let $\alpha_j > 0$ be the importance factor of every job, i.e., $\forall j \in J$.

Let $p_{j,m} \in \{0,1\}$ be the parameters that represent whether the job-j uses the machine-m.

Let $\sigma_{s,m} \in \mathbb{Z}^+$ represent the available number of machines of type-m at shift-s.

$$\min z = \sum_{j \in J} \alpha_j \cdot \left[\sum_{s \in S} s x_{s,j,L} \right]$$

Subject to

$$\sum_{s \in S} s x_{s,j,L} \le d_j, \qquad \forall j \in J_{NL}$$

$$\sum_{j \in J} p_{j,m} x_{s,j,m} \le \sigma_{s,m}, \qquad \forall s \in S, \forall m \in M$$

$$\sum_{s \in S} x_{s,j,m} = 1, \qquad \forall j \in J, \forall m \in M$$

$$p_{j,m} + \sum_{s \in S} s x_{s,j,m} \le \sum_{s \in S} s x_{s,j,m+1}, \qquad \forall j \in J, \forall m = 1,2,3,\dots,L-1$$

$$x_{s,j,m} \in \{0,1\}, \qquad \forall j \in J, \forall s \in S, \forall m \in M$$

See that the expression $\sum_{s \in S} s x_{s,j,m}$ gives the shift job-j on machine-m assigned to. Furthermore, for now, the model assumes that each job represents a fixed number (so called lot size) of products that must be sequentially processed through L machines and hence $p_{j,m}$ indeed means whether job-j is processed on machine-m or not. This model also supports parallel manufacturing on each machine using the parameter $\sigma_{s,m}$.

The code inside the file `Job_Shift_Assigner.ipynb` is written on Google Colab and uses Pyomo model and HiGHS solver for MIPs. It has the model builder functions and a basic integration example. It can take a Pandas DataFrame of products with some attributes and convert into a DataFrame of jobs. Then it can integrate it with some other data and feed into the model builder. Finally, the model solves the problem and the output is then interpreted and integrated back into the DataFrame of jobs. Please check the `.ipynb` file for the code and the commentary for further details of the code.

A prospective future plan is to support for jobs that take less than a shift's duration. As can be seen in the model, model's core assumption is that if a product is to be produced, then it must be in lots; hence, fixed in size which take exactly one shift of one processor. This might be possible by some simple tricks waiting to be implemented straightforwardly. Unfortunately, though, this model cannot support planning of products that take significantly more than one shift on one processor. In some cases, it might be possible to ignore this case or find a way around the problem.

Another to-do is to find a rule-of-thumb for determining the number of shifts, K, for better solving efficiency. This is important since the number of decision variables directly depends on this number. Finding a good minimal K can increase performance tremendously.

Finally, enforcing due times with given – potentially highly restrictive – available machines data often makes the model infeasible. So, J_{NL} set can be used for products that must be on time. It is easy to see that this infeasibility arises due to the fact that there may be some shifts when not many workers are available but many products must be done before a certain deadline. Implementing some sub-algorithm to address this problem, at least partially, however, is on the to-do list.