OF Optimization code

July 4, 2025

1 Mathematical model

We first tried to define a mathematical optimization model which we would then implement in a computer code.

1.1 Variables

- $x(i, m, t) \quad with \quad i = OF\ index\ ,\ m = machine\ index\ ,\ t = time$

$x(i, m, t) = 1$ if the task i is being done on the machine m at the time t

$x(i, m, t) = 0$ otherwise

This variable registers all the necessary information to build the OF time schedule.

$-t_{early}$ Earliness

$-t_{late}$ Lateness

$-nb_{changes}$ Amount of OF changes scheduled on the machines

We also eventually decided to define a fixed changing time of 30 min between 2 OFs. This makes our model easier to implement at first. We could adapt our model to make it more accurate later on.

1.2 Cost Function

The first step was to define a cost function. This function will contain every parameter that will influence decision-making in the OF organization. We chose the following parameters :

- Lateness of the OF
- Earliness of the OF
- Excess of corks produced
- Amount of different OF on the same machine.

The excess of corks produced is due to the fact that we discretized the time.

Therefore we cannot produce the exact number of corks, and have a small excess of corks produced.
The last parameter is not very intuitive, but it allows us to solve different issues :

- Without that parameter, our code would sometimes start an OF, make a pause, and restart it later in order to limit the lateness of the OF. Still, this is not optimal as it would be better to just start the OF later and execute it in one straight part.

- Our code would also tend to execute the OFs simultaneously on two machines, rather than do it on one, resulting in an OF schedule full of small production periods and containing many OF changes on every machine.

We eventually chose the following cost function :

$$Cost = k_1 \sum_i t_{early}(i)^2 + k_2 \sum_i t_{late}(i)^2 + k_3 \cdot nb_{changes} + k_4 \cdot excess$$

$k_1 = 0.01 \quad k_2 = 0.1 \quad k_3 = 1 \quad k_4 = 0.1$
Note that these values were chosen arbitrarily and can easily be modified

1.3 Constraints

Now that we have defined a cost function, we have to translate all physical constraints into a mathematical equation:

1. Only one OF at a time on a machine
2. The amount of corks produced must at least be equal to the quantity commanded
3. We can operate the same OF on maximum two machines at a time, as there are only two printing supports per OF.
4. Double printed corks are printed either once on a double printing machine, or twice on a single printing one.

The first three constraints are easily translated mathematically by:

$$\forall t, m \sum_i x(i, m, t) = 1$$

$$\forall i \sum_{t,m} time\ step \cdot x(i, m, t) \cdot cadence(i, m) \geq nb_{corks\ commanded}$$

$$\forall i, t \sum_m x(i, m, t) \leq 2$$

The last constraint cannot really be translated mathematically but is easily implemented in a computer code.

2 Code structure

2.1 Data extraction

The first step consists of extracting all the necessary data from the db-of excel sheet. Once this is done, we create a list of the different OF. Each OF is assigned to a dictionary containing every necessary information about the OF : due date, quantity left to print, cork type, etc.
We also create a list of the different machines with dictionaries that contain their type (double or simple) and on what type of cork they can work on.

2.2 Mathematical resolution

After defining a time step, we built each variable involved in the problem. We built a "completion time" variable that will enable us to access the lateness/earliness of a task.
With the variables define, we can create the cost functions and impose the constraints defined in the previous section.

2.3 Output

In order to have a readable output, we decided to present the optimal solution as a visual timetable. Each task is represented by a color. The time is represented on the x axis, while the machines used are represented on the y axis. The output could also be an excel sheet, but a visual output makes it easier to verify whether the solution is truly optimal or not.
Once we are sure that our code provides the optimal solution, we will be able to change the output to an excel sheet.

3 What's next ?

- Introducing a variable changing time, depending on the types of corks produced in two consecutive OFs.

- Changing the output to an excel sheet so that it matches the current schedule format used by the factory.

- Improving the code complexity, to have a faster running code, and be able to have a shorter time step.