Workshop: Personalized Gift Handcrafting

October 5, 2020

1 Baseline problem

1.1 Description

We are running a company that produces a particular type of luxury gifts: personalised jewellery with names. These pieces of jewellery are handcrafted.

The production of a given piece of jewellery for a particular name (say "Emma") requires:

- A set of tasks to produce each individual letters (so "E", "m", "m", "a")
- A task to assemble all these letters together for producing the final piece of jewellery

The duration for producing a particular letter depends on the letter itself (for instance producing the letter 'm' is more complex than production the letter 'a' so it will take longer[1]. The duration for producing the different letters is given in the data as a table. In the problem, all times are given in minutes (mn).

The assembly task can be performed only after all the letters have been produced. Its duration is proportional to the number of letters to assemble.

The duration for each letter and the proportionality factor for assembly duration are given in file Data/data.json in JSON format, for instance:

```
{
   "assembly" : 500,
   "letters"  : {
      "A": 1443,
      "B": 1841,
      "C": 1247,
      "D": 1741,
      "E": 1562,
      ...
```

[1]If you are interested in the making-of of this course, for the complexity of letters, I have been using the results of a script counting the number of pixels of the different letters provided here: https://gist.github.com/alexmic/8345076

```
    }
}
```

As an example, Figure 1.1 display the process for producing a piece of jewellery for Emma. The arcs represent precedence constraints between tasks. The blue task in the end is the assembly task.

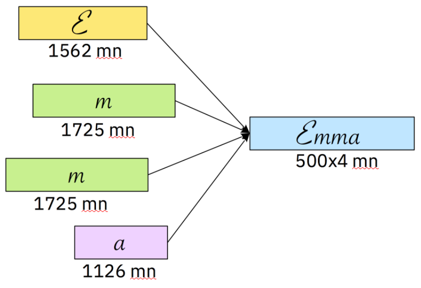

Figure 1: Process for producing Emma's gift

We are on Jan 1, 2019 and our company has collected the set of production orders for the year. A production order is a type of jewellery to produce (so, a name) and a due-date. We assume that all our customers want to have their gift ready for their birthday. Thus a production order is given by a pair (name,birthday).

Orders are defined in file orders.json in JSON format, for instance:

```
{ "orders" : [
    ["Emma",  "15/04"],
    ["Emily", "14/01"],
    ...
    ]
}
```

In the first version of the problem, we suppose that we only have one master jeweller working 24/7 and that the penalty for late delivery of a gift is quadratic: its value is T^2 if T denotes the tardiness in days. For instance if the gift for Emma is only available on 18/04 (so 3 days after Emma's birthday), the related tardiness penalty will be $3^2 = 9$.

We want to schedule the production so as to minimize the total tardiness penalty given that our master jeweller can perform only one task at a time.

1.2 Mathematical formulation

The input data consists of:

- A set $\mathcal{L}$ of letters. Each letter $j \in \mathcal{L}$ is associated a duration D_j.
- An assembly duration (by letter) denoted A.

- A set O of n orders. A given order $i \in O$ is:
  - A name (ex: "Emma"). We denote L_i the multiset of letters of the name on underlying set of letters $\mathcal{L}$. For example $\{E^1, m^2, a^1\}$.
  - A birthday date B_i (assumed to be in mn here)

The mathematical formulation of the baseline problem in CP Optimizer is as follows:

$$
\begin{aligned}
&\min \quad \sum_{i \in O} \lfloor \max(0, \mathrm{endOf}(a_i) - B_i)/1440 \rfloor^2 && && (1)\\
&\mathrm{endBeforeStart}(l_{ij}, a_i) && \forall i \in O, j \in L_i && (2)\\
&\mathrm{noOverlap}([a_i]_{i \in O} \cup [l_{ij}]_{i \in O, j \in L_i}) && && (3)\\
&\text{interval } l_{ij} \text{ size } D_j && \forall i \in O, j \in L_i && (4)\\
&\text{interval } a_i \text{ size } A|L_i| && \forall i \in O && (5)
\end{aligned}
$$

Decision variables. The decision variables of the problem consists of two sets of interval variables. Equation 5 defines a set of interval variables a_i. Variable a_i corresponds to the assembly task of order i, its size is A times the number of letters $|L_i|$. Equation 4 defines a set of interval variables l_{ij} for producing letter j of order i, size of interval l_{ij} is the duration D_j for producing letter j.

Constraints. Precedence constraints (as depicted on Fig. 1.1) are formulated in Equation 2 as endBeforeStart constraints between the intervals l_{ij} for producing each letter and the assembly interval a_i. As the problem involves a unique master jeweler to perform all the tasks, these tasks cannot overlap in time; this is formulated as a noOverlap constraint in Equation 3.

Objective function. The objective function is formulated in Equation 1. It is a sum over all orders i of the tardiness penalty which is the square of the tardiness ($\max(0, \mathrm{endOf}(a_i) - B_i)$) in days (which explains the integer division by $60 * 24 = 1440$).

1.3 Python formulation

A python version of this mathematical model is given in file `V0/gift_production.py`. In the workshop we will work with a "realistic" data set given by the actual first name and birthday date of the students.

2 Problem extensions

In the workshop we will gradually extend the baseline problem with more realistic features. Each extension is identified by a number (from V1 to V8). You will find a skeleton of the python file for each extension in sub-directories V1, ..., V8. Of course do not hesitate to cut&paste the code of previous versions.

2.1 More workers are available

Two master jewelers with identical skills (variant V1)

In this version of the problem we suppose that we have not only one but two master jewelers available for performing the tasks. These two workers have exactly the same skills. Each task must be allocated to one of the workers (the two workers cannot work together on the same task).

<u>Solution:</u> We have two possible ways to model the identical resources: with a cumul function or with explicitly handling resource allocation with optional interval variables, alternative constraints and one noOverlap constraint per worker. As this second model will be necessary for the next variant of the problem, we give here only the formulation with cumul functions.

$$
\begin{aligned}
&\min \quad \sum_{i \in O} \lfloor \max(0, \mathrm{endOf}(a_i) - B_i)/1440 \rfloor^2 \\
&\mathrm{endBeforeStart}(l_{ij}, a_i) && \forall i \in O, j \in L_i \\
&\sum_{i \in O} \mathrm{pulse}(a_i, 1) + \sum_{i \in O, j \in L_i} \mathrm{pulse}(l_{ij}, 1) \leq C && \qquad (1) \\
&\text{interval } l_{ij} \text{ size } D_j && \forall i \in O, j \in L_i \\
&\text{interval } a_i \text{ size } A|L_i| && \forall i \in O
\end{aligned}
$$

The only difference with the baseline problem is that the noOverlap constraint is replaced by a cumul function in Equation 1 that counts the number of workers and is constrained to be lower than the total number of workers C ($C = 2$ in our case).

Master and apprentice jewellers (variant V2)

In this version of the problem we suppose that we have two jewelers available for performing the tasks but they have different skills. One (say Yoda) is a master whereas the second one (say Luke) is an apprentice. The apprentice can perform all the tasks but he works twice slower.

Solution: We suppose that we have a set of workers W. For each worker $k \in W$ we denote r_k the speed ratio of worker k. So for instance if we have 2 workers, the first one (Yoda) with normal speed and the second one (Luke) with speed ratio 2, we will have $r_1 = 1$, $r_2 = 2$. The problem can be formulated as follows.

$$
\begin{array}{lrr}
\min \quad \sum_{i \in O} \lfloor \max(0, \text{endOf}(a_i) - B_i)/1440 \rfloor^2 & & \\
\text{endBeforeStart}(l_{ij}, a_i) & \forall i \in O, j \in L_i & \\
\text{alternative}(l_{ij}, [l_{ij}^k]_{k \in W}) & \forall i \in O, j \in L_i & (1) \\
\text{alternative}(a_i, [a_i^k]_{k \in W}) & \forall i \in O & (2) \\
\text{noOverlap}([a_i^k]_{i \in O} \cup [l_{ij}^k]_{i \in O, j \in L_i}) & \forall k \in W & (3) \\
\text{interval } \; l_{ij} & \forall i \in O, j \in L_i & (4) \\
\text{interval } \; l_{ij}^k \; \text{ opt, size } \; r_k D_j & \forall i \in O, j \in L_i, k \in W & (5) \\
\text{interval } \; a_i & \forall i \in O & (6) \\
\text{interval } \; a_i^k \; \text{ opt, size } \; r_k A |L_i| & \forall i \in O, k \in W & (7)
\end{array}
$$

Two new sets of interval variables are defined: l_{ij}^k (Equation 5) and a_i^k (Equation 7). These intervals are optional and represent the possible allocation of a task (represented respectively by interval variable l_{ij} and a_i) to a given worker $k \in W$. As the duration of the tasks depend on which worker is used, the size of the tasks is specified for each of these optional intervals using the speed ratio r_k of the worker. Note that the size of interval variables l_{ij} and a_i is now left unconstrained (Equations 4 and 6): it will depend on the allocated worker. Alternative constraints (Equations 1 and 2) state that among the optional intervals representing the possible allocations to a worker k, one and only one interval must be present (the one corresponding to the allocated worker) and synchronized (same start and end value) with the task interval. Finally, for each worker k, in Equation 3, a noOverlap constraint is posted on all its tasks. This constraint will ensure that the tasks allocated to worker k (the present intervals) will not overlap.

2.2 Week-end breaks and vacations

In the variants below, we suppose that the two workers of variant V2 do not work during week-ends and each worker has his own vacation plan. The calendar of each worker k (breaks and vacations) is modeled as a stepwise function C_k with value 100% when the worker is available and 0% during the week-ends and vacations.

All tasks can be suspended by breaks (Variant V3)

In this variant, all the tasks (letter production, assembly) are suspended by the week-end breaks and vacation of the worker that perform them.

Solution: The suspension of the tasks by the week-end breaks and vacations can simply be modeled using the calendar step functions as intensity when defining the interval variables in Equations 1 and2.

$$\begin{array}{lr}
\min \quad \sum_{i \in O} \lfloor \max(0, \mathrm{endOf}(a_i) - B_i)/1440 \rfloor^2 & \\
\mathrm{endBeforeStart}(l_{ij}, a_i) & \forall i \in O, j \in L_i \\
\mathrm{alternative}(l_{ij}, [l_{ij}^k]_{k \in W}) & \forall i \in O, j \in L_i \\
\mathrm{alternative}(a_i, [a_i^k]_{k \in W}) & \forall i \in O \\
\mathrm{noOverlap}([a_i^k]_{i \in O} \cup [l_{ij}^k]_{i \in O, j \in L_i}) & \forall k \in W \\
\text{interval } \; l_{ij} & \forall i \in O, j \in L_i \\
\text{interval } \; l_{ij}^k \; \text{ opt, size } \; r_k D_j, \text{ intensity } C_k & \forall i \in O, j \in L_i, k \in W \quad (1) \\
\text{interval } \; a_i & \forall i \in O \\
\text{interval } \; a_i^k \; \text{ opt, size } \; r_k A |L_i|, \text{ intensity } C_k & \forall i \in O, k \in W \quad (2)
\end{array}$$

Some tasks cannot be suspended by breaks (Variant V4)

In this variant of the problem with worker calendars, we additionally consider the constraint that the letter production tasks cannot be suspended by breaks or vacations. They must be performed in a period of time when the worker is continuously available.

Solution: The fact that letter production tasks cannot be suspended by breaks or vacations can be formulated by using forbidExtent constraints (Equation 1).

$$
\begin{array}{llr}
\min & \sum_{i \in O} \lfloor \max(0, \mathrm{endOf}(a_i) - B_i)/1440 \rfloor^2 & \\
\mathrm{endBeforeStart}(l_{ij}, a_i) & & \forall i \in O, j \in L_i \\
\mathrm{alternative}(l_{ij}, [l_{ij}^k]_{k \in W}) & & \forall i \in O, j \in L_i \\
\mathrm{alternative}(a_i, [a_i^k]_{k \in W}) & & \forall i \in O \\
\mathrm{noOverlap}([a_i^k]_{i \in O} \cup [l_{ij}^k]_{i \in O, j \in L_i}) & & \forall k \in W \\
\mathrm{forbidExtent}(l_{ij}^k, C_k) & & \forall i \in O, j \in L_i, k \in W \\
\text{interval } l_{ij} & & \forall i \in O, j \in L_i \\
\text{interval } l_{ij}^k \text{ opt, size } r_k D_j, \text{ intensity } C_k & & \forall i \in O, j \in L_i, k \in W \\
\text{interval } a_i & & \forall i \in O \\
\text{interval } a_i^k \text{ opt, size } r_k A |L_i|, \text{ intensity } C_k & & \forall i \in O, k \in W
\end{array}
\tag{1}
$$

2.3 Batch production (Variant V5)

Until now, each letter were produced one at a time by a given worker. In this variant, the company has decided to invest into some machines that are able to process several instances of the same letter simultaneously (this is often called a batch). Each worker will have his own machine. A machine can process up to 4 instances of the same letter simultaneously (in this case the production of these letters starts and end at the same time). The processing of a batch of letters needs both the machine and the worker.

Solution: We will use the concept of state function to formulate the production of letters by batches. We suppose that each letter $l \in \mathcal{L}$ is indexed by an integer in $[0, |\mathcal{L}| - \infty]$ and that I_{ij} denotes the index of the j^{th} letter of order i.

$$
\begin{array}{lrr}
\min \quad \sum_{i \in O} \lfloor \max(0, \text{endOf}(a_i) - B_i)/1440 \rfloor^2 & & \\
\text{endBeforeStart}(l_{ij}, a_i) & \forall i \in O, j \in L_i & \\
\text{alternative}(l_{ij}, [l_{ij}^k]_{k \in W}) & \forall i \in O, j \in L_i & \\
\text{alternative}(a_i, [a_i^k]_{k \in W}) & \forall i \in O & \\
\text{noOverlap}([a_i^k]_{i \in O}) & \forall k \in W & (1) \\
\text{alwaysEqual}(s^k, l_{ij}^k, I_{ij} + 1, 1, 1) & \forall k \in W, i \in O, j \in L_i & (2) \\
\text{alwaysEqual}(s^k, a_i, 0, 1, 1) & \forall k \in W, i \in O & (3) \\
\sum_{i \in O, j \in L_i} \text{pulse}(l_{ij}^k, 1) \leq Q & \forall k \in W & (4) \\
\text{forbidExtent}(l_{ij}^k, C_k) & \forall i \in O, j \in L_i, k \in W & \\
\text{interval } l_{ij} & \forall i \in O, j \in L_i & \\
\text{interval } l_{ij}^k \text{ opt, size } r_k D_j, \text{ intensity } C_k & \forall i \in O, j \in L_i, k \in W & \\
\text{interval } a_i & \forall i \in O & \\
\text{interval } a_i^k \text{ opt, size } r_k A |L_i|, \text{ intensity } C_k & \forall i \in O, k \in W & \\
\text{stateFunction } s^k & \forall k \in W & (5)
\end{array}
$$

A state function is created in Equation 5 for each worker. It will have state 0 when the worker is performing an assembly task (see the alwaysEqual constraints in Equation 3) and a state in $I_{ij} + 1 \in [1, |\mathcal{L}|]$ depending on the letter produced by the worker (see the alwaysEqual constraints in Equation 2). These constraints prevents two different letters (say 'm' and 'E') to be produced simultaneously because they will require different state of the function. Similarly, an assembly task will not be able to overlap a letter production task. But the model still needs a constraint to forbid the overlap of assembly tasks, this is ensured by Equation 1. Finally, for each worker's machine, constraint in Equation 4 uses a cumul function to ensure that at any point in time, no more than Q instances of the same letter are produced (here $Q = 4$).

2.4 Setup times (Variant V6)

In this variant of the problem, we additionally suppose that the machines cannot instantaneously switch from the production of a batch of letters to a batch of different letters. There is a setup times (one day) to reconfigure the machine. This reconfiguration task is automatic and does not require the worker (so in the meantime, the worker can work on an assembly task for instance).

Solution: Because the setup time only occurs between the letter production activities and only depends on the machine, we see here that is is much easier to consider the worker and its machine as two different resources. We will keep the constraints of the previous model for the worker and create an additional set of state functions for modeling the machine state (with letter production only) and the setup time.

$$
\begin{array}{lr}
\min \quad \sum_{i\in O} \lfloor \max(0, \text{endOf}(a_i) - B_i)/1440 \rfloor^2 & \\
\text{endBeforeStart}(l_{ij}, a_i) & \forall i \in O, j \in L_i \\
\text{alternative}(l_{ij}, [l_{ij}^k]_{k\in W}) & \forall i \in O, j \in L_i \\
\text{alternative}(a_i, [a_i^k]_{k\in W}) & \forall i \in O \\
\text{noOverlap}([a_i^k]_{i\in O}) & \forall k \in W \\
\text{alwaysEqual}(s^k, l_{ij}^k, I_{ij}+1, 1, 1) & \forall k \in W, i \in O, j \in L_i \\
\text{alwaysEqual}(s^k, a_i, 0, 1, 1) & \forall k \in W, i \in O \\
\text{alwaysEqual}(r^k, l_{ij}^k, I_{ij}, 1, 1) & \forall k \in W, i \in O, j \in L_i \quad (1) \\
\sum_{i\in O, j\in L_i} \text{pulse}(l_{ij}^k, 1) \le Q & \forall k \in W \\
\text{forbidExtent}(l_{ij}^k, C_k) & \forall i \in O, j \in L_i, k \in W \\
\text{interval } \ l_{ij} & \forall i \in O, j \in L_i \\
\text{interval } \ l_{ij}^k \ \text{ opt, size } r_k D_j, \text{ intensity } C_k & \forall i \in O, j \in L_i, k \in W \\
\text{interval } \ a_i & \forall i \in O \\
\text{interval } \ a_i^k \ \text{ opt, size } r_k A|L_i|, \text{ intensity } C_k & \forall i \in O, k \in W \\
\text{stateFunction } \ s^k & \forall k \in W \\
\text{stateFunction } \ r^k \ \text{ with } \ M & \forall k \in W \quad (2)
\end{array}
$$

In the model, an additional state function r_k is created for each worker in Equation 1 with a transition matrix M whose value is 0 on the diagonal and T everywhere else (here $T = 24 * 60$). The state of this function represents the letter produced on the machine thanks to the alwaysEqual constraints in Equation 1. The state function will ensure a minimal transition time between states as specified in matrix M.

2.5 Inventories

Limited storage space (Variant V7)

Let's keep on making the problem more realistic. We suppose that once they have been produced, the letters are stored in an inventory until they are assembled. In this variant, we consider that this inventory has a limited capacity and we cannot store more than 20 letters.

Solution:

$$
\begin{array}{lrr}
\min \quad \displaystyle\sum_{i\in O} \lfloor \max(0, \mathrm{endOf}(a_i) - B_i)/1440 \rfloor^2 & & \\
\mathrm{alternative}(l_{ij}, [l_{ij}^k]_{k\in W}) & \forall i \in O, j \in L_i & \\
\mathrm{alternative}(a_i, [a_i^k]_{k\in W}) & \forall i \in O & \\
\mathrm{noOverlap}([a_i^k]_{i\in O}) & \forall k \in W & \\
\mathrm{alwaysEqual}(s^k, l_{ij}^k, I_{ij}+1, 1, 1) & \forall k \in W, i \in O, j \in L_i & \\
\mathrm{alwaysEqual}(s^k, a_i, 0, 1, 1) & \forall k \in W, i \in O & \\
\mathrm{alwaysEqual}(r^k, l_{ij}^k, I_{ij}, 1, 1) & \forall k \in W, i \in O, j \in L_i & \\
\displaystyle\sum_{i\in O, j\in L_i} \mathrm{pulse}(l_{ij}^k, 1) \le Q & \forall k \in W & \\
\mathrm{forbidExtent}(l_{ij}^k, C_k) & \forall i \in O, j \in L_i, k \in W & \\
\mathrm{endAtStart}(l_{ij}, p_{ij}) & \forall i \in O, j \in L_i & (1) \\
\mathrm{endAtStart}(p_{ij}, a_i) & \forall i \in O, j \in L_i & (2) \\
\displaystyle\sum_{i\in O, j\in L_i} \mathrm{pulse}(p_{ij}, 1) \le U & & (3) \\
\mathrm{interval}\ \ l_{ij} & \forall i \in O, j \in L_i & \\
\mathrm{interval}\ \ l_{ij}^k\ \ \mathrm{opt},\ \ \mathrm{size}\ \ r_k D_j,\ \ \mathrm{intensity}\, C_k & \forall i \in O, j \in L_i, k \in W & \\
\mathrm{interval}\ \ p_{ij} & \forall i \in O, j \in L_i & (4) \\
\mathrm{interval}\ \ a_i & \forall i \in O & \\
\mathrm{interval}\ \ a_i^k\ \ \mathrm{opt},\ \ \mathrm{size}\ \ r_k A|L_i|,\ \ \mathrm{intensity}\, C_k & \forall i \in O, k \in W & \\
\mathrm{stateFunction}\ \ s^k & \forall k \in W & \\
\mathrm{stateFunction}\ \ r^k\ \ \mathrm{with}\ \ M & \forall k \in W &
\end{array}
$$

We will create additional interval variables p_{ij} in Equation 4 that represent the interval of time between the end of the letter production (Equation 1) and the start time of the assembly task (Equation 2). Note that as the precedence constraints between p_{ij} and a_i are now redundant with the new precedences so we removed them from the model. The limited capacity of the inventory is formulated in Equation 3 by constraining a cumul function that describes the level of the inventory to be less than U the inventory size (in our case $U = 20$).

Inventory and earliness cost (Variant V8)

Finally, in the last variant of the problem we will work with a more complex version of the objective function that will additionaly take into account:

Earliness cost A term that measures some penalty (in days) for delivering the gifts too early

Inventory cost A term that measures an inventory cost: each letter stored during one day in the inventory incurs a cost of 0.1

The objective function is to minimize the sum of tardiness penalties, earliness and inventory costs.

Solution: So here is an example of formulation of the problem. Only the objective function was changed (in our case, $\alpha = 0.1$).

$$\begin{aligned}
\min \quad & \sum_{i \in O} \lfloor \max(0, \text{endOf}(a_i) - B_i)/1440 \rfloor^2 + \\
& \sum_{i \in O} \lfloor \max(0, B_i - \text{endOf}(a_i))/1440 \rfloor + \\
& \alpha \sum_{i \in O, j \in L_i} \lfloor \text{lengthOf}(p_{ij})/1440 \rfloor
\end{aligned}$$

$$\begin{array}{lr}
\text{alternative}(l_{ij}, [l_{ij}^k]_{k \in W}) & \forall i \in O, j \in L_i \\
\text{alternative}(a_i, [a_i^k]_{k \in W}) & \forall i \in O \\
\text{noOverlap}([a_i^k]_{i \in O}) & \forall k \in W \\
\text{alwaysEqual}(s^k, l_{ij}^k, I_{ij} + 1, 1, 1) & \forall k \in W, i \in O, j \in L_i \\
\text{alwaysEqual}(s^k, a_i, 0, 1, 1) & \forall k \in W, i \in O \\
\text{alwaysEqual}(r^k, l_{ij}^k, I_{ij}, 1, 1) & \forall k \in W, i \in O, j \in L_i \\
\displaystyle\sum_{i \in O, j \in L_i} \text{pulse}(l_{ij}^k, 1) \le Q & \forall k \in W \\
\text{forbidExtent}(l_{ij}^k, C_k) & \forall i \in O, j \in L_i, k \in W \\
\text{endAtStart}(l_{ij}, p_{ij}) & \forall i \in O, j \in L_i \\
\text{endAtStart}(p_{ij}, a_i) & \forall i \in O, j \in L_i \\
\displaystyle\sum_{i \in O, j \in L_i} \text{pulse}(p_{ij}, 1) \le U & \\
\text{interval } \ l_{ij} & \forall i \in O, j \in L_i \\
\text{interval } \ l_{ij}^k \ \text{ opt, size } \ r_k D_j, \ \text{intensity}\, C_k & \forall i \in O, j \in L_i, k \in W \\
\text{interval } \ p_{ij} & \forall i \in O, j \in L_i \\
\text{interval } \ a_i & \forall i \in O \\
\text{interval } \ a_i^k \ \text{ opt, size } \ r_k A |L_i|, \ \text{intensity}\, C_k & \forall i \in O, k \in W \\
\text{stateFunction } \ s^k & \forall k \in W \\
\text{stateFunction } \ r^k \ \text{ with } \ M & \forall k \in W
\end{array}$$