Algorithms for smart scheduling of a DSM enzyme production line

CSE3000 Research Project (2021/22 Q4)

April 2, 2022

1 Introduction

The DSM plant is a batch manufacturing system, in which multiple products (enzymes) are produced. The enzymes are produced using a sequential series of unit operations. Each product is defined by a (unique) recipe, providing a sequential list of operations that must be executed to produce one standard quantity of this product. The recipes define the processing times for all unit operations. The different product lines commonly share equipment. There are several criteria to consider when analyzing the quality of a schedule. A common objective in scheduling tasks is to minimize the make span of the schedule. In practice there are multiple objective criteria, e.g. the management of DSM aims to meet customer demand restricted to due dates and an important objective for the plant schedulers is to minimize idle time of the machines. The aim is to develop an efficient algorithm that solves this task. In this document the different inputs of the scheduling problem are described. Furthermore, a Mixed Integer Linear Programming formulation of the problem is provided (that matches the Python implementation).

2 Input description

2.1 Machines

The DSM plant has different work centers with machines. For the unit operations preparation, filtering, and reception different number of machines are available.

Unit operation	**Available machines**
Preparation	{0, 1, 2}
Filtering	{3, 4, 5, 6}
Reception	{7, 8}

Tabel 1: Machines

2.2 Products

The DSM plant produces different products types, all having their own recipe:

Product type	**Recipe**
Enzyme 0	Preparation → Filtering → Reception
Enzyme 1	Preparation → Filtering
Enzyme 2	Filtering → Reception
Enzyme 3	Preparation → Filtering → Reception
Enzyme 4	Preparation → Filtering → Reception
Enzyme 5	Filtering → Reception

Tabel 2: Recipes

The processing times are given per product type per unit operation:

Product type	Preparation	Filtering	Reception
Enzyme 0	8	4	4
Enzyme 1	3	2	-
Enzyme 2	-	3	3
Enzyme 3	4	6	6
Enzyme 4	5	4	7
Enzyme 5	-	8	3

Tabel 3: Processing times

2.3 Cleaning change-overs

Specialized cleaning is required if two different products are subsequently processed at the same machine. A change-over matrix is defined to specify the change-over times required between product i and product j at machine k. So for all machines we have change-over matrix of the form:

	Enzyme 0	Enzyme 1	Enzyme 2	Enzyme 3	Enzyme 4	Enzyme 5
Enzyme 0	8	4	4			
Enzyme 1	3	2	-			
Enzyme 2	-	3	3			
Enzyme 3	4	6	6			
Enzyme 4	5	4	7			
Enzyme 5	-	8	3			

Tabel 4: Change-over matrix

2.4 Orders

An instance for the optimization problem is defined as a list of orders (jobs), this could be of the following form:

Job	**Product Type**	**Due date**
0	Enzyme 2	8
1	Enzyme 0	20
2	Enzyme 3	18
3	Enzyme 1	10
4	Enzyme 1	20
5	Enzyme 5	14

Tabel 5: Example instance

2.5 Translation to MILP

In the next section we will introduce the MILP-formulation for the scheduling problem. The DSM plant can be seen as a Flexible Job Shop with sequence dependent change-overs. The Flexible Job Shop is well described in the literature (see references from problem description). The processing times are define per product type, per unit operation. If two jobs operations of different product type are subsequently processed by the same machine, specialized cleaning is required. Therefore, change-overs are defined per machine per product combination. In the Instances folder of the GitLab repository, all relevant parameters for the MILP are included in the different files. The instance files contain lists of orders of different sizes (in increasing order). The due dates are included in the instance files, but not used yet in the MILP. This is due to the fact that the due dates are not considered as strict constraints. However, the due dates can be used to consider an alternative objective, e.g. tardiness.

3 MILP

Notation for MILP:

Symbol	Definition
Sets	
$\mathcal{P}$	Unordered set of product types $p \in \mathcal{P}$
$\mathcal{R}$	Unordered set of recipes $r \in \mathcal{R}$
$\mathcal{R}_p$	Recipe for product type $p \in \mathcal{P}$
$\mathcal{J}_p$	Unordered set of jobs that produce 1 standard quantity of product type $p \in \mathcal{P}$
$\mathcal{J}$	Unordered set of jobs with $i \in \mathcal{J}$
$\mathcal{O}$	Unordered set of different unit operations with $j \in \mathcal{O}$
$\mathcal{O}_i$	Sequence (ordered) of operations for producing job i (based on recipe), $\mathcal{O}_{i,l(i)}$ is last element
$\mathcal{M}$	Unordered set of machines with $k \in \mathcal{M}$
$\mathcal{M}_j$	Unordered set of machine alternatives that can execute operation $j \in \mathcal{O}$
$\mathcal{M}_h \cap \mathcal{M}_j$	Unordered set of machine alternatives that can execute operation h and j
Parameters	
d_i	Due date of job $i \in \mathcal{J}$
$p_{i,j,k}$	Processing time of operation $j \in \mathcal{O}$ of job $i \in \mathcal{J}$ at machine $k \in \mathcal{K}$
$co_{g,i,k}$	Change-over time required for first processing job $g \in \mathcal{J}$ and then $i \in \mathcal{J}$ at machine $k \in \mathcal{K}$
Decision variables	
S_{ijk} =	start time for processing operation j of job $i \in \mathcal{J}$ at machine $k \in \mathcal{M}$
C_{ijk} =	completion time for processing operation j of $i \in \mathcal{J}$ at machine $k \in \mathcal{M}$
F_i =	completion time for job $i \in \mathcal{J}$
$X_{i,j,k}$ =	binary variable, equals 1 if operation (i,j) is assigned to machine k
$Y_{i,j,g,h,k}$ =	binary variable, equals 1 if operation (i,j) precedes operation (g,h) at machine k
Objective value	
C_{max} =	$\max_i F_{i,k}$

MILP:

$$\min C_{max}$$

$$\sum_{k \in M_j} X_{ijk} = 1 \qquad \forall i \in J \quad \forall j \in O_i \tag{1}$$

$$S_{i,j,k} + C_{ijk} \leq X_{i,j,k} \cdot L \qquad \forall i \in J \quad \forall j \in O_i \quad \forall k \in M_j \tag{2}$$

$$C_{i,j,k} \geq S_{i,j,k} + p_{i,j,k} - (1 - X_{i,j,k}) \cdot L \qquad \forall i \in J \quad \forall j \in O_i \quad \forall k \in M_j \tag{3}$$

$$S_{i,j,k} \geq C_{g,h,k} + co_{g,i,k} - (2 - X_{i,j,k} - X_{g,h,k} + Y_{i,j,g,h,k}) \cdot L \quad \forall i < g \; \forall j \in O_i \; \forall h \in O_g \forall k \in M_j \cap M_h \tag{4}$$

$$S_{g,h,k} \geq C_{i,j,k} + co_{i,g,k} - (3 - X_{i,j,k} - X_{g,h,k} - Y_{i,j,g,h,k}) \cdot L \quad \forall i < g \; \forall j \in O_i \; \forall h \in O_g \forall k \in M_j \cap M_h \tag{5}$$

$$\sum_{k \in M_j} S_{i,j,k} \geq \sum_{k \in M_j} C_{i,j-1,k} \qquad \forall i \in J \quad \forall j \in O_i - \{O_{i,l(i)}\} \tag{6}$$

$$F_i \geq \sum_{k \in M_j} C_{i,O_{i,l(i)},k} \qquad \forall i \in J \tag{7}$$

$$C_{max} \geq C_i \qquad \forall i \in J \tag{8}$$

Where:

1. Each operation is assigned to only 1 machine.
2. Start time and completion time of operation (i, j) at machine k are set to 0 if they are not assigned to machine k.
3. Disjunctive constraint guarantees that the difference between the start and the completion times equals at least the processing time on machine.
4. Precedence constraint: two jobs cannot be done at one machine at the same time, there must be a precedence relation between two jobs.
5. Precedence constraint: two jobs cannot be done at one machine at the same time (the other way around).
6. Ensures that precedence relations between two operations are not violated, i.e. operation $(i, j-1)$ must always be completed before operation (i, j).
7. Determines the finish time of a job by considering the completion time for the final operation of the job.
8. Make span constraint.