06-665 Process Systems Modeling (Spring 2018)
Project Report

Application of Simultaneous Multi-Product Multi-Stage Batch Processing and Alternative Formulation

By Team 3:

Aditya Chindhade

Eloy Fernandez

Naien He

Wen Zhong

Keywords: Mixed-integer programming; batching and scheduling; chemical batch processes;

Class Modules: 3A: Steady State (SS) Mixed-Integer Programming (MIP)
3B: Multiperiod (MP) Mixed-Integer Programming (MIP)

Table of contents:

List of figures:

List of tables:

Abstract

In this project, we will implement a mixed-integer programming formulation for the simultaneous batching and scheduling in a multi-product and multi-stage plant. We will demonstrate that the simultaneous optimization approach generates better solutions than the conventional approach, where batching and scheduling decisions are carried out sequentially.

To account for non-linearity introduced by simultaneous optimization in a continuous-time approach, we introduce additional batch-selection (binary) and batch-size (continuous) variables. Utilizing general disjunctive programming techniques, demand-satisfaction, and unit-capacity constraints formulated as convex hull, where assignment constraints are active only for the batches that are selected. In addition, we will also implement an alternate formulation to handle dependent changeover costs.

Finally, we will implement two classes of tightening formulations to enhance the model's computational performance. One set will be used to fix a subset of sequencing variables based on pre-calculated feasibilities, while the other will be used to derive and add inequalities based on time windows. We will demonstrate the effect of each tightening formulations on the problem size and tightness. Their overall effectiveness on different types of problems will be studied and discussed.

1. Introduction and Problem Statement

1.1 Motivation

Most batch processing facilities are multi-batch and multi-stage processes, where customer orders are divided into several batches first, then processed through all stages (Pinedo M, 2002). Typically, there are three levels of decisions in a batch processing problem: Lot-sizing → Unit assignment → sequencing (Harjunkoski *et al.,* 2014). In the past, conventional continuous time formulations have not been able to address the simultaneous optimization of all three levels of decisions (Kondili *et al.* 1993; Janak *et al.,* 2004). As a result, conventional methods take a two-step approach that decouples batching decisions from scheduling (unit assignment and sequencing) decisions (Méndez *et al*., 2006).

This two-step approach is only effective when dealing with fixed demand, and when all units have similar capacity (Maravelias and Sung, 2009). As a result, this approach often leads to suboptimal solutions. Moreover, in a two-step approach, batch size and processing time are fixed for scheduling problems after

batch decisions are made. If batching is a part of the optimization, then the processing times must also vary with the batch sizes (Maravelias, 2006). Recently, multiple grid method was proposed for simultaneous batching and scheduling, but the process is only valid for single stages (Castro *et al*., 2006; Castro *et al*., 2007).

1.2 Problem Statement

To address the above-mentioned limitations, the goal of this project is to implement a new continuous time formulation that can simultaneously optimize batching and scheduling decisions. In addition, batch processing time will be treated as a variable dependent on batch sizes. We will consider two base formulations, the global-sequence based model, and the local-sequence based model.

As with a lot of simultaneous optimization approaches, the computational performance of the base model is limited by large degrees of freedom. Therefore, we will implement two classes of tightening formulations in order to increase solution speed.

The finalized model is a mixed integer linear programming (MILP). Different objective functions we account for are the make span, earliness, processing cost and production profit, which will all be used to test the performance of the model under different situations.

1.3 System and Software

The new formulation is tested on 5 batch scheduling examples found in literature. They are solved both by CPLEX (IBM, 2005) in GAMS (GAMS Development Corporation, 2013; Brooke, 1988) and Gurobi (Gurobi Optimization, 2016) in Pyomo (Hart et al., 2011).

2. Literature Survey

Conventional MILP formulations for modeling such processes can be divided into discrete-time (slot based) formulations and continuous-time (precedence based) formulations (Pinto and Grossmann, 1995; Pinto and Grossmann, 1996; Harjunkoski and Grossmann, 2002). In general, continuous-time has several advantages over discrete-time approaches, such as that the solution accuracy doesn't depend on the size of discrete time slots, and that the sequencing and timing decisions are embedded in the precedence decisions, which greatly reduces model complexity (Pinto and Grossmann, 1996; Hui and Gupta, 2000ab).

However, the continuous-time approach requires certain assumptions to work as a MILP. Take example constraints from a conventional approach (Birewar and Grossmann, 1997): for this formulation to work as an MILP, either batch size B_i or batch number n_i has to be a pre-determined parameter. Similarly, either batch number n_i or processing time t_i has to be pre-determined as well (Gupta and Karimi, 2003).

$$\boldsymbol{n_i} \cdot \boldsymbol{B_i} = Q_i \tag{23}$$

$$\sum_{i \in I} \boldsymbol{n_i} \cdot \boldsymbol{\tau_i} + \sum_{i \in I} \sum_{k \in K} SL \cdot NPRS_{ik} \leq H \tag{24}$$

Conventionally, researchers have overcome this difficulty by treating the problem as two separate problems; batching and scheduling. Optimum batching decisions are first solved depending on orders and unit capacities. Then the fixed batch sizes and processing times are fed into the scheduling problem. This approach is applicable when the demand is fixed and the unit capacities for products are the same or similar at each stage. Without these assumptions, the two-step approach will produce sub-optimum solutions (Sundaramoorthy and Maravelias, 2008).

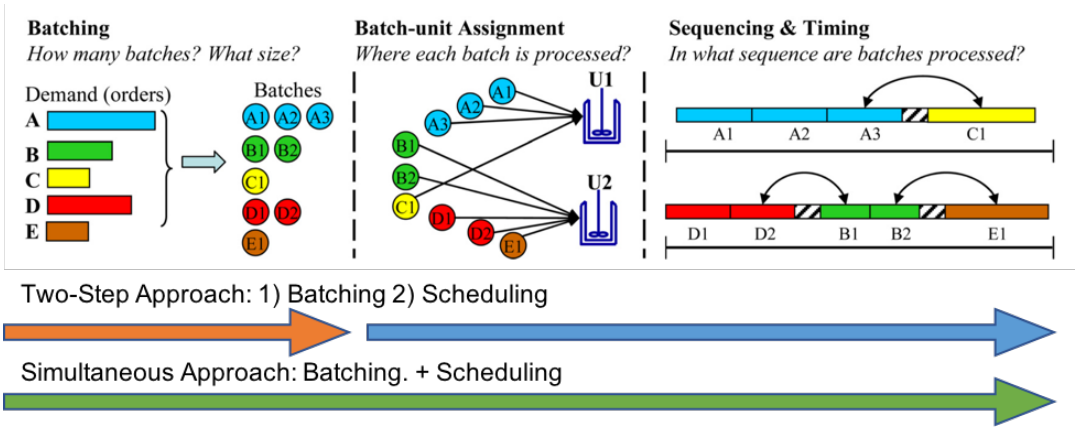

Figure 1. Major decisions in batch process scheduling

The proposed model by Sundaramoorthy and Maravelias allows simultaneous batching and scheduling, where the selection and sizing of batches, assignment of batches to units and sequencing decisions are made simultaneously under the assumptions of no preemption, and that all batches visit all stages and unlimited intermediate storage. The model has two formulations to cover a range of objective functions, including minimization of lateness, earliness, processing costs, make span and maximization of profits and changeover costs, if any. Below is a table of different approaches to solve batching and scheduling problems. This model covers both immediate and global precedence based sequential methods (Harjunkoski *et al*., 2014; Pinto and Grossmann, 1998).

Table 1.Type of mathematical programming for different classes of problems

	Processing characteristics and constraints								
	Changeover times	Changeover costs	Storage restric-tions	Shared utilities	Intermediate due dates	Variable batch sizes	Variable processing times	Holding and utility costs	Time varying utility costs
Sequential									
Precedence-based									
Immediate	✓	✓	✓		✓	✓	✓		
Global	✓		✓		✓	✓	✓		
Time-grid-based									
Discrete (common)			✓	✓	✓	✓		✓	✓
Continuous (unit-specific)	✓	✓			✓	✓	✓		
Network									
Time-grid-based									
Discrete									
Common	✓	✓	✓	✓	✓	✓		✓	✓
Specific	✓		✓		✓	✓		✓	✓
Continuous									
Common	✓	✓	✓	✓	✓	✓	✓		
Specific	✓	✓	✓		✓	✓	✓		

3. Mathematical Models and Methods

3.1 Model Formulation

Given a multistage multiproduct batch plant with: (i) a set of orders $i \in I$ with demand q_i, release/due time r_i/d_i and selling price π_i. (ii) a set of units $j \in J$ with minimum/maximum capacity b_j^{min}/b_j^{max}, fixed/proportional processing time constants $\tau_{ij}^F/\ \tau_{ij}^P$, and fixed/proportional processing costs c_{ij}^F and c_{ij}^P. (iii) a set of stages $k \in K$ with units $j \in J_k$ at stage k. (iv) a set of forbidden units JF_i and forbidden paths $(j, j')\ \in FP$ for all orders. (v) changeover time $\tau_{ii'j}^{ch}$ from a batch of order i to a batch of order i' on unit j.

3.1.1 Number of Batches

To enable the simultaneous optimization of batching and scheduling, we introduce index l to denote the lth potential batch index for product i. Theoretically l can take values up to $+\infty$ (e.g. each batch produces almost nothing and would require an infinite number of batches to make up for the total demand). However, given a specific problem, the range of batches for order i can be calculated using Eqn (1) and (2):

$$l_i^{min} = \left\lceil \frac{q_i}{\hat{b}_i^{max}} \right\rceil \qquad \forall i \tag{1}$$

$$l_i^{max} = \left\lceil \frac{q_i}{\hat{b}_i^{min}} \right\rceil \qquad \forall i \tag{2}$$

where $\hat{b}_i^{max} = \min\limits_{k \in K}[\max\limits_{j \in JA_{ik}}(b_j^{max})]$ and $\hat{b}_i^{min} = \max\limits_{k \in K}[\min\limits_{j \in JA_{ik}}(b_j^{min})]$ are the maximum and minimum feasible batch sizes for order i. Furthermore, we define a range set $L_i = \{1, 2, \dots, l_i^{max}\}$ of potential batches.

3.1.2 Batching and Assignment

Since each product is disaggregated into an unknown number of batches, we will split the total demand q_i into the summation of all the potential batches for product $i \in I$.

$$\sum_{l \in Li} B_{il} \geq q_i \qquad \forall i \tag{3}$$

If both maximum and minimum demand q_i^{max} and q_i^{min} are provided, Eqn (3a) is used instead:

$$q_i^{min} \leq \sum_{l \in Li} B_{il} \leq q_i^{max} \qquad \forall \text{i} \tag{3a}$$

The following convex hull makes sure that if a batch is not selected, its batch size will be forced to 0.

$$\hat{b}_i^{min} Z_{il} \leq B_{il} \leq \hat{b}_i^{max} Z_{il} \qquad \forall i, l \in L_i \tag{4}$$

If a certain batch Z_{il} is active, then this batch can select only one unit $j \in J$ at stage k.

$$Z_{il} = \sum_{j \in JAik} X_{ilj} \qquad \forall i, l \in L_i, k \tag{5}$$

The model further disaggregates the batch sizing variable B_{il} into unit assignment variable $\bar{B}_{ilj}$.

$$B_{il} = \sum_{j \in JAik} \bar{B}_{ilj} \qquad \forall i, l \in L_i, k \tag{6}$$

Similar to Eqn (4), using the convex hull formulation, when a disaggregated batch X_{ilj} is selected, batch size $\bar{B}_{ilj}$ is bounded by the capacity of unit j. If a disaggregated batch X_{ilj} is not selected, $\bar{B}_{ilj}$ must be 0.

$$b_j^{min} X_{ilj} \leq \bar{B}_{ilj} \leq b_j^{max} X_{ilj} \qquad \forall i, l \in L_i, j \tag{7}$$

A symmetry breaking constraint is added to makes sure that Z_{il} is selected first before $Z_{i(l+1)}$.

$$Z_{i(l+1)} \leq Z_{il} \qquad \forall i, l \in L_i \tag{8}$$

3.1.3 Sequencing and Timing

We define a global sequencing binary variable $Y_{ili'l'k}$ to denote the sequencing between batches (i, l) and (i', l') at a given stage k. $Y_{ili'l'k}$ is 1 if batch (i, l) is processed before (i', l') in stage k.

$$X_{ilj} + X_{i'l'j} - 1 \leq Y_{ili'l'k} + Y_{i'l'ilk} \qquad \forall (i, l, i', l') \in IL: i \leq i', k, j \in JA_{ik} \cap JA_{i'k} \tag{9}$$

This constraint is written over all possible sequencing pairs, which consist of either different products, or different batches from the same product. The set IL is defined as $IL = \{i, l \in L_i, i', l' \in L'_i : (i \neq i') \vee ((i = i') \wedge (l \neq l'))\}$. With variable $Y_{ili'l'k}$ activated, we can model the finish time $T_{i'l'k}$ for batch (i', l').

$$T_{i'l'k} \geq T_{ilk} + \sum_{j \in JA_{ik}} \left(\tau^F_{i'j} X_{i'l'j} + \tau^P_{i'j} \bar{B}_{i'l'j} \right) - H(1 - Y_{ili'l'k}) \qquad \forall (i, l, i', k') \in IL, k \tag{10}$$

In the case where changeover time is significant, we add changeover time by using Eqn (10a) instead.

$$T_{i'l'k} \geq T_{ilk} + \sum_{j \in JA_{ik} \cap JA_{i'k}} \left(\tau^F_{i'j} X_{i'l'j} + \tau^P_{i'j} \bar{B}_{i'l'j} + \tau^{ch}_{il'j} X_{i'l'j} \right) - H(1 - Y_{ili'l'k}) \quad \forall (i,l,i',k') \in IL, k \quad (10a)$$

Between two successive stages, the difference in the finish time of batch (i,l) must be greater than its processing time at the latter stage.

$$T_{il(k+1)} \geq T_{ilk} + \sum_{j \in JAi(k+1)} (\tau^F_{ij} X_{ilj} + \tau^P_{ij} \bar{B}_{ilj}) \qquad \forall i,l \in L_i, k < |K| \quad (11)$$

3.1.4 Miscellaneous

For each order $i \in I$, the release time r_i and due time d_i should be satisfied. Moreover, the finish time at each stage k is also constrained by adding processing time at previous stages.

$$T_{ilk} \geq r_i Z_{il} + \sum_{k' \leq k} \sum_{j \in JA_{ik'}} (\tau^F_{ij} X_{ilj} + \tau^P_{ij} \bar{B}_{ilj}) \qquad \forall i,l \in L_i, k \quad (12)$$

$$T_{ilk} \leq d_i Z_{il} - \sum_{k' > k} \sum_{j \in JA_{ik'}} (\tau^F_{ij} X_{ilj} + \tau^P_{ij} \bar{B}_{ilj}) \qquad \forall i,l \in L_i, k \quad (13)$$

Forbidden paths and assignments should also be satisfied.

$$X_{ilj} + X_{ilj'} \leq Z_{il} \qquad \forall \, \mathrm{i,l} \in L_i, (\mathrm{j,j'}) \in \mathrm{FP} \quad (14)$$

$$X_{ilj} = 0 \qquad \forall \, \mathrm{i,l} \in L_i, \mathrm{j} \in JF_i \quad (15)$$

Similar to Eqn (8), batch size variables also requires symmetry breaking.

$$B_{i(l+1)} \leq B_{il} \qquad \forall \, \mathrm{i,l} \in L_i \quad (16)$$

For each order, the selection binary variable Z_{il} is fixed to 1 for a pre-computed for all $l < l_i^{min}$.

$$Z_{il} = 1 \qquad \forall \mathrm{i,l} \leq l_i^{min} \quad (17)$$

3.1.5 Extension Formulation #1: Sequence-Dependent Changeovers

To address the fact that $Y_{ili'l'k}$ does not indicate immediate change, a local sequence model is formulated. We define a new local sequence binary variable $W_{ili'l'j}$ which denotes adjacent batches. Furthermore, we define variables XF_{ilj} and XL_{ilj} which are active if batch (i,l) is assigned first and last respectively to unit j. This formulation uses a technique from the travelling salesman problem (TSP) (Padberg and Rinaldi, 1991).

$$\sum_{i'l' \in L_{i'}} W_{i'l'ilj} + XF_{ilj} = X_{ilj} \qquad \forall \mathrm{i,l} \in \mathrm{L}_i, \mathrm{j} \quad (9a)$$

$$\sum_{i'l' \in L_{i'}} W_{ili'l'j} + XL_{ilj} = X_{ilj} \qquad \forall \mathrm{i,l} \in \mathrm{Li}, \mathrm{j} \quad (9\mathrm{b})$$

Equations (9a) and (9b) represent classic assignment constraints in a TSP. For any given batch (i, l) in a unit j, it will either be the first process (XF_{ilj}), or will change from other processes ($\sum_{i'l' \in L_{i'}} W_{i'l'ilj}$). Similarly, it will either be the last process (XL_{ilj}) or has to change to other processes ($\sum_{i'l' \in L_{i'}} W_{ili'l'j}$). At most only one order can take the place of the first or last order in the sequence.

$$\sum_{i,l \in Li} XF_{ilj} \leq 1 \qquad \forall \text{j} \tag{9c}$$

$$\sum_{i,l \in Li} XL_{ilj} \leq 1 \qquad \forall \text{j} \tag{9d}$$

Similar to Eqn (10), if batch (i', l') immediately follows batch (i, l), then their finish times are constrained:

$$T_{i'l'k} \geq T_{ilk} + \sum_{j \in JAik \cap JAi'k} \left(\tau^F_{i'j} X_{i'l'j} + \tau^P_{i'j} \bar{B}_{i'l'j} + \tau^{ch}_{il'j} W_{ii'l'j} \right) - H\left(1 - \sum_{j \in JAik \cap JAi'k} W_{ili'l'j}\right) \quad \forall (i, l, i', l') \in IL, k \tag{10b}$$

3.1.6 Extension Formulation #2: Tightening A

To enhance the computational performance, we introduce a new set of tightening constraints (Castro, 2005; Castro and Grossmann, 2006) that can be pre-computed. With this we compute the earliest start time (est_{ik}), earliest finish time (eft_{ik}), latest start time (lst_{ik}) and latest finish time (lft_{ik}). Since batch size is unknown before solving, we use a minimum batch size of order i (pre-computed parameter) ${\tau_{ij}}^{min} = \tau^F_{ij} + \tau^P_{ij} \hat{b}_i^{min}$.

Table 2. Illustration of pre-computation on $\boldsymbol{est_{ik}, eft_{ik}, lst_{ik}, lft_{ik}}$ for Example 3

(i,k)	est_{ik}	eft_{ik}
$(A, K1)$	$= r_A = 20$	$= est_{A,K1} + min\,(\tau_{A,J2}, \tau_{A,J3})$ $= 20 + min(4.3, 5.2) = 24.3$
$(A, K2)$	$= r_A + min\,(\tau_{A,J2}, \tau_{A,J3})$ $= 20 + min(4.3, 5.2) = 24.3$	$= est_{A,K1} + min\,(\tau_{A,J4}, \tau_{A,J5}, \tau_{A,J6})$ $= 24.3 + min(3.5, 3.7, 3.3) = 27.6$

(i,k)	lst_{ik}	lft_{ik}
$(A, K1)$	$= lft_{A,K1} - min\,(\tau_{A,J2}, \tau_{A,J3})$ $= 46.7 - min(4.3, 5.2) = 42.3$	$= d_A - min\,(\tau_{A,J4}, \tau_{A,J5}, \tau_{A,J6})$ $= 50 - min(3.5, 3.7, 3.3) = 46.7$
$(A, K2)$	$= lft_{A,K2} - min\,(\tau_{A,J4}, \tau_{A,J5}, \tau_{A,J6})$ $= 50 - min(3.5, 3.7, 3.3) = 46.7$	$= d_A = 50$

If, for a given (i, k) and (i', k'), the earliest finish time for (i, k) is later than the latest start time for (i', k'), it means that the (i, k) cannot change to (i', k').

$$Y_{ili'l'k} = 0 \qquad \forall (i, l, i', l') \in IL, k\text{: eft}_{ik} > \text{lst}_{i'k} \tag{18}$$

$$W_{ili'l'j} = 0 \qquad \forall (i, l, i', l') \in IL, j \in J_k, k\text{: eft}_{ik} > \text{lst}_{i'k} \tag{18a}$$

Furthermore, we introduce $ub_{ii'j}$ which denotes the upper bound of start time of order i in unit j, assuming it is followed by order i'. To ensure the changeover is feasible, we have to make sure both i and i' meet their respective deadlines. This is done with the following equations:

$$ub_{ii'j} = \min \begin{pmatrix} d_i - \sum_{k'>K, j\in JA_ik'} \min\left(\tau_{ij'}{}^{min}\right) - \tau_{ij}{}^{min}, \\ d_{i'} - \sum_{k'>K, j'\in JA_{i'k'}} \min\left(\tau_{i'j'}{}^{min}\right) - \tau_{i'j}{}^{min} - \tau_{ij}{}^{min} \end{pmatrix} \forall \mathrm{i}, \mathrm{i}', \mathrm{j} \quad (19)$$

If $ub_{ii'j}$ is earlier than the earliest start time est_{ik}, it means that this changeover is guaranteed to introduce infeasibility in the system. Therefore, a new set is defined for all the feasible changeovers.

$$P_{i'k} = \left\{\mathrm{i} \in \mathrm{I} : est_{ik} \leq max_{j\in JA_ik'}\left(ub_{ii'j}\right)\right\} \quad \forall \mathrm{i}', \mathrm{k} \quad (20)$$

Table 3. Illustration of pre-computation on $\boldsymbol{P_{i'k}}$ for Example 3

(i, i', k)	j	$ub_{ii'j}$ from di	$ub_{ii'j}$ from $d_{i'}$	$\max_{\mathrm{j}\in\mathrm{JA_{ik}}}\left(ub_{ii'j}\right)$	est_{ik}	Comment
$(A, A, K2)$	$J4$	46.5	43.0	43.3	24.3	$est_{ik} < \max_{\mathrm{j}\in\mathrm{JA_{ik}}}\left(ub_{ii'j}\right)$ Thus, $i \in \mathrm{P}_{i'k}$
	$J5$	46.3	42.6			
	$J6$	46.7	43.3			
$(A, B, K2)$	$J4$	46.5	23.0	23.3	24.3	$est_{ik} > \max_{\mathrm{j}\in\mathrm{JA_{ik}}}\left(ub_{ii'j}\right)$ Thus, $i \notin \mathrm{P}_{i'k}$
	$J5$	46.3	22.6			
	$J6$	46.7	23.3			

If the pair (i, l), (i', l') is not feasible, then we can fix the sequence binary variable to equal 0:

$$Y_{ili'l'k} = 0 \qquad \forall (i, l, i', l') \in IL, k, i \notin \mathrm{P}_{i'k} \quad (21)$$

$$W_{ili'l'k} = 0 \qquad \forall (i, l, i', l') \in IL, , j \in Jk, k, i \notin \mathrm{P}_{i'k} \quad (21a)$$

3.1.7 Extension Formulation #3: Tightening B

Here, we will implement another proposed algorithm to effectively reduce the computation time even further (Maravelias, 2006). We introduced a new parameter, a time window for each unit j, which is bounded by the earliest start time of its first batch (lower bound lbw_{jn}) and finish time of its last batch (upper bound ubw_{jn}). The available processing time left for unit j then becomes $ubw_{jn} - lbw_{jn}$. As a result, the total processing time plus changeover time in unit j has to fit in the available time window.

$$\sum_{i\in IC_{jn}, l\in Li}\left(\tau_{ij}^{F} X_{ilj} + \tau_{ij}^{P} \bar{B}_{ilj}\right) \leq \mathrm{ubw}_{jn} - \mathrm{lbw}_{jn} \quad \forall j, n \in Nj \quad (22)$$

$$\sum_{i\in IC_{jn}, l\in Li}\left(\tau_{ij}^{F} X_{ilj} + \tau_{ij}^{P} \bar{B}_{ilj}\right) + \sum_{\substack{(i,l,i'l')\in IL \\ i,i'\in IC_{jn}}} \tau_{ij}^{ch} W_{ili'l'j} \leq \mathrm{ubw}_{jn} - \mathrm{lbw}_{jn} \quad \forall k, j \in J = J_k, n \in N_j \quad (22a)$$

T*able 4. Preprocessing Algorithm*

for $k \in \{1,2,...|\mathbf{K}|\}$,
  for $j \in \mathbf{J}_k$,
    Initialize auxiliary set: $\mathbf{IC}' = \mathbf{I} \in \mathbf{IA}_j$
    Reset cut number: $n = 0$
    Do while $\mathbf{IC}' \neq \emptyset$,
      Initialize active set: $\mathbf{IC}_{jn} = \mathbf{IC}'$
      Do while $\mathbf{IC}_{jn} \neq \emptyset$ and $\max_{i \in \mathbf{IC}_{jn}} (\mathrm{lft}_{ik}) = \max_{i \in \mathbf{IC}'_{jn}} (\mathrm{lft}_{ik})$,
        Update cut number: $n = n + 1$
        Calculate parameters:
          $\mathrm{lbw}_{jn} = \min_{i \in \mathbf{IC}_{jn}} (\mathrm{est}_{ik})$
          $\mathrm{ubw}_{jn} = \max_{i \in \mathbf{IC}_{jn}} (\mathrm{lft}_{ik})$
          $\mathrm{tail}_{jn} = \min_{i \in \mathbf{IC}_{jn}} \sum_{k' > k} \min_{j' \in \mathbf{JA}_{ik'}} (\tau^{min}_{ij'})$
        Update $\mathbf{IC}_{jn}$ by removing the order(s) with the earliest start time:
          $\mathbf{IC}_{jn} = \mathbf{IC}_{jn} \{i \in \mathbf{I}: \mathrm{est}_{i'k} = \mathrm{lbw}_{jn}\}$
      Update $\mathbf{IC}'$ by removing the order(s) with the latest finish time:
        $\mathbf{IC}' = \mathbf{IC}' \backslash \{i \in \mathbf{I}: \mathrm{lft}_{ik} = \mathrm{ubw}_{jn}\}$
Calculate the number of inequalities for unit j: $\mathbf{N}_j = n$

The main challenge here is to identify as many 'scenarios' as possible, in order to generate as many cuts as possible to tighten the problem. We will use an iterative outer and inner loop approach, to systematically generate all valid cuts by iterating through all possible combinations of products, identifying their time window and applying cuts. The following illustrates one inner loop iteration process.

Table 5. Illustration of Preprocessing Algorithm on Time Window for Unit J1

Cuts	Time Window Upper Bound	Time Window Lower Bound	IC_{jn} (Possible Combinations)
0	80	0	B, C, D, E, F, G, H, I, J. K, L
1	80	5	B, D, G, I, J, K, L
2	80	25	D, G, I. J. K, L
3	80	30	G, J, K, L
4	80	40	J, K
5	76.8	0	B, C, D, E, F, G, H, K, L
6		5	B, D, G, K, L
…	…	…	…
21	16.8	0	C, H
22	17.7	0	C

3.2 Data Source and Solution Methods

In this project, we solved 5 different scenarios in a multi-batch, multi-stage production setting, using data obtained from (Sundaramoorthy and Maravelias, 2008). Scenarios 1 and 2 use small sized base global sequence models to demonstrate its advantage over traditional sequential approaches. Scenarios 3 and 4 demonstrate the speed increase in computational performance by tightening A and B. Scenario 5 applies the local sequence formulation to demonstrate the application when changeover cost is significant.

Table 6. Objective and Constraints for Scenario 1-5

Scenario \ Formulation		Sequential Scheduling	Global Sequence	Global Sequence Tightening A	Global Sequence Tightening A + B	Local Sequence Tightening A + B
1	min	Make Span	Make Span			
	s.t.	Eqn (3) – (17) Fixed Batching	Eqn (3) – (17)			

Scenario \ Formulation		Sequential Scheduling	Global Sequence	Global Sequence Tightening A	Global Sequence Tightening A + B	Local Sequence Tightening A + B
2	min	Earliness	Earliness			
	s.t.	Eqn (3) – (17) Fixed Batching	Eqn (3) – (17)			
3	min		Production Cost	Production Cost	Production Cost	
	s.t.		Eqn (3) – (17)	Eqn (3) – (17) Eqn (8) – (21)	Eqn (3) – (17) Eqn (8) – (21) + (22)	
4	max		Profit	Profit	Profit	
	s.t.		Eqn (3a) – (17)	Eqn (3a) – (17) Eqn (8) – (21)	Eqn (3a) – (17) Eqn (8) – (21) + (22)	
5	min					Production Cost + Changeover Cost
	s.t.					Eqn (3) – (17) (9abcd) (10b) Eqn (8) – (21) + (22) (18a) + (21a) +(22a)

3.2.1 Degrees of Freedom

A simultaneous batching and scheduling problem has three levels of degrees of freedom: (i) the selection and sizing of batches, (ii) the assignment of batches to processing units, and (iii) the sequencing and timing of batches in each unit. The first level corresponds to the batching problem, whereas the second and the third levels comprise the scheduling problem. As a result, the model contains high degrees of freedom. Take scenario 3 (Global Sequence Tightening A + B formulation) as an example:

Table 7. Degrees of Freedom

Product	Stages	Units	Continuous Variable	Binary Variable	Equality Constraint	Inequality Constraint
10	2	6	532	7257	2974	19260

After removing equality constraints, the remaining degrees of freedom is 4815, and these are bounded by 19260 inequality constraints, which means that a lot of those inequalities are soft constraints.

3.2.2 Critical Constraints

Eqn (5): Only one unit can be chosen during each stage. In the context of the problem, we focus on applications in pharmaceuticals and the food industry, which have quality control requirements to make sure batches do not mix.

Eqn (8) and (16): Symmetrical breaking constraints. Symmetry breaking is not required for obtaining the correct solution, but are are critical in increasing the computational performance

Eqn (12) and (13): The order release date and due date. There are treated as hard constraints in our formulation. However, sometimes the entire schedule can be greatly optimized if one of the bottleneck orders is allowed a small slack. Therefore, it is common in industry to assign a penalty value to lateness. For instance, a penalty value could pick the market value of the product, as if to purchase it from a competitor to fulfill the order.

3.2.3 Modelling Approach

In this project, we used general disjunctive programming (GDP) techniques to decompose the problem and subsequently achieve simultaneous optimization. Specifically, we used a convex hull formulation for the decomposition of total production q_i to batch assignment B_{il}, then down to unit assignment $\bar{B}_{ilj}$.

$$\begin{bmatrix} Z_{il} = 1 \\ \begin{pmatrix} X_{ilj} = 1 \\ \hat{b}_i^{min} \le B_{il} \le \hat{b}_i^{\max} \end{pmatrix} \bigvee \begin{pmatrix} X_{ilj} = 0 \\ \bar{B}_{ilj} = 0 \end{pmatrix} \end{bmatrix} \bigvee \begin{bmatrix} Z_{il} = 0 \\ B_{il} = 0 \\ \bar{B}_{ilj} = 0 \end{bmatrix} \qquad \forall i, l, j \qquad (25)$$

Sequencing constraints are written over a high dimension set (i, l, i', l', j), and if we were to use convex hull formulation, it would generate too many constraints. As a result, big-M type formulation is used, where the value of M is set to be the planning horizon to reduce the over-relaxation big-M constraints produce.

$$T_{i'l'k} \ge T_{ilk} + \sum_{j \in JA_{ik}} \left(\tau^F_{i'j} X_{i'l'j} + \tau^P_{i'j} \bar{B}_{i'l'j} \right) - H(1 - Y_{ili'l'k}) \qquad \forall (i, l, i', k') \in IL, k \qquad (10)$$

The other key challenge in this formulation is the modeling of variable processing times. Here we use the typical formulation for discontinuous domains called “fixed charge cost model” to model the discontinuities formed by using a binary variable with a continuous variable.

$$Processing\ Time = \tau^F_{i'j} X_{i'l'j} + \tau^P_{i'j} \bar{B}_{i'l'j} \qquad (26)$$

3.2.4 Software Package

All 5 different formulations, together with 5 sets of data, are solved both by CPLEX (IBM, 2005) in GAMS (GAMS Development Corporation, 2013) and Gurobi (Gurobi Optimization Inc, 2016) in Pyomo (Hart *et al*., 2011). CPLEX is developed by IBM to use simplex algorithms together with branch and bound to solve mixed integer linear programming problems. We picked CPLEX together with GAMS for its reliability in solving large scale problems. Gurobi is a mixed integer linear solver that is fully integrated in Pyomo.

Although it is not as reliable as CPLEX, it fully exploits parallelism in a multi-core computer and has some leading MIP cutting plane routines and heuristics. In this project, we formulated a number of tightening constraints to increase computational performance, and Gurobi was used to further test its performance.

4. Results and Discussion

4.1 Example 1

The first example shows how simultaneously optimizing both batching and scheduling is better than simply optimizing via a sequential approach. The problem is set up with three products A, B, and C, which are to be processed through two stages, each having two units, with the goal to minimize the make span to produce 30 kg of A, 40 kg of B, and 40 kg of C.

When the problem is solved with a standard sequential approach, the total time comes out to 17.2 hours till completion. The model was then changed to simultaneously optimize both batching and scheduling, and the new objective came out to 14.5 hours to completion.

Table 8. Objective Value for Example 1 with Sequential and Simultaneous Approach

	Variables Count		Objective (Min Make Span)
Sequential Approach	38 Continuous	84 Binary	17.2 h
Simultaneous Approach	44 Continuous	90 Binary	14.5 h

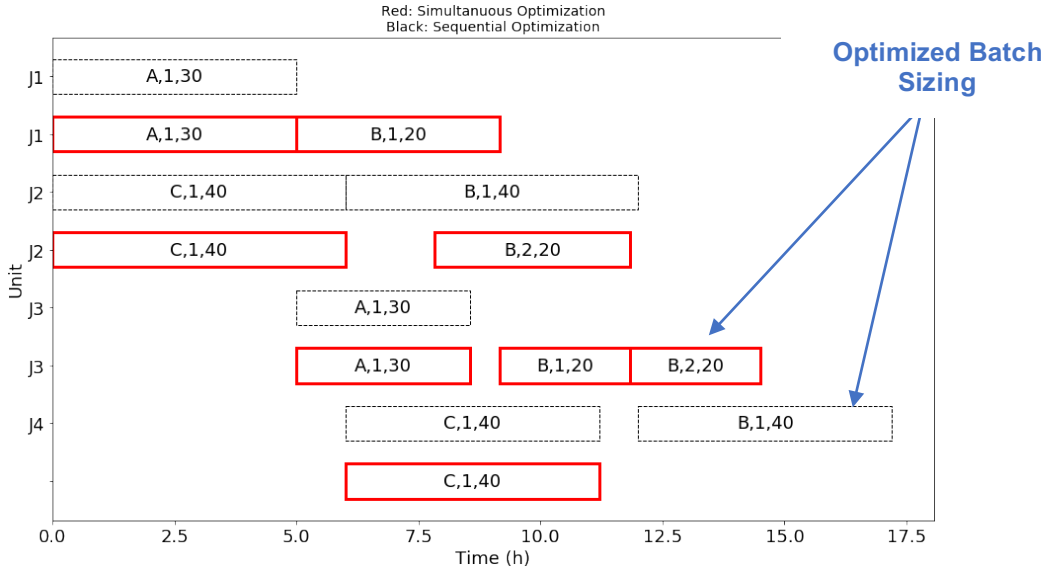

Figure 2. Gantt chart for Example 1

A visual representation of the schedule is shown in the Gantt chart above, which shows when using a sequential approach, the algorithm did not catch that order B can be separated into two parts. Since this

is a basic problem, it is easy to see which option is better, but when the problems get larger, it gets more difficult to visually determine the best schedule without using a simultaneous optimization approach.

4.2 Example 2

The second example illustrates the advantage of using variable processing times in the simultaneous approach, when compared with traditional fixed approaches. The example is set up with three products A, B, and C, which are to be processed in two stages, with two and three units respectively. The overall goal in this example is to minimize the earliness, while producing 30 kg of A, 70 kg of B, and 55 kg of C, at 10 hours, 25 hours, and 20 hours respectively in each stage.

Table 9. Objective Values for Example 2 with Sequential and Simultaneous Approach

	Variables Count		Objective (Min Earliness)
Sequential Approach	38 Continuous	84 Binary	5 h
Simultaneous Approach	89 Continuous	286 Binary	1.56 h

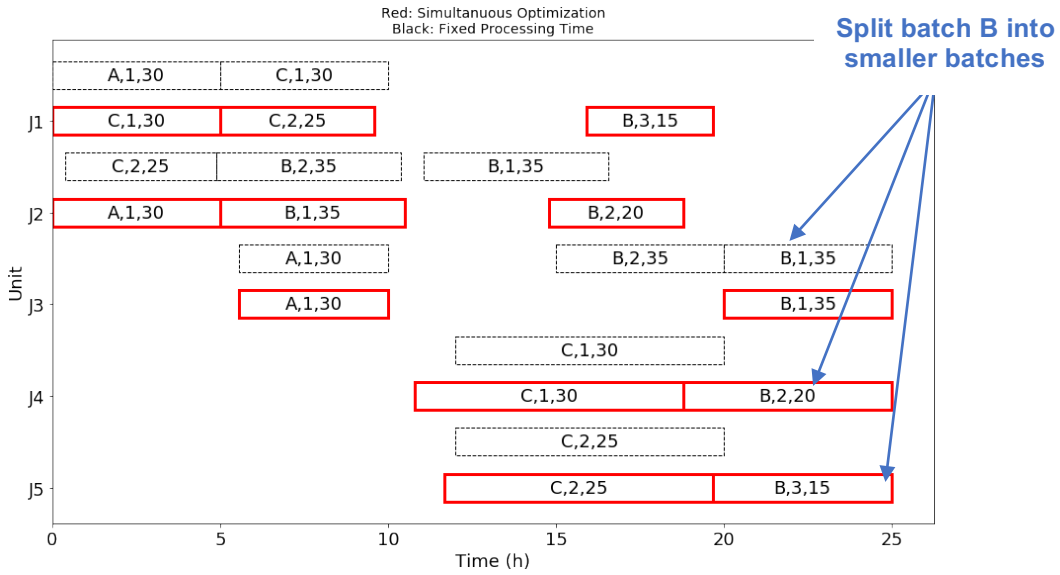

Figure 3. Gantt chart for Example 2

The Gantt chart of both solutions is shown above. The one with constant processing time has an earliness of 5 hours. Note that in a fixed processing time solution, both batches of order B take place on unit J3 even though units J4 and J5 are free from 20 to 25h. This is because the pre-determined batch sizes of order B are greater than the capacities of units J4 or J5. Even if the batch size of order B is reduced, a sequential approach still cannot take advantage of small slacks in complex scheduling problems due to constant processing times.

When we solved the same instance using variable processing times, we obtained a solution with an earliness of 1.56 hours. In this case, the second batch of order B is divided into two batches with shorter processing times, and then was assigned to units J4 and J5, leading to a better solution. The model was then able to modify the processing time by dynamically micro-adjusting batch size, separating the order into different units, and making the overall process more efficient.

In these two basic examples, we showed how we can get a better lateness and earliness by optimizing both batching and scheduling at the same time, and by modeling variable processing times of each batch. With the simultaneous batching and scheduling formulation, we are able to expand to larger models.

4.3 Example 3

In our third example, we started to move onto a larger model. Scenario 3 has 12 products with independent release and due time, two stages and six total units. The objective is to meet customer demands at a minimum production cost. The Gantt chart of the optimal solution is shown below:

Table 10. Objective Value for Example 3 with Simultaneous Approach

	Variables Count		Objective (Min Total Cost)
Simultaneous Approach	532 Continuous	7257 Binary	$3057

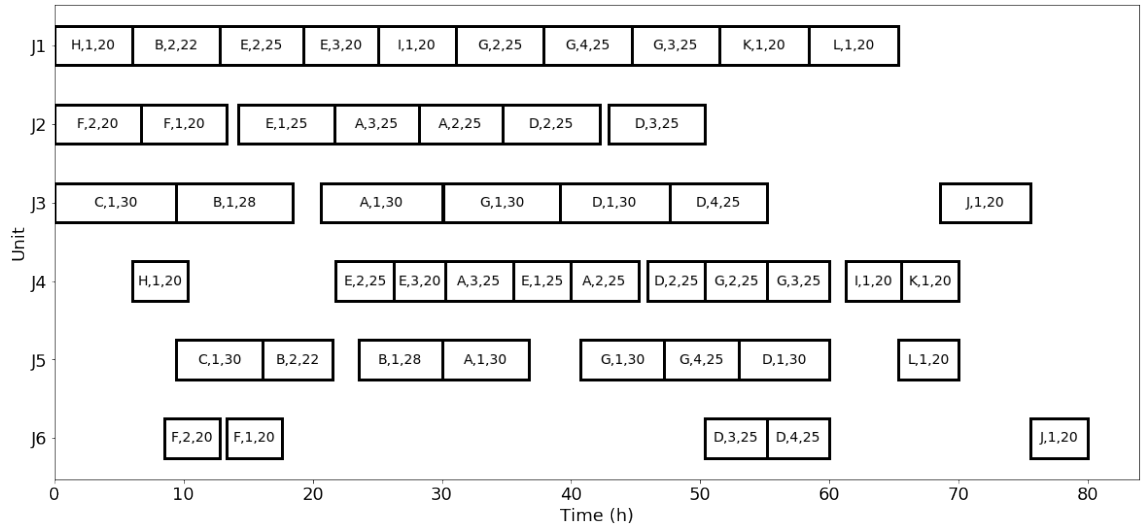

Figure 4. Gantt chart for Example 3

When compared with literature results, we found that although our solution has the same objective value as that in literature, some of the sequences are shifted. By sequentially conducting integer cuts on current solutions, we have proven that this problem has multiple optimal solutions. Although not immediately

obvious, multiple solutions are indeed possible in scenario 3, because the objective is to minimize total cost. As we have discussed in model formulation before, global sequence formulation can only model changeover time, not changeover cost, therefore, it makes sense that a scheduling problem with some slacks could produce multiple solutions as long as the simultaneous batching decision is optimized.

After solving scenario 3 using a global sequence base formulation, we compared the effectiveness of the two tightening methods we presented in section 3.1.6 and 3.1.7.

Table 11. Degree of Freedom and Model Statistic for Example 3

			Global Sequence	Global Sequence Tightening A	Global Sequence Tightening A + B
Variable	Before Pre-solve	Continuous	532		
		Binary	7257		
	After Pre-solve	Continuous	390	390	387
		Binary	**6161**	**5327**	**5136**
Constraints	Equality		**256**	**2974**	2974
	Inequality		19077	**19077**	**19260**
CPLEX Solution Time			267s	136s	52s

* Gurobi with tightening A+B: with 8 core processing average: 12s

As a general observation, in 2008, when scenario 3 was first published, CPLEX could only solve to 0.2% gap after 10000s without tightening the constraints. Ten years later, the same problem was solved in 267s without any computational enhancement. Further, Gurobi in Pyomo solves the model just 12 s. However, as we discussed in section 3.2.4, Gurobi leverages a lot hidden heuristics, and therefore the solution speed is not a definitive indication of good problem formulation. Despite being able to produce surprising results, heuristics often lacks reproducibility, and in a few attempts, we have found that Gurobi solves the model slower than CPLEX.

In terms of model statistics, we could observe that tightening A and B both contribute to the solution speed significantly, while maintaining their distinctive differences. As discussed in section 3.1.6, tightening A works by pre-computing feasible assignments and feasible predecessors. After generating those feasibility sets, the constraint works by setting infeasible variables to 0. This is shown in the increase of equality constraints from 256 to 2974, and reduction of binary variables from 6161 to 5327.

On the other hand, tightening B works by identifying time windows and adding inequality constraints to exclude assignments that exceed the maximum possible time. This is shown here by the increase in

inequality constraints from 19077 to 19260. At the same, binary variables drop from 5327 to 5136. Both methods combined reduce the problem's degree of freedom by 15%, while increasing the number of constraints by another 15%, a greatly improved computational performance.

4.4 Example 4

In our fourth example, we had 8 products with independent release and due time, two stages and six total units. The objective is more complicated as it had to calculate the total production profit (Prasad and Maravelias, 2008). The Gantt chart of the optimal solution is shown below:

Table 12. Objective Value for Example 4 with Simultaneous Approach

	Variables Count		Objective (Max Total Profit)
Simultaneous Approach	487 Continuous	6102 Binary	$2056

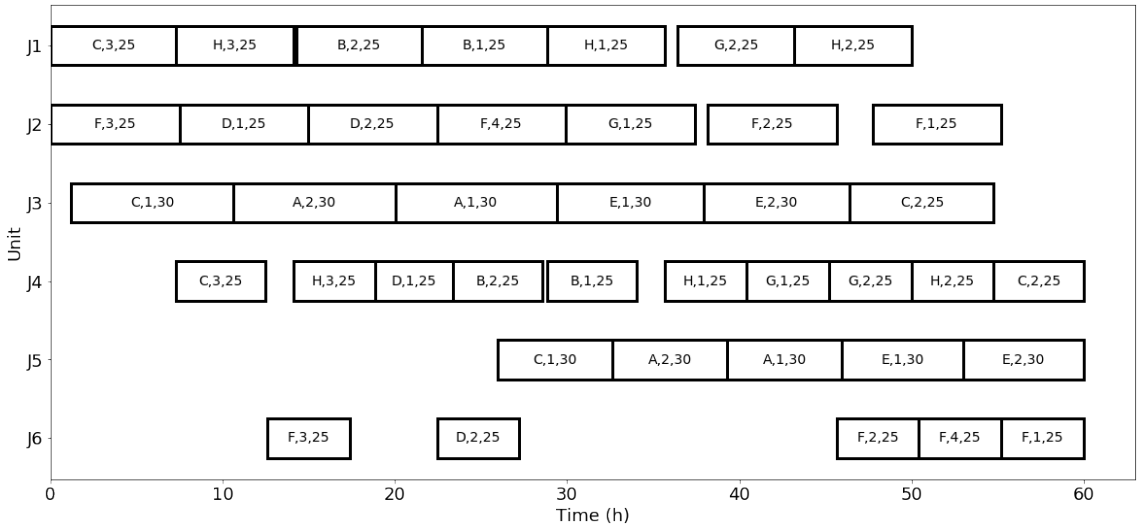

Figure 5. Gantt chart for Example 4

Table 13. Degree of Freedom and Model Statistic for Example 4

			Global Sequence	Global Sequence Tightening A	Global Sequence Tightening A + B
Variable	Before Pre-solve	Continuous		487	
		Binary		6102	
	After Pre-solve	Continuous	**394**	**394**	394
		Binary	**6032**	**6032**	6032
Constraints	Equality		**223**	**223**	223
	Inequality		**16183**	**16183**	16226
CPLEX Solution Time			10000* s	10000* s	520 s

* gap1 = 1.6%, gap2 ~ 0% * Gurobi with tightening A+B: with 8 core processing average: 20s

Similar to example 3, we tested different tightening constraints on the new objective to maximize profit. To our surprise, although the problem size is slightly smaller than example 3, CPLEX could not find a maximum after 10000s. This reflects the impact of different objective functions. With a relatively simple objective function like total cost, example 3 is solved in all three formulations. However, with a complex objective that considers both selling price and production cost, the importance of tighter formulation is emphasized.

Another important limitation we learnt from model statistics is that tightening formulation A could be ineffective depending on specific problem data. In example 4, the reason why tightening formulation A offers little improvement over base case is because it does not generate any effective cuts. It doesn't reduce the number of variables, nor does it increase the number of constraints.

This observation can be further explained by the complexity required to derive tightening constraints A and B. In essence, tightening constraint A catches infeasibility through one step of manipulation. If the data given was of high quality, then it cannot derive any meaningful cuts. On the other hand, tightening formulation B, in this case still increases the number of inequalities cut, which eventually leads to the optimal solution. This can be explained by the fact that tightening formulation B requires a deeper level of data manipulations and covers all potential combinations. This makes tightening formulation B much more effective.

4.5 Example 5

Example 5 illustrates the application of a local sequence, where sequence-dependent changeover costs become significant. In addition to price, demand, release and due time, changeover costs and changeover times are also considered. We processed six orders through two stages, each containing two units. The objective is to minimize the production cost and changeover cost. When compared with the literature result, our solution has different batch assignments and scheduling results. However, the objective value is the same with what literature provided as $2960. The Gantt chart for the optimal value is shown below:

Table 14. Objective Value for Example 5 with Local Sequence Approach

	Variables Count		Objective (Min Total Cost)
Local Sequence	229 Continuous	588 Binary	$2960

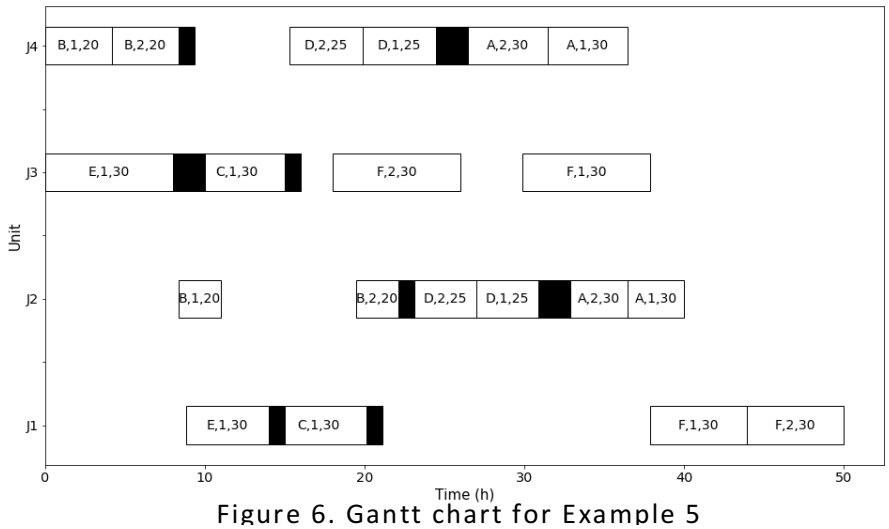

Figure 6. Gantt chart for Example 5

Note that similarly to example 3, this problem has multiple solutions. We have proved that both solutions are valid by fixing the sequence and rerunning the code in GAMS, and still obtaining the same objective function. This example demonstrates the ability to transform this generic global sequence model to account for local sequence cost. Since the majority of the model remains unchanged, we believe that it will show similar performances as example 3 and 4 have.

5. Conclusions and Future Work

5.1 Conclusions

The simultaneous optimization of batching and scheduling decisions formulation is implemented in this project. The challenges in formulating the simultaneous optimization problem as a MILP is addressed using general disjunctive programming formulations. To model batching decisions, a convex hull formulation is adopted to disaggregate each product i into potential batches B_{il}, then further into unit assignments $\bar{B}_{ilj}$, using corresponding binary variables Z_{il} and X_{ilj}.

When modeling scheduling decisions, assignment and sequencing variables and constraints only become active if the corresponding batches are selected. Processing times are subsequently modeled as a function of selection and batch-size variables. To model sequencing of different batches, we implemented both global-sequence and local-sequence.

Through examples 1 and 2, it is clear that this formulation can potentially lead to solutions that are better than the ones obtained by conventional sequential approaches. With simultaneous batching and scheduling, we can make smarter batching decisions by taking advantage of slacks in the overall scheduling model. However, without any further enhancement, the model is found to be too slow to solve when applied to larger problems such as in examples 3 and 4.

To solve larger instances, we implemented two classes of tightening constraints. The first class, A, aims to pre-compute feasibility sets and then set corresponding infeasible variables to 0. The second class, B, aims to pre-compute scenarios to derive corresponding time windows, then to add the inequalities back to the model. We have found that in both examples 3 and 4, class B works better. This can be explained by the fact that class A derives infeasibilities based on simple manipulations of input data, while class B derives inequalities based on complex data manipulations.

5.2 Future Work

One of the biggest challenges this model faces is the computational performance. Since the model has to account for the maximum number of potential batches when creating variables, the size and degrees of freedom become exceedingly large. In reality, a process will rarely make use of all the batches, and as a modeler, we have to consider the worst in order to guarantee optimality.

A potential direction of improving its computational performance could come from the model's inherent structure: demand → batch-selecting binary → unit assignment binary→ sequencing selection binaries. In other words, decisions flow from the higher (selection) level to the lower (sequencing) level. This structure can be exploited in decomposition methods, where the original problem is decomposed into several sub-problems that can be solved in parallel.

Finally, all the examples we formulated have restrictions on due time, which are allowed to be exceeded (Dessouky *et al*. 2009). The formulation can be more complex and universal if we have a penalty for a violation on due dates.

References

Birewar D. B., Grossmann I. E. Simultaneous production planning and scheduling in multiproduct batch plants *Ind Eng Chem Res* **29** 570-580 (1990).

Brooke A., D. Kendrick and A. Meeraus, *GAMS: A User's Guide*. Scientific Press, Palo Alto, Calif. (1988).

Castro P. M., Grossmann I. E. New continuous-time milp model for the short-term scheduling of multistage batch plants *Ind Eng Chem Res* **44** 9175-9190 (2005).

Castro P. M., Grossmann I. E., Novais A. Q. Two new continuous-time models for the scheduling of multi-stage batch plants with sequence dependent changeovers. Ind Eng Chem Res **45** 6210-6626 (2006).

Castro P. M., Erdirik-Dogan M., Grossmann I. E. Simultaneous batching and scheduling of single stage batch plant with parallel units. *AIChe Journal* **54** (2007).

Dessouky M., Kijowski B., Verma S. Simultaneous batching and scheduling for chemical processing with earliness and tardiness penalties. **8** *Production and Operation Management* (2009).

GAMS Development Corporation. General Algebraic Modeling System (GAMS) Release 24.2.1. Washington, DC, USA, (2013).

Gupta A., Karimi I. A. An improved milp formulation for scheduling multi-product multi-stage batch plants. *Ind Eng Chem Res* **42** 2365-2380 (2003).

Gurobi Optimization, Inc. Gurobi optimizer reference manual. (2016).

Harjunkoski I., Grossmann I. E. Decomposition techniques for multi-stage scheduling problems using mixed-integer and constraint programming method. *Comput Chem Eng* **25** 1533-1552 (2002).

Harijunkoski I., Maravelias C., Bongers P., Castro M., Engell S., Grossmann i. E., Hooker J.,... Mendez C. Scope for industrial applications of production scheduling models and solution methods. *Comput Chem Eng* **62** 161-193 (2014)

Hart, W. E., Jean-Paul W., and David L. W. Pyomo: modeling and solving mathematical programs in Python. *Mathematical Programming Computation 3,* **3** 219-260 (2011).

Hui C. H., Gupta A. A novel milp formulation for short-term scheduling of multistage multiproduct batch plants. *Comput Chem Eng* **24** 1611-1617 (2000a).

Hui C. H., Gupta, A. A novel milp formulation for short-term scheduling of multistage multiproduct batch plants with sequence-dependent constraints. *Comput Chem Eng* **24** 2075-2717 (2000b).

IBM, Tech. Rep. IBM ILOG CPLEX user's manual. (2015).

Janak SL, Lin X, Floudas CA. Enhanced continuous-time unit-specific event-based formulation for short-term scheduling of multipurpose batch processes: resource constraints and mixed storage policies. *Ind Eng Chem Res* **43** 2516-2533 (2004).

Kindili E., Pantelides C. C., Sargent R. A general algorithm for short-term scheduling batch operations. i. milp formulation. *Comput Chem Eng* **17** 211-227 (1993)

Maravelias C. T. A decomposition framework for scheduling of single- and multi-stage process. *Comput Chem Eng* **30** 407-420 (2006).

Maravelias C. T., Sung C. Integration of production planning and scheduling: overview, challenges and opportunities. *Comput Chem Eng* **12** 1919-1930 (2009).

Méndez, C. A., Cerda, J. Grossmann, I. E. State-of-the-art review of optimization methods for short-term scheduling of batch processes. *Comput Chem Eng* **30** 913−946 (2006).

Padberg M., Rinaldi G. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problem. *SIAM* **33** 60-100 (1991).

Pinedo M. *Scheduling: theory, algorithms and systems*; New York: Prentice Hall (2002).

Pinto J. M., Grossmann I. E. A continuous time mixed integer linear programming model for short term scheduling of multi-stage batch plants. *Ind Eng Chem Res* **34** 3037-3051 (1995).

Pinto J. M., Grossmann I. E. A novel milp formulation for short-term scheduling of batch plants with preordering constraints. *Ind Eng Chem Res* **35** 338-342 (1996).

Pinto J. M., Grossmann I. E. Assignment and sequencing models for scheduling of process system. *Ind Eng Chem Res* **81** 433-466 (1998).

Sundaramoorthy A., Maravelias C. T. Simultaneous batching and scheduling in multistage multiproduct process. *Ind Eng Chem Res* **47** 1546-1555 (2008).