A Rolling Horizon Approach for Scheduling of Multiproduct Batch Production and Maintenance Using Generalized Disjunctive Programming Models

Ouyang Wu[a], Giancarlo Dalle Ave[b,c], Iiro Harjunkoski[b], Lars Imsland[a,*]

[a] *Department of Engineering Cybernetics, Norwegian University of Science and Technology, 7491 Trondheim, Norway*
[b] *ABB Power Grids Research, Mannheim, Germany*
[c] *Process Dynamics and Operations Group, Department of Biochemical & Chemical Engineering, TU Dortmund, 44221 Dortmund, Germany*

Abstract

This paper considers joint production and maintenance scheduling of a multiproduct batch chemical manufacturing plant. A Generalized Disjunctive Programming-based formulation is proposed for the scheduling problem, integrating additional features inspired by an industrial case study, namely sequence-dependent degradation and limited final product storage tanks. To properly consider maintenance in the context of production scheduling, a long time horizon needs to be considered, resulting in high computational complexity. To this end, a new rolling horizon approach is proposed to find good quality solutions to these scheduling problems with their extended horizons in order to better consider the trade-offs between production and maintenance scheduling. The proposed scheduling formulation and rolling horizon approach are tested using an industrial case study and analyzed with a variety of tuning parameter sets.

Keywords: maintenance scheduling, sequence-dependent degradation, performance decay, rolling horizon method, precedence models

1. Introduction

Batch scheduling is an important topic in the process industries as it is applicable to a wide variety of industrial production plants. These batch production plants often produce multiple products requiring several steps. Furthermore, it is often desirable to combine additional features into the scheduling problems including (but not limited to), storage constraints, equipment condition, and maintenance concerns. Due to the many different complex tradeoffs between these concerns, it is desirable to combine them into a single scheduling model as

*Corresponding author
Email address: lars.imsland@ntnu.no (Lars Imsland)

effective scheduling is a requirement for efficient plant operations in the process industries (Harjunkoski, 2016).

Many types of optimal batch scheduling models exist in literature today. A comprehensive review of these types of models can be found in Méndez et al. (2006). A common method of modeling scheduling problems (and optimization problems in general) is via Generalized Disjunctive Programming (GDP). GDP has been applied to many relevant process industry problems including strip packing (Trespalacios and Grossmann, 2017) and process design (Chen and Grossmann, 2019a). A review of modeling paradigms using GDP can be found in Chen and Grossmann (2019b). A specific example in scheduling comes from Castro and Grossmann (2012) who used GDP to derive a set of generic continuous-time scheduling models and compared them based on computational efficiency.

A key feature of scheduling problems is the type of material transfer and storage policies presented in the plant. In many facilities, the transfer of materials is highly constrained because it requires shared resources, such as the presence of finite storage units. Material and storage policies in batch scheduling plants can be modeled with a variety of formulations. A precedence-based model for storage was proposed by (Sundaramoorthy and Maravelias, 2008). Their problem featured limited storage, both in terms of size and number of vessels, as well as constraints on the time that product could spend in storage. Kilic et al. (2011) used a state-task network model to explicitly model storage vessels. A make-and-pack process with finite intermediate buffer was proposed by Klanke et al. (2020). They used a combined immediate precedence-based model and discrete-time model and solved the problem using an iterative order insertion heuristic.

Another topic of interest to scheduling is to integrate equipment condition and maintenance decisions into production scheduling as resources are shared by maintenance and production processes (e.g. processing units). A continuous-time model for maintenance scheduling in a gas power plant was presented by Castro et al. (2014). Their formulation was derived using GDP and featured scheduling of the turbines as well as of maintenance teams. Turbines were also studied by Xenos et al. (2016), but in the context of a compressor network for a chemical plant. Their discrete-time formulation considered two different washing procedures in order to reduce additional fouling-based energy costs. Combined maintenance and production scheduling to avoid a decrease in production was studied by Vieira et al. (2017). They considered a bio-pharmaceutical process under performance decay, where maintenance must occur before a maximum number of batches is reached. They formulated the problem using a continuous-time resource-task network model and tested the problem with different objectives including maximizing total profit, and minimizing number of maintenance activities. Maintenance and production scheduling of a steel plant was studied by Biondi et al. (2017). They presented a multi-time scale discrete-time model coupled with a remaining useful lifetime model to keep track of the assets' life cycles. A steel plant was also studied by Dalle Ave et al. (2019b) but on a shorter time scale. Their model featured operating mode-dependent degradation with the goal to minimize total operating and maintenance costs.

Due to the complexity of many industrial processes, additional solution algorithms or model reformulations, are needed to solve the mixed-integer scheduling problems in industrially relevant time-frames. One reason that these problems are often difficult to solve is due to symmetry or equivalent solutions. By adding symmetry-breaking constraints, one can reduce the size of the search space. An example of a work that considers symmetry breaking constraints is Trespalacios and Grossmann (2017). In this work, symmetry breaking constraints are applied to regions that can be represented with the selection of two different disjunctive terms. An example from scheduling comes from the work of Baumann and Trautmann (2014), who pointed out that symmetry in short-term scheduling often arises due to identical batches. They removed said symmetry by imposing arbitrary sequences for each group of identical batches, leading to better computational efficiency. Decomposition approaches on the other hand, reduce the size of the original scheduling problem by breaking into smaller pieces which can be solved separately. A two-step iterative method was proposed by Aguirre et al. (2012) to solve a semiconductor manufacturing problem. In their approach, a general scheduling is performed at the first stage, with a more detailed model being used in the second. The solution is then iteratively improved using a reduced MILP at each step. Mean value cross decomposition was applied to a pulp-and-paper as well as a steel case study by Hadera et al. (2019). The method is not guaranteed to converge, however experimentally it was shown to produce high quality solutions quickly.

Time-based decomposition are popular decomposition schemes and they often manifest themselves as rolling horizon (RH) algorithms. Rolling horizon approaches are iterative methods in which a subset of the horizon is studied in detail at every iteration while the rest is represented in an aggregate form. Decisions are fixed in one iteration and the detailed portion of the time horizon is slid for the next iteration until the whole horizon has been scheduled in detail. A classic example of a rolling horizon being applied to scheduling problems is through the work of Dimitriadis et al. (1997). The aggregate model in this case was formulated using weighted sums of corresponding exact variables. Rolling horizon algorithms are often also used to bridge models of different time scales. Li and Ierapetritou (2010) use a rolling horizon approach to bridge the gap between planning and scheduling, where targets for the scheduling horizon come from the higher level planning model. Another work comes from Dalle Ave et al. (2019a). In this work near- and short-term electricity-related concerns were combined into a single scheduling model. The discrete-time rolling horizon algorithm used a non-uniform grid to distinguish between the detailed and aggregate model.

This work builds upon previous works by the same authors. The problem in question is production and maintenance scheduling of a multiproduct chemical manufacturing plant with sequence-dependent degradation (Wu et al., 2020a) and limited storage tanks (Wu et al., 2020b). This work summarizes the previous works and proposes a GDP-based formulation for some of the earlier presented features. Furthermore, the case study is extended to industrially-relevant time horizons in order to better understand the complex tradeoffs between production and maintenance scheduling. To deal with these extended horizons, a novel rolling horizon algorithm is proposed and analyzed for a wide variety of parameter sets.

2. Problem Description

This work considers an industrial scheduling case study in a chemical batch plant. The process in this case study consists of multiple stages with parallel units in some stages. The topology of the batch process is presented in Fig. 1. For each batch run, a set of raw materials are charged and mixed in a monomer make-up vessel referred to as unit U1, which is the first production stage. The monomer fluid is then mixed with oil and transferred through homogenizers to form a monomer emulsion, which is subsequently fed to one of two parallel batch reactors (denoted as units U2 and U3) in the second stage. Emulsion polymerization in the batch reactors turns the monomer emulsion to polymer products. The products are then transferred and stored in temporary tanks for quality checks, which comprises the third production stage. The plant produces multiple grades of product with fixed batch sizes by using various different recipes in the production stages (Stage 1 and Stage 2) .

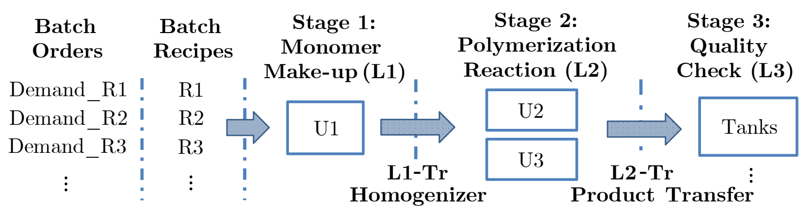

Figure 1: Process topology of case study

One of the types of degradation in the case study is fouling of the batch reactors. The fouling is formed by polymer residuals inside the reactors and the associated heat exchangers. Due to the fouling it is more difficult to cool the reactor, which results in an increased batch time. As the fouling continues to build-up, the pressure drop across the heat exchanger increases. Due to safety concerns, the reactors must be shut down and cleaned before a complete blockage occurs. An upper limit of the pressure drop defines the latest point at which reactor maintenance must occur.

The evolution of fouling in the batch reactors, and the associated maintenance actions, are some of the key bottlenecks in the case study and therefore need to be explicitly considered in the production scheduling problem. A key performance indicator (KPI) of fouling is taken to measure the level of fouling at the start of each batch run in the reactors (Wu et al., 2019a). The sequence of batch recipes that determine the conditions of polymerization is the main factor that affects the rate of fouling evolution from an operational view. A model which describes the sequence-dependent fouling evolution between batches that depends on the batch recipe and the fouling KPI after the previous batch in the same reactor was developed in Wu et al. (2020a).

The tanks in the third stage store multiple batches of products in integer quantities. A storage tank can only be occupied by product of one recipe at a time. A quality check is performed in a tank before the products inside are transferred for further distribution. Since one tank can only store product that is associated to a recipe, the number of tanks will put constraints on the recipes currently in production, and therefore the sequences of the multiproduct batch runs have to meet the corresponding storage constraints.

The batch scheduling model in this paper considers the main features mentioned above to generate practical and accurate solutions for the case study. Since the batching decisions are assumed to have been made elsewhere, the considered decisions in this scheduling model are: (1) assignment of various batches to one of the units in each stage; (2) sequencing of batches considering fouling and storage constraints; (3) scheduling of maintenance operations throughout the course of production.

Nomenclature

Indices

p, p', p''	Production batch
m	Unit
s	Stage
r	Batch recipe
g, g'	Batch group
(r, g)	Recipe-specific group assigned to a storage tank

Sets

P	Set of production orders
P_r	Subset of P using batch recipe r
P_{sub}^n	Subset of P for scheduling in the nth time horizon[a]
P_{new}	Subset of P that are not scheduled during RH iterations[a]
P_{fix}	Subset of P that are sequentially fixed during RH iterations[a]

[a]See Algorithm 1

Sets

P^n_{fix}	Subset of P that are sequentially fixed in nth time horizon[a]
S	Stages of production units
S_{st}	Stage of storage tanks
S_f	Subset of S containing units in M_f
R	Batch recipes
M	Units of production
M_{st}	Storage tanks
M_f	Subset of M affected by degradation
M_s	Subset of M in stage s
G_r	Set of groups consisting of r-recipe batches

Parameters

tp_{pm}	Fixed processing time of batch run p at unit m
ts_m	Time of availability in unit m
tc	Time cost for maintenance in unit $m \in M_f$
tr_{ps}	Time for material transfer of batch run p in stage s to the next stage
t_{qc}	Time for quality check and product transfer in storage tanks
FI_m	Initial KPI value of fouling in unit $m \in M_f$
Fc	KPI value after cleaning
F_{max}	Threshold of fouling KPI in in unit $m \in M_f$
Af_{pm}, Bf_{pm}	Recipe-specific parameters of fouling model for batch run p in unit $m \in M_f$
A_{pm}	Proportional parameter for extra processing time of batch run p due to fouling in unit $m \in M_f$
λ	Weight parameter
T_{HP}	Length of time horizon[a]
C_{init}	Initial condition and availability of units[a]

Continuous variables

Ts_{ps}	Start time of batch run p in stage s
$Ts_{(r,g)}$	Start time of group (r, g) in a storage tank
$Te_{(r,g)}$	End time of group (r, g) in a storage tank
Te_{ps}	End time of batch run p in stage s
Tp_{ps}	Processing time of batch run p in stage s
Tf_p	Extra processing time of batch run p in unit $m \in M_f$ due to degradation
Tc_p	Time for maintenance right before batch p in unit $m \in M_f$
f_{pm}	Fouling KPI of unit $m \in M_f$ at the beginning of batch p
fe_m	Fouling KPI of unit $m \in M_f$ after finishing all batch runs
MS	Makespan

Boolean variables

$XG_{pp'm}$	Sequencing decision for batch p preceding batch p' in unit $m \in M \setminus M_f$

[a]See Algorithm 1

Boolean variables

$XG^1_{pp'm}$	Assignment decision for at most one of batches p and p' in unit $m \in M \setminus M_f$
Y_{pm}	Assignment decision of batch p in unit $m \in M$
$X^g_{pp's}$	Sequencing decision for batch p preceding batch p' in stage $s \in S \setminus S_f$
$X_{pp's}$	Sequencing decision for batch p immediately preceding batch p' in a certain unit of stage $s \in S_f$
X^F_{pm}	Sequencing decision of batch p in the first place of unit $m \in M_f$
X^L_{pm}	Sequencing decision of batch p in the last place of unit $m \in M_f$
$X^{f1}_{pp'm}$	Sequencing decision for batch p immediately preceding batch p' in unit $m \in M_f$
X^{f2}_{pm}	Sequencing decision for maintenance immediately preceding batch p in unit $m \in M_f$
X^{f3}_{pm}	Sequencing decision for batch p in the first place of unit $m \in M_f$
X^{f4}_{pm}	Assignment decision of batch p in a unit other than $m \in M_f$
Z_p	Decision of maintenance immediate before batch i in a certain unit $m \in M_f$
$X^{im1}_{pp's}$	Sequencing decision for batch p immediately preceding batch p' in the same unit in stage $s \in S_f$ without maintenance in between
$X^{im2}_{pp's}$	Sequencing decision for batch p immediately preceding batch p' in the same unit in stage $s \in S_f$ with maintenance in between
XF^{im3}_{pm}	Sequencing decision for batch p in the first place of unit $m \in M_f$ without maintenance occurring before
XF^{im4}_{pm}	Sequencing decision for batch p in the first place of unit $m \in M_f$ with maintenance occurring before
XL^{im3}_{pm}	Sequencing decision for batch p in the last place of unit $m \in M_f$ without maintenance occurring before
XL^{im4}_{pm}	Sequencing decision for batch p in the last place of unit $m \in M_f$ with maintenance occurring before
X^{L0}_m	No assignment of batches in unit $m \in M_f$

3. GDP-based Formulation

This section presents a GDP-based formulation for the scheduling problem that is described in Section 2. The indices, sets, parameters and variables in the formulation are summarized in the Nomenclature section.

In the scheduling model, a set of batches are assigned to one of the parallel units in each production stage and are sequenced as a combination of batch recipes in each unit; these batches generate multiple grades of products or semi-finished products to be transferred to the next stage. The sequences of these batch runs in the production stages can be described using two types of precedence concepts. In Méndez and Cerdá (2003); Castro and Grossmann (2012), a so-called general precedence formulation which considers Boolean variables $XG_{pp'm}$ and $XG^1_{pp'm}$, such that $XG_{pp'm}$ is true if batch p and p' are both assigned to unit m and batch p is sequenced before batch p'; $XG^1_{pp'm}$ is true if at least one of batch p and batch p' is not assigned to unit m. These Boolean variables represent a set of individual terms of disjunctions for sequencing any two batches in a stage. A set of disjunctive constraints describe how the timing of two batches in a stage are connected using the exclusive OR operator $\veebar$, which yields the following GDP-based constraints:

$$\begin{bmatrix} XG_{p'pm} \\ Te_{p's} + tr_{p's} \leqslant Ts_{ps} \end{bmatrix} \veebar \begin{bmatrix} XG_{pp'm} \\ Te_{ps} + tr_{ps} \leqslant Ts_{p's} \end{bmatrix} \veebar \; XG^1_{p'pm} \;, \qquad \forall\, p, p' \in P : p' < p,\; m \in M_s,\; s \in S \setminus S_f \tag{1}$$

where the disjunctive constraint $XG^1_{p'pm}$ is empty because no timing constraints exist for two batches assigned to different parallel units; Ts_{ps} and Te_{ps} denote the start and end time of batch p in stage s, and $tr_{p's}$ is the transfer time of products or semi-finished products to the next stage. $XG_{pp'm}$ is further represented using types of Boolean variables $X^g_{pp's}$ and Y_{pm} as Eqs. (2) and (3) show, where $X^g_{pp's}$ is true if batch p starts earlier than batch p' in stage s, and Y_{pm} is true if batch p is assigned to unit m. Equation (4) presents the constraints that batch p is assigned to one of the units in stage s. This type of precedence model is used for the production stages that have no degradation issues ($s \in S \setminus S_f$), which has the advantage of fewer variables compared to other formulations.

$$XG_{pp'm} \iff X^g_{pp's} \wedge Y_{pm} \wedge Y_{p'm},\; \forall\, p \neq p' \in P,\; s \in S \setminus S_f,\; m \in M_s \tag{2}$$

$$XG_{p'pm} \iff \neg X^g_{pp's} \wedge Y_{pm} \wedge Y_{p'm},\; \forall\, p \neq p' \in P,\; s \in S \setminus S_f,\; m \in M_s \tag{3}$$

$$\sum_{m \in M_s} Y_{pm} = 1,\; \forall\, p \in P,\; s \in S \tag{4}$$

The second type of precedence formulations for batch sequencing uses the concept of immediate precedence (Gupta and Karimi, 2003). This formulation considers Boolean variables $X_{pp's}$ to be true if batch p and batch p' are assigned to the same unit and batch p' is immediately sequenced after batch p'. In regards to to disjunctions for scheduling batch p in stage s, two other types of disjunctive terms are represented using Boolean variables X^F_{pm} and X^L_{pm}; X^F_{pm} is true if batch p is in the first place of the sequence in unit m, while X^L_{pm} is true if batch p is in the last place of the sequence in unit m. The disjunctions and the corresponding disjunctive constraints for the timing of batches are

$$\underset{p' \neq p \in P}{\veebar} \begin{bmatrix} X_{p'ps} \\ Te_{p's} + tr_{p's} \leqslant Ts_{ps} \end{bmatrix} \underset{m \in M_s}{\veebar} X^F_{pm},\; \forall\, p \in P,\; s \in S_f \tag{5}$$

$$\underset{p' \neq p \in P}{\veebar} \begin{bmatrix} X_{pp's} \\ Te_{ps} + tr_{ps} \leqslant Ts_{p's} \end{bmatrix} \underset{m \in M_s}{\veebar} X^L_{pm},\; \forall\, p \in P,\; s \in S_f \tag{6}$$

where terms X^F_{pm} and X^L_{pm} present no disjunctive constraints for the timing of batch p in unit m given other existing parallel units. Boolean variables $X_{pp's}$ and Y_{pm} are associated in Eq. (7) to put constraints on the assignment of two consecutive batches in one unit.

$$Y_{pm} \leqslant Y_{p'm} + 1 - X_{p'ps} - X_{pp's},\ \forall\ p \neq p' \in P,\ m \in M_s,\ s \in S_f \tag{7}$$

In the stage of batch reactors, sequences of recipe-specific batch runs influence the evolution of fouling in the reactors, and an immediate precedence formulation can be used to represent the fouling evolution from batch to batch according to a specific batch sequence. This formulation considers types of Boolean variables: $X^{f1}_{pp'm}$ is true if batch p and batch p' are assigned to unit m and batch p is immediately sequenced before batch p'; X^{f2}_{pm} is true if a maintenance is carried out right before batch p in unit m; X^{f3}_{pm} is true if batch p is in the first place of the sequence in unit m and no maintenance happens right before batch p in the same unit; X^{f4}_{pm} is true if batch p is not assigned to unit m. The Boolean variables describe individual terms of disjunctions for the fouling evolution in batch p, and the GDP-based constraints that model the disjunctions are:

$$\begin{aligned} \underline{\bigvee}_{p' \neq p \in P} \begin{bmatrix} X^{f1}_{p'pm} \\ f_{pm} = Af_{p'm} \cdot f_{p'm} + Bf_{p'm} \end{bmatrix} \underline{\vee} \begin{bmatrix} X^{f2}_{pm} \\ f_{pm} = F_c \end{bmatrix} \\ \underline{\vee} \begin{bmatrix} X^{f3}_{pm} \\ f_{pm} = FI_m \end{bmatrix} \underline{\vee} \begin{bmatrix} X^{f4}_{pm} \\ f_{pm} = 0 \end{bmatrix},\ \forall\ p \in P,\ m \in M_f \end{aligned} \tag{8}$$

Here, f_{pm} denotes the fouling KPI of unit $m \in M_f$ at the beginning of batch p, and M_f refers to the set of batch reactors; f_{pm} is computed from the fouling indicator of the immediately preceding batch $f_{p'm}$ using a recipe-specifc degradation model $\{Af_{p'm}, Bf_{p'm}\}$ when $X^{f1}_{pp'm}$ is true (Wu et al., 2019b); f_{pm} is reverted back to F_c when the unit is cleaned right before batch p and $X^{f2}_{pp'm}$ is true; f_{pm} equals to the initial fouling condition FI_m when $X^{f3}_{pp'm}$ is true; f_{pm} equals zero when batch p is not assigned to unit m. An allowed threshold of fouling F_{max} is set as the upper-bound of f_{pm}

$$f_{pm} \leqslant F_{max},\ \forall\ p \in P,\ m \in M_f$$

The Boolean variables in Eq. (8) are further represented using the Boolean variables from Eqs. (2) and (5) and yield

$$X^{f1}_{pp'm} \iff \neg Z_p \wedge X_{pp's} \wedge Y_{p'm} \wedge Y_{pm},\ \forall\ p \neq p' \in P,\ s \in S_f,\ m \in M_s \tag{9}$$

$$X^{f2}_{pm} \iff Z_p \wedge Y_{pm},\ \forall\ p \in P,\ m \in M_f \tag{10}$$

$$X^{f3}_{pm} \iff \neg Z_p \wedge X^F_{pm},\ \forall\ p \in P,\ m \in M_f \tag{11}$$

$$X^{f4}_{pm} \iff \neg Y_{pm},\ \forall\ p \in P,\ m \in M_f \tag{12}$$

where, Boolean variable Z_p is true if batch p is assigned to a unit and a maintenance is immediately sequenced before batch p in the same unit.

To consider maintenance in addition to the sequences of batches in the reactors, an immediate precedence formulation extends the GDP-based logic constraints in Eqs. (5) and (6) to

$$\underset{p'\neq p\in P}{\veebar}\begin{bmatrix} X^{im1}_{p'ps} \\ Te_{p's}+tr_{p's}\leqslant Ts_{ps}\end{bmatrix}\underset{p'\neq p\in P}{\veebar}\begin{bmatrix} X^{im2}_{p'ps} \\ Te_{p's}+tr_{p's}+tc\leqslant Ts_{ps}\end{bmatrix}$$
$$\underset{m\in M_s}{\veebar} XF^{im3}_{pm}\underset{m\in M_s}{\veebar} XF^{im4}_{pm},\qquad \forall\ p\in P,\ s\in S_f \tag{13}$$
$$\underset{p'\neq p\in P}{\veebar}\begin{bmatrix} X^{im1}_{pp's} \\ Te_{ps}+tr_{ps}\leqslant Ts_{p's}\end{bmatrix}\underset{p'\neq p\in P}{\veebar}\begin{bmatrix} X^{im2}_{pp's} \\ Te_{ps}+tr_{ps}+tc\leqslant Ts_{p's}\end{bmatrix}$$
$$\underset{m\in M_s}{\veebar} XL^{im3}_{pm}\underset{m\in M_s}{\veebar} XL^{im4}_{pm},\qquad \forall\ p\in P,\ s\in S_f \tag{14}$$

where Boolean variable $X^{im1}_{p'ps}$ is true if batch p' is immediately sequenced before batch p in the same unit and no maintenance is carried out between them, while $X^{im2}_{p'ps}$ is true if batch p' is immediately sequenced before batch p in the same unit and a maintenance task is performed between them. The Boolean variable XF^{im3}_{pm} is true if batch p is sequenced in the first position of unit m without maintenance occurring before; Boolean variable XF^{im4}_{pm} is true if batch p is sequenced in the first place of unit m and a maintenance operation is scheduled right before batch p. Similarly, the Boolean variables XL^{im3}_{pm} and XL^{im4}_{pm} are defined that XL^{im3}_{pm} is true if batch p is sequenced in the last place of unit m without maintenance in advance; XL^{im4}_{pm} is true if batch p is sequenced in the last place of unit m with maintenance performed right before batch p. These Boolean variables are equivalent to the logic expressions using the Boolean variables from Eqs. (5) and (6) and Z_p, as Eqs. (15) to (20) show.

$$X^{im1}_{p'ps} \iff \neg Z_p\wedge X_{p'ps},\ \forall\ p\neq p'\in P,\ s\in S_f \tag{15}$$
$$X^{im2}_{p'ps} \iff Z_p\wedge X_{p'ps},\ \forall\ p\neq p'\in P,\ s\in S_f \tag{16}$$
$$XF^{im3}_{pm} \iff \neg Z_p\wedge X^{F}_{pm},\ \forall\ p\in P,\ m\in M_f \tag{17}$$
$$XF^{im4}_{pm} \iff Z_p\wedge X^{F}_{pm},\ \forall\ p\in P,\ m\in M_f \tag{18}$$
$$XL^{im3}_{pm} \iff \neg Z_p\wedge X^{L}_{pm},\ \forall\ p\in P,\ m\in M_f \tag{19}$$
$$XL^{im4}_{pm} \iff Z_p\wedge X^{L}_{pm},\ \forall\ p\in P,\ m\in M_f \tag{20}$$

The disjunctions for the assignment of batch p in stage s are modeled as the GDP-based logic constraints

$$\underset{\substack{m\in M_s,\\ s\in S_f}}{\veebar}\begin{bmatrix} Y_{pm} \\ Tp_{ps}=tp_{pm}+Tf_p \\ Ts_{ps}\geqslant ts_m+Tc_p\end{bmatrix}\underset{\substack{m\in M_s,\\ s\in S\setminus S_f}}{\veebar}\begin{bmatrix} Y_{pm} \\ Tp_{ps}=tp_{pm} \\ Ts_{ps}\geqslant ts_m\end{bmatrix},\ \forall\ p\in P \tag{21}$$

where Tp_{ps} denotes the processing time of batch p in stage s and it is determined by Boolean variable Y_{pm}; tp_{pm} is the unit-specific production time of batch p in unit m, and Tf_p is the extra processing time because of the fouling in the batch reactors, which is estimated to be proportional to fouling KPIs as Eq. (22) shows. The start and end time of batch p are constrained according to Tp_{ps} in Eq. (23). The start time Ts_{ps} of batch p is always later than the time ts_m that unit m becomes available when Y_{pm} is true, and Tc_p is the cleaning time before batch p as Eq. (25) shows. These disjunctive constraints can be reformulated directly as MILP constraints as presented in Eqs. (24) and (26).

$$Tf_p = \sum_{m \in M_f} A_{pm} \cdot f_{pm},\ \forall\ p \in P \tag{22}$$

$$Ts_{ps} + Tp_{ps} = Te_{ps},\ \forall\ p \in P,\ s \in S \tag{23}$$

$$Tp_{ps} = \begin{cases} \sum_{m \in M_s} tp_{pm} \cdot Y_{pm} + Tf_p, & \forall\ p \in P,\ s \in S_f \\ \sum_{m \in M_s} tp_{pm} \cdot Y_{pm}, & \forall\ p \in P,\ s \in S \setminus S_f \end{cases} \tag{24}$$

$$Tc_p = tc \cdot Z_p,\ \forall\ p \in P \tag{25}$$

$$Ts_{ps} \geqslant \begin{cases} \sum_{m \in M_s} ts_m Y_{pm} + Tc_p, & \forall\ p \in P,\ s \in S_f \\ \sum_{m \in M_s} ts_m Y_{pm}, & \forall\ p \in P,\ s \in S \setminus S_f \end{cases} \tag{26}$$

In the final stage, the storage and quality check stage, products generated from batch runs in the production stages are transferred to one of the storage tanks, and a quality check is performed once the tank is full of products from the same recipe. A concept of groups is introduced to represent batches that generate the same grade of product to be stored in a tank together for the final quality check (Wu et al., 2020b). Each group has index (r, g) to denote a type of recipe r and an index number of groups g that belong to recipe r. These groups are associated with batches through Boolean variables $Y_{p(r,g)}$, $Y^f_{p(r,g)}$ and $Y^l_{p(r,g)}$. $Y_{p(r,g)}$ is true if batch p is assigned to group (r, g); $Y^f_{p(r,g)}$ is true if batch p is in the first place of the batch sequence in group (r, g), and $Y^l_{p(r,g)}$ is true if batch p is in the last place of the batch sequence in group (r, g). In the considered scenario, all batches generate the same quantity of product. Therefore, the same number of batches of any product type are needed to fill up the tanks before the final quality check and transfer of products out of storage can be performed. The number of batches in a group is fixed and corresponds to the tank volume. The number of groups is predefined giving a predefined batch set in the scheduling model, and $Y_{p(r,g)}$, $Y^f_{p(r,g)}$ and $Y^l_{p(r,g)}$ are predefined to assign corresponding batches in the groups. Product transfer of batches from the production stages to the storage stage are therefore represented as assignment and sequencing of groups in the tanks. Similar to the sequencing constraints of batches in Eq. (1) a general precedence formulation considers disjunctions for sequencing of any two of the groups, and Boolean variables $XG_{(r',g')(r,g)m}$ and $XG1_{(r',g')(r,g)m}$ describe individual terms in each disjunction. $XG_{(r',g')(r,g)m}$ is true when group (r', g') is sequenced before group (r, g) in tank $m \in M_{st}$;

$XG1_{(r',g')(r,g)m}$ is true if any one of the two groups is not assigned in tank m. The GDP-based constraints for these disjunctions are

$$\begin{bmatrix} XG_{(r',g')(r,g)m} \\ Te_{(r',g')} \leqslant Ts_{(r,g)} \end{bmatrix} \veebar \begin{bmatrix} XG_{(r,g)(r',g')m} \\ Te_{(r,g)} \leqslant Ts_{(r',g')} \end{bmatrix} \veebar \; XG1_{(r',g')(r,g)}$$
$$\forall\, r, r' \in R,\; g \in G_r,\; g' \in G_{r'} : g' < g,\; m \in M_{st} \tag{27}$$

where $Ts_{(r,g)}$ and $Te_{(r,g)}$ are the start and end times of group (r, g) following the above disjunctive timing constraints of the groups. The timing of groups are also associated with the timing of batches in the groups. These are determined with the known first and last place batches within the group,

$$Te_{(r,g)} \geqslant \sum_{p \in P_r} (Te_{p(s-1)} + t_{qc}) \cdot Y^{l}_{p(r,g)},\; \forall\, r \in R,\; g \in G_r, s \in S_{st} \tag{28}$$

$$Ts_{(r,g)} = \sum_{p \in P_r} Te_{p(s-1)} \cdot Y^{f}_{p(r,g)},\; \forall\, r \in R,\; g \in G_r, s \in S_{st} \tag{29}$$

where $Te_{p(s-1)}$ is the end time of batch p in the upsteam stage of the tanks; t_{qc} denotes the time of quality check and product transfer from tanks to product transport or other storage places.

The scheduling optimization formulation considers two terms in the objective function. The first is the Makespan (MS) of the schedule,

$$MS \geqslant Te_{(r,g)},\; \forall\, r \in R,\; g \in G_r \tag{30}$$

The other candidate is the final fouling KPIs of the batch reactors. The final fouling KPIs are obtained using GDP-based constraints, as

$$\veebar_{p \in P} \begin{bmatrix} X^{L}_{pm} \\ fe_m = Af_{pm} \cdot f_{pm} + Bf_{pm} \end{bmatrix} \veebar \begin{bmatrix} X^{L0}_{m} \\ fe_m = FI_m \end{bmatrix},\; \forall\, m \in M_f \tag{31}$$

In the sequence of each batch reactor, the disjunctive constraints in Eq. (31) calculate the value of fouling KPI after the last batch in the sequence, or when X^{L0}_m is true if no batches are scheduled in unit m so that the final fouling KPI equals the initial fouling KPI of unit m. Equation (31) prevents a cleaning task from being scheduled at the end of the horizon for two main reasons. The first is due to the model itself; one of the goals of the model is to minimize the makespan of the schedule (more information about the objective will be given alongside the case studies in Section 5). Unless a much larger emphasis is placed on the fouling objective over the makespan, an "unnecessary" maintenance task will never be added to the end of the horizon. Additionally, in actual operation the scheduler will be operating in a true moving-horizon fashion, instead of a shrinking horizon manner as studied in this paper. If a maintenance task is needed at the end of the current horizon (but is not present in the current version of the schedule) and a new order comes in, the algorithm would schedule the maintenance task before production of the new task in the new horizon.

To solve the GDP-based models, one approach is to reformulate the logic programming models as MIP models and to solve using conventional MIP solvers. The common reformulation methods include the big-M method and the convex-hull reformulation. Castro and Grossmann (2012) present examples of applying reformulation to GDP-based formulations for batch scheduling. Pyomo is a collection of optimization modelling packages in Python that supports disjunctions (Hart et al., 2017). GDP-based models can be solved in Python using Pyomo through the use of automated problem transformations, converting the GDP model to a MIP model.

4. Rolling Horizon Algorithm

To schedule a large number of batches of various product grades, the size of the proposed MILP problem becomes too large to solve to optimality in a reasonable time period using an MILP solver. Instead of solving a single large-size MILP problem, a rolling horizon algorithm decomposes the original scheduling problem into many smaller MILP problems. This is accomplished by grouping scheduling tasks into many smaller time horizons. In each time horizon, a sub-scheduling problem (SubMILP) with a smaller set of batches is formulated and it is solved to optimality within a much shorter period of time than the full-space model. The batches that are scheduled within the current sub-schedule horizon are then fixed in the original scheduling problem, and the algorithm moves on to the next sub-scheduling problem.

Algorithm 1 Rolling horizon algorithm

function RH-MILP(P, C_{init}, T_{HP})
    $\mathrm{Sol_d} \leftarrow$ MasterMILP(P, C_{init})
    $P_{new} \leftarrow P$, $P_{fix} \leftarrow \emptyset$, $C^1_{init} \leftarrow C_{init}$
    $P^1_{sub} \leftarrow$ InitSubMILP(P_{new}, C^1_{init}, T_{HP})
    repeat $n = 2, 3, 4...$
        ($\mathrm{Sol^n}$, $P^n_{fix}, C^{n+1}_{init}$) $\leftarrow$ SolveSubMILP(P^n_{sub}, C^n_{init}, T_{HP})
        $\mathrm{Sol_d} \leftarrow$ FixDiscreteVar($\mathrm{Sol_d}$, P_{fix}, $\mathrm{Sol^n}$, P^n_{fix})
        $P_{fix} \leftarrow P_{fix} \cup P^n_{fix}$, $P_{new} \leftarrow P_{new} \setminus P^n_{fix}$
        $P^{n+1}_{sub} \leftarrow$ InitSubMILP(P_{new}, C^{n+1}_{init}, T_{HP})
    until $P^{n+1}_{sub} = P_{new}$
    $\mathrm{Sol^{n+1}} \leftarrow$ SolveMILP(P^{n+1}_{sub}, C^{n+1}_{init})
    $\mathrm{Sol_d} \leftarrow$ FixDiscreteVar($\mathrm{Sol_d}$, P_{fix}, $\mathrm{Sol^{n+1}}$, P^{n+1}_{sub})
    $\mathrm{Sol_{RH}} \leftarrow$ SolveMasterFixedLP(P, C_{init}, $\mathrm{Sol_d}$)
    return $\mathrm{Sol_{RH}}$
end function

The main RH algorithm is presented in Algorithm 1. The initial condition and availability of units is denoted as C_{init}. A set of batch runs denoted as P are predefined to be scheduled according to the production target. In the RH algorithm, SubMILP is formulated using the proposed model in Section 3 and schedule a set of batch runs in the nth time horizon (defined as P_{sub}^{n}), given C_{init}^{n} which specifies the corresponding initial condition of the sub-problem with superscript n. C_{init}^{n} includes the fouling KPIs FI_{m}^{n} at the beginning of the current time horizon, the exact time that the units become available ts_{m}^{n}, and the status of the tanks (whether tanks are already filled with one of the product grades and how many batches of product are stored in the tanks). The status of the tanks are presented as a group set $(r,g)_{init}^{n}$, such that if group $(r,g) \in (r,g)_{init}^{n}$ was assigned to tank $m \in M_s$ ($Y_{(r,g)m}$ is true) in the previous time horizon, and the start time of the group $Ts_{(r,g)}$ is fixed according to the first batch in group (r,g) that is scheduled in the previous time horizon. P_{sub} is calculated via function InitSubMILP giving the set of the unscheduled batch runs, denoted by P_{new}, C_{init} and the length of horizon periods T_{HP}, which is described in Algorithm 2. Function SolveSubMILP solves $\mathrm{SubMILP^{n}}$ and calculates the set of batch runs P_{fix}^{n} that are scheduled in the nth time horizon. P_{fix} refers to the batch runs that have already been fixed at a previous point in the iterative process. $\mathrm{Sol_{d}}$ refers to the discrete variables of the master problem of the RH algorithm denoted as MasterMILP(P, C_{init}). The variables in $\mathrm{Sol_{d}}$ that relate to batch runs in P_{fix}^{n} or additional batch runs in P_{fix} are assigned with fixed binary values according to $\mathrm{Sol^{n}}$ the solution of $\mathrm{SubMILP^{n}}$ by calling function FixDiscreteVar. In the last iteration of the RH algorithm, P_{sub}^{n+1} equals P_{new}, and the remaining batches of P_{new} are scheduled according to $\mathrm{Sol^{n+1}}$ by fixing the corresponding variables in $\mathrm{Sol_{d}}$. The RH solution denoted as $\mathrm{Sol_{RH}}$ is obtained by solving MasterMILP with fixed discrete variables ($\mathrm{Sol_{d}}$) using an LP solver.

Algorithm 2 Initialization of SubMILP

function InitSubMILP(P_{new}, C_{init}, T_{HP})
    $P_{sub} \leftarrow \emptyset$
    for $r \in R$ **do**
        Sol $\leftarrow$ SolveMILP(P_{new}^{r}, C_{init})
        for $p \in P_{new}^{r}, s \in S$ **do**
            if $\mathrm{Sol}.Ts_{ps} \leqslant \max_{m \in M_s} (C_{init}.ts_m) + T_{HP}$ **then**
                $P_{sub} \leftarrow P_{sub} \cup p$
            end if
        end for
    end for
    return P_{sub}
end function

Algorithm 2 describes the procedures for calculating P_{sub}. P_{sub} is a subset of P_{new} that determines all allowable combinations of recipe types and batch sequences in a given time horizon. The size of P_{sub} determines the problem size of SubMILP. To find a smaller size of P_{sub}, a series of relatively smaller MILP problems are formulated to only schedule batch runs of the same recipe within

Algorithm 3 Solve SubMILP with extended results

function SolveSubMILP(P_{sub}, C_{init}, T_{HP})
    $P_{fix} \leftarrow \emptyset$
    Sol $\leftarrow$ SolveMILP(P_{sub}, C_{init})
    for $p \in P_{sub}, s \in S$ **do**
        if Sol.$Ts_{ps} \leqslant \max_{m \in M_s} (C_{init}.ts_m) + T_{HP}$ **then**
            $P_{fix} \leftarrow P_{fix} \cup p$
        end if
    end for
    $C_{init}^{new} \leftarrow$ CalculateCinit(Sol, P_{fix})
    return (Sol, P_{fix}, C_{init}^{new})
end function

the time horizon. P_{new}^r, a subset of P_{new}, is defined as $\{p | p \in P_{new}, p \in P_r\}$, $\forall\ r \in R$. Function SolveMILP(P_{new}^r, C_{init}) refers to the process of applying a MILP solver to a SubMILP defined by P_{new}^r and C_{init} and it returns the optimal solution (Sol). The time horizon is defined for multiple stages with the start time given by the availability ts_m of the units in the same stage given by C_{init} and the length parameter T_{HP}. By further checking the start times of batches in the solution Sol.Ts_{ps}, batch runs that start before the end time of the time horizon in any stage s are added to P_{sub}. Therefore, P_{sub} contains all batch runs of the optimal single-recipe scheduling sequences within the time horizon and allows for all combinations of recipe-specific batch sequences in the time horizon.

Algorithm 3 presents the procedures to calculate P_{fix} and C_{init}^{new} based on the optimal solution of the sub-problem. Firstly, SubMILP built by P_{sub} and C_{init} is solved optimally using an MILP solver. Because P_{sub} contains many more batch runs to provide all possibilities of recipe-specific batch sequences within the time horizon, P_{fix} as a subset of P_{sub} is introduced to denote the set of batch runs that are scheduled within the time horizon according to solution Sol. Furthermore, the initial condition for the scheduling in the next time horizon C_{init}^{new} is computed by checking unit condition in solution Sol after finishing all batch runs in P_{fix}.

5. Computational Results

In this section, the proposed rolling horizon method and the GDP-based scheduling formulation are tested and analyzed for a set of tuning parameters. The optimization models and the rolling horizon method are implemented using Pyomo (version 5.7) in Python 3.7. The GDP-based formulation is automatically transformed into MILP models using big-M reformulations via Pyomo's automatic transformation and are solved using Gurobi 9.0. The tests were run on a 2.3GHz 36 core machine with 192GB of RAM.

5.1. Application of the rolling horizon method to the scheduling problem

The rolling horizon approach in Algorithm 1 is applied to solve a relatively large scheduling problem. The problem instance is generated from the aforementioned case study and formulated as an MILP scheduling problem. In the problem instance, a set of 36 predefined batches with three recipes (each recipe has 12 batches) will be scheduled, representing set P in the rolling horizon algorithm. The cleaning tasks are the main maintenance actions performed in the batch reactors and are scheduled along with batch production runs to prevent the values of the fouling KPIs from becoming too large. In the storage stage, two parallel tanks store products generated from batch runs in the production stages; all batches have the same production size, and each of the tanks stores up to three batches of products of the same grade before the final quality check. This leads to 12 groups of products to be stored and checked in the storage stage, and each group is associated with three batches of one recipe. The processing time in unit U1 for different recipes is in the range from two to three hours, and the processing time in the batch reactors is 6-7 hours, based on the type of recipe and the current state of fouling. Cleaning of the fouling requires a two-day shutdown of the reactors. Material transfers between two neighboring stages takes 1.5-2 hours, depending on the type of unit. Quality checks in the storage tanks take roughly six hours. The production time for a given production target can be up to two weeks. The master MILP for this problem instance has a size of 58687 rows, 2112 columns and 163381 nonzeros with 220 continuous variables and 1668 binary variables. Solving this master MILP using an MILP solver is difficult and requires high computational effort.

A multiobjective optimization function is considered in the rolling horizon framework. The objective function is a convex combination of makespan and the final fouling KPIs of the batch reactors. The two terms in Eq. (32) are correlated and result in a scheduling sequence that aims to end in an as clean as possible condition for the non-bottleneck reactor.

$$\min \quad \lambda \cdot \mathrm{MS}^n + (1-\lambda) \cdot \sum_{m \in M_2} fe_m^n \tag{32}$$

The value of the weight parameter λ is between zero and one. The object function becomes minimization of Makespan when λ equals one, and the objective function puts more weight on minimizing the final fouling KPIs when the value of λ decreases. The weight parameter λ is taken as a tuning parameter for the performance of the rolling horizon framework. The other tuning parameter considered in the test is the length of time horizons T_{HP}.

The tuning parameters are assigned to a set of values in the tests of the rolling horizon approach. In Table 1, five sets of scheduling results are generated from the rolling horizon method using different values of T_{HP} and a fixed value of λ. The values of T_{HP} are 400min up to 800min in increments of 100min. The result in No.5 has the largest value of T_{HP} and presents the smallest value of the objective function in Table 1. The solutions of No.1, No.3 and No.5 in Table 1 are presented in Gantt charts along with figures of fouling KPI curves

No.	T_{HP} (min)	fe_{U2}	fe_{U3}	MS (min)	Obj. value	Solution time (sec)
1	400	3.07	2.19	13491	12142.69	551
2	500	3.07	2.19	13491	12142.69	624
3	600	3.35	2.01	13806	12426.52	6405
4	700	3.85	2.28	13470	12124.15	12966
5	800	4.25	2.19	13393	12054.41	11430

Table 1: Results of running rolling horizon method; weight parameter (λ) is 0.9

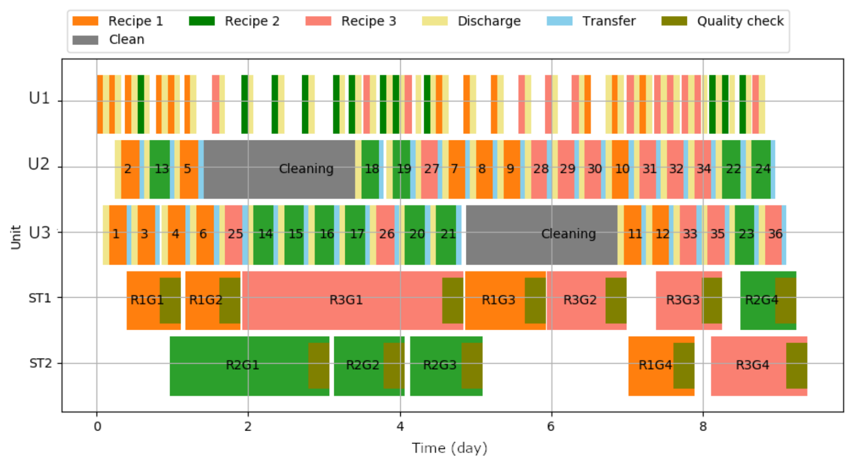

Figure 2: Gantt chart in RH result given λ: 0.9, T_{HP}: 400

as Figs. 2 to 7 show. One of the main differences between these solutions are the timing of cleaning tasks in the two reactors. In the solution of No.5, the cleaning tasks are scheduled relatively earlier than the other two solutions: the cleaning task for unit U2 is scheduled after one batch run in the solution of No.5 as Fig. 4 shows; Fig. 2 illustrates that the cleaning task for unit U2 in the solutions of No.1 is scheduled after three batch runs, and the one in the solution No.3 is scheduled after four batch runs as presented in Fig. 3. The fouling KPI curves in Figs. 5 to 7 are associated with the sequences of batches and cleaning. The results show that the solution of No.5 has the largest value of the final fouling KPI in unit U2, and the earlier scheduled cleaning task is one of the main contributing factors. Despite the effect of recipe sequences on the fouling evolution, more batch runs are scheduled after the cleaning task in unit M2, which in general results in more fouling in the batch reactor by the end of the scheduled production.

Moreover, the results in Table 1 show that solutions with larger values of T_{HP} tend to have better solutions with smaller values of the objective func-

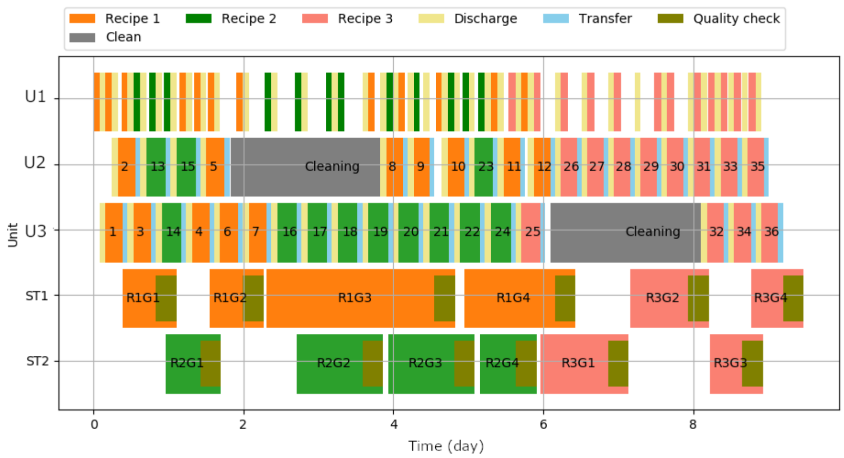

Figure 3: Gantt chart in RH result given λ: 0.9, T_{HP}: 600

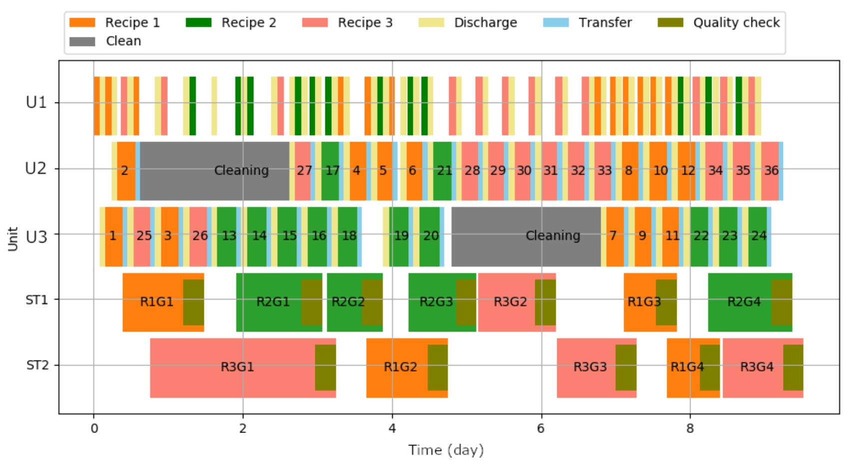

Figure 4: Gantt chart of RH result given λ: 0.9, T_{HP}: 800

tion. The rolling horizon method does not generate the optimal solutions but computes optimal solutions in each iteration and yields a sub-optimal complete solution after all the iterations. The sub-scheduling problems with larger T_{HP} take more batches into account and therefore are more likely to generate better local solutions. However, this is not always valid, and one exception can be found in the solution of No.3. This solution has a much larger value of the objective function than other solutions. Comparing with other solutions in Table 1, one of the main differences in the solution of No.3 is described in the previous paragraph that the cleaning tasks of No.3 are scheduled timely later

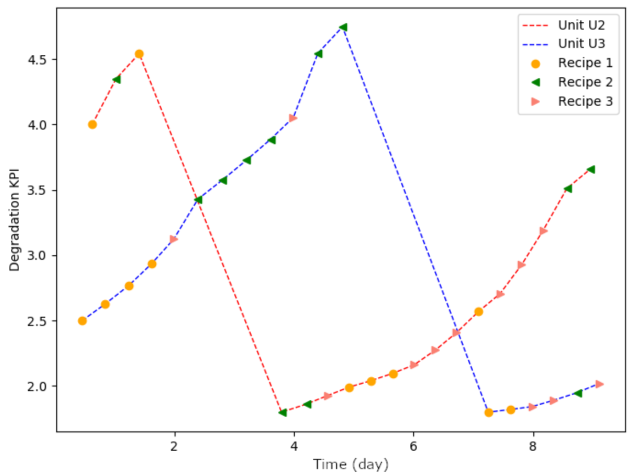

Figure 5: Fouling evolution in RH result given λ: 0.9, T_{HP}: 400

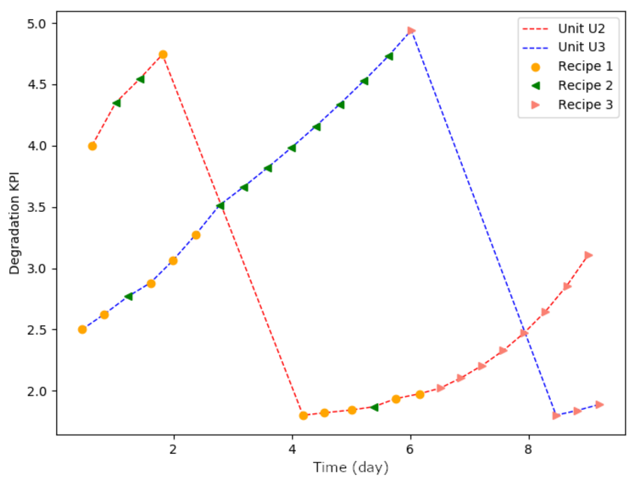

Figure 6: Fouling evolution in RH result given: λ: 0.9, T_{HP}: 600

than ones in other solutions.

To illustrate the different solutions generated in the iterations, the results in the second iteration of No.1 and No.3 are presented in Figs. 8 and 9. The results in the first iteration of both solutions P^1_{fix} are the same: batch 1 and batch 2 are

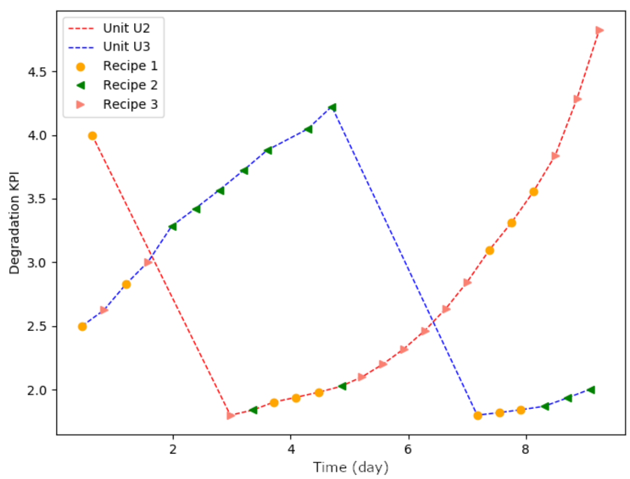

Figure 7: Fouling evolution in RH result given λ: 0.9, T_{HP}: 800

fixed in the first time horizon, which generates the same initial conditions C^2_{init} for the scheduling in the second iteration of the both solutions of the rolling horizon method. A larger value of T_{HP} in No.3 leads to a larger set P^2_{sub} which has 13 batches compared with 10 batches in P^2_{sub} of result No.1. The differences in the scheduling problems of the second iteration yield two local solutions from the rolling horizon. From this solution, the first three batch runs are fixed leading to different P^2_{fix}; these are batches 3, 13 and 4 in No.1 and batches 3, 13 and 15 in No.3 as Figs. 2 and 3 present. In the iterations after, No.1 and No.3 generate different local solutions P^n_{fix} as they are solving completely different sub-scheduling problems.

The solution time is total time over all iterations and increases as the size of the scheduling problems in each iteration increases. According to the results in Table 1, the solution time increases from 624 seconds to 6405 seconds when T_{HP} increases from 500 minutes to 600 minutes. Another increase in the scale of solution time is when T_{HP} increases from 600 minutes to 700 minutes, and the solution time increases from 6405 seconds to 12966 seconds. Moreover, the solutions generated by No.1 and No.2 are the same with similar solution times. No.4 and No.5 take the same order of magnitude of solution time. This is because the increase in T_{HP} of the two pairs of solutions does not enable adding more batch runs in the iterations, and therefore the size of scheduling problems that depend on the number of batches of each iterations are nearly the same.

A set of RH tests were also performed using different values of λ and a fixed value of T_{HP}. The results are presented in Table 2, in which seven sets of λ

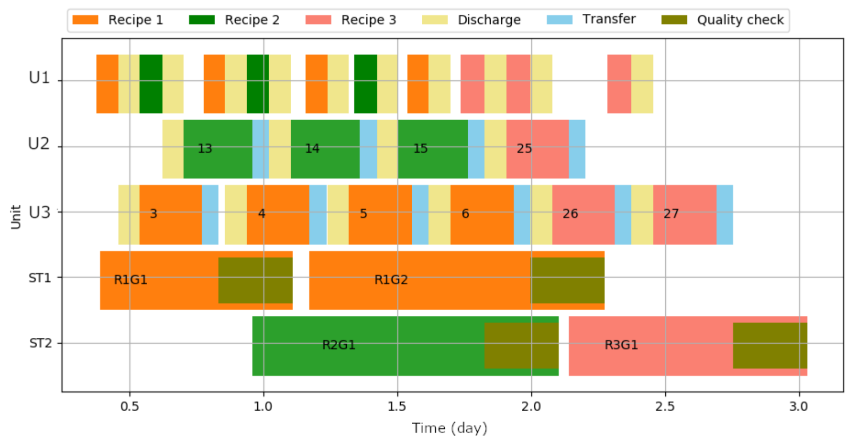

Figure 8: Gantt chart in RH result given λ: 0.9, T_{HP}: 400 at 2nd iteration

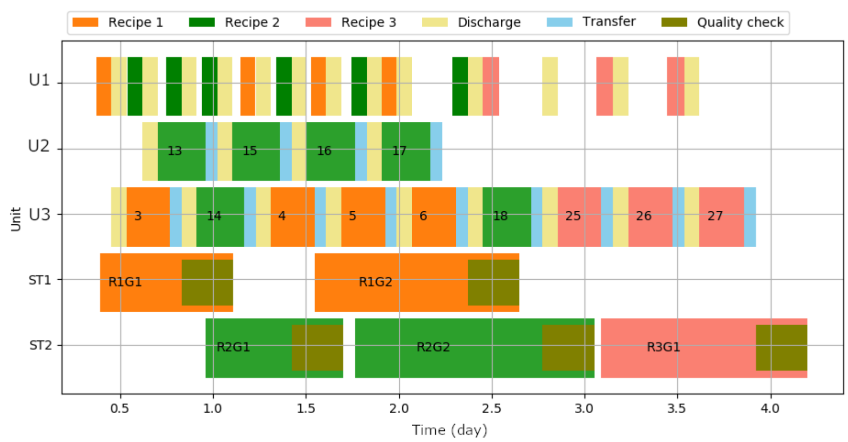

Figure 9: Gantt chart in RH result given λ: 0.9, T_{HP}: 600 at 2nd iteration

are from 0.3, up to 0.9 and T_{HP} set to 800. The key results including MS and final fouling KPI are presented in Table 2. Since the weight parameter in the objective function of the tests are different, the values of objective functions of the tests are not presented in Table 2 as they cannot be directly compared. Among these solutions, No.7 has the smallest value of MS, and No.1 and No.2 have relatively larger MS. This intuitively can be explained as the value of the weight parameter. As λ increases, the relative weight on MS also increases, and solutions of the different time horizons tend to have smaller MS in each iteration. These solutions finally result in smaller MS of the overall solutions as can be

No.	λ	fe_{U2}	fe_{U3}	MS (min)	Solution time (sec)
1	0.3	4.07	2.25	13620	12446
2	0.4	4.95	2.02	13723	12314
3	0.5	4.27	2.19	13404	13245
4	0.6	4.03	2.26	13617	13211
5	0.7	3.54	2.26	13453	10316
6	0.8	5.19	2.03	13679	12948
7	0.9	4.25	2.19	13393	11430

Table 2: Results of running rolling horizon method; T_{HP} is 800

seen in No.7 and No.1. However, local solutions that have smaller MS do not always generate smaller MS of the overall solution, as seen in the cases between No.1 and No.2, the ones between No.3 and No.4 and the comparison between No. 5 and No.6. The overall solution presented by the RH solutions from each iteration always have a gap to the globally optimal solution. Different gaps in the solutions in Table 2 make the weight parameter less sensitive in emphasizing one of the multi-objective terms in the overall performance. Furthermore, while both the makespan and the fouling KPIs depend on the sequence of products, there is a base-level threshold for both of these values that must always be incurred even if the globally optimal sequence is determined leaving a relatively narrow (though still significant) band for improvement. It is sometimes difficult to see this trend due to the loss of optimality at each iteration. On the other hand, changing λ results in different solutions per iteration and results in different feasible solutions. A practical view of using various values of λ in the RH runs is to provide a selection of good solutions from these feasible solutions. For example, the solutions of No.3, No.5 and No.7 in Table 2 are relatively better than other solutions by looking at both fouling KPIs and MS.

In an attempt to reduce the computational efficiency of the algorithm we attempted to omit some problem details, in this case the additional time due to fouling, and add them back in a post-processing heuristic. This is implemented by modifying Eq. (24) as follows:

$$Tp_{ps} = \sum_{m \in M_s} tp_{pm} \cdot Y_{pm}, \ \forall \ p \in P, \ s \in S$$

in the formulations for the sub-problems in Algorithm 1. The solutions for each iteration are then corrected to be feasible and integrated into the master problem in Algorithm 1. The results are presented in Table 3. It was observed that the overall computation time of the algorithm was improved comparing with the results in Table 2, and the solutions with longer time of sub-period $T_{HP} = 1000$ are obtained within around two hours. Meanwhile, this variant RH method generated good feasible solutions for cases No.3 and No.5; especially, No.5 ends up with smaller values of fouling KPIs, while MS is nearly 1.5 hours shorter than No.6. Therefore, this variant RH method helps to generate good feasible solutions and can solve the problem with acceptable computation time.

No.	λ	T_{HP} (min)	fe_{U2}	fe_{U3}	MS (min)	Solution time (sec)
1	0.4	800	3.86	2.14	13669	3696
2	0.6	800	3.86	2.14	13669	3694
3	0.8	800	3.86	2.14	13570	3702
4	0.4	1000	3.87	1.91	13793	7813
5	0.6	1000	1.87	2.37	13577	4393
6	0.8	1000	2.34	1.95	13694	6721

Table 3: Results of running rolling horizon method with simplified sub-problems

6. Conclusions

Optimized batch scheduling is key for the profitability of many process industries. These production plants produce multiple products in a multi-stage environment. Due to the tight coupling between production scheduling, maintenance scheduling, and product storage it is desirable to combine these concerns into a single optimization problem. In this work, an industrial scheduling case study was analyzed that considers sequence-dependent degradation, restorative maintenance, and limited product storage. The scheduling problem was modeled using a continuous-time GDP-based model. One of the issues with combining maintenance and production scheduling is that the maintenance concerns occur on a different time scale than production scheduling. Merging the two into a single model is intractable using a pure mathematical programming approach. To overcome this, a rolling horizon algorithm was proposed. The rolling horizon algorithm breaks the time horizon into smaller periods where soft scheduling targets for each period are predefined using a heuristic. Tasks that are unable to be completed in the current window are passed back to the heuristic to be considered when predefining tasks for the subsequent window. This is performed iteratively until the entire schedule has been calculated in detail. Results show that the proposed approach can yield good quality solutions quite quickly depending on the choice of rolling horizon parameters. It is not easy however to determine what good parameters are *a priori*.

This points to a few directions for future research. The question remains of how to automatically tune the rolling horizon algorithm to determine optimal parameter sets without extensive off-line numerical testing. Other aspects of the production site could also be incorporated into the scheduling model. For example, planning decisions such as the ordering and timing of raw material deliveries could improve overall site coordination. Uncertainty also plays a role in all industrial scheduling. In this case study, two major sources of uncertainty exist: batch timings, and fouling evolution. These could potentially be integrated into the scheduling model as stochastic parameters.

Acknowledgement

Financial support is gratefully acknowledged from the Marie Skłodowska Curie Horizon 2020 EID-ITN project "PROcess NeTwork Optimization for efficient and sustainable operation of Europe's process industries taking machinery condition and process performance into account – PRONTO", Grant agreement No 675215.

The anonymous reviewers are gratefully acknowledged for suggestions that significantly improved the RH algorithm.

References

Aguirre, A.M., Méndez, C.A., Gutierrez, G., Prada, C.D., 2012. An improvement-based milp optimization approach to complex aws scheduling. Computers & Chemical Engineering 47, 217 – 226. URL: http://www.sciencedirect.com/science/article/pii/S0098135412002207, doi:https://doi.org/10.1016/j.compchemeng.2012.06.036. fOCAPO 2012.

Baumann, P., Trautmann, N., 2014. A hybrid method for large-scale short-term scheduling of make-and-pack production processes. European journal of operational research 236, 718–735.

Biondi, M., Sand, G., Harjunkoski, I., 2017. Optimization of multipurpose process plant operations: A multi-time-scale maintenance and production scheduling approach. Computers & Chemical Engineering 99, 325–339.

Castro, P.M., Grossmann, I.E., 2012. Generalized disjunctive programming as a systematic modeling framework to derive scheduling formulations. Industrial & Engineering Chemistry Research 51, 5781–5792.

Castro, P.M., Grossmann, I.E., Veldhuizen, P., Esplin, D., 2014. Optimal maintenance scheduling of a gas engine power plant using generalized disjunctive programming. AIChE journal 60, 2083–2097.

Chen, Q., Grossmann, I.E., 2019a. Effective generalized disjunctive programming models for modular process synthesis. Industrial & Engineering Chemistry Research 58, 5873–5886. URL: https://doi.org/10.1021/acs.iecr.8b04600, doi:10.1021/acs.iecr.8b04600, arXiv:https://doi.org/10.1021/acs.iecr.8b04600.

Chen, Q., Grossmann, I.E., 2019b. Modern modeling paradigms using generalized disjunctive programming. Processes 7, 839.

Dalle Ave, G., Harjunkoski, I., Engell, S., 2019a. A non-uniform grid approach for scheduling considering electricity load tracking and future load prediction. Computers & Chemical Engineering 129, 106506. URL: http://www.sciencedirect.com/science/article/pii/S0098135419300183, doi:https://doi.org/10.1016/j.compchemeng.2019.06.031.

Dalle Ave, G., Hernandez, J., Harjunkoski, I., Onofri, L., Engell, S., 2019b. Demand side management scheduling formulation for a steel plant considering electrode degradation. IFAC-PapersOnLine 52, 691–696.

Dimitriadis, A., Shah, N., Pantelides, C., 1997. Rtn-based rolling horizon algorithms for medium term scheduling of multipurpose plants. Computers & Chemical Engineering 21, S1061 – S1066. URL: http://www.sciencedirect.com/science/article/pii/S0098135497876430, doi:https://doi.org/10.1016/S0098-1354(97)87643-0. supplement to Computers and Chemical Engineering.

Gupta, S., Karimi, I., 2003. An improved milp formulation for scheduling multiproduct, multistage batch plants. Industrial & engineering chemistry research 42, 2365–2380.

Hadera, H., Ekström, J., Sand, G., Mäntysaari, J., Harjunkoski, I., Engell, S., 2019. Integration of production scheduling and energy-cost optimization using mean value cross decomposition. Computers & Chemical Engineering 129, 106436. URL: http://www.sciencedirect.com/science/article/pii/S0098135418311451, doi:https://doi.org/10.1016/j.compchemeng.2019.05.002.

Harjunkoski, I., 2016. Deploying scheduling solutions in an industrial environment. Computers & Chemical Engineering 91, 127–135.

Hart, W.E., Laird, C.D., Watson, J.P., Woodruff, D.L., Hackebeil, G.A., Nicholson, B.L., Siirola, J.D., 2017. Pyomo-optimization modeling in python. volume 67. Springer.

Kilic, O.A., van Donk, D.P., Wijngaard, J., 2011. A discrete time formulation for batch processes with storage capacity and storage time limitations. Computers & Chemical Engineering 35, 622–629.

Klanke, C., Yfantis, V., Corominas, F., Engell, S., 2020. Scheduling of a large-scale industrial make-and-pack process with finite intermediate buffer using discrete-time and precedence-based models, in: Eden, M.R., Ierapetritou, M.G., Towler, G.P. (Eds.), 30th European Symposium on Computer Aided Process Engineering (ESCAPE 30). Elsevier. volume Accepted of *Computer Aided Chemical Engineering*.

Li, Z., Ierapetritou, M.G., 2010. Rolling horizon based planning and scheduling integration with production capacity consideration. Chemical Engineering Science 65, 5887 – 5900. URL: http://www.sciencedirect.com/science/article/pii/S000925091000477X, doi:https://doi.org/10.1016/j.ces.2010.08.010.

Méndez, C.A., Cerdá, J., 2003. An milp continuous-time framework for short-term scheduling of multipurpose batch processes under different operation strategies. Optimization and Engineering 4, 7–22.

Méndez, C.A., Cerdá, J., Grossmann, I.E., Harjunkoski, I., Fahl, M., 2006. State-of-the-art review of optimization methods for short-term scheduling of batch processes. Computers & Chemical Engineering 30, 913–946.

Sundaramoorthy, A., Maravelias, C.T., 2008. Modeling of storage in batching and scheduling of multistage processes. Industrial & Engineering Chemistry Research 47, 6648–6660.

Trespalacios, F., Grossmann, I.E., 2017. Symmetry breaking for generalized disjunctive programming formulation of the strip packing problem. Annals of Operations Research 258, 747–759.

Vieira, M., Pinto-Varela, T., Barbosa-Póvoa, A.P., 2017. Production and maintenance planning optimisation in biopharmaceutical processes under performance decay using a continuous-time formulation: A multi-objective approach. Computers & Chemical Engineering 107, 111–139.

Wu, O., Bouaswaig, A., Imsland, L., Schneider, S.M., Roth, M., Leira, F.M., 2019a. Campaign-based modeling for degradation evolution in batch processes using a multiway partial least squares approach. Computers & Chemical Engineering 128, 117–127.

Wu, O., Dalle Ave, G., Harjunkoski, I., Bouaswaig, A., Schneider, S.M., Roth, M., Imsland, L., 2020a. Optimal production and maintenance scheduling for a multiproduct batch plant considering degradation. Computers & Chemical Engineering 135, 106734. URL: http://www.sciencedirect.com/science/article/pii/S0098135419309706, doi:https://doi.org/10.1016/j.compchemeng.2020.106734.

Wu, O., Dalle Ave, G., Harjunkoski, I., Bouaswaig, A., Schneider, S.M., Roth, M., Imsland, L., 2020b. Short-term multiproduct batch scheduling considering storage features, in: IFAC World Congress 2020 Germany. Elsevier. volume Accepted of *IFAC*.

Wu, O., Dalle Ave, G., Harjunkoski, I., Imsland, L., Schneider, S.M., Bouaswaig, A.E.F., Roth, M., 2019b. Short-term scheduling of a multipurpose batch plant considering degradation effects, in: Computer Aided Chemical Engineering. Elsevier. volume 46, pp. 1213–1218.

Xenos, D.P., Kopanos, G.M., Cicciotti, M., Thornhill, N.F., 2016. Operational optimization of networks of compressors considering condition-based maintenance. Computers & Chemical Engineering 84, 117–131.