MIXED-INTEGER PROGRAMMING MODELS AND SOLUTION METHODS FOR CHEMICAL PRODUCTION SCHEDULING

by

HO JAE LEE

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

(Chemical Engineering)

at the

UNIVERSITY OF WISCONSIN-MADISON

2019

Date of final oral examination: October 30, 2019

The thesis is approved by the following members of the Final Oral Committee:

Christos T. Maravelias, Professor, Chemical and Biological Engineering
Reid Van Lehn, Assistant Professor, Chemical and Biological Engineering
Laura Albert, Professor, Industrial and Systems Engineering
Victor M. Zavala, Associate Professor, Chemical and Biological Engineering

ABSTRACT

Optimization-based chemical production scheduling allows for efficient utilization of available assets and brings significant operational benefits including reduction in costs. Unfortunately, application of such techniques to industrial settings is challenging due to multiple reasons: (i) the optimization models need to be general to accommodate different production processes, (ii) the solution of such models need to be quick to allow for frequent updates to the schedules, and (iii) the models should be capable of providing multiple alternative schedules for the practitioners to compare and implement. The goal of this work is to address the aforementioned challenges and bring optimization-based scheduling techniques closer to industrial applications.

First, we develop mathematical programming models for simultaneous batching and scheduling in general sequential production environment while taking into account various process features including storage policies and limited shared utilities. The models are based on novel modeling approaches which allow for exploitation of instance characteristics, thus leading to solution of large-scale instances.

Second, we develop a novel framework for a solution algorithm that harnesses the advantages of discrete- and continuous-time scheduling models. Specifically, we propose an algorithm that has modeling flexibility and computational efficiency of discrete-time models, as well as high solution accuracy of their continuous counterparts. We investigate in detail how the algorithm can be improved and extended to solve real-world industrial instances that are thought to be computationally near impossible if transitional methods were to be used.

Finally, we develop systematic methods to generate multiple alternative schedules, specifically to account for modeling simplifications introduced in the scheduling models and plant nervousness when revising schedules. We generate alternative schedules by quantifying specific characteristics of a schedule using explicitly defined metrics, which are favored at different degrees by penalizing them in the objective function with varying penalty weights. We show that, by leveraging penalty weights, schedules with desirable properties can be readily found.

ACKNOWLEDGMENTS

First and foremost, I would like to express my sincerest gratitude to my advisor, Professor Christos T. Maravelias, for supporting and guiding me throughout my PhD study. As a perfect mentor, he has taught me what it is to be a truly passionate and dedicated researcher. All I have achieved throughout this journey would not have been possible without him.

Besides my advisor, I would like to thank the rest of my thesis committee, Professors Victor M. Zavala, Reid Van Lehn, and Laura Albert, for attending my defense and providing valuable insights from various perspectives.

Many thanks to the members of Maravelias group for engaging discussions and providing constructive feedback. It is a privilege to have worked with such friendly and talented people. My special thanks go to Lingxun Kong for being a supportive friend. The numerous discussions I had with him were truly joyful moments.

I also want to thank Hanall for her patience and understanding. It was her that enabled me to push through the hardships along the way. Finally, I dedicate this thesis to my loving family. Every step that I took in this journey is the result of their endless and unconditional support.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1

INTRODUCTION

1.1 CHEMICAL PRODUCTION SCHEDULING

Chemical manufacturing facilities consist of a series of operations that compete for limited resources, such as process units, utilities and manpower. Through production scheduling, favorable allocation of these limited resources to various production tasks over time can be achieved (Pinedo, 2012). In various manufacturing sectors, including pharmaceuticals, food and drinks, specialty chemicals and petrochemical industries, production scheduling brings significant benefits in terms of asset utilization, customer satisfaction and cost reduction. For instance, Kelly and Mann (2003) showed that, scheduling based on optimization methods can potentially bring multi-million dollars of reduction in annual operations costs in a medium sized refinery. Furthermore, Wassick (2009) has shown that enterprise-wise cost reduction can be achieved in a chemical production complex through a more efficient coordination between the interconnected subsystems.

Hence, a wide range of methods for chemical production scheduling have been studied: (i) algorithms based on heuristics (e.g. dispatching rules), (ii) metaheuristics (e.g. genetic evolution, particle swarm algorithms), and (iii) exact methods (e.g. mathematical programming, constraint programming), as well as different types of hybrid and decomposition approaches. Among the various approaches, the ones based on mathematical programming have been the most prominent (Reklaitis, 1996; Pinto and Grossmann, 1998; Kallrath, 2002; Floudas and Lin, 2004; Mendez et al., 2006; Maravelias, 2012; Harjunkoski et al., 2014; Georgiadis et al., 2019).

The vast majority of mathematical programming based approaches employ mixed-integer programming (MIP) formulations. Specifically, discrete decisions such as allocation and start of tasks are modeled via binary variables, while as continuous decisions such as batch sizes and inventory levels are modeled via non-negative continuous variables. The formulations utilize constraints to ensure the feasibility of the schedule by enforcing material balance, unit allocation and capacity constraints as well as other process-specific characteristics.

Unfortunately, however, such MIP scheduling models are NP-hard problems, and thus, are extremely difficult to solve. Therefore, the main focus in the process systems engineering (PSE) community has been to develop efficient models and solution methods to enable solution of scheduling models for industrial-scale instances.

1.2 PRODUCTION ENVIRONMENTS

Chemical processes are generally classified into three types of production environments (Maravelias, 2012). First, in the sequential environment, strict material handling restrictions are present where batch mixing or splitting is not allowed, and each task consumes and produces one material. Hence, the entities that are being modeled throughout the process are batches, leading to a batch-based modeling approach. Second, in the network environment, batches of materials can be mixed and split, and each task is modeled to consume and produce one or more materials. Problems in network environments have been addressed using material-based models in which material inventory levels are tracked throughout the horizon. The third environment is the hybrid environment where different material handling restrictions are present. While the majority of approaches are focused on modeling either sequential or network environments, models that can address problems in hybrid environments were also proposed (Sundaramoorthy and Maravelias, 2011a; Velez and Maravelias, 2013b). We note that the aforementioned environment-based classification is not dependent on the structure of the facility, but rather on the material handling restrictions present within the facility.

The commonly known as State-Task Network (STN) and Resource-Task Network (RTN) representations were first proposed by Pantelides and coworkers for the scheduling in network environments (Kondili et al., 1993; Shah et al., 1993; Pantelides, 1994). Production facilities are represented as networks with state (resource) and task nodes connected by arcs describing the consumption and production of states (resources) by each task. The original models employed discrete representation of time, but various models based on continuous-time representation have also been proposed (Schilling and Pantelides, 1996; Papageorgiou and Pantelides, 1996a; Ierapetritou and Floudas, 1998; Mockus and Reklaitis, 1999; Lee et al., 2001; Castro et al., 2001; Giannelos and Georgiadis, 2002; Maravelias and Grossmann, 2003b; Wu and Ierapetritou, 2004; Castro et al., 2004; Sundaramoorthy and Karimi, 2005; Shaik and Floudas, 2008; Gimenez et al., 2009; Susarla et al., 2010). The proposed models enabled consideration of various process features including variable processing times, temporary storage of material within units, and different storage policies. Moreover, model based on mixed-time representation (Maravelias, 2005; Westerlund et al., 2007) and models that adopt discrete, but non-uniform time grids (Velez and Maravelias, 2013c, 2015) were also proposed. We note that the aforementioned models allow flexible material handling (i.e. batch mixing/splitting), thus cannot be used to solve scheduling problems in sequential facilities.

MIP scheduling models for the sequential environment, where each batch goes through series of predetermined stages while its integrity is preserved, are based on a batch-based approach. The majority of these models adopt continuous-time representation, in which the timing of each time point or equivalently the length of each slot is subject to optimization. Largely, two modeling approaches have been adopted: precedence-based approaches (Hui et al., 2000; Mendez et al., 2001; Gupta and Karimi, 2003; Prasad and Maravelias, 2008; Sundaramoorthy and Maravelias, 2008a,b; Kopanos et al., 2009) and time-grid-based approaches (Pinto and Grossmann, 1995; Chen et al., 2002; Castro and Grossmann, 2005; Liu and Karimi, 2007, 2008; Capon-Garcia et al., 2009). More recently, models based on discrete-time representation where the time horizon is divided into uniform time intervals are proposed (Sundaramoorthy et al., 2009; Merchan et al., 2016; Lee and Maravelias, 2017a,b). This allowed for a more efficient modeling of process features, including time-varying availability and cost of limited shared resources and linear modeling of inventory and backlog profiles.

1.3 TIME REPRESENTATIONS: DISCRETE AND CONTINUOUS

One of the key attributes for classifying MIP scheduling models is the time representation. Both discrete- and continuous-time representations for scheduling models in the network environment have distinctive advantages and disadvantages (see Figure 1.1 for a short summary). A detailed review on the comparison between the two can be found in the literature (Floudas and Lin, 2004; Sundaramoorthy and Maravelias, 2011b).

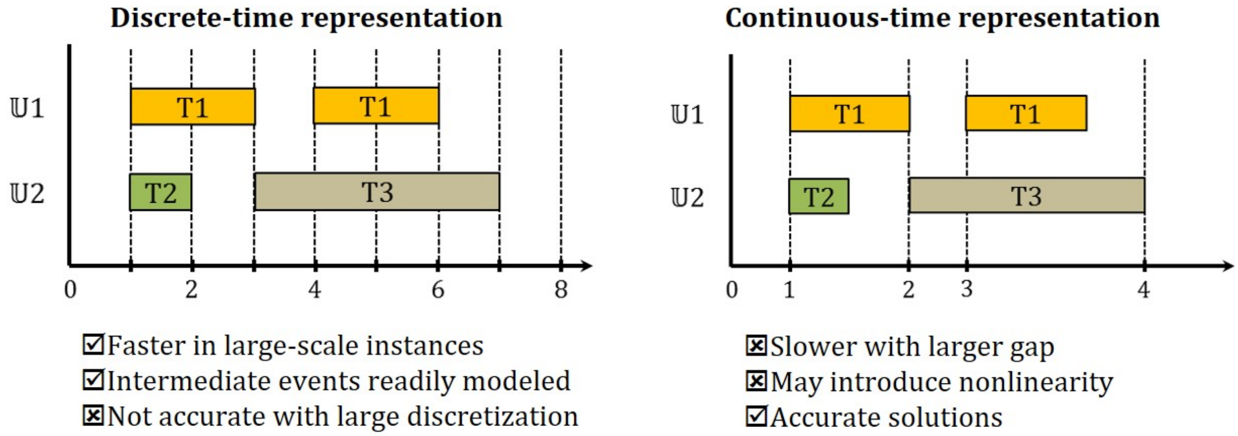

Discrete-time representation
U1
U2
T1
T1
T2
T3
0
2
4
6
8
☑Faster in large-scale instances
☑Intermediate events readily modeled
☒Not accurate with large discretization
Continuous-time representation
U1
U2
T1
T1
T2
T3
0
1
2
3
4
☒Slower with larger gap
☒May introduce nonlinearity
☑Accurate solutions

Figure 1.1: Advantages and disadvantages of discrete- and continuous-time representations.

One of the strengths of discrete-time formulations is that the common discrete-time grid adopted provides a reference grid of time for all the shared resources, such as material and utilities. This allows various profiles, such as consumption level of utilities and inventory/backlog levels, to be monitored and efficiently modeled without introduction of nonlinear constraints (Zyngier and Kelly, 2009; Velez and Maravelias, 2014). Another advantage is that time-varying problem data can be readily accommodated without any additional computational cost. Furthermore, through extensive computational studies, it is shown that the discrete-time models provide better solutions (i.e. better objective and smaller gap) than their continuous counterpart in large-scale instances (Sundaramoorthy and Maravelias, 2011b; Merchan et al., 2016). However, the main limitations are the discrete approximation of time and the large size of the resulting models. Specifically, if small discretization is needed to guarantee optimality, the model might become intractable, while if larger discretization steps are chosen, the model might provide schedules that are not accurate due to discretization errors. In addition, some of the processing features that continuous-time models can readily handle (e.g. variable processing times) cannot be modeled efficiently.

On the other hand, the main advantage of the continuous-time formulations is the accuracy of the solutions. Specifically, given sufficient number of time points, continuous-time models are guaranteed to find the optimal solution. However, the number of time points required is not known a priori, thus an iterative procedure is needed where the problem is solved while increasing the number of time points until no improvement in the objective is observed. Furthermore, they are not as efficient as discrete-time formulations in modeling processing features such as intermediate deliveries and demands, limited shared utilities, and accommodating time-varying problem data.

1.4 SOLUTION METHODS

In addition to the efforts devoted to the development of more efficient MIP scheduling models, the PSE community also focused their research on developing solution methods.

First, tightening methods where constraints are added to the models to improve the integrality gap are proposed. Some are based on generation of valid inequality based on preprocessing (Pinto and Grossmann, 1995) and others are based on problem data propagation algorithms (Velez et al., 2013; Merchan et al., 2013; Merchan and Maravelias, 2016). Decomposition methods, where a large problem is divided into smaller subproblems, including methods based on Lagrangian relaxation (Wu and Ierapetritou, 2003; Calfa et al., 2013), iterative solution of smaller problems (Maravelias and Grossmann, 2003a; Castro et al., 2009a), hierarchical decision-making (Bassett et al., 1996; Kelly and Zyngier, 2008), and integration of different solution methods (Jain and Grossmann, 2001; Harjunkoski and Grossmann, 2002; Maravelias and Grossmann, 2004; Roe et al., 2005; Maravelias, 2006), are proposed. In addition, reformulation based on variable disaggregation (Sahinidis and Grossmann, 1991; Yee and Shah, 1998) and variable aggregation (Velez and Maravelias, 2013d), as well as algorithms for parallel computing (Subrahmanyam et al., 1996; Ferris et al., 2009; Velez and Maravelias, 2013a) are investigated. Finally, solution algorithms that merges the advantages of formulations based on the two time representations are proposed (Lee and Maravelias, 2018, 2019a,b).

1.5 THESIS OUTLINE

The goal of this thesis is to bring optimization-based scheduling techniques one step closer to industrial applications. To achieve this, we aim to develop (i) general models that account for different process characteristics, (ii) solution methods that enable solution of difficult scheduling instances, and (iii) systematic approaches for generation of alternative schedules.

The remainder of the thesis is structured as follows. First, in Chapter 2, we present a general model for scheduling in multipurpose environment. Specifically, we introduce two models based on different modeling approaches and provide extensions to account for various storage policies and limited shared utilities. Next, in Chapters 3-5, we present a solution algorithm that harnesses the advantages of discrete- and continuous-time scheduling models, namely discrete-continuous algorithm (DCA). Specifically, in Chapter 3, we introduce the framework and mathematical formulations. In Chapter 4, we propose systematic approaches to decide the key model parameters of DCA, which lead to significant improvements to the computational speed and solution quality. In Chapter 5, we generalize DCA to account for a wide range of process characteristics commonly observed in different manufacturing sectors. In Chapter 6, we present systematic methods to generate alternative schedules to account for modeling simplifications and plant nervousness. Finally, in Chapter 7, we provide conclusions and discuss future research directions.

The notations used in each chapter are presented at the end of the chapter. We use lowercase italic letters for indices, uppercase bold letters for sets, lowercase Greek letters for parameters, and uppercase italics for decision variables.

2

SIMULTANEOUS BATCHING AND SCHEDULING IN MULTIPURPOSE FACILITIES

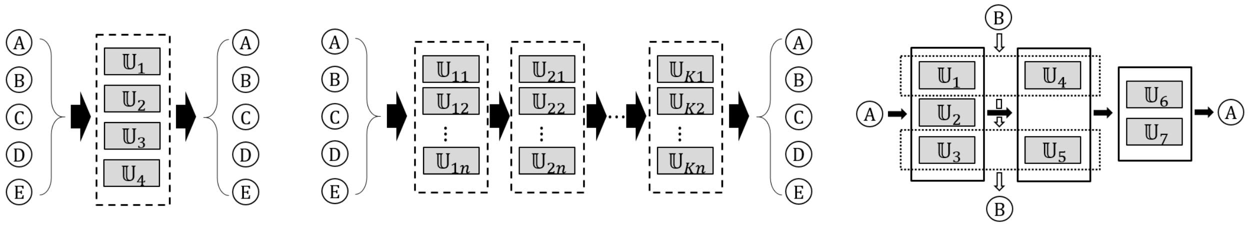
2.1 MOTIVATION

The sequential production environment is one of the most common in batch manufacturing facilities. It exhibits a series of stages in which batches have to go through without being mixed, split or recycled. Among problems in sequential environments, multistage batch plants are the facilities that have received the most attention. However, only a limited amount of studies have been conducted on the more general multipurpose batch plants.

Furthermore, most of the existing approaches for scheduling in sequential environments are based on several restrictive assumptions. Specifically, batching decisions are assumed to be decoupled from scheduling decisions, often leading to suboptimal solutions. This assumption may be reasonable in relatively simple processes, where good batching decisions can be readily identified. However, as the complexity increases, making batching decisions without considering their scheduling aspects may lead to inefficient utilization of resources. Another simplifying assumption commonly made is that there are always storage vessels available for intermediate products, i.e. unlimited intermediate storage (UIS). However, this assumption is not applicable to many of the process industries. Accordingly, the goal of this chapter is to develop mathematical programming models for scheduling in multipurpose facilities that can address the aforementioned limitations. We develop two mixed-integer programming (MIP) models that adopt two different modeling approaches. Furthermore, we discuss the computational performance of the two approaches.

2.2 SEQUENTIAL PRODUCTION ENVIRONMENT

Sequential facilities are similar to discrete manufacturing facilities (e.g. semiconductor, electronics) in that discrete entities are processed through a sequence of different stages. Hence, sequential environments can be further divided into three subtypes according to the number of stages, and the complexity of the production route (Maravelias, 2012). The first subtype is the single-stage batch plant, where each batch goes through a single stage consisting of one or more parallel units. The second subtype is the multistage batch plant, where each batch is processed through identical series of stages consisting of multiple parallel units, similar to flexible flow-shop studied in the Operations Research (OR) community. The third subtype is the multipurpose batch plant, where, similar to job-shop, each batch has to be processed through a series of *product-specific* stages. The three subtypes of sequential environment are shown in Figure 2.1. Despite the similarities, scheduling of chemical processes in sequential environment are often different from discrete manufacturing scheduling due to processing features such as utility requirements and storage policies.



(a) Single-stage (b) Multistage (c) Multipurpose

Figure 2.1: Types of sequential production environment. (a) Each batch goes through a single stage consisting of multiple parallel units. (b) Each batch goes through a sequence of stages consisting of multiple parallel units. The set of units is partitioned among the stages. (c) Each batch goes through product-specific stages, each consisting of multipurpose units. Each unit may belong to different stages of different products.

While problems in single-stage facilities have been studied (Cerda et al., 1997; Dessouky and Kijowski, 1997; Liu et al., 2010), most work has focused on multistage facilities. Broadly, two major modeling approaches have been adopted: precedence-based and time-grid-based.

Precedence-based models employ precedence binaries to explicitly determine the sequence in which each batch is processed in each stage or unit (Hui et al., 2000; Mendez et al., 2001; Mendez and Cerda, 2002; Harjunkoski and Grossmann, 2002; Gupta and Karimi, 2003; Maravelias, 2006; Panek et al., 2008; Prasad and Maravelias, 2008; Sundaramoorthy and Maravelias, 2008a,b; Kopanos et al., 2009). Depending on how the precedence binaries are defined, precedence-based models are further classified into global-precedence and local-precedence based models. Time-grid-based models adopt time slots/grids onto which batches are assigned/mapped, implicitly enforcing the precedence relationships between each pair of batches (Pinto and Grossmann, 1995; Chen et al., 2002; Castro and Grossmann, 2005; Liu and Karimi, 2007, 2008; Capon-Garcia et al., 2009). They can be further classified into unit-specific and common time-grid models. Time-grid-based models tend to be tighter and computationally more effective than precedence-based models, but require an iterative procedure to determine the number of time slots/periods necessary to represent the optimal solution. While the vast majority of work has adopted continuous-time representations, discrete-time models that can readily account for process features, such as time-varying availability and cost of limited shared resources, were also proposed (Sundaramoorthy et al., 2009; Merchan et al., 2016).

Scheduling in multipurpose facilities, on the other hand, has received limited attention. Kim et al. (2002) proposed a completion time algorithm for scheduling in facilities with re-entries, while Ferrer-Nadal et al. (2008) proposed a continuous-time model with global precedence binaries that can consider material transfer operations. Lee and Maravelias (2017a) recently proposed discrete-time models for scheduling in multipurpose facilities considering limited intermediate storage.

In terms of storage policies, Ku and Karimi (1988) and Kim et al. (1996) considered various storage policies for multistage facilities with a single unit per stage (i.e. flow-shop). This was then extended to multipurpose facilities with a single unit per stage (i.e. job-shop) (Kim et al., 2000). Liu and Karimi (2008) addressed no intermediate storage (NIS) in multistage facilities, and Mendez and Cerda (2003) proposed a continuous-time model that considers different storage policies in multipurpose facilities.

Many of the aforementioned studies are based on the assumption that batching decisions are known *a priori*. Mendez et al. proposed a two-stage approach for single-stage facilities, in which batching and scheduling problems are sequentially solved (Mendez et al., 2000). Later, a model that simultaneously considers batching and scheduling decisions in single-stage facilities was proposed by Castro et al. (2008). In multistage facilities, a model for simultaneous batching and scheduling was first proposed by Prasad and Maravelias (2008), while models that can account for various process features, such as variable processing times, storage policies, utilities and changeovers have been proposed (Sundaramoorthy and Maravelias, 2008a,b; Sundaramoorthy et al., 2009; Castro et al., 2009b). Simultaneous batching and scheduling for batch plants operating in a campaign-mode was introduced (Fumero et al., 2014).

2.3 PROBLEM STATEMENT AND TIME REPRESENTATION

We study the simultaneous batching and scheduling in multipurpose batch plants with limited intermediate storage. Given product orders, batching decisions (i.e. number of batches and batch sizes) have to be made, and each batch has to then go through order-specific stages (dependent on the product recipe), consisting of one or more multipurpose units, which may belong to more than one order-specific stages. Furthermore, *re-entry* may exist, in which a unit may belong to multiple stages of the production sequence of an order. Multiple orders of the same product may be given, which results in multiple orders with an identical recipe. Batches of intermediates between stages can either be temporarily stored in a compatible storage vessel or sent to the next stage directly.

We make the following assumptions: (i) preemption is not allowed (i.e. once started, tasks cannot be interrupted); (ii) processing times are independent of the batch size; (iii) all data are deterministic; (iv) storage vessels have large enough capacity to store any size of compatible batch; and (v) batch size does not change through production stages. The extensions where the fourth and fifth assumptions are relaxed will be discussed in section 2.6.

The following are the sets, indices and parameters that we use to define the problem (see Figure 2.2).

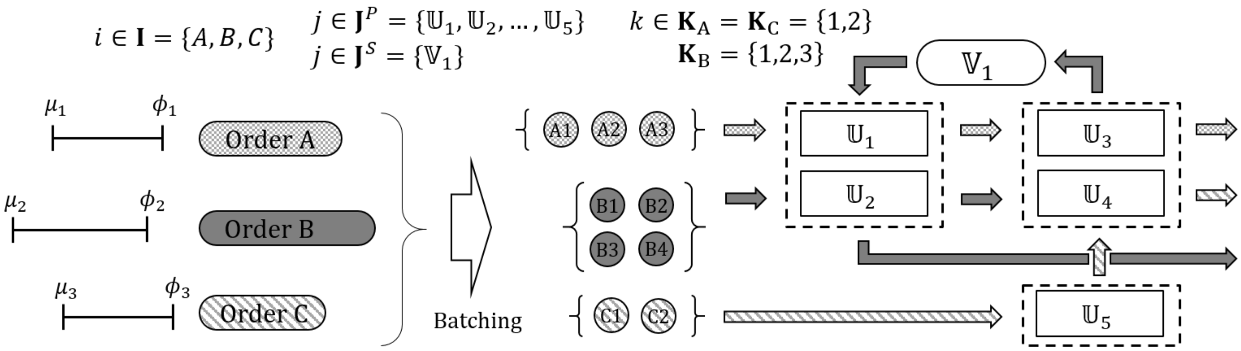
Indices/sets

$i \in \mathbf{I}$	Orders
$j \in \mathbf{J}/\mathbf{J}^P/\mathbf{J}^S$	Units/processing units/storage vessels

$k \in \mathbf{K}$	Stages
$\mathbf{J}_{ik}$	Process units that can process order i on stage k
$\mathbf{J}^{S}_{ik}$	Storage vessels that can store a batch of order i that has been processed on stage k
$\mathbf{K}_i$	Stages on which order i is processed
$\mathbf{K}^{-}_{ik}/\mathbf{K}^{+}_{ik}$	Stages on which order i have to be processed before/after stage k

Parameters

α_{ijk}	Cost of processing a batch of order i on stage k in unit j
$\beta^{min}_{j}/\beta^{max}_{j}$	Minimum/maximum capacity of unit j
κ^{LS}_{i}	Last stage of order i
μ_i	Release time of order i
ξ_i	Amount due of order i
τ_{ijk}	Processing time of order i on stage k in unit j
ϕ_i	Due date of order i



$i \in \mathbf{I} = \{A, B, C\}$
$j \in \mathbf{J}^P = \{\mathbb{U}_1, \mathbb{U}_2, \ldots, \mathbb{U}_5\}$
$j \in \mathbf{J}^S = \{\mathbb{V}_1\}$
$k \in \mathbf{K}_A = \mathbf{K}_C = \{1,2\}$
$\mathbf{K}_B = \{1,2,3\}$
$\mathbb{V}_1$
μ_1
ϕ_1
Order A
μ_2
ϕ_2
Order B
μ_3
ϕ_3
Order C
Batching
A1
A2
A3
B1
B2
B3
B4
C1
C2
$\mathbb{U}_1$
$\mathbb{U}_2$
$\mathbb{U}_3$
$\mathbb{U}_4$
$\mathbb{U}_5$

Figure 2.2: Representation of a multipurpose facility consisting of five units and a storage vessel. Total of three orders are to be met, each of which goes through at most three production stages.

Figure 2.2 shows an example of a multipurpose facility. It consists of 5 processing units ($j \in \mathbf{J}^P = \{\mathbb{U}_1, \mathbb{U}_2, \ldots, \mathbb{U}_5\}$), and a storage vessel ($j \in \mathbf{J}^S = \{\mathbb{V}_1\}$). There are three orders ($i \in \mathbf{I} = \{A, B, C\}$) with release and due dates, μ_i and ϕ_i. Each order has to go through order-specific stages defined as $\mathbf{K}_A = \mathbf{K}_C = \{1, 2\}$ and $\mathbf{K}_B = \{1, 2, 3\}$. The compatible units and vessels for each stage of each order are $\mathbf{J}_{A,1} = \mathbf{J}_{B,1} = \mathbf{J}_{B,3} = \{\mathbb{U}_1, \mathbb{U}_2\}$, $\mathbf{J}_{A,2} = \mathbf{J}_{B,2} = \mathbf{J}_{C,2} = \{\mathbb{U}_3, \mathbb{U}_4\}$,

$\mathbf{J}_{C,1} = \{\mathbb{U}_5\}$, and $\mathbf{J}^S_{B,2} = \{\mathbb{V}_1\}$.

A discrete-time representation is adopted. The time horizon is divided into uniform intervals of length δ to form a total of T time periods. We introduce set $t \in \mathbf{T} = \{0, 1, 2, ..., T\}$ to denote the T+1 time points, from the earliest release date (i.e. $\eta^0 = min_{i \in \mathbf{I}}\, \mu_i$) to the latest due date (i.e. $\eta^F = max_{i \in \mathbf{I}}\, \phi_i$) as shown in Figure 2.3. The discretization step δ is usually chosen as the greatest common factor of the time related data, but greater values of δ can be chosen if computational tractability is favored over accuracy.

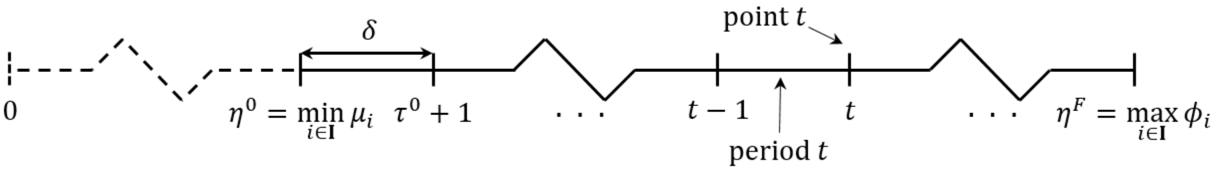

δ
point t
0
$\eta^0 = \min_{i \in \mathbf{I}} \mu_i$
$\tau^0 + 1$
. . .
$t-1$
t
. . .
$\eta^F = \max_{i \in \mathbf{I}} \phi_i$
period t

Figure 2.3: Common grid with discrete-time representation adopted in the proposed models.

Given the order release and due dates, the earliest start (ε_{ik}) and latest finish (λ_{ik}) times of the batches of order i on stage $k \in \mathbf{K}_i$ can be calculated.

$$\varepsilon_{ik} = \mu_i + \sum_{k' \in \mathbf{K}^-_{ik}} \min_{j \in \mathbf{J}_{ik'}} \tau_{ijk'} \quad \forall\, i, k \in \mathbf{K}_i \tag{2.1}$$

$$\lambda_{ik} = \phi_i - \sum_{k' \in \mathbf{K}^+_{ik}} \min_{j \in \mathbf{J}_{ik'}} \tau_{ijk'} \quad \forall\, i, k \in \mathbf{K}_i \tag{2.2}$$

Set $\mathbf{T}_{ijk}$ denotes the set of feasible time points for the batches of order i on stage $k \in \mathbf{K}_i$ executed in unit $j \in \mathbf{J}_{ik}$. In addition, $\mathbf{I}_{jkt}$ is the set of orders on stage k that can start in unit j at time point t; and $\mathbf{J}_{ikt}$ is the set of units in which batches of order i on stage $k \in \mathbf{K}_i$ can start at time point t.

$$\mathbf{T}_{ijk} = \{t \in \mathbf{T} | \varepsilon_{ik} \le t \le \lambda_{ik} - \tau_{ijk}\} \quad \forall\, i, k \in \mathbf{K}_i, j \in \mathbf{J}_{ik} \tag{2.3}$$

$$\mathbf{I}_{jkt} = \{i \in \mathbf{I}_{jk} | \varepsilon_{ik} \le t \le \lambda_{ik} - \tau_{ijk}\} \quad \forall\, j, k, t \tag{2.4}$$

$$\mathbf{J}_{ikt} = \{j \in \mathbf{J}_{ik} | \varepsilon_{ik} \le t \le \lambda_{ik} - \tau_{ijk}\} \quad \forall\, i, k \in \mathbf{K}_i, t \tag{2.5}$$

These sets are used to enforce release and due date constraints, and also reduce the total number of variables. We note that index t and ε_{ik} and λ_{ik} have time-interval units, based on δ, rather than actual time values (e.g. 30min, 1hr, etc.).

2.4 MODEL BASED ON EXPLICIT LABELING OF BATCHES

2.4.1 *Modeling approach*

A straight forward approach is to explicitly label the batches of an order and treat them separately. To achieve this, we introduce a new set $l \in \mathbf{L}$ to denote batches of a given order, and label each batch with a double index, (i, l). Although each batch is treated and scheduled individually, they are linked through the order satisfaction constraint, as well as the unit utilization constraint as will be shown in the next subsection. In addition to the previously introduced sets/indices, the following are used in the proposed model.

Sets/indices

$l \in \mathbf{L}_i$ Batches of order i

Variables

$X^L_{iljkt} \in \{0, 1\}$ =1 if batch $l \in \mathbf{L}_i$ on stage k starts in unit j at time point t

$X^S_{ijkt} \in \{0, 1\}$ =1 if any batch of order i on stage k is being stored in vessel j during time period t

$Y^L_{il} \in \{0, 1\}$ =1 if batch $l \in \mathbf{L}_i$ is selected

$S^L_{ilkt} \in \{0, 1\}$ =1 if material produced by batch $l \in \mathbf{L}_i$ on stage k is being stored during time period t

$N_i \in \mathbb{Z}_+$ Number of batches of order i

$B_{il} \in \mathbb{R}_+$ Batch size of $l \in \mathbf{L}_i$

2.4.2 *Mathematical Model*

If batch $l \in \mathbf{L}_i$ is selected (i.e. $Y^L_{il} = 1$), it is processed on every stage $k \in \mathbf{K}_i$.

$$\sum_{j \in \mathbf{J}_{ik}} \sum_{t \in \mathbf{T}_{ijk}} X^L_{iljkt} = Y^L_{il} \quad \forall\, i, l \in \mathbf{L}_i, k \in \mathbf{K}_i \tag{2.6}$$

A clique constraint for each unit and time point is written to prevent multiple batches being processed at the same time.

$$\sum_{k} \sum_{i \in \mathbf{I}_{jk}} \sum_{l \in \mathbf{L}_i} \sum_{t'=t,\ t' \in \mathbf{T}_{ijk}}^{t-\tau_{ijk}+1} X^L_{iljkt'} \leq 1 \quad \forall\, j \in \mathbf{J}^P, t \tag{2.7}$$

where $\mathbf{I}_{jk}$ denotes the set of orders on stage k that can be processed in unit j. The summation over $t^{'} \in \mathbf{T}_{ijk}$ is not required, but reduces the number of variables.

The sequencing of a batch between stages is implicitly enforced through the inventory balance constraint.

$$S^L_{ilk(t+1)} = S^L_{ilkt} + \sum_{j \in \mathbf{J}_{ik(t-\tau_{ijk})}} X^L_{iljk(t-\tau_{ijk})} - \sum_{j \in \mathbf{J}_{i(k+1)t}} X^L_{ilj(k+1)t} \quad \forall\, i, l \in \mathbf{L}_i, k \in \mathbf{K}_i, t \in \mathbf{T} \setminus \{\eta^F\} \tag{2.8}$$

which, since the inventory variable, S^L_{ilkt}, is defined as binary, also enforces the no batch mixing/splitting restriction. The integrality of the inventory variable can be relaxed (i.e. defined as $S^L_{ilkt} \in [0, 1]$), since X^L_{iljkt} is binary.

Similar to Eq. 2.7, we can write a clique constraint for the storage vessels to ensure that at most one compatible batch is stored during any time period.

$$\sum_{k} \sum_{i \in \mathbf{I}^S_{jk}} X^S_{ijkt} \leq 1 \quad \forall\, j \in \mathbf{J}^S, t \tag{2.9}$$

where $\mathbf{I}^S_{jk}$ is the set of orders that can be stored in vessel j after being processed on stage k.

In addition, when the inventory of a batch is nonzero, the batch is stored in one of the compatible vessels.

$$\sum_{l \in \mathbf{L}_i} S^L_{ilkt} = \sum_{j \in \mathbf{J}^S_{ik}} X^S_{ijkt} \quad \forall\, i, k \in \mathbf{K}_i \setminus \{\kappa_i^{LS}\}, t \in \{t | \varepsilon_{ik} \leq t \leq \lambda_{ik}\} \tag{2.10}$$

Bounds on the number of batches needed to meet an order can be determined based on the calculation of the effective maximum and minimum batch sizes of the order, taking into account the maximum and minimum unit capacities (Eqs. 2.11, 2.12). However, a more realistic minimum batch size (Eq. 2.13) is used to generate a tighter bound, since the processing times are independent of the batch size (Prasad and Maravelias, 2008) (see Figure 2.4). The bounds generated are shown in Eq. 2.14.

$$\beta_i^{eff,max} = \min_{k \in \mathbf{K}_i} (\max_{j \in \mathbf{J}_{ik}} \beta_j^{max}) \quad \forall i \tag{2.11}$$

$$\beta_i^{eff,min} = \max_{k \in \mathbf{K}_i} (\min_{j \in \mathbf{J}_{ik}} \beta_j^{min}) \quad \forall i \tag{2.12}$$

$$\hat{\beta}_i^{eff,min} = \min_{k \in \mathbf{K}_i} (\min_{j \in \mathbf{J}_{ik}} \beta_j^{max}) \quad \forall i \tag{2.13}$$

$$\left\lceil \xi_i / \beta_i^{eff,max} \right\rceil \leq N_i \leq \left\lceil \xi_i / \hat{\beta}_i^{eff,min} \right\rceil \quad \forall\ i \tag{2.14}$$

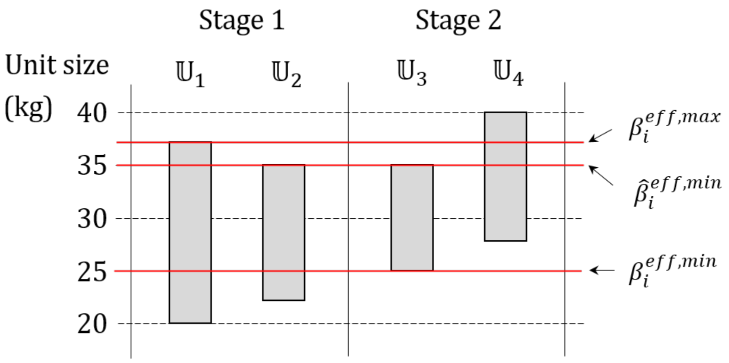


Figure 2.4: Calculation of effective minimum ($\beta_i^{eff,min}$) and maximum ($\beta_i^{eff,max}$) batch sizes for a given order with 2 stages and 2 units in each stage. A more realistic minimum batch size is given as $\hat{\beta}_i^{eff,min}$.

The total number of batches of order i, N_i, is equal to the total number of selected batches for the order.

$$\sum_{l \in \mathbf{L}_i} Y^L_{il} = N_i \quad \forall\ i \tag{2.15}$$

Eq. 2.16 constrains the size of the batch $l \in \mathbf{L}_i$ (i.e. B_{il}),

$$\beta_j^{min} \sum_{t \in \mathbf{T}_{ijk}} X^L_{iljkt} \leq B_{il} \leq \beta_j^{max} + (\max_{j' \in \mathbf{J}_{ik}}(\beta_{j'}^{max}) - \beta_j^{max})(1 - \sum_{t \in \mathbf{T}_{ijk}} X^L_{iljkt}) \tag{2.16}$$

$$\forall\, i, l \in \mathbf{L}_i, k \in \mathbf{K}_i, j \in \mathbf{J}_{ik}$$

The sum of batch sizes of all selected batches of order i has to meet order i.

$$\sum_{l \in \mathbf{L}_i} B_{il} \geq \xi_i \;\; \forall\, i \tag{2.17}$$

Furthermore, the symmetry in selecting batches can be removed by the following symmetry breaking constraints.

$$Y^L_{il} \geq Y^L_{i(l+1)} \;\; \forall\, i, l \in \mathbf{L}_i \tag{2.18}$$

$$B_{il} \geq B_{i(l+1)} \;\; \forall\, i, l \in \mathbf{L}_i \tag{2.19}$$

The model based on explicit labeling of batches consists of Eqs. 2.6-2.10, 2.11, 2.13-2.19, which will be referred to as M1 in the following.

2.4.3 *Objective functions*

We consider three objective functions: earliness, makespan and cost minimization. In earliness minimization, we minimize the summation of the time differences between the due date and the actual finish time of all batches.

$$\min \delta \sum_i \sum_{l \in \mathbf{L}_i} \left\{ \phi_i Y^L_{il} - \sum_{j \in \mathbf{J}_{i\kappa_i^{LS}}} \sum_{t \in \mathbf{T}_{ij\kappa_i^{LS}}} \left(t + \tau_{ijk} \right) X^L_{ilj\kappa_i^{LS}t} \right\} \tag{2.20}$$

In makespan minimization, we ensure that $MS \in \mathbb{R}_+$ is greater than or equal to the finish time of all the batches in their last stages.

$$\min MS \tag{2.21}$$

$$MS \geq \sum_{j \in \mathbf{J}_{i\kappa_i^{LS}}} \sum_{t \in \mathbf{T}_{ij\kappa_i^{LS}}} \delta(t + \tau_{ij\kappa_i^{LS}}) X^L_{ilj\kappa_i^{LS}t} \;\; \forall\, i, l \in \mathbf{L}_i \tag{2.22}$$

For cost minimization, we introduce α_{ijk}, the processing cost of batch of order i on stage k being processed in unit j; α^U_{rt}, the cost of utility r during time period t; and α^{Inv}_{ijk}, the inventory cost of batch of order i on stage k stored in vessel j. The objective function is,

$$\min \sum_i \sum_{l \in \mathbf{L}_i} \sum_{k \in \mathbf{K}_i} \sum_{j \in \mathbf{J}_{ik}} \sum_{t \in \mathbf{T}_{ijk}} \alpha_{ijk} X^L_{iljkt} + \sum_r \sum_t \alpha^U_{rt} U_{rt} + \sum_i \sum_{k \in \mathbf{K}_i} \sum_{j \in \mathbf{J}^S_{ik}} \alpha^{Inv}_{ijk} X^S_{ijkt} \tag{2.23}$$

where $r \in \mathbf{R}$ denotes the set of utilities, and U_{rt} denotes the amount of utility r consumed during time period t (see Section 2.6).

2.5 MODEL BASED ON BATCH SIZE INTERVALS

2.5.1 *Modeling approach*

The second model is based on the concept of batch size intervals. We introduce the following:

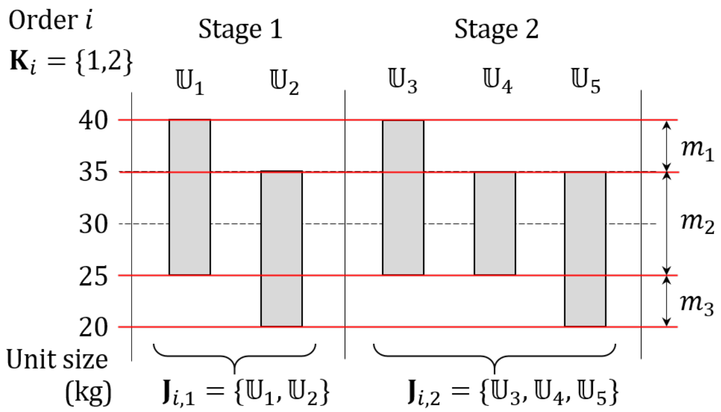
Sets/indices

$l \in \mathbf{L}_i$	Batches of order i
$m \in \mathbf{M}$	Batch size intervals
$\mathbf{M}_i$	Batch size intervals for order i
$\mathbf{M}_{ij}$	Batch size intervals for order i when processed in unit j
$\mathbf{J}_{imk}$	Process units that can process order i on stage k assigned to batch size interval m

Parameters

$\zeta^{min}_{im} / \zeta^{max}_{im}$	Lower/upper endpoint of batch size interval m of order i

A set of batch size intervals for a given order i (i.e. $\mathbf{M}_i$) is a set of intervals that partition the entire batch size domain (from effective minimum to maximum batch sizes), such that the routing of units that corresponds to each interval is different from adjacent intervals. To illustrate, we consider an order i, which has to be processed on two stages (see Figure 2.5). The effective maximum and minimum batch sizes are 40kg and 20kg, respectively. We can find three intervals that correspond to different unit routings. Hence, the set of batch size intervals for order i is given as $\mathbf{M}_i = \{m_1, m_2, m_3\}$, where each interval index, $m \in \mathbf{M}_i$, corresponds to a batch size interval $[\zeta_{im}^{min}, \zeta_{im}^{max}]$ (e.g. $\zeta_{im_1}^{min} = 35kg$, $\zeta_{im_1}^{max} = 40kg$).



Interval	Unit routing
m_1	$\{\mathbb{U}_1\} \rightarrow \{\mathbb{U}_3\}$
m_2	$\{\mathbb{U}_1, \mathbb{U}_2\} \rightarrow \{\mathbb{U}_3, \mathbb{U}_4, \mathbb{U}_5\}$
m_3	$\{\mathbb{U}_2\} \rightarrow \{\mathbb{U}_5\}$
$\mathbf{M}_i = \{m_1, m_2, m_3\}$	

Figure 2.5: An illustration of batch size intervals of order i, processed on two stages consisting of 2 and 3 units, respectively. $\mathbf{M}_i = \{m_1, m_2, m_3\}$ defines the set of batch size intervals for this order, where each batch size interval correspond to different unit routings.

Given an order i, set $\mathbf{M}_i$ and the corresponding intervals are unique. For example, if we denote interval indices that correspond to intervals 25-30kg and 30-35kg as m_2^1 and m_2^2, respectively, intervals in set $\{m_1, m_2^1, m_2^2, m_3\}$ span the full domain, but m_2^1 and m_2^2 have identical unit routings. Similarly, if we denote the interval index that corresponds to interval 25-40kg as m_{1+2}, the intervals in set $\{m_{1+2}, m_3\}$ span the full domain, but the unit routing of m_{1+2} cannot be defined. Thus, both sets violate the definition of $\mathbf{M}_i$.

Lastly, $\mathbf{M}_{ij}$ is the subset of batch size intervals for order i whose corresponding routing include $j \in \mathbf{J}_i$ (e.g. $\mathbf{M}_{i,\mathbb{U}_2} = \{m_2, m_3\}$); and $\mathbf{J}_{imk}$ is the subset of units $j \in \mathbf{J}_{ik}$ that can process a batch assigned to batch size interval $m \in \mathbf{M}_i$ (e.g. $\mathbf{J}_{i,m_1,1} = \{\mathbb{U}_1\}$).

Order satisfaction can be modeled through the selection and assignment of batches to intervals $m \in \mathbf{M}_i$, such that the total amount processed meets the due amount. Once a batch is assigned to an interval, scheduling of this batch is straight forward, since the unit routing for the batch is uniquely defined.

The following are the variables used in the proposed model.

Variables

$X^M_{imjkt} \in \{0,1\}$ =1 if a batch of order i on stage k with batch size interval m starts in unit j at time point t

$X^S_{ijkt} \in \{0,1\}$ =1 if any batch of order i on stage k is being stored in vessel j during time period t

$Y^M_{ilm} \in \{0,1\}$ =1 if batch $l \in \mathbf{L}_i$ assigned to interval m is selected

$S^M_{imkt} \in \mathbb{Z}_+$ Inventory of batch of order i assigned to interval m on stage k during time period t

$N_i \in \mathbb{Z}_+$ Number of batches of order i

$N^M_{im} \in \mathbb{Z}_+$ Number of batches of order i assigned to interval m

$B_{il} \in \mathbb{R}_+$ Batch size of $l \in \mathbf{L}_i$

$B^M_{ilm} \in \mathbb{R}_+$ Batch size of $l \in \mathbf{L}_i$ assigned to interval m

2.5.2 *Identification of batch size intervals*

Identifying the set of batch size intervals for every order is not a trivial task, so we propose an algorithm to do so (see Figure 2.6). It is based on the idea that interval endpoints are equal to either the maximum or minimum capacity of a processing unit that can process the order. The algorithm iterates through the set of possible interval endpoints, from the effective maximum batch size to the effective minimum batch size. To do this, we introduce sets, $\mathbf{J}^{max}_i$ and $\mathbf{J}^{min}_i$, which initialize as $\mathbf{J}_i$, but are modified throughout the algorithm.

The algorithm is initialized by setting the upper endpoint of the first interval as the effective maximum batch size. In the first step (i.e. 1a/1b), the unit with the capacity that has been selected as the upper endpoint of the current batch size interval is removed from sets $\mathbf{J}^{max}_i$ and $\mathbf{J}^{min}_i$. In the second step, the lower endpoint of the current interval is set as the largest value among the maximum capacities of units in $\mathbf{J}^{max}_i$ and minimum capacities of units in $\mathbf{J}^{min}_i$. The third step (i.e. 3a/3b) is then performed to move on to the next interval or try a new upper endpoint for the current interval, according to whether the current interval is a valid interval or

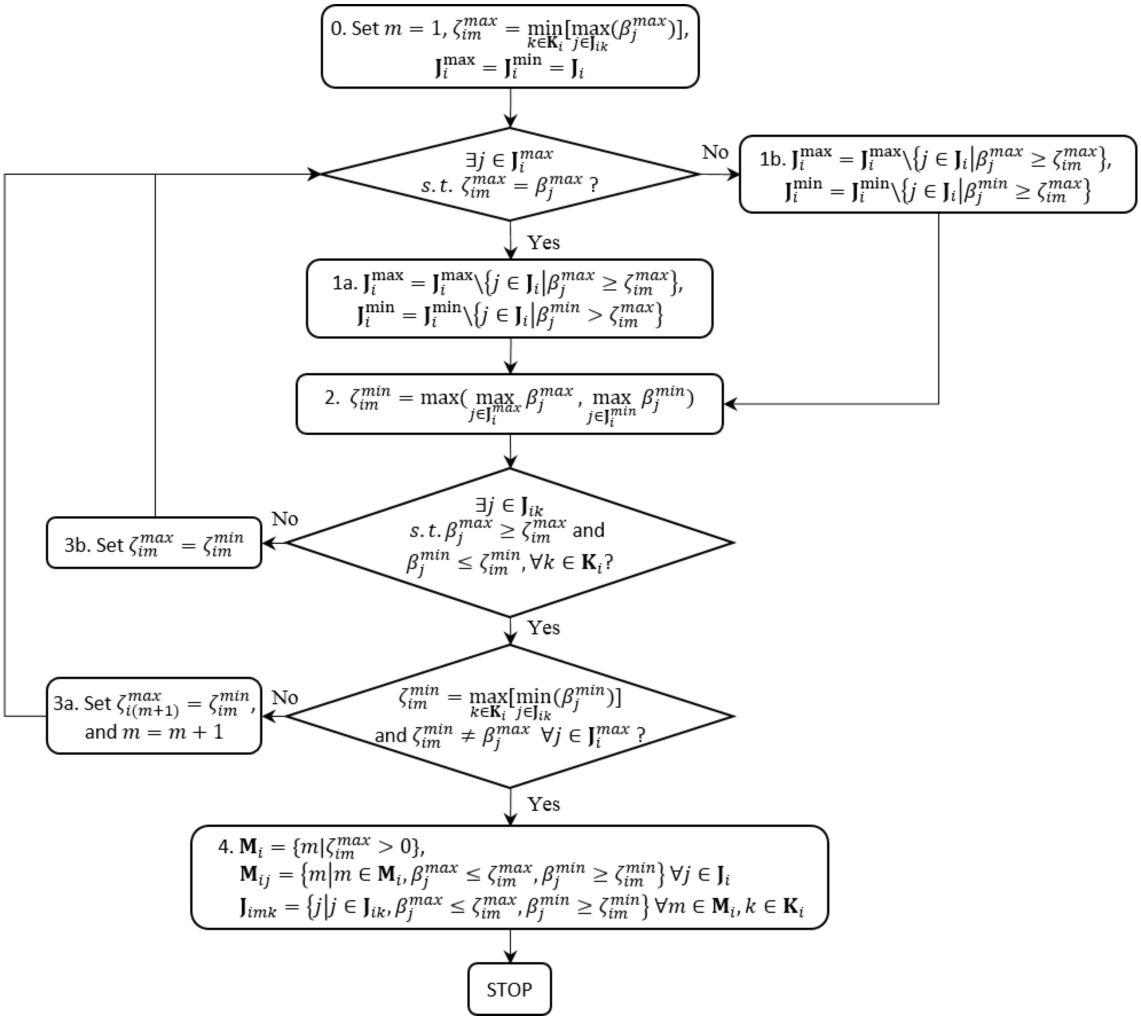

0. Set $m = 1$, $\zeta_{im}^{max} = \min_{k\in\mathbf{K}_i}[\max_{j\in\mathbf{J}_{ik}}(\beta_j^{max})]$,
$\mathbf{J}_i^{\max} = \mathbf{J}_i^{\min} = \mathbf{J}_i$
$\exists j \in \mathbf{J}_i^{max}$
$s.t.\ \zeta_{im}^{max} = \beta_j^{max}$?
No
1b. $\mathbf{J}_i^{\max} = \mathbf{J}_i^{\max} \backslash \{j \in \mathbf{J}_i | \beta_j^{max} \geq \zeta_{im}^{max}\}$,
$\mathbf{J}_i^{\min} = \mathbf{J}_i^{\min} \backslash \{j \in \mathbf{J}_i | \beta_j^{min} \geq \zeta_{im}^{max}\}$
Yes
1a. $\mathbf{J}_i^{\max} = \mathbf{J}_i^{\max} \backslash \{j \in \mathbf{J}_i | \beta_j^{max} \geq \zeta_{im}^{max}\}$,
$\mathbf{J}_i^{\min} = \mathbf{J}_i^{\min} \backslash \{j \in \mathbf{J}_i | \beta_j^{min} > \zeta_{im}^{max}\}$
2. $\zeta_{im}^{min} = \max(\max_{j\in\mathbf{J}_i^{max}} \beta_j^{max}, \max_{j\in\mathbf{J}_i^{min}} \beta_j^{min})$
3b. Set $\zeta_{im}^{max} = \zeta_{im}^{min}$
No
$\exists j \in \mathbf{J}_{ik}$
$s.t.\ \beta_j^{max} \geq \zeta_{im}^{max}$ and
$\beta_j^{min} \leq \zeta_{im}^{min}, \forall k \in \mathbf{K}_i$?
Yes
3a. Set $\zeta_{i(m+1)}^{max} = \zeta_{im}^{min}$,
and $m = m + 1$
No
$\zeta_{im}^{min} = \max_{k\in\mathbf{K}_i}[\min_{j\in\mathbf{J}_{ik}}(\beta_j^{min})]$
and $\zeta_{im}^{min} \neq \beta_j^{max}\ \forall j \in \mathbf{J}_i^{max}$?
Yes
4. $\mathbf{M}_i = \{m | \zeta_{im}^{max} > 0\}$,
$\mathbf{M}_{ij} = \{m | m \in \mathbf{M}_i, \beta_j^{max} \leq \zeta_{im}^{max}, \beta_j^{min} \geq \zeta_{im}^{min}\}\ \forall j \in \mathbf{J}_i$
$\mathbf{J}_{imk} = \{j | j \in \mathbf{J}_{ik}, \beta_j^{max} \leq \zeta_{im}^{max}, \beta_j^{min} \geq \zeta_{im}^{min}\}\ \forall m \in \mathbf{M}_i, k \in \mathbf{K}_i$
STOP

Figure 2.6: Algorithm to obtain sets $\mathbf{M}_i, \mathbf{M}_{ij}, \mathbf{J}_{imk}$, and parameters ζ_{im}^{min}, ζ_{im}^{max} for a given order i. The full information on batch size intervals of a multipurpose facility can be obtained by performing this algorithm for all $i \in \mathbf{I}$.

not. If the termination criterion is met (i.e. the lower endpoint of the last interval equals to the effective minimum batch size), the required sets and parameters are generated at the fourth step, based on the obtained intervals.

To illustrate, the intervals for the facility shown in Figure 2.5 are identified by initializing $\zeta_{im_1}^{max} = 40kg$, and $\mathbf{J}_i^{max} = \mathbf{J}_i^{min} = \{\mathbb{U}_1, ..., \mathbb{U}_5\}$. In the first step, units with maximum and minimum capacities of $40kg$ ($= \zeta_{im_1}^{max}$) are removed from $\mathbf{J}_i^{max}$ and $\mathbf{J}_i^{min}$, respectively: $\mathbf{J}_i^{max} = \{\mathbb{U}_2, \mathbb{U}_4, \mathbb{U}_5\}$, $\mathbf{J}_i^{min} = \{\mathbb{U}_1, ..., \mathbb{U}_5\}$. In the second step, lower endpoint of interval m_1 is selected as the maximum among the candidate capacities: $\zeta_{im_1}^{min} = 35kg$. This is a valid interval, therefore the algorithm proceeds to the next interval, repeating until it obtains all the intervals. Finally, we generate the required sets and parameters based on the identified intervals (e.g. $\mathbf{M}_i = \{m_1, m_2, m_3\}$).

2.5.3 *Mathematical Model*

The number of batches processed on each stage assigned to a given batch size interval is equal to the number of batches of order i assigned to the interval.

$$\sum_{j \in \mathbf{J}_{imk}} \sum_{t \in \mathbf{T}_{ijk}} X^M_{imjkt} = N^M_{im} \quad \forall\, i, m \in \mathbf{M}_i, k \in \mathbf{K}_i \tag{2.24}$$

Unit utilization is enforced through clique constraints.

$$\sum_k \sum_{i \in \mathbf{I}_{jk}} \sum_{m \in \mathbf{M}_{ij}} \sum_{t'=t,\ t' \in \mathbf{T}_{ijk}}^{t-\tau_{ijk}+1} X^M_{imjkt'} \leq 1 \quad \forall\, j \in \mathbf{J}^P, t \tag{2.25}$$

Similar to the previous model, batch sequencing is implicitly enforced through the inventory balance constraint. However, the major difference compared to Eq. 2.8 is that the inventory variable, S^M_{imkt} is defined as a non-negative integer variable as opposed to binary.

$$S^M_{imk(t+1)} = S^M_{imkt} + \sum_{j \in \mathbf{J}_{ik(t-\tau_{ijk})} \cap \mathbf{J}_{imk}} X^M_{imjk(t-\tau_{ijk})} - \sum_{j \in \mathbf{J}_{i(k+1)t} \cap \mathbf{J}_{im(k+1)}} X^M_{imj(k+1)t}$$
$$\forall\, i, m \in \mathbf{M}_i, k \in \mathbf{K}_i, t \in \mathbf{T} \setminus \{\eta^F\} \tag{2.26}$$

In the previous model, the identity of each batch is strictly determined via (i, l), and variables, X^L_{iljkt} and S^L_{ilkt}, could only be activated by the corresponding batch. However, in this model, since we allow multiple batches to be assigned to the same interval, multiple batches can activate X^M_{imjkt} and S^M_{imkt}, possibly resulting in an inventory of more than one batch during a given time period. S^M_{imkt} can also be defined as a non-negative continuous variable as discussed previously.

Eq. 2.27 ensures that the number of batches being stored in compatible vessels is equal to the inventory of the batches of order i.

$$\sum_{m \in \mathbf{M}_i} S^M_{imkt} = \sum_{j \in \mathbf{J}^S_{ik}} X^S_{ijkt} \quad \forall\, i, k \in \mathbf{K}_i \setminus \{\kappa^{LS}_i\}, t \in \{t | \varepsilon_{ik} \leq t \leq \lambda_{ik}\} \tag{2.27}$$

We bound the total number of batches using Eq. 2.14. In addition, for a given order, the total number of batches processed is equal to number of batches assigned to each interval.

$$\sum_{l \in \mathbf{L}_i} Y^M_{ilm} = N^M_{im} \ \ \forall\, i, m \in \mathbf{M}_i \tag{2.28}$$

$$\sum_{m \in \mathbf{M}_i} N^M_{im} = N_i \ \ \forall\, i \tag{2.29}$$

Furthermore, a batch $l \in \mathbf{L}_i$ is assigned to at most one interval.

$$\sum_{m \in \mathbf{M}_i} Y^M_{ilm} \leq 1 \ \ \forall\, i, l \in \mathbf{L}_i \tag{2.30}$$

If a batch is assigned to an interval, then B^M_{ilm} is bounded as follows.

$$\zeta^{min}_{im} Y^M_{ilm} \leq B^M_{ilm} \leq \zeta^{max}_{im} Y^M_{ilm} \ \ \forall\, i, l \in \mathbf{L}_i, m \in \mathbf{M}_i \tag{2.31}$$

Since a batch can only be assigned to one interval (i.e. Eq. 2.30), the following holds.

$$\sum_{m \in \mathbf{M}_i} B^M_{ilm} = B_{il} \ \ \forall\, i, l \in \mathbf{L}_i \tag{2.32}$$

The order satisfaction and the symmetry breaking constraints are identical to the one of previous model (Eq. 2.17-2.19).

The following tightening constraint ensures order satisfaction.

$$\sum_{l \in \mathbf{L}_i} \sum_{m \in \mathbf{M}_i} \zeta^{max}_{im} Y^M_{ilm} \geq \xi_i \ \ \forall\, i \tag{2.33}$$

The model based on batch size intervals consists of Eqs. 2.9, 2.11, 2.13, 2.14, 2.17-2.19, 2.24-2.33, and will be referred to as M2.

2.5.4 *Objective functions*

The earliness and cost minimization objective functions are similar to the previous model (Eq. 2.20, 2.23).

$$\min \delta \sum_{i} \sum_{m\in\mathbf{M}_i} \left\{ \phi_i \sum_{l\in\mathbf{L}_i} Y^M_{ilm} - \sum_{j\in\mathbf{J}_{im\kappa_i^{LS}}} \sum_{t\in\mathbf{T}_{ij\kappa_i^{LS}}} \left(t+\tau_{ijk}\right) X^M_{imj\kappa_i^{LS}t} \right\} \tag{2.34}$$

$$\min \sum_{i} \sum_{m\in\mathbf{M}_i} \sum_{k\in\mathbf{K}_i} \sum_{j\in\mathbf{J}_{imk}} \sum_{t\in\mathbf{T}_{ijk}} \alpha_{ijk} X^M_{imjkt} + \sum_{r}\sum_{t} \alpha^{Ut}_{rt} U_{rt} + \sum_{i} \sum_{k\in\mathbf{K}_i} \sum_{j\in\mathbf{J}^S_{ik}} \alpha^{Inv}_{ijk} X^S_{ijkt} \tag{2.35}$$

As discussed previously, the main decision variable, X^M_{imjkt}, can be activated by multiple batches of a given order, which makes the modeling of makespan difficult, because the finish time of each batch cannot be explicitly expressed using X^M_{imjkt} variable. To address this, we disaggregate the Eq. 2.22 in time and write it for every time point.

$$MS \geq \sum_{j\in\mathbf{J}_{im\kappa_i^{LS}}} \delta(t+\tau_{ij\kappa_i^{LS}}) X^M_{imj\kappa_i^{LS}t} \quad \forall i, m \in \mathbf{M}_i, t \tag{2.36}$$

This constraint, however, provides a weak relaxation, since most of the RHS terms reduce to zero in any feasible solution. To reduce the number of constraints and improve the relaxation, we use partial disaggregation in time,

$$MS \geq \sum_{t'=\varepsilon_{i\kappa_i^{LS}}+(\theta-1)\tau_{ij\kappa_i^{LS}},\ t'\in\mathbf{T}_{ij\kappa_i^{LS}}}^{\varepsilon_{i\kappa_i^{LS}}+\theta\tau_{ij\kappa_i^{LS}}-1} \delta\left(t'+\tau_{ij\kappa_i^{LS}}\right) X^M_{imj\kappa_i^{LS}t'} \tag{2.37}$$

$$\forall i, m \in \mathbf{M}_i, j \in \mathbf{J}_{im\kappa_i^{LS}}, \theta = 1, 2, ..., \Theta_{ij}$$

The summation over t' is performed over a smaller range than the processing time of the batch assigned to the unit, which ensures that at most one X^M_{imjkt} is nonzero within this range. The constraint is written for $\theta = 1, 2, ..., \Theta_{ij}$, where Θ_{ij} is the total number of time ranges available for order i when processed in unit $j \in \mathbf{J}_{i\kappa_i^{LS}}$, (i.e. $\Theta_{ij} := \left\lceil \dfrac{\lambda_{i\kappa_i^{LS}} - \tau_{ij\kappa_i^{LS}} - \varepsilon_{i\kappa_i^{LS}}}{\tau_{ij\kappa_i^{LS}}} \right\rceil$). An illustration of partial disaggregation in time of the makespan constraint is shown in Figure 2.7.

For difficult makespan minimization problems, we can provide a tighter lower bound of MS

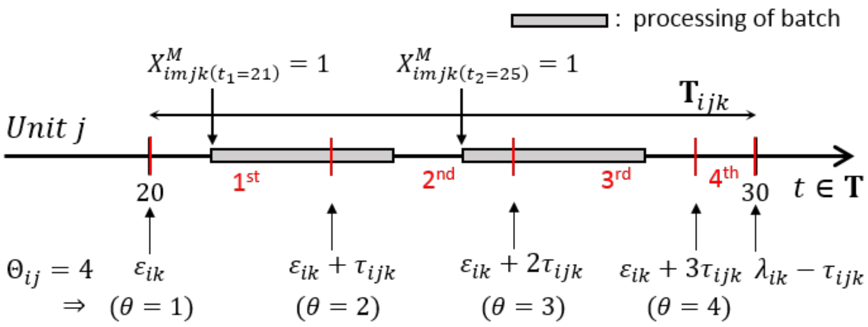


Figure 2.7: An example of partial time discretization of the makespan constraint. Given i, m, j, and $k(= \kappa_i^{LS})$ with $\tau_{ijk} = 3$, $\varepsilon_{ik} = 20$, and $\lambda_{ik} = 33$, total of four time ranges are available (i.e. $\Theta_{ij} = 4$). Note that at most one X^M_{imjkt} can take value of 1 within a single range.

with the minimum total processing time divided by the total number of processing units. This constraint is valid in both models.

$$MS \geq \frac{1}{|\mathbf{J}^P|} \sum_i \sum_{k \in \mathbf{K}_i} \min_{j \in \mathbf{J}_{ik}} \tau_{ijk} \tag{2.38}$$

2.6 EXTENSIONS AND REMARKS

2.6.1 *Limited shared resources*

The use of a common time-grid allows for monitoring of total consumption rate of shared resources (e.g. steam, manpower, etc.), enables linear modeling of time-varying availability and cost of such resources.

We introduce parameter ρ_{ijkt} to the denote fixed amount of utility r required to process a batch of order i on stage $k \in \mathbf{K}_i$ in unit $j \in \mathbf{J}_{ik}$. The availability and cost of utility r during time period $t \in \mathbf{T} \setminus \{0\}$ are given as ψ_{rt} and α^U_{rt}, respectively. We introduce $U_{rt} \in \mathbb{R}_+$, to denote the total amount of utility r consumed during period t. The utility constraints for models M1 and M2 are Eq. 2.39 and Eq. 2.40, respectively, while Eq. 2.41 bounds the total amount of utility r

being consumed during time period t for both models.

$$U_{r(t+1)} = U_{rt} + \sum_{i} \sum_{l\in\mathbf{L}_i} \sum_{k\in\mathbf{K}_i} \sum_{j\in\mathbf{J}_{ikt}} \rho_{ijkr} X^{L}_{iljkt} - \sum_{i} \sum_{l\in\mathbf{L}_i} \sum_{k\in\mathbf{K}_i} \sum_{j\in\mathbf{J}_{ik(t-\tau_{ijk})}} \rho_{ijkr} X^{L}_{iljk(t-\tau_{ijk})} \qquad \forall r, t \in \mathbf{T} \setminus \{\eta^F\} \tag{2.39}$$

$$U_{r(t+1)} = U_{rt} + \sum_{i} \sum_{m\in\mathbf{M}_i} \sum_{k\in\mathbf{K}_i} \sum_{j\in\mathbf{J}_{ikt}} \rho_{ijkr} X^{M}_{imjkt} - \sum_{i} \sum_{m\in\mathbf{M}_i} \sum_{k\in\mathbf{K}_i} \sum_{j\in\mathbf{J}_{ik(t-\tau_{ijk})}} \rho_{ijkr} X^{M}_{imjk(t-\tau_{ijk})} \qquad \forall r, t \in \mathbf{T} \setminus \{\eta^F\} \tag{2.40}$$

$$U_{rt} \leq \psi_{rt} \quad \forall r, t \in \mathbf{T} \setminus \{\eta^0\} \tag{2.41}$$

2.6.2 *Storage vessel capacities*

So far we have assumed that the capacity of the storage vessels is large enough to store any compatible batch. We propose an extension to model facilities with limited vessel capacities. Let the minimum and maximum storage capacity of vessel $j \in \mathbf{J}^S$ be denoted as β^{min}_j and β^{max}_j, respectively. For M1, since the storage of a batch in a vessel is now dependent on the batch size and the vessel capacity, we introduce $X^S_{iljkt} \in \{0, 1\}$, to denote storage of batch $l \in \mathbf{L}_i$ on stage k in vessel j during time period t. Then, if a batch is stored in a vessel, its batch size has to be within the vessel capacity.

$$\beta^{min}_j X^S_{iljkt} \leq B_{il} \leq \beta^{max}_j + (\max_{j'\in\mathbf{J}^S_{ik}} \beta^{max}_{j'} - \beta^{max}_j)(1 - X^S_{iljkt}) \qquad \forall\, i, l \in \mathbf{L}_i, j \in \mathbf{J}^S_{ik}, k \in \mathbf{K}_i, t \in \mathbf{T}^S_{ik} \tag{2.42}$$

Furthermore, Eq. 2.9 and Eq. 2.10 are rewritten in terms of the newly introduced storage variable.

$$\sum_{k} \sum_{i\in\mathbf{I}^S_{jk}} \sum_{l\in\mathbf{L}_i} X^S_{iljkt} \leq 1 \quad \forall\, j \in \mathbf{J}^S, t \tag{2.43}$$

$$S^L_{ilkt} = \sum_{j\in\mathbf{J}^S_{ik}} X^S_{iljkt} \quad \forall\, i, l \in \mathbf{L}_i, k \in \mathbf{K}_i \setminus \{\kappa^{LS}_i\}, t \in \mathbf{T}^S_{ik} \tag{2.44}$$

where $\mathbf{T}^S_{ik}$ is the set of feasible time periods in which order $i \in \mathbf{I}$ on stage $k \in \mathbf{K}_i$ can be stored in any compatible vessel.

$$\mathbf{T}_{ik}^{S} = \{t \in \mathbf{T} | \varepsilon_{ik} + \min_{j \in \mathbf{J}_{ik}} \tau_{ijk} + 1 \leq t \leq \lambda_{ik}\} \;\; \forall\, i, k \in \mathbf{K}_i \tag{2.45}$$

In M2, the sets of batch size intervals were generated based on the routing of processing units, not including the vessels. However, with limited vessel capacities, routing of vessels (in addition to units) has to be considered. Therefore, the interval generation algorithm is modified to include vessel capacities as well (see Appendix A.1). Furthermore, we introduce $X_{imjkt}^{S} \in \{0,1\}$, to denote storage of a batch of order i assigned to batch size interval m on stage k in vessel j during time period t and re-write Eq. 2.9 and Eq. 2.27 as follows.

$$\sum_{k \in \mathbf{K}} \sum_{i \in \mathbf{I}_{jk}^{S}} \sum_{m \in \mathbf{M}_{ij}} X_{imjkt}^{S} \leq 1 \;\; \forall\, j \in \mathbf{J}^{S}, t \tag{2.46}$$

$$S_{imkt}^{M} = \sum_{j \in \mathbf{J}_{imk}^{S}} X_{imjkt}^{S} \quad \forall\, i, m \in \mathbf{M}_i, k \in \mathbf{K}_i \setminus \{\kappa_i^{LS}\}, t \in \{t | \varepsilon_{ik} \leq t \leq \lambda_{ik}\} \tag{2.47}$$

2.6.3 *Stage-dependent batch sizes*

We consider the case where the batch size changes as it moves through the production stages. This case may appear in industries such as fine chemicals and pharmaceuticals, where a large batch of raw material is processed through multiple stages to produce a smaller batch of refined product. We define the batch size of a given stage as the inlet batch size, and we choose the size of the final product as the basis. In other words, we define B_{il} as the size of the final product, and the unit capacities on each stage are normalized with respect to it. To do this, we introduce a new parameter, χ_{ik}, to denote the relative size of the final product with respect to the batch size on stage k. If σ_{ik} is the ratio between the batch size of order i prior and after being processed on stage k, then χ_{ik} is defined as follows (see Figure 2.8).

$$\chi_{ik} = \prod_{k' \geq k}^{\kappa_i^{LS}} \sigma_{ik'} \;\; \forall\, i, k \in \mathbf{K}_i \tag{2.48}$$

First, we calculate the effective maximum/minimum and realistic minimum batch sizes (Eqs.

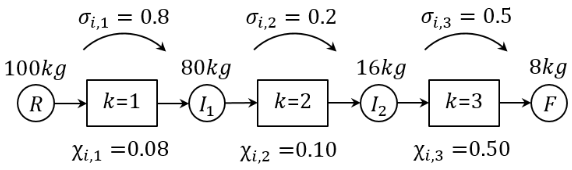


Figure 2.8: Illustration of a process with stage-dependent batch sizes.

2.11-2.13) using the normalized unit capacities as follows.

$$\beta_i^{eff,max} = \min_{k\in\mathbf{K}_i}(\max_{j\in\mathbf{J}_{ik}} \chi_{ik}\beta_j^{max}) \ \ \forall i \tag{2.49}$$

$$\beta_i^{eff,min} = \max_{k\in\mathbf{K}_i}(\min_{j\in\mathbf{J}_{ik}} \chi_{ik}\beta_j^{min}) \ \ \forall i \tag{2.50}$$

$$\hat{\beta}_i^{eff,min} = \min_{k\in\mathbf{K}_i}(\min_{j\in\mathbf{J}_{ik}} \chi_{ik}\beta_j^{max}) \ \ \forall i \tag{2.51}$$

The batch size constraints for M1 (i.e. Eqs. 2.16, 2.42) are rewritten as follows.

$$\chi_{ik}\beta_j^{min}\sum_{t\in\mathbf{T}_{ijk}} X^L_{iljkt} \le B_{il} \le \chi_{ik}\beta_j^{max} + (\max_{j'\in\mathbf{J}_{ik}} \chi_{ik}\beta_{j'}^{max} - \chi_{ik}\beta_j^{max})(1 - \sum_{t\in\mathbf{T}_{ijk}} X^L_{iljkt}) \\ \forall\ i, l \in \mathbf{L}_i, k \in \mathbf{K}_i, j \in \mathbf{J}_{ik} \tag{2.52}$$

$$\chi_{ik}\beta_j^{min} X^S_{iljkt} \le B_{il} \le \chi_{ik}\beta_j^{max} + (\max_{j'\in\mathbf{J}^S_{ik}} \chi_{ik}\beta_{j'}^{max} - \chi_{ik}\beta_j^{max})(1 - X^S_{iljkt}) \\ \forall\ i, l \in \mathbf{L}_i, j \in \mathbf{J}^S_{ik}, k \in \mathbf{K}_i, t \in \mathbf{T}^S_{ik} \tag{2.53}$$

For M2, the batch size interval algorithm has to be modified to account for the stage-dependent batch sizes (see Figure 2.9). The algorithm now iterates through the normalized unit capacities using sets $\mathbf{P}_i^{max}$ and $\mathbf{P}_i^{min}$. Furthermore, we generate the set of batch size intervals for order i on stage k when processed in unit j, denoted as $\mathbf{M}_{ijk}$.

Finally, the unit utilization constraint (i.e. Eq. 2.25) is modified as follows,

$$\sum_{k}\sum_{i\in\mathbf{I}_{jk}}\sum_{m\in\mathbf{M}_{ijk}}\sum_{t'=t,\ t'\in\mathbf{T}_{ijk}}^{t-\tau_{ijk}+1} X^M_{imjkt'} \le 1 \ \ \forall\ j \in \mathbf{J}^P, t \tag{2.54}$$

In cases where storage capacities are considered, the algorithm is further modified (see Ap-

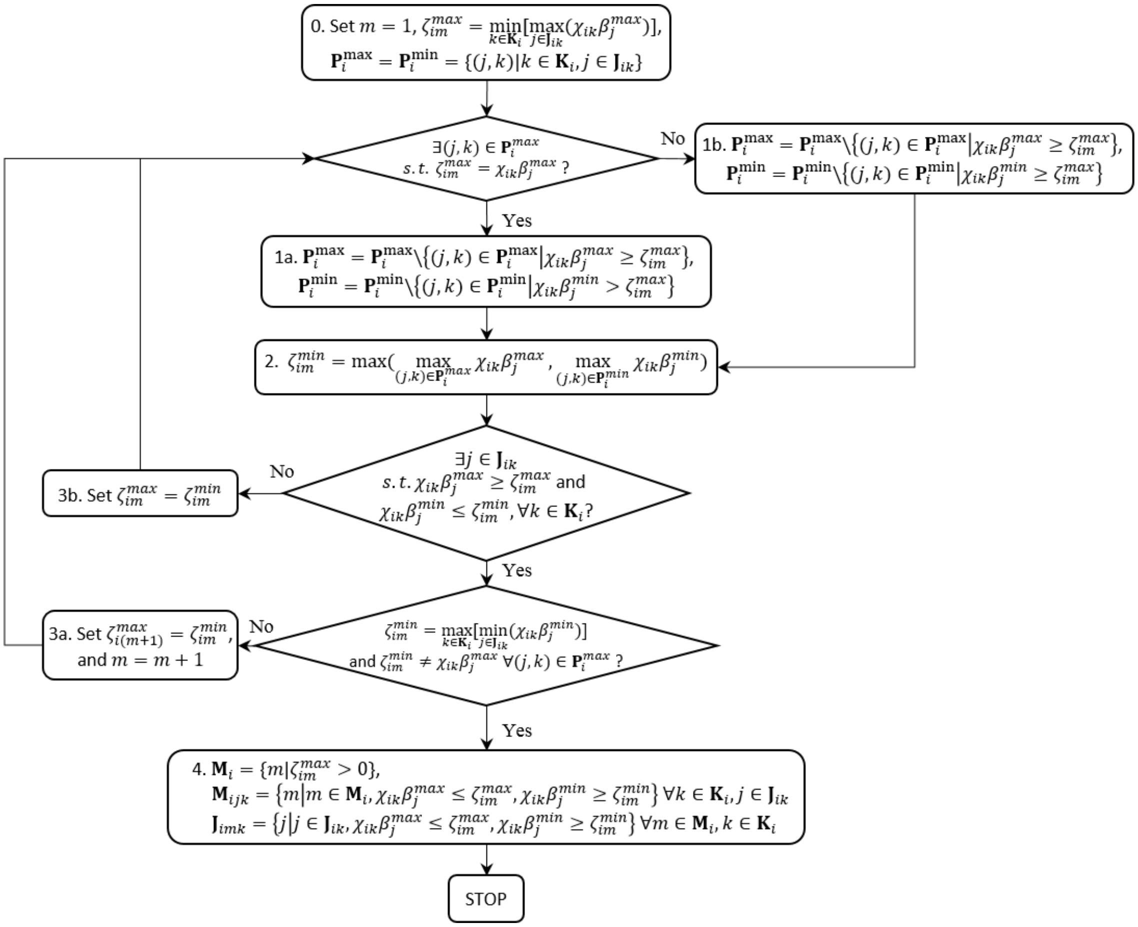

0. Set $m = 1$, $\zeta_{im}^{max} = \min_{k\in\mathbf{K}_i}[\max_{j\in\mathbf{J}_{ik}}(\chi_{ik}\beta_j^{max})]$,
$\mathbf{P}_i^{\max} = \mathbf{P}_i^{\min} = \{(j,k)|k \in \mathbf{K}_i, j \in \mathbf{J}_{ik}\}$
$\exists (j,k) \in \mathbf{P}_i^{max}$ $s.t.\ \zeta_{im}^{max} = \chi_{ik}\beta_j^{max}$?
No
1b. $\mathbf{P}_i^{\max} = \mathbf{P}_i^{\max}\backslash\{(j,k) \in \mathbf{P}_i^{\max}|\chi_{ik}\beta_j^{max} \geq \zeta_{im}^{max}\}$,
$\mathbf{P}_i^{\min} = \mathbf{P}_i^{\min}\backslash\{(j,k) \in \mathbf{P}_i^{\min}|\chi_{ik}\beta_j^{min} \geq \zeta_{im}^{max}\}$
Yes
1a. $\mathbf{P}_i^{\max} = \mathbf{P}_i^{\max}\backslash\{(j,k) \in \mathbf{P}_i^{\max}|\chi_{ik}\beta_j^{max} \geq \zeta_{im}^{max}\}$,
$\mathbf{P}_i^{\min} = \mathbf{P}_i^{\min}\backslash\{(j,k) \in \mathbf{P}_i^{\min}|\chi_{ik}\beta_j^{min} > \zeta_{im}^{max}\}$
2. $\zeta_{im}^{min} = \max(\max_{(j,k)\in\mathbf{P}_i^{max}} \chi_{ik}\beta_j^{max}, \max_{(j,k)\in\mathbf{P}_i^{min}} \chi_{ik}\beta_j^{min})$
$\exists j \in \mathbf{J}_{ik}$
$s.t.\ \chi_{ik}\beta_j^{max} \geq \zeta_{im}^{max}$ and
$\chi_{ik}\beta_j^{min} \leq \zeta_{im}^{min}, \forall k \in \mathbf{K}_i$?
No
3b. Set $\zeta_{im}^{max} = \zeta_{im}^{min}$
Yes
$\zeta_{im}^{min} = \max_{k\in\mathbf{K}_i}[\min_{j\in\mathbf{J}_{ik}}(\chi_{ik}\beta_j^{min})]$
and $\zeta_{im}^{min} \neq \chi_{ik}\beta_j^{max}\ \forall (j,k) \in \mathbf{P}_i^{max}$?
No
3a. Set $\zeta_{i(m+1)}^{max} = \zeta_{im}^{min}$,
and $m = m + 1$
Yes
4. $\mathbf{M}_i = \{m|\zeta_{im}^{max} > 0\}$,
$\mathbf{M}_{ijk} = \{m|m \in \mathbf{M}_i, \chi_{ik}\beta_j^{max} \leq \zeta_{im}^{max}, \chi_{ik}\beta_j^{min} \geq \zeta_{im}^{min}\}\ \forall k \in \mathbf{K}_i, j \in \mathbf{J}_{ik}$
$\mathbf{J}_{imk} = \{j|j \in \mathbf{J}_{ik}, \chi_{ik}\beta_j^{max} \leq \zeta_{im}^{max}, \chi_{ik}\beta_j^{min} \geq \zeta_{im}^{min}\}\ \forall m \in \mathbf{M}_i, k \in \mathbf{K}_i$
STOP

Figure 2.9: Modified algorithm for stage-dependent batch sizes.

pendix A.2), and the vessel utilization constraint (i.e. Eq. 2.47) becomes,

$$\sum_{k}\sum_{i\in\mathbf{I}_{jk}^{S}}\sum_{m\in\mathbf{M}_{ijk}^{S}} X_{imjkt}^{S} \leq 1 \quad \forall\, j \in \mathbf{J}^{S}, t \tag{2.55}$$

where $\mathbf{M}_{ijk}^{S}$ is the set of batch size intervals for order i on stage k when stored in unit j.

The model formulations for each of the extension are summarized in Table 2.1. Equations under limited storage capacity and limited shared utilities show the changes from the base case model.

2.6.4 *Remarks*

The two proposed models are based on different approaches. M1 is based on the idea of disaggregating each order into batches which are treated independently. M2 is based on the concept

Table 2.1: Model formulations

Model	Stage-dependent Batch Size	Base Model	Storage Capacity (−)	Storage Capacity (+)	Utility (+)
M1	No	2.6-2.10, 2.11, 2.13-2.19	2.9, 2.10	2.42-2.45	2.39, 2.41
	Yes	2.6-2.10, 2.14, 2.6, 2.17-2.19, 2.49, 2.51, 2.52	2.9, 2.10	2.43-2.45, 2.53	2.39, 2.41
M2	No	2.9, 2.11, 2.13, 2.14, 2.14, 2.17-2.19, 2.24-2.33	2.9, 2.27	2.46, 2.47	2.40, 2.41
	Yes	2.9, 2.14, 2.17-2.19, 2.24, 2.26-2.33, 2.49, 2.51, 2.54	2.9, 2.27	2.47, 2.55	2.40, 2.41

Equations are removed (−), or added (+) to the base model

of batch size intervals and assignment of batches to the intervals.

M1 is likely to perform better in instances where the range number of potential batches (see Eq. 2.14) is tight, because this leads to fewer X^{L}_{iljkt} variables.

On the other hand, in instances where the capacity of the units are rather uniform, M2 should be more efficient. This is because uniform capacity of the units yields a small number of batch size intervals, which significantly reduces the number of X^{M}_{imjkt} variables. This situation is likely to appear in practice, where (i) only a few different types of units are installed for a given stage, and/or (ii) all units throughout production stages have similar (normalized) capacities. Furthermore, the performance of M2 does not depend on the range number of potential batches.

The comparison between the performance of the two models in different situations is provided in the section 2.8.

2.7 ILLUSTRATIVE EXAMPLES

We present several examples to illustrate the modeling flexibility and the applicability of the proposed models. The first example shows how simultaneous batching and scheduling can lead to surprisingly better solutions as compared to the traditional sequential approach. The second to fourth examples showcase how the proposed extensions, limited shared resources, storage with limited vessel capacities, and stage-dependent batch sizes can be implemented. All the runs in this section are formulated in GAMS 24.5.4 and solved using CPLEX 12.6.2 on a computer with 8GB RAM and Intel Core i7-4770 processor at 3.40GHz running on Windows 10. Discretization length of 1 hour is used.

2.7.1 *Simultaneous batching and scheduling*

We consider a multipurpose facility with 6 units, 2 storage vessels and 3 stages. Four orders with $\xi_i = 100kg \ \forall i$, have to be met. The production recipe of each order, unit capacities and processing times are given in Table 2.2. For complete information, refer to the supplementary information. We consider a makespan minimization instance, and solve using sequential and simultaneous approaches.

Table 2.2: Process data of example 1 (τ_{ijk}) [hr]

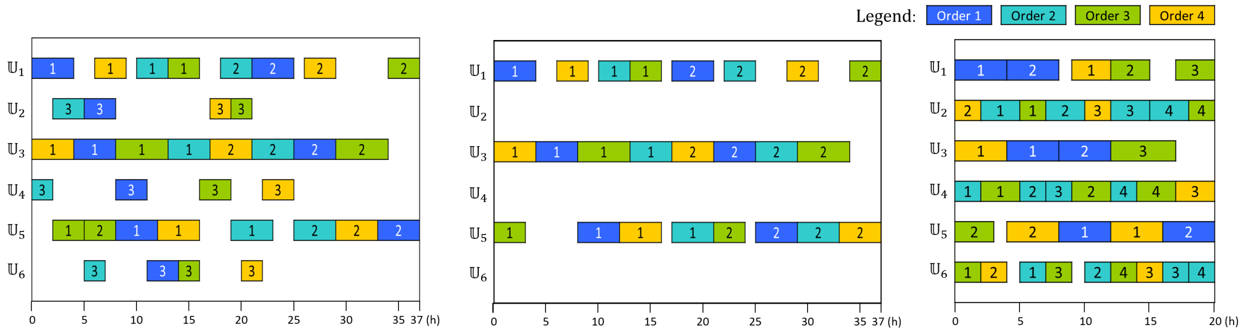
Order	Stage	$\mathbb{U}_1$	$\mathbb{U}_2$	$\mathbb{U}_3$	$\mathbb{U}_4$	$\mathbb{U}_5$	$\mathbb{U}_6$
		30-50(kg)	10-30(kg)	30-50(kg)	10-30(kg)	30-50(kg)	10-30(kg)
	k_1	4	3				
i_1	k_2			4	3		
	k_3					4	3
	k_1	3			2		
i_2	k_2		3	4			
	k_3					4	2
	k_1					3	2
i_3	k_2			5	3		
	k_3	3	2				
	k_1		2	4			
i_4	k_2	3					2
	k_3				3	4	

For the sequential approach, based on the unit capacity information, the effective minimum/maximum batch size for each order are 30kg and 50kg, respectively (i.e. $\hat{\beta}_i^{eff,min} = 30$kg, $\beta_i^{eff,max} = 50$kg). Using an average value of 40kg for the standard batch size, we obtain two batches of 40kg and one batch of 20kg for each order, satisfying the 100kg demand. We also investigate the case where two batches of 50kg for all orders are employed. The results are shown in Table 2.3. The optimal schedules for the instances are shown in Figure 2.10.

The results show that if the batching and scheduling decisions are decoupled, then a makespan of 37 hr is obtained in the optimal solution, while the same orders can be satisfied in only 20 hr if the decisions are made simultaneously. This large difference is because the batching decisions made in the sequential approach lead to an uneven work load distribution among units, since the two larger batches (i.e. 40kg or 50kg) can only be performed on a limited set of units (i.e.$j \in \{\mathbb{U}_1, \mathbb{U}_3, \mathbb{U}_5\}$). On the other hand, in the simultaneous approach, the batching deci-

Table 2.3: Computational statistics of example 1

	Sequential (40/40/20)		Sequential (50/50)		Simultaneous	
	M1	M2	M1	M2	M1	M2
Constraints	3,473	3,003	3,473	3,003	3,473	3,003
Discrete Variables	4,748	2,718	4,748	2,718	4,748	2,718
Continuous Variables	2,438	1,882	2,438	1,882	2,438	1,882
Objective Value (hr)	37	37	37	37	20	20
LP relaxation	8.52	2.23	23.65	4.6	5.61	1.35
CPU (s)	4.94	1.17	7.55	1.55	61.01	16.17
Nodes	0	0	0	23	5,099	2,365
Gap (%)	-	-	-	-	-	-



(a) Sequential (40kg/40kg/20kg)

(b) Sequential (50kg/50kg)

(c) Simultaneous

Figure 2.10: Optimal schedules of example 1 with (a) sequential batching and scheduling with 40/40/20kg batching (M2), (b) sequential batching and scheduling with 50/50kg batching (M2), and (c) simultaneous batching and scheduling (M2). The color code represent different orders and the numbers in each box represent batch l. The makespan is reduced from 37 hr to 20 hr if batching and scheduling decisions are made simultaneously.

sions are made such that small and large units are evenly utilized to fully exploit the available resources. We additionally note that in the simultaneous approach the total processing time is increased due to the increased number of batches (e.g. 12 or 8 batches increased to 13), but the makespan is reduced significantly.

Table 2.4: Batching decisions for example 1 [kg]

	Sequential					Simultaneous			
Orders	l_1	l_2	l_3	l_1	l_2	l_1	l_2	l_3	l_4
i_1	40	40	20	50	50	50	50		
i_2	40	40	20	50	50	30	30	30	10
i_3	40	40	20	50	50	30	30	30	10
i_4	40	40	20	50	50	40	30	20	

2.7.2 *Limited shared utilities*

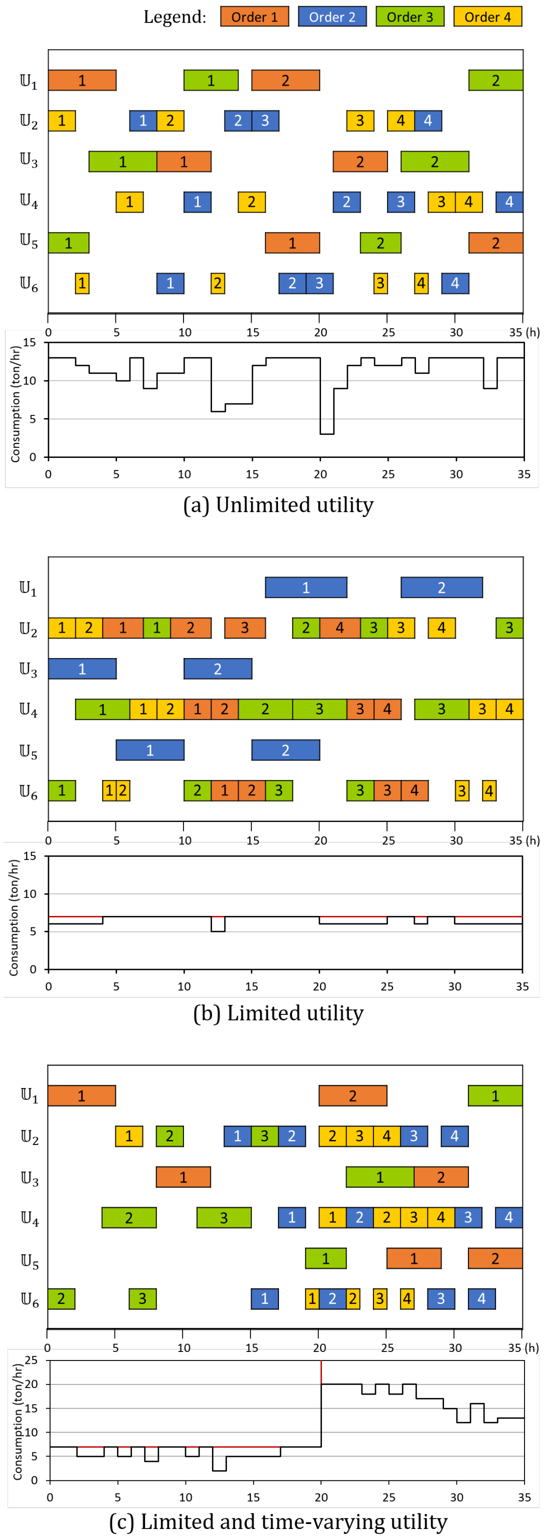
We consider a facility with 6 units, 2 vessels, 3 stages and 4 orders, each for 100kg. Every task requires high pressure steam (HS) at a cost of \$1/(ton/hr). The processing time, cost and utility requirement are shown in Table 2.5 (see supplementary information for full data). We consider three cost minimization instances: (a) unlimited HS available, (b) availability of HS limited to 7ton/hr, and (c) availability of HS is limited to 7ton/hr during the first 20hr and unlimited afterwards. We use both M1 and M2 under utility constraints. The results and the optimal schedules are shown in Table 2.6 and Figure 2.11, respectively. The results show that M2 obtained the optimal solutions in reasonable time for all instances, while M1 failed to prove optimality in the second and third instances.

Table 2.5: Process data of example 2 (τ_{ijk} [hr] /α_{ijk} [\$]/$\rho_{ijk,HS}$ [\$/(ton/hr)])

Order	Stage	$\mathbb{U}_1$	$\mathbb{U}_2$	$\mathbb{U}_3$	$\mathbb{U}_4$	$\mathbb{U}_5$	$\mathbb{U}_6$
		35-50(kg)	10-30(kg)	35-50(kg)	10-30(kg)	35-50(kg)	10-30(kg)
i_1	k_1	5/25/5	3/20/2				
	k_2			4/25/5	2/20/2		
	k_3					4/20/5	2/15/2
i_2	k_1	2/10/3	5/30/1				
	k_2					5/30/1	2/10/3
	k_3	6/35/2		2/15/4			
i_3	k_1					3/10/5	2/10/2
	k_2			5/20/6	4/20/2		
	k_3	4/20/4	2/10/2				
i_4	k_1		2/10/3	5/30/2			
	k_2	3/15/1					1/5/2
	k_3				2/10/4	5/25/2	

Table 2.6: Computational statistics of example 2

	Unlimited		Limited		Time-varying	
	M1	M2	M1	M2	M1	M2
Constraints	2,523	1,591	2,523	1,591	2,523	1,591
Discrete Variables	3,172	1,824	3,172	1,824	3,172	1,824
Continuous Variables	1,753	937	1,753	937	1,753	937
Objective Value (\$)	876	876	920	898	880	877
LP relaxation	442.0	740.2	442.0	740.2	442.0	740.2
CPU (s)	2486.25	0.47	10,800.14	41.74	10,800.22	4.08
Nodes	885,525	0	333,432	2,520	920,475	2,514
Gap (%)	-	-	18.2	-	9.8	-



(a) Unlimited utility

(b) Limited utility

(c) Limited and time-varying utility

Figure 2.11: Optimal schedule and utility consumption profile of example 2 with (a) unlimited utility (M2), (b) limited utility (M2), and (c) utility with time-varying availability (M2). The color code represent different orders and the numbers in each box represent batch l.

Table 2.7: Batching decisions for example 2 [kg]

	Unlimited				Limited				Time-varying			
Orders	l_1	l_2	l_3	l_4	l_1	l_2	l_3	l_4	l_1	l_2	l_3	l_4
i_1	50	50			30	30	20	20	50	50		
i_2	30	30	20	20	50	50			30	30	20	20
i_3	50	50			30	30	20	20	50	30	30	
i_4	30	30	20	20	30	30	20	20	30	30	20	20

In terms of the effect of the utility constraints, the first instance (unlimited HS) results in the least total cost ($876). As can be seen in Figure 2.11(a), batches are evenly spread among the processing units. In the second instance (HS availability equal to 7ton/hr), we observe that batches start to get allocated to more energy efficient but smaller units (i.e. $j \in \{\mathbb{U}_2, \mathbb{U}_4, \mathbb{U}_6\}$), even though this leads to a larger number of batches (see Table 2.7). Interestingly, processing larger batch sizes is more efficient for order i_2, and thus the batching decisions change in the opposite direction where larger but fewer batches are selected. This instance has the largest total cost of $898, due to higher processing cost. In the last instance (Figure 2.11(c)), we can see that there are significantly fewer batches processed in the first 20 hr, where the availability of HS is limited to 7ton/hr. More batches are processed after 20hr, because there are options available for processing in more energy inefficient, but low cost units. We see that batching decisions are almost the same as in the first instance, except for order i_3 (e.g. 50kg-30kg-20kg instead of 50kg-50kg), due to fully occupied units after 20 hr. As a result, the total cost increases to $877.

2.7.3 *Capacity-constrained storage vessels*

We consider a facility with 6 units, 2 vessels and 3 stages, and five orders (100kg each), to be delivered within 40 hour (i.e. $\phi_i = 40hr \; \forall i$). The batch processing time and the batch-vessel compatibility are given in Table 2.8. Two instances are studied. In the first, we assume that large enough vessel capacity is available to store any compatible batch, while in the second instance we take vessel capacities into consideration. The objective is earliness minimization.

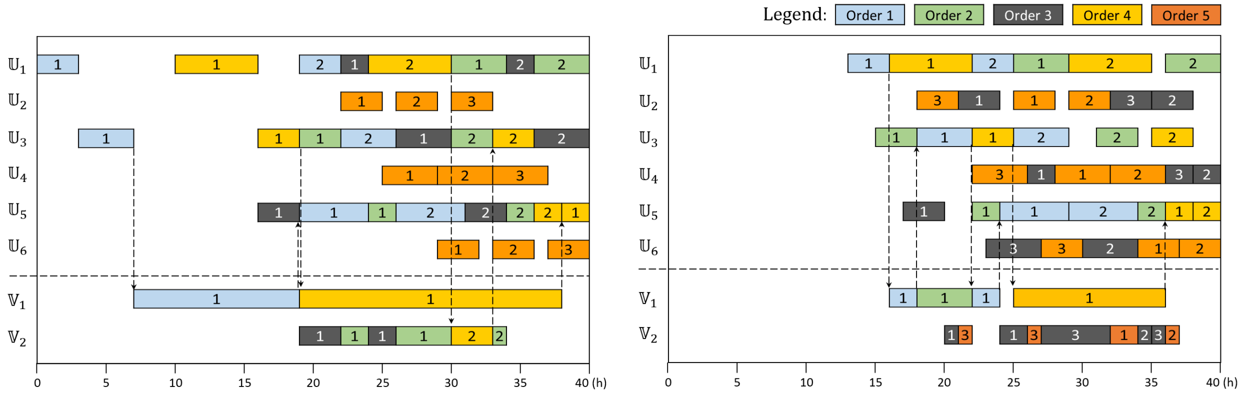
Since a new storage variable X^S_{iljkt} and additional constraints are required to account for vessel capacity, the resulting models are larger (Table 2.9), and the computational time for the second instance is longer for both models. The best strategy to minimize earliness is to select larger

Table 2.8: Process data of example 3 (τ_{ijk}) [hr]

Order	Stage	$\mathbb{U}_1$ 10-50(kg)	$\mathbb{U}_2$ 10-40(kg)	$\mathbb{U}_3$ 10-50(kg)	$\mathbb{U}_4$ 10-40(kg)	$\mathbb{U}_5$ 10-50(kg)	$\mathbb{U}_6$ 10-40(kg)	$\mathbb{V}_1$ 0-50(kg)	$\mathbb{V}_2$ 0-40(kg)
i_1	k_1	2	3					✓	✓
	k_2			3	4			✓	✓
	k_3					4	5	✓	✓
i_2	k_1			5	3			✓	✓
	k_2					3	2	✓	✓
	k_3	4	4					✓	✓
i_3	k_1					4	3	✓	✓
	k_2	3	2					✓	✓
	k_3			2	4			✓	✓
i_4	k_1	4	6					✓	✓
	k_2			2	3			✓	✓
	k_3					2	2	✓	✓
i_5	k_1	3	5					✓	✓
	k_2			4	6			✓	✓
	k_3					3	4	✓	✓

Table 2.9: Computational statistics of example 3

	w/o capacity		w/ capacity	
	M1	M2	M1	M2
Constraints	2,755	2,030	9,807	2,382
Discrete Variables	4,112	2,687	6,572	3,302
Continuous Variables	1,841	1,266	1,841	1,266
Objective Value (hr)	55	55	57	57
LP relaxation	22.12	34.71	22.12	35.25
CPU (s)	20.02	3.66	114.75	3.53
Nodes	1,217	369	3,828	429
Gap (%)	-		-	-



(a) Not considering vessel capacity

(b) Considering vessel capacity

Figure 2.12: Optimal schedule of example 3 (a) without vessel capacity (M1), and (b) with vessel capacity (M1). The Gantt chart also shows when and where each batch is stored, the flow between units and vessels depicted with dashed arrow (only shown for orders i_1 and i_4 for simplicity). The color code represent different orders and the numbers in each box represent batch l.

Table 2.10: Batching decisions for example 3 [kg]

	w/o capacity			w/ capacity		
Orders	l_1	l_2	l_3	l_1	l_2	l_3
i_1	50	50		50	50	
i_2	50	50		50	50	
i_3	50	50		40	40	20
i_4	50	50		50	50	
i_5	40	40	20	40	40	20

batch sizes, which leads to few batches per order (see batching decisions in Table 2.10), except for the fifth order which is divided into smaller batches, because units with higher capacity (i.e. $j \in \{\mathbb{U}_1, \mathbb{U}_3, \mathbb{U}_5\}$) are fully occupied, as shown in Figure 2.12(a). Furthermore, both vessels are actively used for storing 50kg batches to maximize flexibility. As a result, the minimum total earliness is 55 hr.

In the second instance, while the capacity of vessel $\mathbb{V}_1$ remains the same, the capacity of the second vessel $\mathbb{V}_2$ is reduced to 40kg, which becomes binding and thereby leads to smaller and more batches. This is reflected in the optimal schedule where the number of batches of order i_3 increases from 2 to 3 (Table 2.10) with smaller batch sizes. As can be seen in Figure 2.12(b) batches of order i_3 are processed on the smaller units (i.e. $j \in \{\mathbb{U}_2, \mathbb{U}_4, \mathbb{U}_6\}$), and occasionally stored in $\mathbb{V}_2$. Consequently, the earliness increases from 55 hr to 57 hr. More importantly, different batching decision as well as batch-unit assignments are made in the optimal schedule.

2.7.4 *Stage-dependent batch sizes*

We consider a facility consisting of 6 units, 2 vessels, 3 stages, and 4 orders (100kg each). The process information and the information on how batch sizes change through the stages are given in Table 2.11. We solve a makespan minimization problem with both models, and solution statistics are given in Table 2.12.

Model M2 resulted in a larger formulation, because the stage-dependent batch sizes led to relatively non-uniform unit capacities for some of the orders. As a result, M1 proved optimality faster than M2. The batching decisions are affected by how the batch sizes change throughout the stages. Specifically, while batch sizes of order i_4 do not change, order i_3 decreases threefold, thus requiring more number of batches to meet the demand (see Table 2.13). The optimal schedule obtained by M2 is shown in Figure 2.13.

Table 2.11: Process data of example 4

Order	Stage	χ_{ik}	σ_{ik}	$\mathbb{U}1$	$\mathbb{U}2$	$\mathbb{U}3$	$\mathbb{U}4$	$\mathbb{U}5$	$\mathbb{U}6$
				10-125(kg)	10-75(kg)	10-100(kg)	10-60(kg)	10-75(kg)	10-45(kg)
i_1	k_1	0.4	0.8	4	2				
	k_2	0.5	0.75			4	2		
	k_3	0.67	0.67					4	2
i_2	k_1	0.5	1	3	1				
	k_2	0.5	1			2	4		
	k_3	0.5	0.5					4	1
i_3	k_1	0.33	0.83	3	1				
	k_2	0.4	0.48			5	2		
	k_3	0.83	0.83					3	1
i_4	k_1	1	1	2	4				
	k_2	1	1			3	1		
	k_3	1	1					2	4

Table 2.12: Computational statistics of example 4

	M1	M2
Constraints	3,642	5,233
Discrete Variables	5,249	5,519
Continuous Variables	2,590	3,442
Objective Value (hr)	18	18
LP relaxation	5.13	0.90
CPU (s)	20.11	41.89
Nodes	2,499	1,661
Gap (%)	-	-

Table 2.13: Batching decisions for example 4 [kg]

	M1					M2				
Orders	l_1	l_2	l_3	l_4	l_5	l_1	l_2	l_3	l_4	l_5
i_1	50	25	25			50	30	22.5		
i_2	37.5	22.5	22.5	22.5		37.5	37.5	22.5	22.5	
i_3	24	24	24	24	24	24	20	20	18	18
i_4	60	40				60	40			

2.8 COMPUTATIONAL STUDY

We study the effect of (i) the total number of batch size intervals (i.e. $M := \sum_{i \in \mathbf{I}} |\mathbf{M}_i|$), and (ii) the expected number of batches for each order (i.e. $L := \lceil \xi_i / \hat{\beta}_i^{eff,min} \rceil \ \forall i$) on the performance

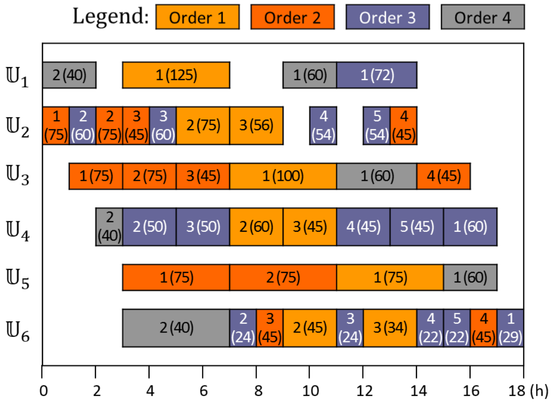


Figure 2.13: Optimal schedule of example 4. The color code represent different orders and the numbers in each box represent batch l and its (size (kg)).

of the models, random instances are generated and tested using makespan minimization as the objective function. In addition, we examine the effect of the aforementioned characteristics on a large-scale instance. A total of 760 runs are performed, using CPLEX 12.6.2 via GAMS 24.5.4 on a cluster with Intel Xeon (E5520) processors at 2.27 HGz and 16GB of RAM running on CentosOS Linux 7. A resource limit of 10,800 seconds and discretization length of 1 hour is used for all the runs. Default CPLEX options were used except for section 2.8.4, where option ObjDef=0.999 is used.

2.8.1 *Effect of total number of batch size intervals*

All instances are generated based on a single facility with the same number of orders, units and vessels. A total of 5 orders with amounts of 60kg, 60kg, 70kg, 80kg, 70kg, respectively, are considered. There are 5 process units, 3 storage vessels, and each order has to be processed on up to 4 stages (i.e. $\max_{i \in \mathbf{I}} |\mathbf{K}_i| = 4$). With the aforementioned parameters fixed, instances with different number of batch size intervals (M), are generated by randomly varying the following factors: (i) number of stages for each order (i.e. $\mathbf{K}_i$), (ii) unit and vessel compatibility (i.e. $\mathbf{J}_{ik}/\mathbf{J}^S_{ik}$), and (iii) unit capacities (i.e. $\beta_j^{min}/\beta_j^{max}$) and their uniformity. We note that the uniformity of the unit capacities is controlled by adjusting the candidates in the sampling pool, and the sampling pool is generated such that the expected number of batches for each order are fixed to 2 (i.e. $L = 2 \ \forall i$). We generated 10 instances for each value of M ranging from 5 to 34 (i.e. a total of 30 sets of instances), resulting in a total of 300 instances. Other parameters including release/due dates and processing time/cost are also randomly generated. Refer to supplementary information for the full algorithm used for the generation of instances.

All 300 instances are solved using both models. Next, the average solution times for the models for each M are obtained by taking average among the 10 instances. Then, the ratio of average solution time and the average number of discrete variables of M2 to M1 is calculated and plotted as a function of M (Figure 2.14).

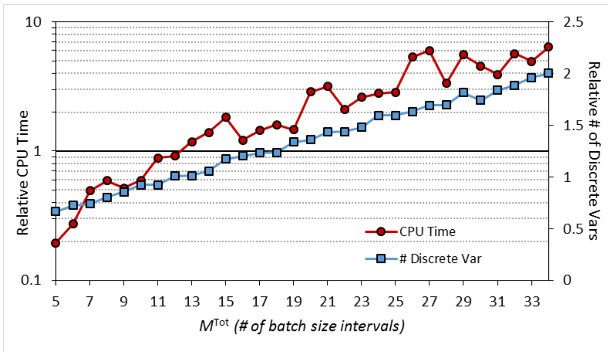


Figure 2.14: Ratio of average solution times and number of discrete variables between two models (i.e. M2:M1) as a function of number of batch size intervals. M2 is superior at low values of M. M1 becomes faster as M increases above 13.

The results show a clear trend. At low M, M2 is smaller and faster. However, the ratio of the model size grows approximately linearly as M increases, and as a result, M1 becomes faster than M2 at high M. The "turn over" point is observed approximately at $M = 13$. However, we note that $M = 13$ does not generally mark the threshold in which M2 becomes faster, because other characteristics of the scheduling instance affect the performance. However, the trend shown in Figure 2.14 holds generally.

2.8.2 *Effect of expected number of batches for each order*

The facility used in the previous section is considered. This time, however, the total number of intervals is 10 (i.e. $M = 10$) and the following factors are randomly varied to generate instances with different values of L: (i) order sizes, ξ_i, and (ii) unit capacities, $\beta_j^{min}/\beta_j^{max}$. We generate 10 instances for each value of L, ranging from 1 to 8 (total of 8 sets of instances), resulting in 80 total instances. Other parameters are randomly generated as previously.

All 80 instances are solved using both models. The two models are then compared using average solution time and number of discrete variables as previously (Figure 2.15). We observe that M1 is superior at low L, while M2 becomes faster as L increases. Interestingly, while M1 is faster at values of L below 4, its size is actually greater than M1 at $L = 3$.

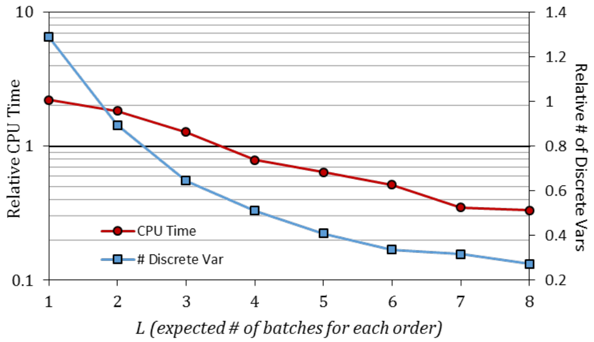


Figure 2.15: Ratio of average solution times and number of discrete variables between two models (i.e. M2:M1) as a function of expected number of batches for each order. M1 is superior at low values of L. M2 becomes faster as L increase above 4.

2.8.3 *General performance*

To evaluate the general performance of the two models, a performance chart using results from all 380 instances is generated (Figure 2.16). In the chart, the abscissa corresponds to the ratio of the solution times of a particular instance between a model and the fastest model, denoted as R. The ordinate corresponds to the percentage of instances solved to optimality within a given R.

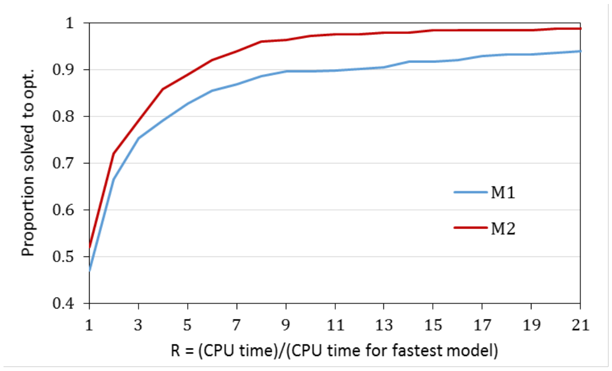


Figure 2.16: The performance chart for the two models. M2 solved 98.8% of the instances to optimality, while M1 solved 93.8%.

While model M2 appears to be faster than M1, it is important to note that among the instances that couldn't be solved to optimality by M2, some of them are solved to optimality by M1 very efficiently, and vice versa. Hence, the performance of the models may vary significantly according to the characteristics of the instance at hand. Thus, the analysis presented in Section 2.6.4 can be very useful in determining which of the two models should be used when scheduling operations in a given facility.

2.8.4 *Large-scale instance*

A multipurpose plant with 10 units, 3 vessels and 4 stages (i.e. $\max_{i \in \mathbf{I}} |\mathbf{K}_i| = 4$) is considered. A total of 30 orders between 50-100kg need to be satisfied. Other process information, including processing time/cost, unit/vessel compatibility, required number of stages for each order, release/due dates are randomly generated (see supplementary information for full data). Minimum and maximum unit capacities are sampled from {10kg, 20kg} and {40kg, 50kg}, respectively. Due to the limited candidates for unit capacities in the sampling pool, the instance exhibits a small number of batch size intervals. Both M1 and M2 are used to solve this instance with all three objective functions. The results are shown in Table 2.14.

Table 2.14: Computational statistics of large-scale instances

	Cost		Earliness		Makespan	
	M1	M2	M1	M2	M1	M2
Constraints	67,621	75,935	67,861	75,965	40,284	49,017
Discrete Variables	138,302	113,015	138,302	113,015	100,484	84,149
Continuous Variables	53,868	63,277	53,898	63,307	29,509	34,742
Objective Value ($ or hr)	5,080	5,080	11	10	140	107
LP relaxation	3,658.90	5,013.75	4.00	4.53	38.57	32.90
CPU (s)	5,383.56	120.66	10,800.01	4094.02	10,800.32	10,800.84
Nodes	197	0	965	419	108	192
Gap (%)	-	-	9.1	-	68.6	58.3

In general, although M2 has more number of constraints than M1, it has fewer discrete variables due to small M. As a result, in cost minimization, M2 yields the optimal solution of $5,080 in around two minutes, while M1 took around 90 minutes. Moreover, in earliness minimization, M2 found the optimal solution of 10 hr, while M1 failed to prove optimality within 3 hours. The optimal schedule for cost minimization objective obtained by M2 is shown in Figure 2.17.

Since makespan minimization is a difficult objective function, we use the iterative scheme proposed by Maravelias and Grossmann (2003a), where we first solve the instance with a shorter time horizon (i.e. lower bound of *MS*) and increase the horizon length in each iteration until a feasible solution is obtained. When the horizon length was set to 140 hr, a feasible solution of 107 hr with 58.3% gap was obtained using M2, while M1 obtained 140 hr with a larger gap of 68.6%. However, given the fact that the models became immediately infeasible when 70 hr horizon is used, we can safely assume that the actual gap of the feasible solutions are at most 50.0% and 34.6% for M1 and M2, respectively.

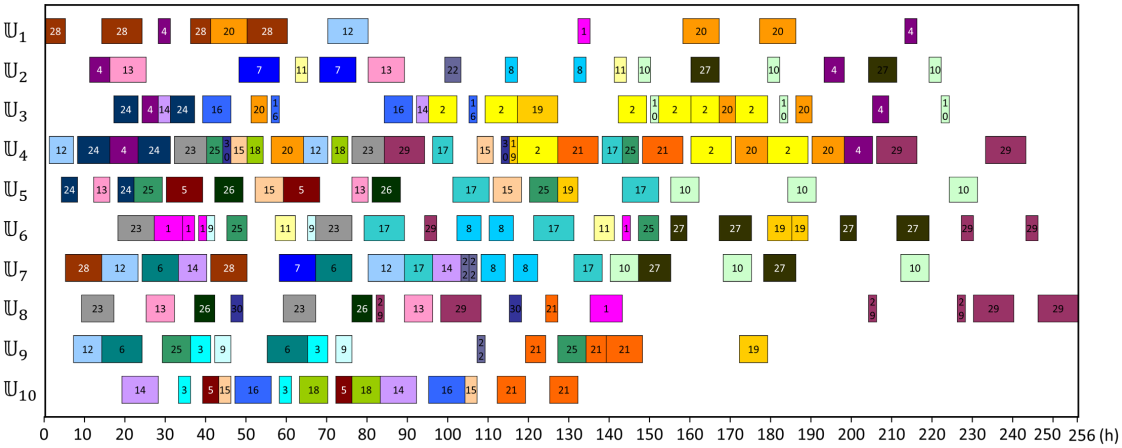


Figure 2.17: Optimal schedule obtained for the large-scale instance with cost minimization objective.

From this instance, we can see that identifying the characteristics of the scheduling instances becomes more important as the problem size increases. Furthermore, exploiting instance characteristics allows us to solve even large scale instances (e.g. ∼70,000 constraints, >100,000 discrete-variables) with easier objective functions in realistic time frames.

2.9 CONCLUSIONS

We studied the simultaneous batching and scheduling in multipurpose batch plants with storage constraints, a class of problems which has not been studied extensively. Two new discrete-time mixed-integer programming models were proposed based on two different modeling approaches: (i) explicit labeling of batches, and (ii) identification of feasible batch size intervals for each unit routing. The first model employs order disaggregation into batches and each batch is explicitly labeled through the introduction of an additional index $l \in \mathbf{L}$. The disaggregated batches are then independently scheduled. The second model is based on the concept of batch size intervals to which each batch is assigned. To identify the set of batch size intervals for every order, we propose an algorithm which has to be run just once for a given facility. Furthermore, we proposed extensions to address limited shared utilities with time-varying availability and cost, storage with capacity constraints, and stage-dependent batch sizes.

Through illustrative examples, we showed that simultaneously considering batching decisions may lead to significantly improved solutions. We also showed that considering utility constraints may have an important impact on the quality of the solution; i.e, the optimal batching decisions obtained were widely different compared to when utility constraints were ignored.

In terms of computational performance, we showed that in instances where unit capacities are relatively uniform, the number of batch size intervals is small and the second model is superior. On the other hand, when the order sizes are relatively small compared to the batch sizes, the first model is more effective. These observations, which are due to the different modeling approaches that were employed, allow us to exploit instance characteristics, which in turn allows us to and solve large-scale instances.

2.10 NOMENCLATURE

Indices/sets

$i \in \mathbf{I}$	Orders
$j \in \mathbf{J}/\mathbf{J}^P/\mathbf{J}^S$	Units/processing units/storage vessels
$k \in \mathbf{K}$	Stages
$l \in \mathbf{L}$	Batches
$m \in \mathbf{M}$	Batch size intervals
$r \in \mathbf{R}$	Resources
$t \in \mathbf{T}$	Time points/periods

Subsets

$\mathbf{I}_{jk}$	Orders on stage k that can be processed in unit j
$\mathbf{I}^S_{jk}$	Orders that can be stored in vessel j after being processed on stage k
$\mathbf{I}_{jkt}$	Orders on stage k that can start in unit j at time point t
$\mathbf{J}_i$	Process units that can process order i
$\mathbf{J}^S_i$	Storage vessels that can store order i
$\mathbf{J}^{max}_i/\mathbf{J}^{min}_i$	Remaining candidate units for batch size interval generation algorithm
$\mathbf{J}_{ik}$	Process units that can process order i on stage k
$\mathbf{J}^S_{ik}$	Storage vessels that can store a batch of order i that has been processed on stage k
$\mathbf{J}_{ikt}$	Process units in which order i on stage k can start at time point t
$\mathbf{J}_{imk}$	Process units that can process order i on stage k assigned to batch size interval m
$\mathbf{J}^S_{imk}$	Storage vessels that can store order i on stage k assigned to batch size interval m

$\mathbf{K}_i$	Stages on which order i is processed
$\mathbf{K}_{ik}^{-}/\mathbf{K}_{ik}^{+}$	Stages on which order i have to be processed before/after stage k
$\mathbf{L}_i$	Batches of order i
$\mathbf{M}_i$	Batch size intervals for order i
$\mathbf{M}_{ij}$	Batch size intervals for order i when processed in unit j
$\mathbf{M}_{ijk}/\mathbf{M}_{ijk}^{S}$	Batch size intervals for order i on stage k processed/stored in unit/vessel j
$\mathbf{P}_i^{max}/\mathbf{P}_i^{min}$	Remaining candidate units for algorithm with stage-dependent batch sizes
$\mathbf{T}_{ijk}$	Set of feasible time points for order i on stage k to start in unit j
$\mathbf{T}_{ik}^{S}$	Set of feasible time periods in which order i on stage k can be stored in any compatible vessel

Parameters

α_{ijk}	Cost of processing a batch of order i on stage k in unit j
α_{rt}^{U}	Cost of resource r during time period t
α_{ijk}^{Inv}	Inventory cost of batch of order i on stage k stored in vessel j
$\beta_j^{min}/\beta_j^{max}$	Minimum/maximum capacity of unit or vessel j
$\beta_i^{eff,min}/\beta_i^{eff,max}$	Effective minimum/maximum batch sizes of order i
$\hat{\beta}_i^{eff,min}$	Realistic minimum batch size of order i
δ	Time discretization length
ε_{ik}	Earliest start time of order i on stage k
$\zeta_{im}^{min}/\zeta_{im}^{max}$	Lower/upper endpoint of batch size interval m of order i
η^0/η^F	Start/end of the scheduling horizon (i.e. $\eta^0 = min_{i\in\mathbf{I}}\,\mu_i$, $\eta^F = max_{i\in\mathbf{I}}\,\phi_i$)

Θ_{ij} Number of time ranges of order i on its last stage when processed in unit j (Eq. 2.37)

κ_i^{LS} Last stage of order i

λ_{ik} Latest finish time of order i on stage k

μ_i Release time of order i

ξ_i Amount due of order i

σ_{ik} Yield of order i on stage k

ρ_{ijkt} Utility r required to process a batch of order i on stage k in unit j

τ_{ijk} Processing time of order i on stage k in unit j

ϕ_i Due date of order i

χ_{ik} Relative size of the final product to batch size on stage k of order i

ψ_{rt} Availability of resource r during time period t

Binary variables

X_{iljkt}^{L} =1 if batch $l \in \mathbf{L}_i$ on stage k starts in unit j at time point t

X_{imjkt}^{M} =1 if order i on stage k with assigned to interval m starts in unit j at time point t

X_{ijkt}^{S} =1 if any batch of order i on stage k is being stored in vessel j during time period t

X_{iljkt}^{S} =1 if batch $l \in \mathbf{L}_i$ on stage k is being stored in vessel j during time period t (M1: Extension2)

X_{imjkt}^{S} =1 if a batch of order i on stage k assigned to interval m is being stored in vessel j during time period t (M2: Extension2)

Y_{il}^{L} =1 if batch $l \in \mathbf{L}_i$ of order i is selected

Y_{ilm}^{M} =1 if batch $l \in \mathbf{L}_i$ assigned to interval m is selected

S_{ilkt}^{L} =1 if material produced by batch $l \in \mathbf{L}_i$ on stage k is being stored during time period t

Non-negative integer variables

S^M_{imkt} inventory of batch of order i assigned to interval m on stage k during time period t

N_i number of batches of order i

N^M_{im} number of batches of order i assigned to interval m

Non-negative continuous variables

B_{il} batch size of $l \in \mathbf{L}_i$

B^M_{ilm} batch size of $l \in \mathbf{L}_i$ assigned to interval m

MS makespan

U_{rt} amount of resource r being consumed during time period t

3

DISCRETE-CONTINUOUS ALGORITHM: FRAMEWORK & MATHEMATICAL FORMULATIONS

3.1 MOTIVATION

One of the key attributes for classifying MIP scheduling models is the time representation. Both discrete- and continuous-time representations for scheduling models in the network environment have distinctive advantages and disadvantages.

One of the main strengths of discrete-time models comes from the fact that the location of every time point is known *a priori*, which allows for straightforward modeling of intermediate events, such as time-varying resource availability and intermediate release/due dates (Kondili et al., 1993; Shah et al., 1993; Lee and Maravelias, 2017b). Furthermore, it allows linear modeling of inventory, backlog and utility costs, and complex logistics (Zyngier and Kelly, 2009). In addition, discrete-time models can handle intermediate events during the execution of a task as proposed by Pantelides (1994). Discrete-time models generally assume fixed processing time, although variable processing times can be modeled through introduction of task modes (Kondili et al., 1993). The major drawback of discrete-time models is related to data inaccuracy. In many instances, relatively large discretization step lengths are chosen for tractability, resulting in approximation errors in the converted problem data, which may lead to inferior solutions. Another facet to this issue is that the selection of discretization step length in such cases is mainly based on user intuition, due to the lack of systematic methods. This issue is rather important, because the quality of the solution depends on the choice of the discretization step length, as will be illustrated throughout the paper.

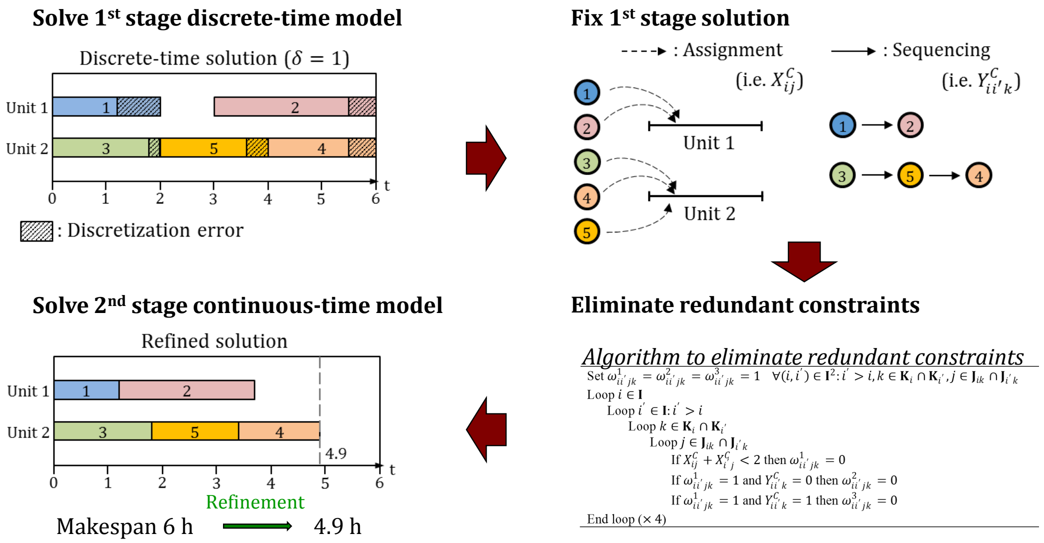
On the other hand, the main advantage of continuous-time models is solution accuracy. Moreover, continuous-time models require significantly fewer time intervals to represent the best solution, leading to smaller models, though in terms of computational performance, continuous-time models are shown to be inferior to their discrete-time counterparts, especially for large-scale instances (Sundaramoorthy and Maravelias, 2011b). Furthermore, modeling intermediate events is not as efficient in continuous-time representation, because it requires additional time points and discrete variables, and/or leads to nonlinearities.

Accordingly, the purpose of this work is to propose a solution method that allows us to harness the advantages of both discrete- and continuous-time mixed-integer programming (MIP) scheduling models, while overcoming their limitations. Specifically, we discuss a method, in which we: (1) obtain an approximate, in terms of events timing, solution using a discrete-time formulation; (2) map this solution onto continuous-time material- and unit-specific grids; and (3) obtain an accurate solution solving a reduced continuous-time formulation based on the grids generated in step 2. The discrete-time model used in the first stage allows us to obtain a high quality solution quickly, as well as consider a wide range of features that cannot be addressed effectively using continuous-time models (e.g., limited shared resources with time-varying resource availability, and intermediate deliveries/demands). The batching and task-assignment decisions of this solution are mapped onto time grids which are used for the formulation of a continuous-time model which is an LP, since batching and assignment decisions are fixed. Thus, the originally obtained solution is *refined* and improved via the fine-tuning of the timing of events and batch sizes.

3.2 BACKGROUND

3.2.1 *Solution refinement in the sequential environment*

In order to obtain accurate solutions based on discrete-time formulations, a solution refinement method for discrete-time models for scheduling in multistage facilities was proposed (Merchan et al., 2016). The proposed method solves the problem in two stages: (1) obtain discrete-time solutions using any discrete-time model, and (2) with the assignments and sequencing binaries fixed, re-solve for the timing of batch processes using a continuous-time model (see Figure 3.1). This method converts a solution into a more accurate one by removing the discretization error with minimal computational expense. In addition, this method allows for larger discretization steps to be used in the first stage without significant loss of accuracy in the final solution, thus enabling the solution of some difficult instances.



```
Algorithm to eliminate redundant constraints
Set $\omega^1_{ii'jk} = \omega^2_{ii'jk} = \omega^3_{ii'jk} = 1 \quad \forall (i,i') \in \mathbf{I}^2 : i' > i, k \in \mathbf{K}_i \cap \mathbf{K}_{i'}, j \in \mathbf{J}_{ik} \cap \mathbf{J}_{i'k}$
Loop $i \in \mathbf{I}$
    Loop $i' \in \mathbf{I} : i' > i$
        Loop $k \in \mathbf{K}_i \cap \mathbf{K}_{i'}$
            Loop $j \in \mathbf{J}_{ik} \cap \mathbf{J}_{i'k}$
                If $X^C_{ij} + X^C_{i'j} < 2$ then $\omega^1_{ii'jk} = 0$
                If $\omega^1_{ii'jk} = 1$ and $Y^C_{ii'k} = 0$ then $\omega^2_{ii'jk} = 0$
                If $\omega^1_{ii'jk} = 1$ and $Y^C_{ii'k} = 1$ then $\omega^3_{ii'jk} = 0$
End loop (× 4)
```

Figure 3.1: Illustration of the solution refinement method for discrete-time models for scheduling in multistage facilities (Merchan et al., 2016).

The solution refinement method for the sequential environment exploits the fact that there are known precedence relations between the upstream and downstream tasks for a given product/batch. Once assignment of batches to units and sequencing of batches within each unit are obtained, the second stage model, which simply ensures that the upstream tasks finish before the downstream tasks begin, can be formulated as an LP.

3.2.2 *Problem statement*

We study the problem of short-term chemical production scheduling in the network environment. We are given:

1) production facility data (e.g. facility layout, unit capacities)

2) production recipes (e.g. mixing rules, processing times)

3) production costs (e.g. processing, inventory holding)

4) resource availability (e.g. raw material, utility)

5) customer demand (e.g. demand amount and dates).

We aim to solve for the optimal schedule using different objectives, such as minimization of cost/makespan or maximization of profit. Intermediate delivery of raw materials and customer demands may be present. The processing time and cost of tasks may be dependent on their batch sizes. The availability of utilities may be limited and time-varying. We assume unlimited dedicated storage for each material, though the method can be modified to address limited capacity.

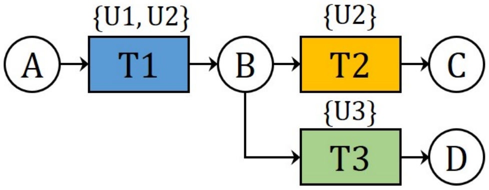


Figure 3.2: A sample network consisting of 4 states (i.e. $\mathbf{K} = \{A, B, C, D\}$), 3 tasks (i.e. $\mathbf{I} = \{T1, T2, T3\}$) performed on 3 units (i.e. $\mathbf{J} = \{U1, U2, U3\}$).

We adopt the STN representation (Kondili et al., 1993) (see Figure 3.2 for an example). As a part of the problem data, we introduce the concept of incident points, to denote time points where intermediate incidents (e.g. change in utility availability, intermediate due dates) occur (see Figure 3.3). The following indices, sets and parameters are used to describe the problem data.

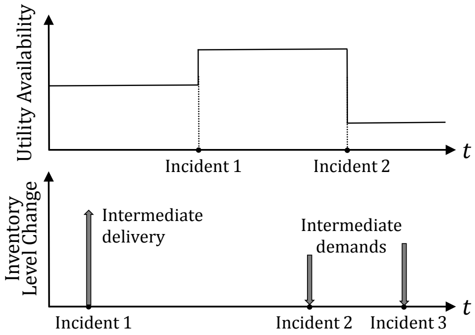


Figure 3.3: Examples of incident points that are used to describe the occurrences of intermediate incidents given by the problem data.

Indices/sets

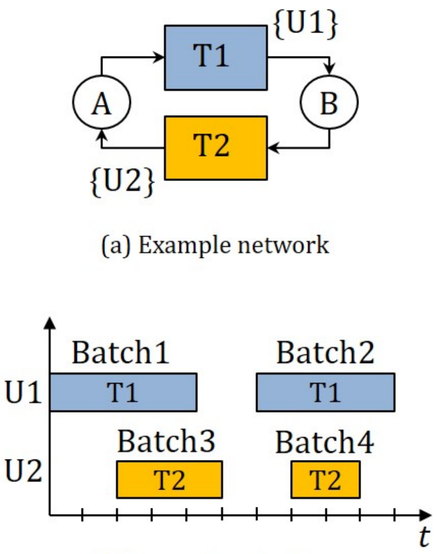
$i \in \mathbf{I}$	Tasks
$j \in \mathbf{J}$	Units
$k \in \mathbf{K}$	Materials
$r \in \mathbf{R}$	Utilities
$p \in \mathbf{P}$	Incident points
$\mathbf{I}_j$	Tasks that can be performed in unit j
$\mathbf{I}_k^+ / \mathbf{I}_k^-$	Tasks that produce/consume material k
$\mathbf{I}_r$	Tasks that require utility r
$\mathbf{J}_i$	Units that can perform task i

Parameters

$\alpha_{ij}^F / \alpha_{ij}^V$	Fixed/variable cost of performing task i in unit j
$\beta_j^{min} / \beta_j^{max}$	Minimum/maximum capacity of unit j
γ_k	Initial inventory of material k
H	Horizon length
(ξ_{kp}, t_{kp}^{ξ})	Amount/time of p^{th} delivery (<0) or demand (>0) of material k
π_k	Price of material k
ρ_{ik}	Fraction of material k produced (>0) or consumed (<0) by task i
ρ_{ir}^U	Amount of utility r required per time period by task i
$\tau_{ij}^F / \tau_{ij}^V$	Fixed/variable processing time of task i performed in unit j
ψ_r^{Ini}	Initial availability of utility r
$(\psi_{pr}^{Var}, t_{pr}^{\psi})$	Amount/time of p^{th} change in availability of utility r

3.3 PROPOSED APPROACH

While direct sequencing relations between tasks exist in the sequential environment, this is not the case in the network environment. This is because different tasks may share one or more materials as their feed or product, and generally multiple batches are executed for each task. For example, upstream and downstream relations between tasks might not exist in some networks (Figure 3.4(a)), and sequencing relations between multiple executed batches are not as straightforward (Figure 3.4(b)). Hence, a different approach is needed to identify relative sequence of batches to achieve feasible material balance in the final solution.



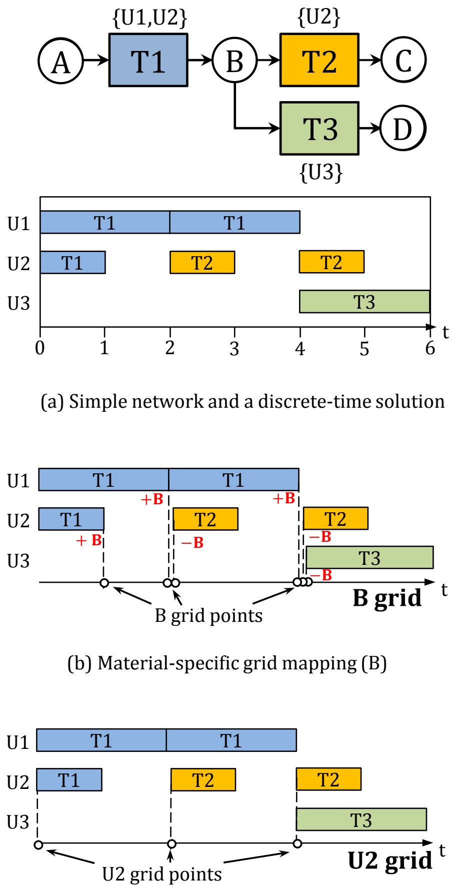
(a) Example network

(b) Example schedule

Figure 3.4: Illustration of difficulty in identifying relative sequence between batches of tasks: (a) Example network, and (b) Example schedule.

The proposed solution method consists of three stages: (i) obtain discrete-time solutions using a discrete-time MIP model, (ii) map the solution on to continuous-time grids using the mapping algorithm, and (iii) re-solve for the timing and size of each batch using a continuous-time LP model. The second stage mapping algorithm identifies the relative sequence between batches, and maps the discrete-time solution onto the two types of continuous-time grids that we introduce: material- and unit-specific grids. The two grids have different purposes.

Material-specific grids are introduced to ensure material balance in the refined solution. The batches of tasks (i.e. executed batches of given tasks) that produce or consume a material are mapped onto the material grid. For instance, assuming Figure 3.5a is the discrete-time solution that we obtained by solving an instance of the sample network (Figure 3.2), batches of tasks that produce (i.e. T1) and consume (i.e. T2, T3) material B are mapped onto material B grid as shown in Figure 3.5b. Since a task may produce and consume one or more materials, each executed batch may be mapped onto multiple material-specific grids. During the mapping process, the



(a) Simple network and a discrete-time solution

(b) Material-specific grid mapping (B)

(c) Unit-specific grid mapping (U2)

Figure 3.5: An illustration of the mapping algorithm: (a) a simple network and a discrete-time solution, and mapping of the solution to (b) material-specific grid and grid points, and (c) unit-specific grid and grid points.

sequence relations of batches of tasks that consume and produce each material are identified and recorded by the algorithm.

Unit-specific grids are introduced to ensure unit utilization constraint in the refined solution. The batches of tasks that have been assigned to each unit in the first stage discrete-time solution are mapped onto the unit grid. The algorithm identifies and records the assignment of batches to units and their relative sequence within each unit. For instance, the batches of tasks shown in the discrete-time solution in Figure 3.5a are mapped onto U2 grid in Figure 3.5c.

The batches that are mapped to the grids are assigned specific material or unit grid point (e.g. (k, m) or (j, m)), and the assignment and sequencing relations are stored in the algorithmic sets and parameters (e.g. $\mathbf{M}_k^P$, f_{ikm}^1, ...) (see Section 3.4.2 for details).

The identified assignment and sequencing relations are used to ensure feasible material balance and unit utilization in the third stage model. Feasible material balance is achieved by enforcing batches of tasks that produce a material to finish before the start of the next upcoming batch that consumes the material. Feasible unit utilization is achieved by ensuring that the batches assigned to a unit follow the same sequence and do not overlap in the refined solution. Figure 3.6 shows an overview of the proposed method.

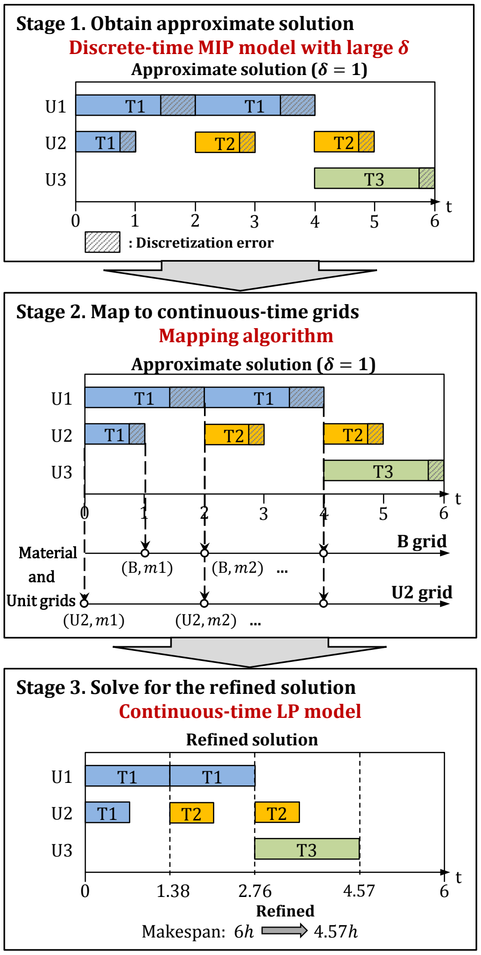


Figure 3.6: An illustration of the three stages of discrete continuous algorithm.

3.4 DISCRETE-CONTINUOUS ALGORITHM

3.4.1 *First stage discrete-time MIP scheduling model*

We use the STN model proposed by Shah et al. (1993) as the first stage discrete-time scheduling model. The model adopts an uniformly discretized (i.e. intervals of δ) common time grid. We introduce a set of time points denoted as $n \in \mathbf{N} = \{0, 1, 2, ..., \bar{H}\}$. We define time period n as the interval between time points $n-1$ and n. We assume that the processing times are fixed as the maximum processing times (i.e. processing times with the largest batch sizes) to ensure feasibility. If needed, variable processing times can be considered by either using subtasks, where each task is divided into multiple subtasks which can process different range of batch sizes with different processing times (Kondili et al., 1993; Sundaramoorthy and Maravelias, 2011b); or by using a mixed-time grid, where variable processing times are mapped onto a discrete-time grid (Maravelias, 2005). The converted data in terms of δ, followed by the decision variables of the model are as follows.

Parameters for discrete-time grid

δ Step size for discretization of time

$\bar{H}$ Horizon length (i.e. $\bar{H} = \lfloor H/\delta \rfloor$)

$\bar{\xi}_{kn}$ delivery (<0) or demand (>0) amount of material k at time point n (i.e. $\bar{\xi}_{kn} = \xi_{kp} \ \forall (n,p)$: $n = \lceil t^{\xi}_{kp}/\delta \rceil$ for delivery and $n = \lfloor t^{\xi}_{kp}/\delta \rfloor$ for demand)

$\bar{\tau}_{ij}$ Estimated processing time of task i performed in unit j (i.e. $\bar{\tau}_{ij} = \lceil (\tau^{F}_{ij} + \beta^{max}_{j} \tau^{V}_{ij})/\delta \rceil$)

Variables

$X_{ijn} \in \{0,1\}$ =1 if task i begins in unit j at time point n

$B^{D}_{ijn} \in \mathbb{R}_{+}$ Batch size of task i processed in unit j starting at time point n

$MS \in \mathbb{R}_{+}$ Makespan

$S^{D}_{kn} \in \mathbb{R}_{+}$ Inventory of material k during time period n

The model consists of unit utilization (Eq. 3.1), unit capacity (Eq. 3.2), and material balance

(Eq. 3.3) constraints.

$$\sum_{i \in \mathbf{I}_j} \sum_{n' \geq n - \bar{\tau}_{ij} + 1}^{n} X_{ijn'} \leq 1 \quad \forall\, j, n \tag{3.1}$$

$$\beta_j^{min} X_{ijn} \leq B_{ijn}^D \leq \beta_j^{max} X_{ijn} \quad \forall\, i, j \in \mathbf{J}_i, n \tag{3.2}$$

$$S_{k(n+1)}^D = S_{kn}^D + \sum_{i \in \mathbf{I}_k^+} \rho_{ik} \sum_{j \in \mathbf{J}_i} B_{ij(n-\bar{\tau}_{ij})}^D + \sum_{i \in \mathbf{I}_k^-} \rho_{ik} \sum_{j \in \mathbf{J}_i} B_{ijn}^D - \bar{\bar{\xi}}_{kn} \quad \forall\, k, n \tag{3.3}$$

In this work, we consider three objective functions: minimization of makespan (Eq. 3.4), cost (Eq. 3.5), and maximization of profit (Eq. 3.6).

$$min\ MS: \ MS \geq \sum_i \sum_{j \in \mathbf{J}_i} (n + \bar{\tau}_{ij}) X_{ijn} \quad \forall\ n \tag{3.4}$$

$$min\ \sum_i \sum_{j \in \mathbf{J}_i} \sum_n (\alpha_{ij}^F X_{ijn} + \alpha_{ij}^V B_{ijn}^D) \tag{3.5}$$

$$max\ \sum_k \pi_k (S_{k(\bar{H}+1)}^D + \sum_n \bar{\bar{\xi}}_{kn}) - \sum_i \sum_{j \in \mathbf{J}_i} \sum_n (\alpha_{ij}^F X_{ijn} + \alpha_{ij}^V B_{ijn}^D) \tag{3.6}$$

In the first stage of the solution refinement method, we solve for the optimal schedule using the STN model.

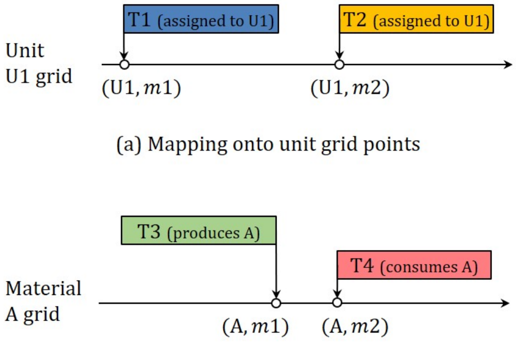
3.4.2 *Second stage mapping algorithm*

The mapping algorithm presented in this section maps the discrete-time solution (i.e. X_{ijn}) onto the material- and unit-specific grids and extracts all the necessary information (e.g. batch-unit assignment, sequencing of batches of tasks within each material and unit grid) which will be used later in the third stage continuous-time LP model. We introduce $m \in \mathbf{M}$ to denote the set of grid points on the unit- and material-specific grids.

Index pair (j, m) is used to denote the m^{th} grid point on unit j grid. Figure 3.7a shows an example of how two batches of tasks (i.e. T1 and T2) that have been assigned to unit U1 are mapped onto the unit U1 grid. As can be seen from the figure, the start of the batches are mapped to a grid point.

Index pair (k, m) is used to denote the m^{th} grid point on the material A grid. Figure 3.7b shows an example of batches of tasks that produce (i.e. T3) and consume (i.e. T4) material

A being mapped onto material A grid. Since materials are produced at the end of a task and consumed at the beginning, we map the end of the batch that produce material A and the start of the batch that consume to the material grid. In addition, we introduce $\mathbf{M}_A^P/\mathbf{M}_A^C$ to denote the set of material grid points on material A grid at which the material is produced/consumed.



(a) Mapping onto unit grid points

(b) Mapping onto material grid points

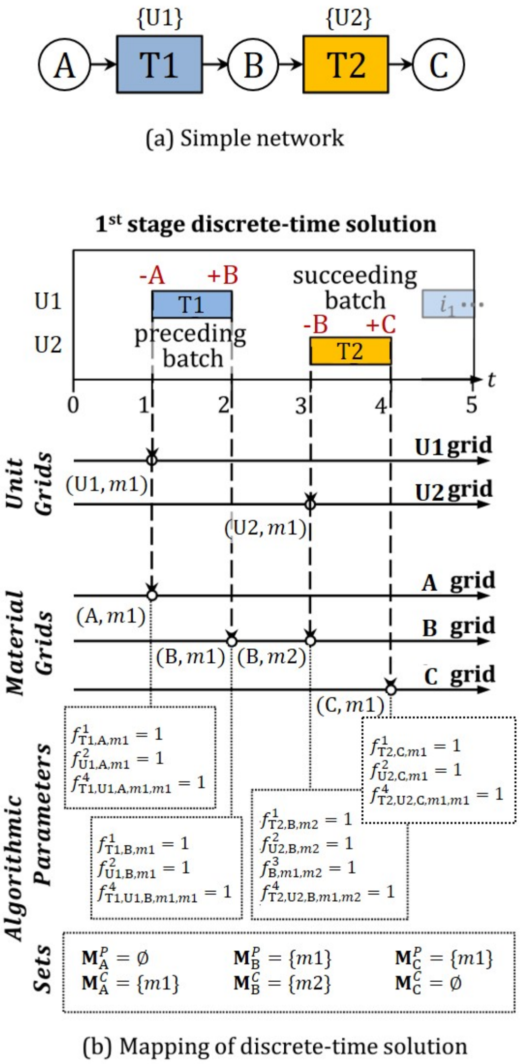
Figure 3.7: Illustration of mapping onto (a) unit and (b) material grid points.

We introduce several algorithmic parameters to store all the necessary information extracted from the discrete-time solution:

Algorithmic parameters

M_j^U	Number of tasks performed in unit j
M_k^M	Number of tasks that produce/consume material k performed
$f_{ikm}^1 \in \{0,1\}$	One if a batch of task i is mapped to material grid point (k,m)
$f_{jkm}^2 \in \{0,1\}$	One if a batch mapped to material grid point (k,m) is performed in unit j
$f_{kmm'}^3 \in \{0,1\}$	One if point where material k is produced (i.e. m) is before point where it is consumed (i.e. $m^{'}$)
$f_{ijkmm'}^4 \in \{0,1\}$	One if a batch of task i is mapped to unit grid point (j,m) and material grid point $(k,m^{'})$

Information regarding assignments of batches of tasks to unit- and material-specific grids is stored in parameters f_{ikm}^1, f_{jkm}^2 and $f_{ijkmm'}^4$. In addition, relative location of batches assigned to each grid is stored using index m. On the other hand, information regarding sequencing of *preceding* and *succeeding batches* for a given material k is stored in parameter $f_{kmm'}^3$. Given a batch of task $i \in \mathbf{I}_k^-$, we define the *preceding batches* as batches that produce material k before the given batch starts. The batch that consumes k is then defined as the *succeeding batch* for those preceding batches. See Figure 3.8 for an illustrative example of mapping and the obtained algorithmic sets and parameters.



(a) Simple network

(b) Mapping of discrete-time solution

Figure 3.8: Illustrative example of the mapping algorithm performed on a simple network: (a) a simple network, and (b) a discrete-time solution and its mapping.

The mapping algorithm is shown in Table 3.1. After it is initialized in line 0, the algorithm scans through all the time points starting from $n = 0$ to the end of the horizon (line 1). For each material k (line 2), it searches for batches of tasks that produce and consume the material (line 3-4 and 15-16). If it detects one (line 5 and 17), it makes the appropriate mapping to the unit- and material-specific grids by activating the algorithmic parameters (lines 6-11 and 18-24). As can be seen in line 5 and 17, the grid points that produce a given material are always mapped before the points that consume, if both are mapped from the same time point n.

Table 3.1: Mapping algorithm

```
0:  **Initialize:** $M_j^U = 0 \ \forall j, \ M_k^M = 0 \ \forall k, \ f^{1 \sim 4} = 0, \ \mathbf{M}_k^P = \mathbf{M}_k^C = \emptyset \ \forall k$
1:  **Loop** $n \in \mathbf{N}$
2:    **Loop** $k \in \mathbf{K}$
3:      **Loop** $i \in \mathbf{I}_k^+$
4:        **Loop** $j \in \mathbf{J}_i$
5:          **If** $X_{ij(n-\bar{\tau}_{ij})} = 1$
6:            $M_k^M \leftarrow M_k^M + 1$
7:            $M_j^U \leftarrow \sum_{n' < n} \sum_{i' \in \mathbf{I}_j} X_{i'jn'}$
8:            $\mathbf{M}_k^P \leftarrow \mathbf{M}_k^P \cup \{M_k^M\}$
9:            $f^1_{ik,m(=M_k^M)} \leftarrow 1$
10:           $f^2_{jk,m(=M_k^M)} \leftarrow 1$
11:           $f^4_{ijk,m(=M_j^U),m'(=M_k^M)} \leftarrow 1$
12:         **EndIf**
13:     **EndLoop x2**
15:     **Loop** $i \in \mathbf{I}_k^-$
16:       **Loop** $j \in \mathbf{J}_i$
17:         **If** $X_{ijn} = 1$
18:           $M_k^M \leftarrow M_k^M + 1$
19:           $M_j^U \leftarrow \sum_{n' \leq n} \sum_{i' \in \mathbf{I}_j} X_{i'jn'}$
20:           $\mathbf{M}_k^C \leftarrow \mathbf{M}_k^C \cup \{M_k^M\}$
21:           $f^1_{ik,m(=M_k^M)} \leftarrow 1$
22:           $f^2_{jk,m(=M_k^M)} \leftarrow 1$
23:           $f^3_{kmm'} \leftarrow 1 \ \forall m < M_k^M, \ m \in \mathbf{M}_k^P, \ m' = M_k^M$
24:           $f^4_{ijk,m(=M_j^U),m'(=M_k^M)} \leftarrow 1$
25:         **EndIf**
26: **EndLoop x4**
```

3.4.3 *Third stage continuous-time LP refinement model*

Following are the non-negative continuous decision variables used in the model:

Variables

$B^C_{km} \in \mathbb{R}_+$	Batch size of the task mapped to material grid point (k, m)
$S^C_{km} \in \mathbb{R}_+$	Inventory level of material k at material grid point (k, m)
$T^U_{jm} \in \mathbb{R}_+$	Timing of unit grid point (j, m)
$T^M_{km} \in \mathbb{R}_+$	Timing of material grid point (k, m)

The following constraint enforces the sequencing between preceding and succeeding batches. Given a material grid, the finish time of preceding batches should be less than or equal to the start time of the succeeding batches.

$$T^M_{km} \leq T^M_{km'} \quad \forall k, m, m^{'} : f^3_{kmm'} = 1 \tag{3.7}$$

We enforce the sequencing between the batches of tasks assigned to the same unit using the following constraint. It takes variable processing time into account.

$$T^U_{jm} + (\tau^F_{ij} + \tau^V_{ij} B^C_{km'}) \leq T^U_{j(m+1)} \quad \forall i, j, k, m, m^{'} : f^4_{ijkmm'} = 1 \tag{3.8}$$

Since we mapped each batch onto both unit- and material-specific grids, we need the time matching constraint between unit and material grid points. Eq. 3.9-3.10 ensure that the timing of each batch is identical across all grids. We write the constraint for batches that consume (Eq. 3.9) and produce (Eq. 3.10) material k separately, depending on how they are mapped onto the grids (see Subsection 3.4.2).

$$T^U_{jm} = T^M_{km'} \quad \forall i, j, k, m, m^{'} \in \mathbf{M}^C_k : f^4_{ijkmm'} = 1 \tag{3.9}$$

$$T^U_{jm} = T^M_{km'} - (\tau^F_{ij} + \tau^V_{ij} B^C_{km'}) \quad \forall i, j, k, m, m^{'} \in \mathbf{M}^P_k : f^4_{ijkmm'} = 1 \tag{3.10}$$

The unit capacity and batch size matching constraints are enforced by Eq. 3.11-3.12. Batch size matching constraint is necessary, because a single batch of a task can be mapped onto multiple material grids since it may consume and produce one or more materials.

$$\beta_j^{min} \leq B_{km}^C \leq \beta_j^{max} \;\; \forall j,k,m : f_{jkm}^2 = 1 \tag{3.11}$$

$$B_{km'}^C = B_{k'm''}^C \;\; \forall i,j,k,k^{'},m,m^{'},m^{''} : f_{ijkmm'}^4 = f_{ijk'mm''}^4 = 1,\; k < k^{'} \tag{3.12}$$

The inventory balance equation is written such that the inventory level of material k at grid point m is equal to the sum of initial inventory and the amount produced/consumed up to that point. However, it is only written for grid points at which the material is consumed (i.e. $m \in \mathbf{M}_k^C$), to ensure inventory levels do not drop below zero.

$$S_{km}^C = \gamma_k + \sum_{m > m' \in \mathbf{M}_k^P} \sum_{i \in \mathbf{I}_k^+ : f_{ikm'}^1 = 1} \rho_{ik} B_{km'}^C + \sum_{m \geq m' \in \mathbf{M}_k^C} \sum_{i \in \mathbf{I}_k^- : f_{ikm'}^1 = 1} \rho_{ik} B_{km'}^C \;\; \forall k, m \in \mathbf{M}_k^C \tag{3.13}$$

Assuming all the demand is at the end of the scheduling horizon, Eq. 3.14 ensures that the final inventory of each material is greater than or equal to the total demand. Extension to intermediate demands will be presented in Section 3.5.

$$S_{k,M_k^M}^C \geq \sum_p \xi_{kp} \;\; \forall k \tag{3.14}$$

For makespan minimization, we add the following constraint to ensure that the makespan is greater than or equal to the time each material is produced.

$$min MS : MS \geq T_{km}^M \;\; \forall k, m \in \mathbf{M}_k^P \tag{3.15}$$

We minimize total processing cost for cost minimization.

$$min \sum_i \sum_{j \in \mathbf{J}_i} \sum_k \sum_{\substack{m \in \mathbf{M}_k^P \\ f_{ikm}^1 = f_{jkm}^2 = 1}} \frac{1}{|K_i^+|} (\alpha_{ij}^F + \alpha_{ij}^V B_{km}^C) \tag{3.16}$$

We negate the effect of double counting the cost of tasks that are mapped to multiple material grids by dividing the whole term by the number of the materials produced by each given task.

When maximizing profit, we assume that all the final products can be sold at the end of the time horizon.

$$max \sum_{k} \pi_k S^C_{k,M^M_k} - \sum_{i} \sum_{j \in \mathbf{J}_i} \sum_{k} \sum_{\substack{m \in \mathbf{M}^P_k \\ f^1_{ikm}=f^2_{jkm}=1}} \frac{1}{|\mathbf{K}^+_i|} (\alpha^F_{ij} + \alpha^V_{ij} B^C_{km}) \tag{3.17}$$

The first stage MIP model may yield a solution which, although optimal if the approximate times are used, leads to a suboptimal solution for the original instance after the third phase is executed. In these cases, we employ a relaxation of the original problem in the first stage; specifically, we formulate a model using a longer scheduling horizon and potentially longer deadlines (see discussion in Subsection 3.5.3).

3.5 EXTENSIONS

3.5.1 *Intermediate deliveries and demands*

In cases where intermediate deliveries and demands are present, modifications to the algorithm (refer to Appendix B.1) and the third stage LP model are made. In terms of material grid k, intermediate deliveries and demands can simply be treated as batches of production and consumption tasks, respectively. The only differences are that these tasks are not assigned to any units, and have zero processing times. Otherwise, they are treated as other batches of production and consumption tasks, and thus the sequencing constraints (identified by the algorithm) are enforced in the third stage LP model. We introduce an additional binary algorithmic parameter, f^5_{kmp}, which gets assigned a value of one if the p^{th} delivery or demand occurs at material grid point (k, m).

Unlike the timing of actual batches that are mapped to unit- and material-specific grids and are optimized in the third stage LP model, the timings of intermediate deliveries and demands must be fixed to the values given in the problem data.

$$T^M_{km} = t^{\xi}_{kp} \;\; \forall \, k, m, p : f^5_{kmp} = 1 \tag{3.18}$$

The inventory balance constraint now includes deliveries and demands. Furthermore, the demand satisfaction constraint (Eq. 3.14) is removed.

$$S^C_{km} = \gamma_k + \sum_{\substack{m' < m \\ m' \in \mathbf{M}^P_k}} \Big(\sum_{\substack{i \in \mathbf{I}^+_k \\ f^1_{ikm'} = 1}} \rho_{ik} B^C_{km'} - \sum_{p : f^5_{km'p} = 1} \xi_{kp} \Big) + \sum_{\substack{m' \leq m \\ m' \in \mathbf{M}^C_k}} \Big(\sum_{\substack{i \in \mathbf{I}^-_k \\ f^1_{ikm'} = 1}} \rho_{ik} B^C_{km'} - \sum_{p : f^5_{km'p} = 1} \xi_{kp} \Big) \qquad \forall k, m \in \mathbf{M}^C_k \tag{3.19}$$

3.5.2 *Limited shared utilities*

Limited shared utility can be modeled in the first stage MIP model by adding the following utility balance constraint.

$$U^D_{rn} = \sum_i \sum_{j \in \mathbf{J}_i} \sum_{n' \geq n - \bar{\tau}_{ij}}^{n-1} \rho^U_{ir} X_{ijn'} \leq \bar{\psi}_{rn} \quad \forall\, r, n \tag{3.20}$$

where U^D_{rn} is the amount of utility r being consumed during time period n, and $\bar{\psi}_{rn}$ is the total amount of utility r available during time period n (see Nomenclature).

Utilities are consumed at the beginning of each task, and the same amounts are released at the end. Thus, in principle, the utilities can be treated similar to materials, except that the type and the amount of utility being consumed and produced (released) by a given task are always the same.

Following the same idea as with the material-specific grids, we introduce utility-specific grids onto which the batches of tasks that consume and release utilities are mapped. The mapping algorithm identifies the relative sequence of consumption and release of utility on each utility grid, which then will be used in the third stage LP model to ensure feasible utility balance. The time-varying availability of utility can be modeled similar to intermediate deliveries and demands of utility: increase and decrease in availability are modeled as utility being delivered and shipped, respectively.

We use utility grid point (r, m) to denote the m^{th} grid point on utility r grid. Furthermore, we introduce the following algorithmic parameters to store the aforementioned information. See Appendix B.2 for the mapping algorithm.

Algorithmic parameters for limited shared utilities

M_r^R	Number of tasks that consumed or produced utility r
$f^6_{irm} \in \{0,1\}$	One if task i is mapped to utility grid point (r,m)
$f^7_{rmm^{'}} \in \{0,1\}$	One if point where utility r is released (i.e. m) is before point where it is consumed (i.e. $m^{'}$)
$f^8_{ijrmm^{'}} \in \{0,1\}$	One if task i is mapped to unit grid point (j,m) and utility grid point $(r,m^{'})$
$f^9_{rmp} \in \{0,1\}$	One if p^{th} change in the availability of utility r occurs at utility grid point (r,m)

In the third stage LP model, we introduce a non-negative continuous variable T^R_{rm} to denote the time of the utility grid point (r,m). The following constraint ensures the sequencing between the utility release and consumption times.

$$T^R_{rm} \leq T^R_{rm^{'}} \quad \forall\, m, m^{'}, r : f^7_{rmm^{'}} = 1 \tag{3.21}$$

We match the timing of utility grid points and unit grid points through Eqs. 3.22,3.23.

$$T^U_{jm} = T^R_{rm^{'}} \quad \forall i, j, r, m, m^{'} \in \mathbf{M}^{R,C}_r : f^8_{ijrmm^{'}} = 1 \tag{3.22}$$

$$T^U_{jm} = T^R_{rm^{''}} - (\tau^F_{ij} + \tau^V_{ij} B^C_{km^{'}}) \quad \forall i, j, k, r, m, m^{'}, m^{''} \in \mathbf{M}^{R,P}_r : f^4_{ijkmm^{'}} = f^8_{ijrmm^{''}} = 1 \tag{3.23}$$

We fix the times at which the availability of utilities change to the times given by the problem data.

$$T^R_{rm} = t^{\psi}_{pr} \quad \forall\, m, p, r : f^9_{rmp} = 1 \tag{3.24}$$

The utility balance constraint ensures that the utility consumption does not exceed the total availability.

$$U^C_{rm} = \psi^{Ini}_r + \sum_{\substack{m^{'}<m \\ m^{'}\in \mathbf{M}^{R,P}_r}} \Big(\sum_{\substack{i\in \mathbf{I}_r \\ f^6_{irm^{'}}=1}} \rho^U_{ir} + \sum_{p:f^9_{rm^{'}p}=1} \psi^{Var}_{pr} \Big) - \sum_{\substack{m^{'}\leq m \\ m^{'}\in \mathbf{M}^{R,C}_r}} \Big(\sum_{\substack{i\in \mathbf{I}_r \\ f^6_{irm^{'}}=1}} \rho^U_{ir} - \sum_{p:f^9_{rm^{'}p}=1} \psi^{Var}_{pr} \Big) \quad \forall\, r, m \in \mathbf{M}^{R,C}_r \tag{3.25}$$

where U^C_{rm} is a non-negative continuous variable that denotes the amount of available utility r at utility grid point (r,m).

3.5.3 *Infeasibility of first-stage model*

Given a feasible instance (i.e. an instance for which a feasible solution exists if the accurate processing times are used), the approximation of processing times in the first stage yields a model which is also feasible for most problem classes. In profit maximization problems with no orders or orders that can be met easily (so excess capacity is used for excess sales), the first stage problem is always feasible. In makespan minimization problems, which typically have no deadlines, the first stage problem will always be feasible provided that a sufficiently long horizon is considered. The first stage problem can become infeasible when a set of orders with tight deadlines have to be met at, say, minimum cost. First, we note that this case can be readily addressed by replacing deadlines with due dates and introducing backlogs, something which is in fact necessary and routinely done in practice, regardless of the model employed, simply because the given deadlines may also lead to an infeasible continuous-time model. Second, if the specific case where tight deadlines have to be considered, then a feasible first stage model is obtained by increasing the horizon and deadlines. This "relaxation" allows for the identification of first stage solutions which, although violate the original due dates, will lead to feasible solutions in the third phase of the algorithm.

In summary, all problems classes and instances can be addressed through backlog introduction and/or horizon and deadline relaxation. We also note that these two methods can be used to not only address the first stage model infeasibility, but to also find first-stage decisions that lead to better third-stage solutions. Therefore, we plan to study them in more depth in the future.

3.6 APPLICATIONS

All the runs in this section are performed using CPLEX 12.6.2 via GAMS 24.5.4 on a cluster with Intel Xeon (E5520) processors at 2.27 GHz and 16GB of RAM running on CentosOS Linux 7. Default options with resource limit of 5h are used unless stated otherwise.

3.6.1 *Illustrative examples*

We consider the benchmark example from Kondili et al. (1993) to illustrate the accuracy and versatility of the proposed method. For simplicity, we consider makespan minimization objective.

In the first example, we modify the processing times such that they have highly irregular fractional values (see Table 3.4), in which discrete-time formulations cannot efficiently handle without significant loss in the solution accuracy. We assume that the processing times are fixed. The horizon length is 40h, and 100kg of P1 and 200kg of P2 are demanded at the end of the horizon. The network structure and other process information can be found in the supplementary information.

Table 3.4: Processing times of example 1, τ_{ij} [h]

	Heater	Reactor1	Reactor2	Still
	0-100(kg)	0-80(kg)	0-50(kg)	0-200(kg)
Heating	1.35			
Rxn 1		2.71	3.81	
Rxn 2		1.97	2.31	
Rxn 3		1.11	0.94	
Separ.				2.55

We solve this example with the proposed method using $\delta = 0.5h$ in the first stage. We also solve this example with a continuous-time model proposed by Sundaramoorthy and Karimi (2005), which is known as one of the best performing continuous-time models in the literature. The computational results are shown in Table 3.5. We do not show the computational statistics of the third stage LP model (solved in <0.01s). The Gantt chart for the first stage solution and the refined solution are shown in Figure 3.9.

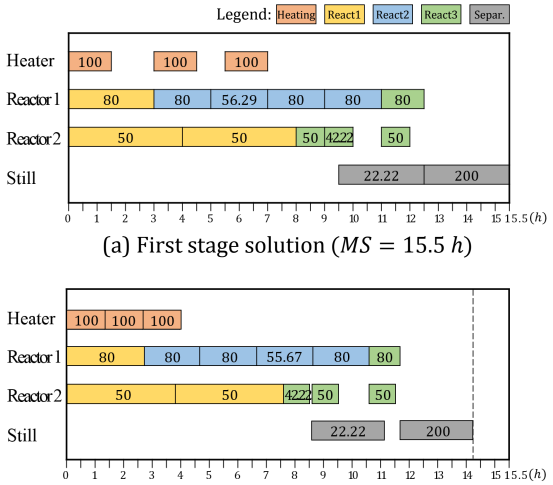
The results show that both the proposed method (Discrete-Continous Algorithm, or DCA) and the continuous-time model (CM) obtained the same objective of 14.25h. However, while CM failed to prove optimality in 5h, DCA obtained the same objective in less than a second (refined from 15.5h). If this example was to be solved solely with a discrete-time model to optimality, $\delta = 0.01h$ has to be used making the problem intractable. Here, we used a significantly larger δ, and still were able to obtain a solution as good as the one obtained by CM. From this example, we can see how effective the proposed method is in handling unfavorable and irregular processing times, one of the largest disadvantages of discrete-time formulations.

Table 3.5: Computational statistics of example 1

	DCA	CM
δ or #Slots	0.5	15
Constraints	2711	1264
Discrete Variables	648	176
Continuous Variables	2063	1087
LP relaxation	0.963	9.151
Objective (h)	15.5	14.25^a
(Refined)	14.25	
CPU (s)	0.91	18000.42^*
Nodes	37	$> 10^7$
Gap (%)	-	7.64

[a] *Found after 341.70 s*

**Reached resource limit*



(a) First stage solution ($MS = 15.5\ h$)

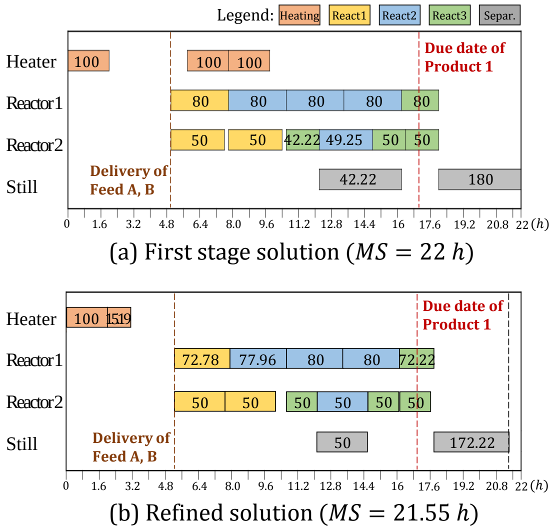
(b) Refined solution ($MS = 14.25\ h$)

Figure 3.9: Gantt chart of (a) first stage solution, and (b) refined solution of example 1. The numbers in the charts represent batch sizes (kg).

In the second example, we consider the same network with both variable processing times and intermediate deliveries and demands. The initial inventory of the raw materials F1, F2 and F3 are 500, 0 and 0kg, respectively, and 500kg of F2 and F3 are delivered at 5h. Demands for P1 and P2 are 100kg at 17h and 200kg at the end of the horizon, respectively. Other process information can be found in the supplementary information. Since CM cannot accommodate intermediate delivery and demands, only DCA is used. The results are shown in Table 3.6 and Figure 3.10. As shown in the results, both variable processing times and intermediate delivery and demands can be readily implemented without any additional computational cost.

Table 3.6: Computational statistics of example 2

	DCA
δ	0.2
Constraints	6735
Discrete Variables	1608
Continuous Variables	3428
LP relaxation	0.826
Objective (h)	22
(Refined)	21.552
CPU (s)	4.33
Nodes	1950
Gap (%)	-



(a) First stage solution ($MS = 22\ h$)

(b) Refined solution ($MS = 21.55\ h$)

Figure 3.10: Gantt chart of (a) first stage solution, and (b) refined solution of example 2. The numbers in the charts represent batch sizes (kg).

In example 3, we consider the same network as example 2 with limited utility availability. Two type of resources are available: high pressure steam (HS) and cooling water (CW). The utility requirement for each task is shown in Table 3.7. We solve two instances: (i) fixed availability of HS (10 units/h) and CW (8 units/h), and (ii) time-varying availability of HS and CW (see Figure 3.11 for details). The first instance is solved with both DCA and CM, while solely DCA is used to solve the second instance, because CM cannot be effectively modified to model time-varying availability of shared resources. The results are shown in Table 3.8. The Gantt charts of the first stage and the refined solutions for both instances are shown in Figure 3.11.

In the first instance, DCA obtained the same objective in around 6s, while CM took more

Table 3.7: Utility requirements of example 3, ρ_{ir}^{U} [units/h]

	Heating	Rxn 1	Rxn 2	Rxn 3	Separ.
HS	10				5
CW		4	2	2	3

Table 3.8: Computational statistics of example 3

Utility Availability	Fixed		Time-varying
	DCA	CM	DCA
δ or #Slots	0.2	15	0.2
Constraints	7137	1294	7137
Discrete Variables	1608	176	1608
Continuous Variables	3830	1125	3830
LP relaxation	0.395	11.359	0.412
Objective (h)	17	16.552[a]	18.8
(Refined)	16.552		18.052
CPU (s)	6.22	18000.11*	9.50
Nodes	967	$> 10^7$	2206
Gap (%)	-	11.05	-

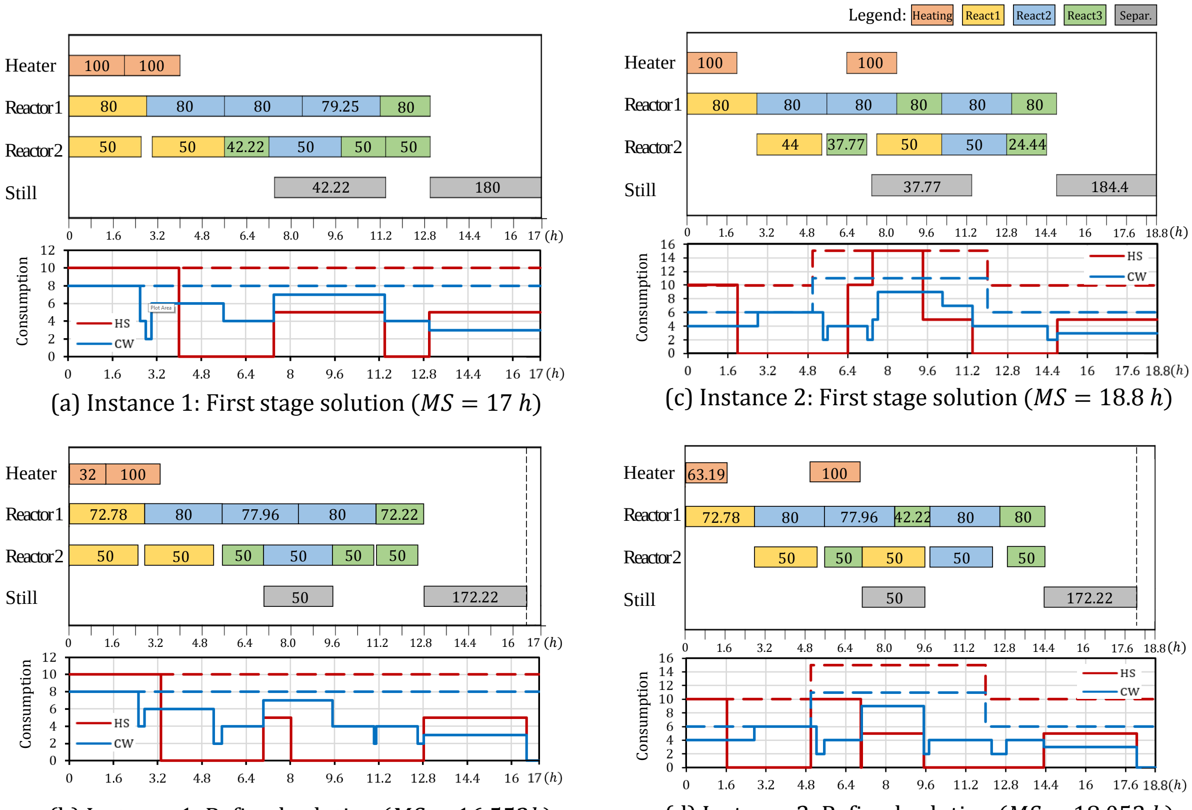
[a] *Found after 40.73 s*

* *Reached resource limit*

than 40 s to find the solution and failed to close the gap within 5h. In the second instance, the lower amount of CW available at the beginning of the horizon led to a more constrained problem (although the availability increases after 5h), leading to a longer makespan of 18.052h. However, no additional computational effort was needed to consider time-varying availability of shared resources.

3.6.2 *Computational performance*

In this section we study the computational performance of the proposed method on difficult examples based on larger networks and longer horizons. We consider three benchmark examples from the literature. In the first example (denoted as K), we consider a modified network from Kallrath (2002) with makespan minimization as the objective. In the second (denoted as MP), we consider a modified network from Maravelias and Papalamprou (2009) with cost minimization as the objective. Lastly, we consider a modified network from Papageorgiou and Pantelides (1996b) with profit maximization as the objective (denoted as PP). The STN representation of the net-



(a) Instance 1: First stage solution ($MS = 17\ h$)

(c) Instance 2: First stage solution ($MS = 18.8\ h$)

(b) Instance 1: Refined solution ($MS = 16.552h$)

(d) Instance 2: Refined solution ($MS = 18.052\ h$)

Figure 3.11: Gantt chart of first stage and refined solutions of the first (i.e. (a), (b)) and second (i.e. (c), (d)) instances of example 3. The numbers in the charts represent batch sizes (kg).

works and their complete process information can be found in the supplementary information.

The discretization step lengths of DCA for each example is chosen such that the first stage MIP models are tractable, and the number of time slots for CM are chosen such that they are most likely sufficient to represent the optimal solutions. The first stage MIP models of examples K and MP are solved to optimality without relaxing the horizon length. In example PP, we allowed 5% optimality gap and relaxed the horizon length by 5%. The computational statistics are shown in Table 3.9. To visualize the performance of both models, the evolution of the upper and lower bounds of CM over normalized time (with respect to the CPU time of DCA) are shown in Figure 3.12. To illustrate the complexity of the examples, optimal schedule for example PP is shown in Figure 3.13.

The results show the effectiveness of the proposed solution method in obtaining the best known or close to best known solution in a very short amount of time. As can be seen from Table 3.9, although CM is significantly smaller in size compared to DCA, it was unable to close

Table 3.9: Computational statistics of examples K, MP and PP

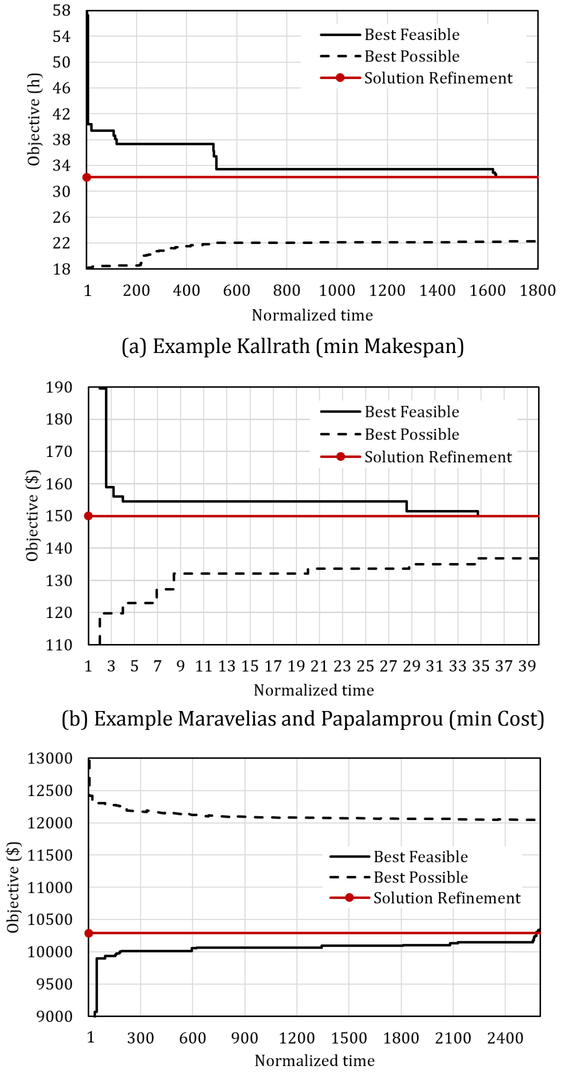
	K (min Makespan)		MP (min Cost)		PP (max Profit)	
	DCA	CM	DCA	CM	DCA	CM
δ or #Slots	1	20	2	20	2	40
Constraints	6359	4891	7092	7118	5105	9377
Discrete Variables	1775	671	2257	968	1408	1192
Continuous Variables	3145	3891	3994	5643	3164	7784
LP relaxation	1.43	18.08	118.49	118.49	10786.44	12419.48
Objective (h or $)	37	32.2	150	150	10336.08	10488.41
(Refined)	32.2		150		10289.44	
CPU (s)	10.00	18000.22*	2.41	18000.32*	2.89	18000.13*
Nodes	2408	432469	48327	80723	383	92988
Gap (%)	-	30.90	-	1.75	4.01	14.60

**Reached resource limit*

the gap within 5h for all examples. Furthermore, Figure 3.12 shows that CM required significant amount of time to find the objective that was found almost instantaneously by DCA. Specifically for example K, DCA found a solution with an objective of 32.2h (refined from 37h) in 10s, while CM required more than 4.5h to find the same feasible solution. In example MP, DCA found the best known solution of $150 in less than 3s, while CM required around 80s. Interestingly in example PP, a solution of $10336.08 was found in the first stage of DCA and was refined to a worse objective of $10289.44. This is because of the longer horizon that was considered, which led to an overestimation of the profit in the first place. In this example, CM found a slightly better feasible solution of $10488.41 than DCA, but it took more than 2h to find a solution that is better than the one found by DCA in less than 3s.

3.7 CONCLUSIONS

Discrete-time MIP models for chemical production scheduling have two major modeling advantages: (1) the ability to handle a wide range of processing features at no additional computational cost (e.g., modeling of time-varying resource availability and cost, and intermediate shipments and orders); and (2) can be readily modified to account for additional processing features. Furthermore, recent advances in computing, general MIP solvers, and domain-specific solution methods, have enabled us to address industrial-scale problems using discrete-time formulations



(a) Example Kallrath (min Makespan)

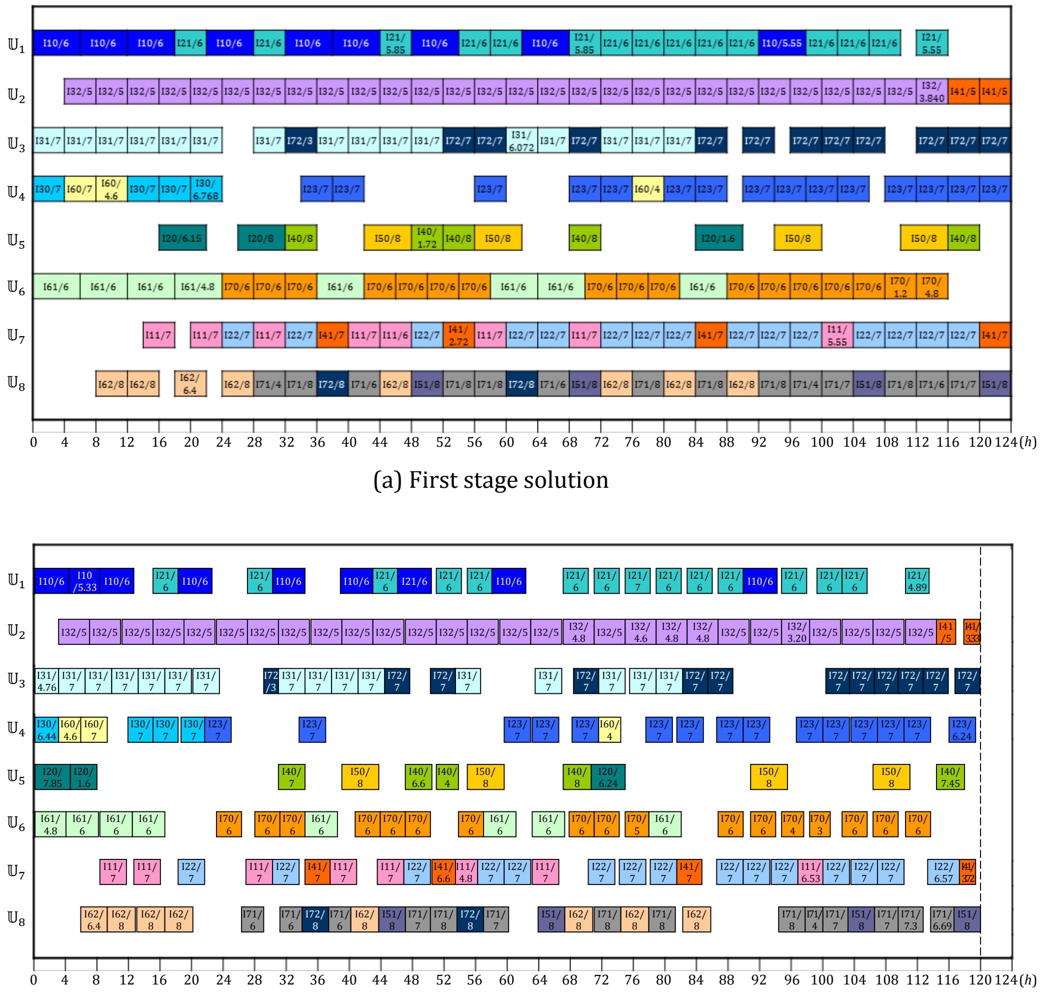
(b) Example Maravelias and Papalamprou (min Cost)

(c) Example Papageorgiou and Pantelides (max Profit)

Figure 3.12: The evolution of upper and lower bounds of CM over the normalized time for each benchmark example. CM takes more than (a) 1600, (b) 34, and (c) 2500 times the time it took for DCA to find the same objective, respectively.

in reasonable time frames. However, these models have one major disadvantage, namely, the poor accuracy of the obtained solutions, which has kept the debate open. In fact, the comparison between discrete- and continuous-time scheduling models has been one of the most debated topics in the area of chemical production scheduling, if not the entire process systems engineering.

In this chapter, we proposed a solution algorithm that allows us to exploit the advantages of both time representations, while overcoming their limitations. The method consists of three stages. In the first stage, a discrete-time model is used to quickly obtain an approximate solution.



(a) First stage solution

(b) Refined solution

Figure 3.13: Gantt chart of (a) first stage and (b) refined solutions of example PP (120h horizon). The numbers in the charts represent tasks/batch sizes (kg)

In the second stage, the first stage solution is mapped onto newly introduced unit- and material-specific grids, which are then used to formulate a continuous-time model. In the third stage, an LP model is solved to obtain an accurate, and potentially improved, solution based on the batching, assignment and relative batch sequencing information form the first stage solution.

The proposed method maintains the major advantage of discrete-time models, namely, the ability to address many classes of problems with a wide range of features, while addressing its major disadvantage, namely, the accuracy of the obtained solutions. In fact, through the conceptually new unit-, material-, and utility-specific grids, this challenge is reduced to merely solving an LP. The method was shown to return high quality solutions while being substantially faster than all previously proposed approaches.

Finally, the proposed framework offers flexibility in terms of both computational enhancements and modeling extensions. For example, the computational efficiency of the algorithm can be enhanced by carefully selecting the level of discretization in the first stage model. Accordingly, the development of problem- and instance-specific computational enhancements as well as the formulation of problem-specific models in the first and third stages of the algorithm will be the topic of the following chapters.

3.8 Nomenclature

Indices/sets

$i \in \mathbf{I}$	Tasks
$j \in \mathbf{J}$	Units
$k \in \mathbf{K}$	Materials
$m \in \mathbf{M}$	Unit, material and utility grid points
$n \in \mathbf{N}$	Time points or periods
$p \in \mathbf{P}$	Incident points
$r \in \mathbf{R}$	Utilities

Subsets

$\mathbf{I}_j$	Tasks that can be performed in unit *j*
$\mathbf{I}_k^+ / \mathbf{I}_k^-$	Tasks that produce/consume material *k*
$\mathbf{I}_r$	Tasks that require utility *r*
$\mathbf{J}_i$	Units that can perform task *i*
$\mathbf{M}_k^P / \mathbf{M}_k^C$	Material grid points at which material *k* is produced/consumed
$\mathbf{M}_r^{R,P} / \mathbf{M}_r^{R,C}$	Utility grid points at which utility *r* is released/consumed

Parameters

$\alpha_{ij}^F / \alpha_{ij}^V$	Fixed/variable cost of performing task *i* in unit *j*
$\beta_j^{min} / \beta_j^{max}$	Minimum/maximum capacity of unit *j*
γ_k	Initial inventory of material *k*
H	Horizon length
(ξ_{kp}, t_{kp}^{ξ})	Amount/time of p^{th} delivery (<0) or demand (>0) of material *k*
π_k	Price of material *k*
ρ_{ik}	Fraction of material *k* produced (>0) or consumed (<0) by task *i*
ρ_{ir}^U	Amount of utility *r* required per time period by task *i*
$\tau_{ij}^F / \tau_{ij}^V$	Fixed/variable processing time of task *i* performed in unit *j*

ψ_r^{Ini} Initial availability of utility r

$(\psi_{pr}^{Var}, t_{pr}^{\psi})$ Amount/time of p^{th} change in availability of utility r

Parameters for discrete-time grid

δ Step size for discretization of time

$\bar{H}$ Horizon length (i.e. $\bar{H} = \lfloor H/\delta \rfloor$)

$\bar{\xi}_{kn}$ delivery (<0) or demand (>0) amount of material k at time point n (i.e. $\bar{\xi}_{kn} = \xi_{kp} \;\; \forall (n,p) :$ $n = \lceil t_{kp}^{\xi}/\delta \rceil$ for delivery and $n = \lfloor t_{kp}^{\xi}/\delta \rfloor$ for demand)

$\bar{\tau}_{ij}$ Estimated processing time of task i performed in unit j (i.e. $\bar{\tau}_{ij} = \lceil (\tau_{ij}^{F} + \beta_j^{max}\tau_{ij}^{V})/\delta \rceil$)

$\bar{\psi}_{rn}$ Total amount of utility r available during time period n (i.e. $\bar{\psi}_{rn} = \psi^{Ini} + \sum_{p \in \mathbf{P}_n^{\psi}} \psi_{pr}^{Var} \; \forall n,$ where $\mathbf{P}_n^{\psi} = \{p | (\lfloor t_{pr}^{\psi}/\delta \rfloor \leq n$ and $\psi_{pr}^{Var} < 0)$ or $(\lceil t_{pr}^{\psi}/\delta \rceil \leq n$ and $\psi^{Var} > 0)\}$

Algorithmic parameters

M_j^U Number of tasks performed in unit j up to current iteration

M_k^M Number of tasks that produce/consume material k performed up to current iteration

M_r^R Number of tasks that consumed or produced utility r up to current iteration

$f_{ikm}^1 \in \{0,1\}$ =1 if a batch of task i is mapped to material grid point (k,m)

$f_{jkm}^2 \in \{0,1\}$ =1 if a batch mapped to material grid point (k,m) is performed in unit j

$f_{kmm'}^3 \in \{0,1\}$ =1 if point where material k is produced (i.e. m) is before point where it is consumed (i.e. $m^{'}$)

$f_{ijkmm'}^4 \in \{0,1\}$ =1 if a batch of task i is mapped to unit grid point (j,m) and material grid point $(k,m^{'})$

f_{kmp}^5 =1 if p^{th} delivery or demand occurs at material grid point (k,m)

$f_{irm}^6 \in \{0,1\}$ =1 if a batch of task i is mapped to utility grid point (r,m)

$f_{rmm'}^7 \in \{0,1\}$ =1 if point where utility r is released (i.e. m) is before point where it is consumed (i.e. $m^{'}$)

$f^8_{ijrmm'} \in \{0,1\}$ =1 if a batch of task i is mapped to unit grid point (j,m) and utility grid point $(r,m^{'})$

$f^9_{rmp} \in \{0,1\}$ =1 if p^{th} change in availability in utility occurs at utility grid point (r,m)

Binary variable

X_{ijn} =1 if task i begins in unit j at time point n

Non-negative continuous variables

B^C_{km} Batch size of the task mapped to material grid point (k,m)

B^D_{ijn} Batch size of task i processed in unit j starting at time point n

MS Makespan

S^C_{km} Inventory level of material k at material grid point (k,m)

S^D_{kn} Inventory of material k during time period n

T^M_{km} Timing of material grid point (k,m)

T^R_{rm} Timing of utility grid point (r,m)

T^U_{jm} Timing of unit grid point (j,m)

U^C_{rm} Amount of available utility r at utility grid point (r,m)

U^D_{rn} Amount of utility r being consumed during time period n

4

DISCRETE-CONTINUOUS ALGORITHM: MODEL PARAMETERS

4.1 MOTIVATION

As illustrated in the Chapters 3, Discrete-Continuous Algorithm (DCA) offers the modeling flexibility and computational efficiency of discrete-time models, while retaining the accuracy of continuous-time models. More importantly, the framework provides flexibility in choosing the model parameters, namely the discretization step length and horizon relaxation, which, if chosen carefully, may lead to further computational enhancements and improvements in the solution quality. However, finding the *best* set of parameters for general scheduling instances is not a trivial task.

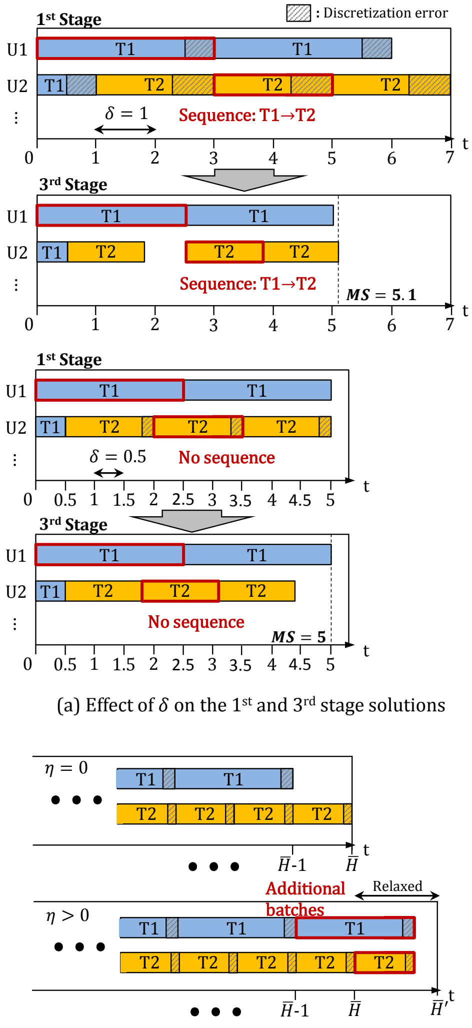
Accordingly, the goal of this work is to provide a thorough discussion on the role of these two parameters, and propose systematic methods of finding the *best* set of parameters for general scheduling instances. It is important to note that the proposed methods are applicable not only to the DCA framework, but also to any discrete-time scheduling formulations.

4.2 BACKGROUND

4.2.1 *Key user-defined parameters*

One of the major strengths of DCA lies in its flexibility. Appropriate selection of parameters δ and η, can dramatically improve the performance of the algorithm.

The choice of δ directly impacts the size of the first stage model, which is computationally the most expensive stage. While a large δ leads to a shorter computational time, it may lead to a significantly inaccurate first stage solution. A smaller δ, on the other hand, leads to better approximations of the time-related data, yielding a first stage solution that is likely to be closer to the optimal solution of the original problem. Figure 4.1a illustrates this point: although we may end up with a smaller model when $\delta = 1$ h is used, $\delta = 0.5$ h obtains a better first stage objective value (i.e. makespan of 5 h) as well as a different sequencing of batches (highlighted in red), which is one of the attributes that determines the quality of the third stage solution.



(a) Effect of δ on the 1st and 3rd stage solutions

(b) Effect of η on the 1st stage solution

Figure 4.1: Effect of the user-defined parameters on the first stage solution.

The choice of η is more relevant in profit maximization and cost minimization problems as discussed earlier. In the case of profit maximization, the number of batches in the first stage solution influences the quality of the third stage solution, because no new batches can be added in the third stage. As illustrated in Figure 4.1b, a relaxed horizon (i.e. $\eta > 0$) allows for additional batches to be executed, while the opportunity is lost in the case of no relaxation. In the case of cost minimization, appropriately chosen η allows the first stage model to find the solution with the batch-unit assignments with the lowest cost, which could be impossible if the original horizon was used, due to approximation errors.

4.2.2 *DCA solution cutoff*

One important functionality of the third stage of DCA is to *cut off* the batches in the first stage solution that are infeasible to the original problem. This is important when the horizon is relaxed in the profit maximization problem, where an excessive number of batches may be scheduled. In such cases, the cut off procedure is triggered, where the third stage model is solved with a sufficiently long horizon and a penalty term to prevent unnecessary delays, thus "pushing" the execution of batches as early as possible; and the batches that finish beyond the original horizon are identified and removed from the schedule.

4.3 MOTIVATING EXAMPLES

In this section, we illustrate how different choices of δ and η lead to widely different first and third stage solutions, and why finding the *best* combination is not as obvious as it appears.

4.3.1 *Example 1: Choosing the discretization step length*

We consider the network shown in Figure 4.2. Each task is performed on a task-dedicated unit, and the processing times are *regular* (see Table 4.1). Orders of 30 kg of both final products, C and E, are due at the end of the 40 h horizon. The objective is to minimize makespan.

To be able to provably find the best solution with a discrete-time model, we have to use a δ equal to the greatest common factor of the time-related data, i.e., $\delta = 0.1$ h. However, when using DCA, the largest δ that does not compromise the quality of the solution should be used for

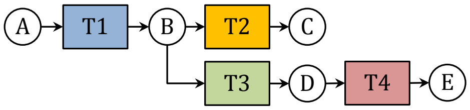


Figure 4.2: A simple network for the motivating examples.

Table 4.1: Processing times of example 1-3, τ_{ij}^{F} [h]

	Example 1				Example 2				Example 3		
	U1	U2	U3	U4	U1	U2	U3	U4	U1	U2	U3
[kg]	0-10	0-10	0-10	0-10	0-40	0-10	0-10	0-20	0-40	0-10	0-10
T1	3.1				2.5				2.5		
T2		2				2.5				2.5	
T3			4				3.9			3.9	
T4				2				5.8			5.8

computational efficiency. The interesting challenge is to figure out which δ is best suited for this case.

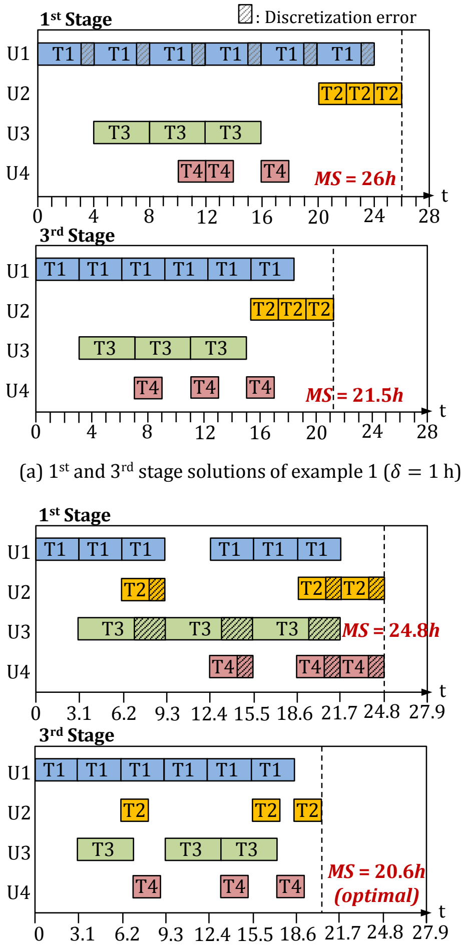
One intuitive heuristic method for finding a suitable δ, is to find a δ that divides the time-related data with the least amount of error. In this regard, we use $\delta = 1$ h to solve this example, and the obtained solutions are shown in Figure 4.3a. The first stage solution has a makespan of 26 h, which is refined to 21.5 h in the third stage. The only approximation error is in the processing time of task T1. Overall, the choice of $\delta = 1$ h seems to be reasonable in terms of approximating the problem data.

Next, we consider a more aggressive choice, $\delta = 3.1$ h. This choice seems to be inappropriate, as it leads to large approximation errors in the processing times of most tasks. Surprisingly, however, when the example is solved using $\delta = 3.1$ h, better solutions with objective values of 24.8 h and 20.6 h are obtained in the first and third stage, respectively (Figure 4.3b). In fact, 20.6 h is the best makespan for this example.

4.3.2 *Example 2: Importance of relaxation of horizon*

We consider the network from the previous example (Figure 4.2) with different processing times (see Table 4.1). The prices of the two final products, C and E, are $5/kg and $100/kg, respectively. The objective is to maximize profit over a scheduling horizon of 40 h.

We solve the example with two choices of δ, namely 1.25 h and 3.9 h. Intuitively, the former



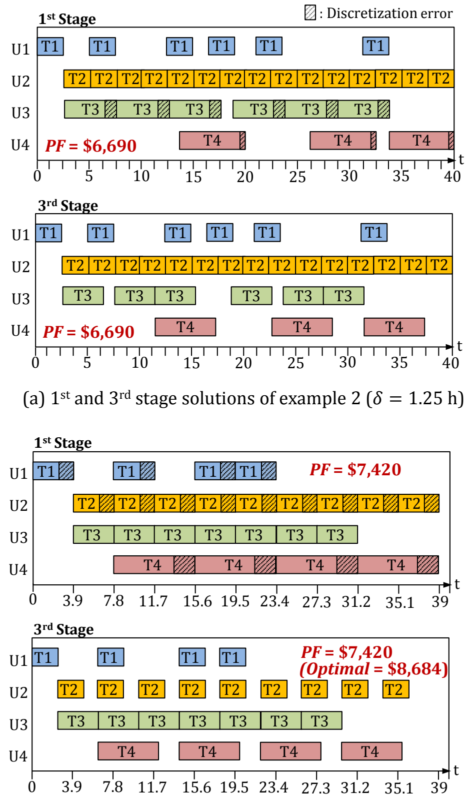
(a) 1st and 3rd stage solutions of example 1 (δ = 1 h)

(b) 1st and 3rd stage solutions of example 1 (δ = 3.1 h)

Figure 4.3: The first and third stage solutions obtained for motivating example 1 using (a) $\delta = 1$h, and (b) $\delta = 3.1$ h.

seems to be a better choice, because it provides better approximation of the processing times and leads to a finer grid. The results are shown in Figure 4.4. Once again, counter-intuitively, DCA with $\delta = 3.9$ h finds a better solution ($7,420) compared to when $\delta = 1.25$ h is used ($6,690). Nonetheless, the solution is still far away from the best solution of this example ($8,684), mainly due to discretization errors that lead to the first stage solutions with fewer batches in certain units (e.g. U2).

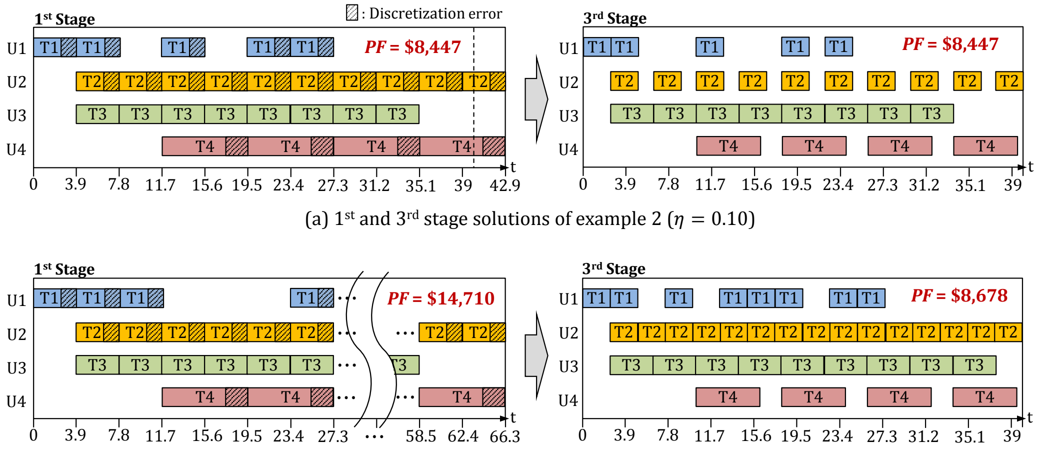
To address this issue, we solve the instance using $\delta = 3.9$ h with two horizon relaxations, $\eta = 0.1$ and 0.75. The solutions are shown in Figure 4.5. The results show that as we increase η, more batches are scheduled, leading to a solution ($8,678) that is closer to the best solution. The results suggest that the impact of the effectively shorter horizon, due to discretization errors,



(a) 1st and 3rd stage solutions of example 2 ($\delta = 1.25$ h)

(b) 1st and 3rd stage solutions of example 1 ($\delta = 3.9$ h)

Figure 4.4: The first and third stage solutions obtained for motivating example 2 using (a) $\delta = 1.25$ h, and (b) $\delta = 3.9$ h.



(a) 1st and 3rd stage solutions of example 2 ($\eta = 0.10$)

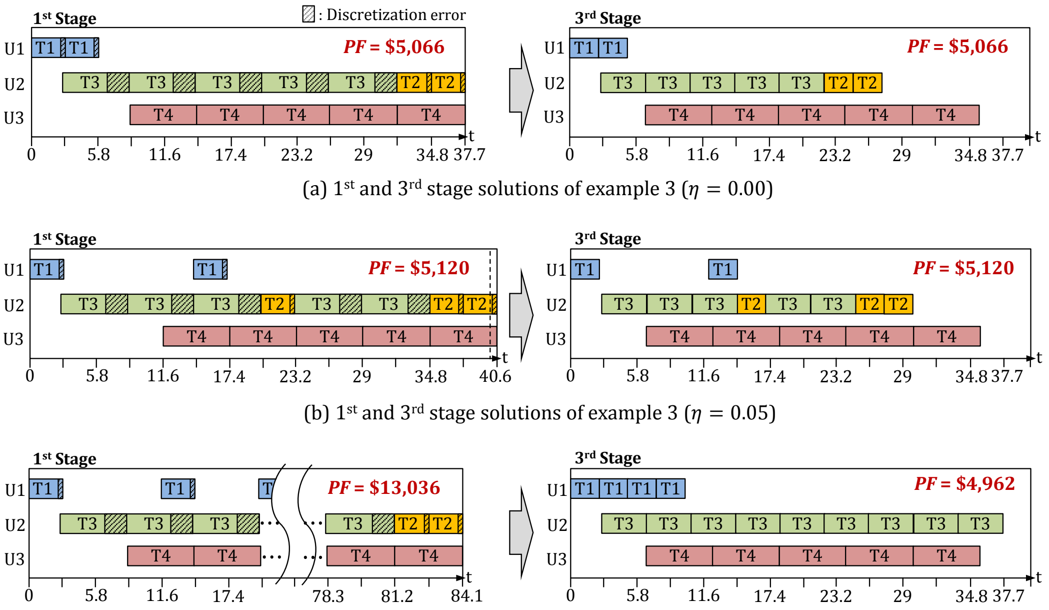
(b) 1st and 3rd stage solutions of example 2 ($\eta = 0.75$)

Figure 4.5: The first and third stage solutions obtained for motivating example 2 with $\delta = 3.9$ h and (a) $\eta = 0.10$, and (b) $\eta = 0.75$.

can be mitigated by using a nonzero η. Furthermore, it seems that solution quality improves as the horizon relaxation increases because the solution with the optimal number of batches can be found, after the excessive batches are cut off in the third stage (see Subsection 4.2.2). However, as we will see in the next example, this is not necessarily the case.

4.3.3 *Example 3: Choosing the horizon relaxation*

We consider the same network (Figure 4.2), except that tasks T2 and T3 are now executed in the same unit as shown in Table 4.1. The objective is to maximize profit within a horizon of 40 h.



(a) 1st and 3rd stage solutions of example 3 ($\eta = 0.00$)

(b) 1st and 3rd stage solutions of example 3 ($\eta = 0.05$)

(c) 1st and 3rd stage solutions of example 3 ($\eta = 1.20$)

Figure 4.6: The first and third stage solutions obtained for motivating example 3 with $\delta = 2.9$ h and (a) $\eta = 0.00$, (b) $\eta = 0.05$, and (c) $\eta = 1.20$.

We solve this example with $\delta = 2.9$ h and three different η's, namely 0, 0.05, and 1.2. The results are shown in Figure 4.6. With no relaxation (Figure 4.6a), we obtain a schedule with unit U2 relatively idle. Although no additional batches of T3 can be carried out (due to the limited time for the downstream task T4), additional batches of T2 can be executed to improve the objective value. This is exactly what happens when $\eta = 0.05$ is used (Figure 4.6b). However, if the horizon is relaxed too much (Figure 4.6c), an inferior third stage solution is obtained. This is because the task that leads to a more valuable product (i.e. T3) is given priority over T2, which leads to the third stage solution with multiple batches of T3 that are not converted to the final product, negatively impacting the solution quality. This example suggests that a more systematic analysis is necessary to identify the appropriate horizon relaxation.

4.4 DISCRETIZATION STEP LENGTH SELECTION

4.4.1 *Proposed approach*

Finding a δ that yields a good first stage solution is important, because a strong correlation between the quality of the first and the third stage solutions exists as will be shown later in this section. To study how to select a δ for a specific instance, we introduce the concept of error evaluation function, denoted as $\varepsilon(\delta)$, which calculates the approximation error in the time-related data, introduced by a chosen δ. In an ideal scenario, $\varepsilon(\delta)$ should be a metric for predicting the quality of the first stage solution, as shown in Figure 4.7: δ's that lead to smaller errors result in better solutions.

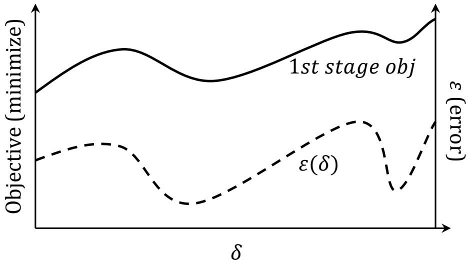


Figure 4.7: An illustration of the error evaluation function that projects the behavior of the first stage objective value as a function of δ.

To be able to accurately predict the quality of the first stage solution as a function of δ, we consider various factors: (i) processing times of tasks, (ii) idleness (or conversely, *busyness*) of units, (iii) *value* of tasks, and (iv) bottleneck units.

4.4.2 *Error evaluation functions*

We introduce a new parameter, E_{ij}, to denote the error in approximating the processing time of task i assigned to unit j.

$$E_{ij} = \left|\tau_{ij} - \delta\bar{\tau}_{ij}\right| \quad \forall i, j \in \mathbf{J}_i \tag{4.1}$$

Error in approximation of processing times

The first error evaluation function, denoted as $\varepsilon^E(\delta)$, only considers the error in the approximation of the processing times.

$$\varepsilon^E(\delta) = \frac{1}{\delta}\sum_i \sum_{j \in \mathbf{J}_i} E_{ij} \tag{4.2}$$

Selection based on this error evaluation function closely resembles the heuristic method, in which δ's are chosen such that the time-related data modulo δ are small.

Error weighted by busyness of units

The second function, ε^B, weights the approximation errors of the processing times using the concept of unit *busyness*, defined as the total time a unit is engaged processing the assigned batches. This is based on the intuition that errors in relatively idle units have lower impact on the quality of the solution than errors in busier units. However, since the busyness of units are not known *a priori*, we obtain approximate values by solving an LP relaxation of the first stage model with δ selected as the largest common factor of the time-related data:

$$Busy_j = \sum_{i \in \mathbf{I}_j} \sum_t (\tau^F_{ij} + \tau^V_{ij}\beta^{max}_j) X^*_{ijt} \quad \forall j \tag{4.3}$$

where X^*_{ijt} is the solution of the LP relaxed problem ($X_{ijt} = 1$ if task i starts in unit j at time point t).

In addition, we take into account the contribution of each task $i \in \mathbf{I}_j$ towards the busyness of unit j, by weighting with the number of batches of task i executed in unit j, denoted as N^*_{ij}.

$$N^*_{ij} = \sum_t X^*_{ijt} \quad \forall i, j \in \mathbf{J}_i \tag{4.4}$$

Using $Busy_j$ and N^*_{ij}, we weight the approximation errors of the processing times.

$$\varepsilon^B(\delta) = \frac{1}{\delta}\sum_i \sum_{j \in \mathbf{J}_i} \omega^B_{ij} E_{ij}\ , \quad \omega^B_{ij} = (\frac{N^*_{ij}}{\sum_i N^*_{ij}})(\frac{Busy_j}{\sum_j Busy_j}) \tag{4.5}$$

Error weighted by value of tasks

In profit maximization problems, we introduce the concept of the *value* of a task to denote the profit-wise contribution of a batch of a given task. The underlying idea is that errors in approximating processing times of tasks that are more valuable may have higher impact on the quality of the solution. For example, consider a network with two tasks that produce two final products with widely different selling prices. The task that produces the product of the larger value is more valuable, and would have higher impact on the solution quality, assuming the differences in other factors (e.g. batch size, processing costs) are negligible.

Given a network, the value of task i, denoted by Val_i, can be calculated prior to solving the problem through a value propagation algorithm, which back propagates the price of the final product to evaluate the contribution of each task to the profit, considering unit capacities and the consumption/production ratio of materials of each task (see Appendix C). The error evaluation function is defined as follows.

$$\varepsilon^V(\delta) = \frac{1}{\delta} \sum_i \sum_{j \in \mathbf{J}_i} \omega_i^V E_{ij} \,, \quad \omega_i^V = (\frac{Val_i}{\sum_i Val_i}) \tag{4.6}$$

Error weighted by unit demand

The *demand* of a unit is a concept we introduce to measure the improvement in the objective function value when the unit assignment (i.e. clique) constraints (Eq. 3.1) are allowed to be marginally violated. For instance, an idle unit has a demand of zero, because having an additional unit of the same type does not improve the objective value, while the bottleneck unit has nonzero demand, because having one more of such unit will improve the objective value. Although not identical, this concept is similar to the shadow prices of constrained resources in an LP.

The demand of unit j, denoted as Dem_j, can be obtained by solving the LP relaxation of the first stage model. Specifically, we define it as the summation of the dual variables, $Cliq^*_{jt}$, of the unit's clique constraints (Eq. 3.1) over time:

$$Dem_j = \sum_t Cliq^*_{jt} \quad \forall j \tag{4.7}$$

Similar to ε^B, we take into account the contribution of each task $i \in \mathbf{I}_j$:

$$\varepsilon^D(\delta) = \frac{1}{\delta}\sum_i \sum_{j \in \mathbf{J}_i} \omega^D_{ij} E_{ij} \ , \quad \omega^D_{ij} = (\frac{N^*_{ij}}{\sum_i N^*_{ij}})(\frac{Dem_j}{\sum_j Dem_j}) \tag{4.8}$$

4.4.3 *Function comparison*

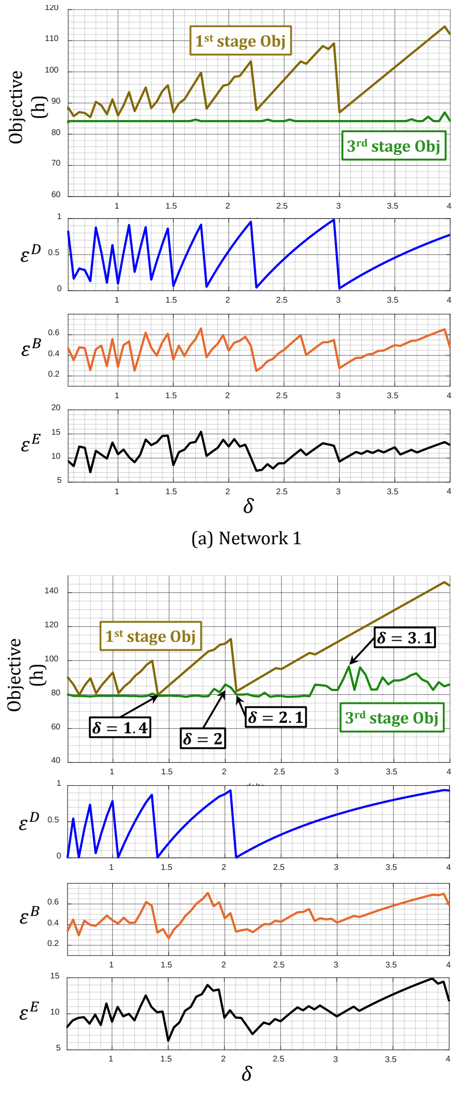
We test the proposed error evaluation functions using two networks from the literature: Kallrath (2002) and Papageorgiou and Pantelides (1996b). The networks are modified such that the processing times are irregular (e.g. 2.9 h, 4.4 h); specific details can be found in Lee and Maravelias (2019a). Although we only consider two networks in this section, similar results are obtained for other networks (data not shown).

A total of 6 instances are studied based on two networks and three objective functions. For each instance, we evaluate how accurately the proposed functions predict the behavior of the objective values. To do this, we solve each instance with varying values of δ and plot the objective values along with the error evaluation functions (recall Figure 4.7). The results are presented and analyzed for each objective function.

Makespan minimization

First, as shown in Figure 4.8, the first stage objective value (brown line) shows a repeated behavior: gradual increase up to a local maximum followed by a sudden drop to a local minimum. The interval in which this behavior is repeated increases with δ and is dependent on the network. This behavior is observed because there are certain values of δ such that the processing times modulo δ are small. Specifically, when δ approaches these values from below, the processing times are overestimated by nearly a full δ, resulting in large errors, while the error drops to zero as soon as δ reaches them. The sudden changes in errors are reflected in the behavior of the first stage objective value.

Second, the third stage objective value (green line) does not change significantly at low values of δ, but fluctuates more at larger values, especially for the second network. Importantly, we note that all the inferior third stage solutions are observed when δ's that lead to inferior first stage solutions are used (e.g. $\delta = 2$ h and 3.1 h) (Figure 4.8b). Conversely, δ's that lead to the local



(a) Network 1

(b) Network 2

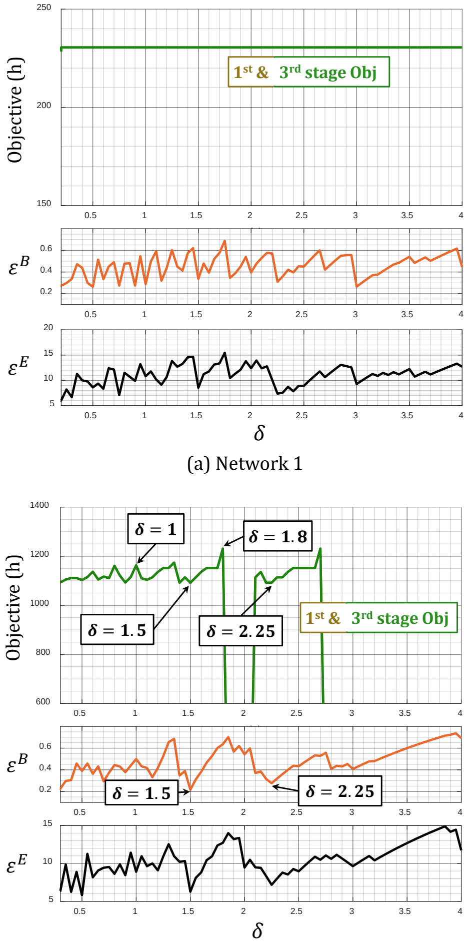
Figure 4.8: The optimal objective values for the first and third stage models and error evaluation functions with respect to δ (makespan minimization).

minima of first stage objective values, lead to good third stage solutions (e.g. $\delta = 1.4$ h and 2.1 h).

In terms of the error evaluation functions, ε^D accurately predicts the behavior of the first stage objective value. Specifically, the local minima/maxima of the first stage objective value are obtained exactly at δ's that result in local minima/maxima of ε^D. Function ε^B is not as accurate, as some of the δ's that lead to its local minima yield inferior first stage objective values. Finally, ε^E, which resembles the traditional heuristic method, does not appear to be correlated with the first stage objective value.

Following the aforementioned observations, the δ's for makespan minimization problem should be selected as the ones that result in local minima of ε^D.

Cost minimization



(a) Network 1

(b) Network 2

Figure 4.9: The optimal objective values for the first and third stage models and error evaluation functions with respect to δ (cost minimization).

In instances with a sufficiently long horizon, the optimal assignment of batches to units with the least processing cost is feasible. Thus, regardless of the choice of δ, the solutions with such assignments are feasible and optimal for the first stage model, resulting in a constant objective value, with respect to δ, as shown in Figure 4.9a. The first and third stage objective values coincide, because the first stage solution is always feasible and optimal in the third, assuming zero horizon relaxation.

Once the horizon becomes binding, the choice of δ becomes important, as it not only determines the quality of the solutions, but also the feasibility of the first and third stage models (see Figure 4.9b). In the second network, the instance is feasible when solved with δ's ranging from 0.3 to 1.8 h and from 2.1 to 2.7 h, while any other δ makes the first stage model infeasible.

If the batch-unit assignments with the lowest cost are infeasible, the clique constraints in the LP relaxed problem become binding. Otherwise, the clique constraints are not binding (i.e. all duals are zero), thereby error evaluation function ε^D cannot be used. Hence, we compare the other two functions, ε^E and ε^B, which show similar behavior and can be used to identify the δ's that result in the best objective values. In Figure 4.9b, we observe that $\delta = 1.5$ h and 2.25 h are good choices. When different δ's are chosen (e.g. 1 h and 1.8 h), inferior objective values are obtained. The first stage model is infeasible only when δ's that lead to large ε^B are chosen.

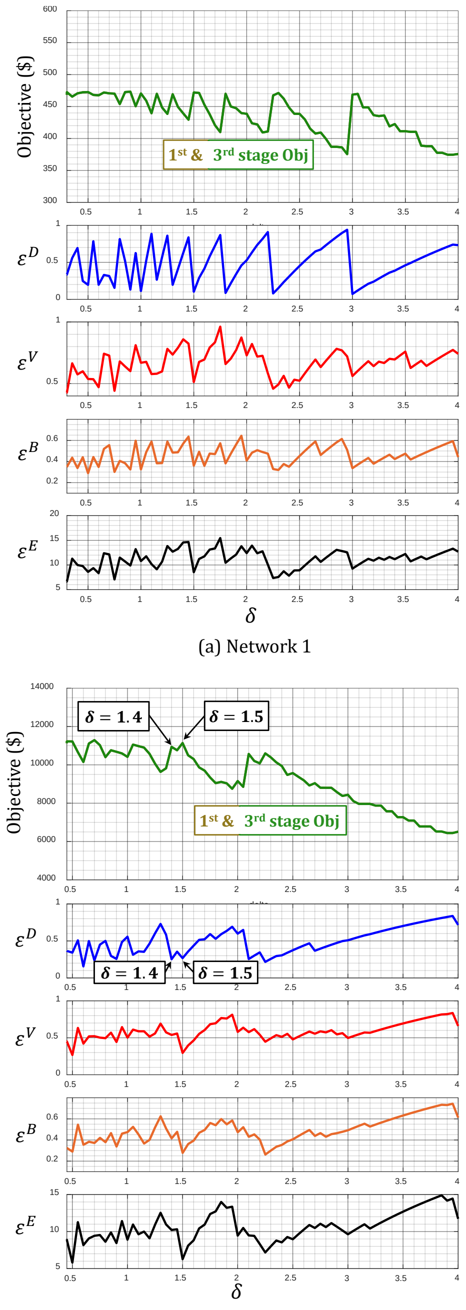
All things considered, it is suggested that δ's that result in local minima of ε^B be used for cost minimization instances.

Profit maximization

In profit maximization instances, the optimal first stage solution is optimal in the third, causing the objective values of both stages to coincide as shown in Figure 4.10. Although the objective values decay in general with increasing δ, they exhibit oscillatory behavior similar to the one observed in makespan minimization instances.

As in the case of makespan minimization, ε^D best predicts the quality of the first stage solution, and thus the third stage solution: δ's that result in local minima of ε^D result in local maxima of the objective values (e.g. δ =1.4 h and 1.5 h in Figure 4.10b). The other three functions (ε^E, ε^B and ε^V) are not good predictors, as they fail to identify several δ's that result in local maxima of the objective values, while falsely identifying ones that do not.

Therefore, the selection of δ's for profit maximization instances should be based on ε^D.



(a) Network 1

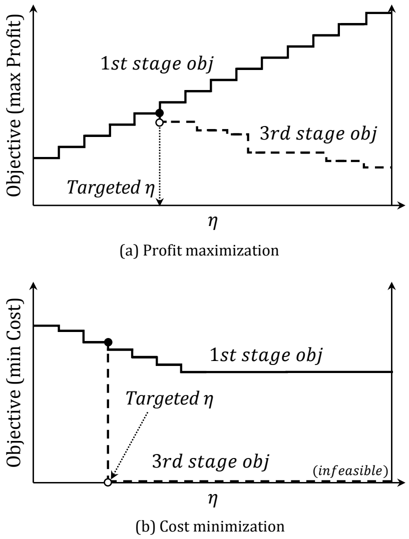
(b) Network 2

Figure 4.10: The optimal objective values for the first and third stage models and error evaluation functions with respect to δ (profit maximization).

4.5 HORIZON LENGTH RELAXATION SELECTION

4.5.1 *Proposed approach*

Appropriately chosen horizon relaxation, η, compensates for the effectively reduced horizon caused by the approximation errors of the time-related data. For a given instance, the *targeted* η can be found by solving the instance repeatedly with increasing η and plotting the first and third stage objective values as a function of η (see Figure 4.11). The point where the first and third stage objective values start to differ (profit maximization), or where the third stage model becomes infeasible (cost minimization) is the targeted η that our methods are aiming to estimate. Note that in this section, we assume that δ is already determined using the methods in the previous section.

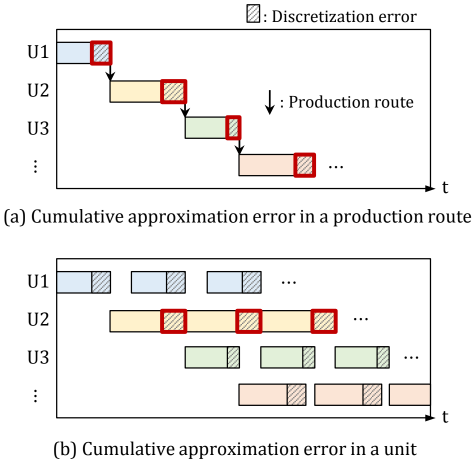


(a) Profit maximization

(b) Cost minimization

Figure 4.11: The evolution of the first and third stage objective values as a function of η for (a) profit maximization, and (b) cost minimization.

Intuitively, we expect that η depends on the cumulative approximation error in each production route (i.e. the sequence of tasks required to produce a final product from raw materials) as shown in Figure 4.12a. In general, however, multiple batches of tasks in a production route are executed (Figure 4.12b), and as a result, the cumulative approximation error in a specific unit (U2 in this case) becomes more important. Furthermore, the effect of earliest start time and latest finish time of that unit becomes negligible as longer horizons are considered. We introduce the concept of *cumulative error function* to calculate the contribution of cumulative approximation error in each unit to η.



(a) Cumulative approximation error in a production route

(b) Cumulative approximation error in a unit

Figure 4.12: Illustration of cumulative approximation error in (a) a production route, and (b) a unit.

4.5.2 *Cumulative error functions*

Let ε_η denote the cumulative error function. Then, the horizon relaxation, η, is calculated by

$$\eta = \tilde{\varepsilon}_\eta / H \tag{4.9}$$

where $\tilde{\varepsilon}_\eta$ $(= \sigma\varepsilon_\eta)$ is the scaled cumulative error function, with σ being the scaling factor. We note that $\sigma = 1$ is used in most cases, except for a special case as will be discussed later.

The first function, ε_η^{B1}, is defined as the cumulative approximation error of the busiest unit. We define a singleton $\mathbf{J}^{BMax} = \{j | Busy_j \geq Busy_{j'} \ \forall j'\}$, containing the unit with the maximum $Busy_j$.

$$\varepsilon_\eta^{B1} = \sum_{j \in \mathbf{J}^{BMax}} \sum_{i \in \mathbf{I}_j} E_{ij} N_{ij}^* \tag{4.10}$$

We could also consider weighting the cumulative approximation error based on busyness.

$$\varepsilon_\eta^{B2} = \frac{1}{\sum_j Busy_j} \sum_j Busy_j \sum_{i \in \mathbf{I}_j} E_{ij} N_{ij}^* \tag{4.11}$$

Similarly, we define ε_η^{D1} and ε_η^{D2} as the cumulative error functions that consider cumulative approximation error of the most demanded unit, and the cumulative approximation error weighted by the demand of each unit, respectively. If $\mathbf{J}^{DMax} = \{j | Dem_j \geq Dem_{j'} \; \forall j'\}$, then we obtain:

$$\varepsilon_\eta^{D1} = \sum_{j \in \mathbf{J}^{DMax}} \sum_{i \in \mathbf{I}_j} E_{ij} N_{ij}^* \tag{4.12}$$

$$\varepsilon_\eta^{D2} = \frac{1}{\sum_j Dem_j} \sum_j Dem_j \sum_{i \in \mathbf{I}_j} E_{ij} N_{ij}^* \tag{4.13}$$

Finally, we consider ε_η^{BE}, which is based on the difference between the maximum total processing time (i.e. busyness+error) among the units and the horizon.

$$\varepsilon_\eta^{BE} = \max_j \{Busy_j + \sum_{i \in \mathbf{I}_j} E_{ij} N_{ij}^*\} - H \tag{4.14}$$

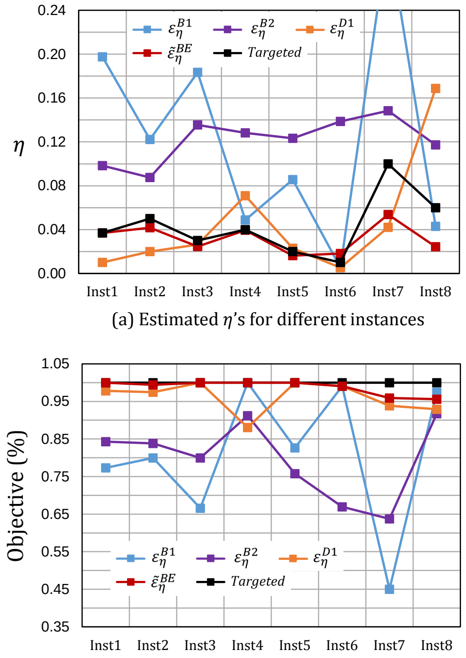
4.5.3 *Cumulative error function comparison*

Profit maximization

The proposed cumulative error functions are evaluated using eight instances. As the targeted η's tend to be smaller than the ones estimated by these functions, we also investigate the cumulative error functions scaled by a factor of one over the number of demanded units (i.e. $\sigma = \frac{1}{\sum_{(j:Dem_j > 0)} 1}$).

Figure 4.13a shows the η's (colored lines) for each instance calculated using different ε_η's, as well as the targeted η (black line). A scaling factor of one is used for all ε_η except for $\tilde{\varepsilon}_\eta^{BE}$, where the aforementioned scaling factor is applied. To simplify the presentation, we purposefully omit the tilde notation for ε_η's with scaling factor of one throughout the chapter. Figure 4.13b shows the percentile of the third stage objective values obtained with the estimated η's with respect to the one obtained with the targeted η's.

First we notice that, in general, it is better to be conservative when relaxing the horizon. For example, in instance 2, η's obtained using ε_η^{D1} (orange line) and ε_η^{B2} (purple line), underestimates and overestimates the targeted η by a similar margin, respectively (Figure 4.13a). However, the corresponding third stage objective value obtained based on ε_η^{D1} is more than 15% better than the one obtained based on ε_η^{B2} (Figure 4.13b). Similar results can be observed in instance 8.



(a) Estimated η's for different instances

(b) Objectives obtained by the estimated η's

Figure 4.13: Profit maximization: (a) Estimated η's based on different cumulative error functions, and (b) the corresponding third stage objective values for different instances.

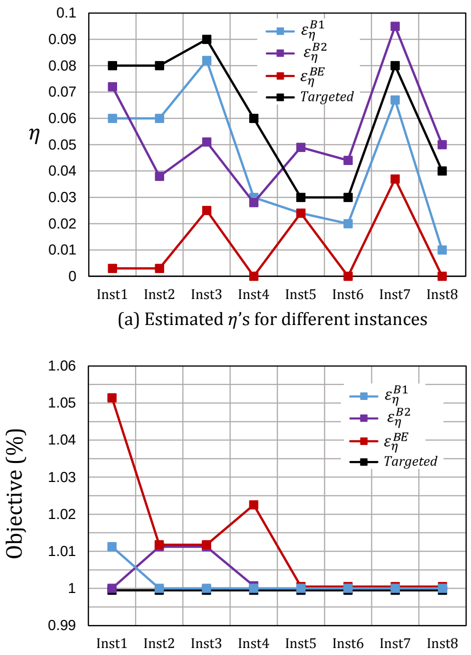
Furthermore, in cases where the targeted η's are overestimated by a large margin, significantly inferior third stage solutions are obtained (e.g., when using relaxations predicted based on ε_η^{B1} and ε_η^{B2}).

The targeted η's are best estimated when $\tilde{\varepsilon}_\eta^{BE}$ (red line) is used, often times yielding the same objective values as the targeted η's. We note that different combinations of cumulative error functions and scaling factors were tested, but they did not provide estimations that are as good (data not shown).

Cost minimization

Since the clique constraints of the first stage model become binding in limited cases (recall 4.4.3), only the cumulative error functions that consider busyness of units, namely ε_η^{B1}, ε_η^{B2} and ε_η^{BE}, are tested. The results are shown in Figure 4.14.

Unlike profit maximization, overestimating the targeted η should be avoided, because it will lead to infeasible third stage model; e.g., ε_η^{B2} (purple line) Inst5-Inst8. On the other hand, underestimating the targeted η's seems to be acceptable in many cases as the corresponding third



(a) Estimated η's for different instances

(b) Objectives obtained by the estimated η's

Figure 4.14: Cost minimization: (a) Estimated η's based on different cumulative error functions, and (b) the corresponding third stage objective values for different instances.

stage objective values are of high quality regardless; e.g., ε_η^{BE} (red line) Inst5-Inst8. All things considered, ε_η^{B1} leads to the best estimations, often giving η's that are close to the targeted ones, returning identical (or similar) third stage objective values to the ones found with the targeted η's.

4.6 COMPUTATIONAL STUDY

All the runs in this section are performed using CPLEX 12.6.2 via GAMS 24.9.2 on a cluster with Intel Xeon (E5520) processors at 2.27 GHz and 16GB of RAM running on CentosOS Linux 7. Default options with a resource limit of 1h are used unless stated otherwise.

4.6.1 *Motivating examples revisited*

For comparison, we solve the examples with DCA and two other approaches, namely a discrete-time model (Shah et al., 1993) with δ chosen based on heuristic methods and one of the most effective continuous-time model (Sundaramoorthy and Karimi, 2005), denoted as DM and CM, respectively.

Example 1

We consider the simple network in Figure 4.2 with makespan minimization as the objective function. We plot the first and third stage objective values obtained at different values of δ, along with the error evaluation function, ε^D (see Figure 4.15).

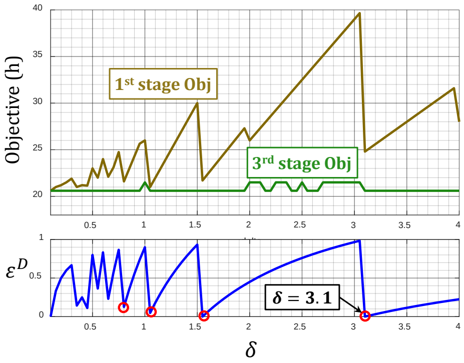


Figure 4.15: Selection of δ for motivating example 1 (makespan minimization).

We observe that the local minima of the first stage objective value are accurately predicted by ε^D. Based on the graph, discretizations of 0.8, 1.05, 1.55 and 3.1 h are expected to lead to good solutions, but we select the largest, 3.1 h, for computational performance. This results in a good first stage solution (24.8 h), and thus a good third stage solution (20.6 h), which turns out to be the best solution for this instance. In contrast, the δ that would have been chosen based on traditional heuristic methods, 1 h, leads to a large value of ε^D, resulting in inferior first and third stage solutions of 26 h and 21.5 h, respectively.

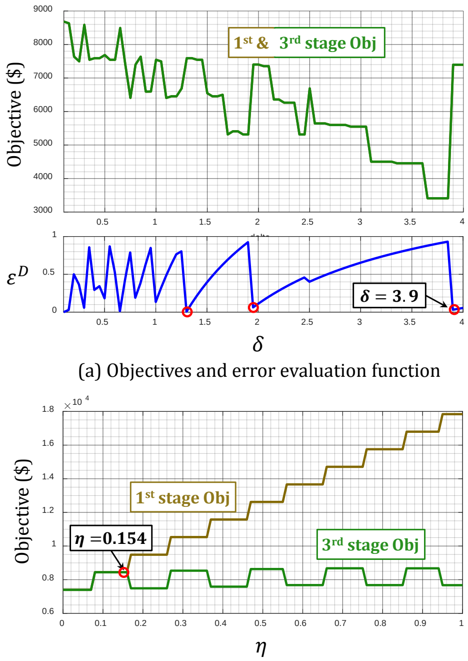
In terms of computational time, DCA with $\delta = 3.1$ h is two orders of magnitude faster than CM, while providing a solution of the same quality (see Table 4.2).

Example 2

We consider the same network with profit maximization as the objective function. Among the δ's that lead to local minima of ε^D (Figure 4.16a), we choose 3.9 h for computational efficiency. Although the objective value returned ($7,402) is better than the one obtained using $\delta = 1.25$ h ($6,690), it is not as good as the best for the instance ($8,684).

Table 4.2: Computational statistics of motivating examples

	Example 1			Example 2			Example 3		
	DM	CM	DCA	DM	CM	DCA	DM	CM	DCA
δ or #Slots	1	15	3.1	1.25	20	3.9	1	20	2.9
Constraints	878	776	280	579	776	211	677	967	250
Disc. Var.	168	116	56	136	116	52	168	137	64
Cont. Var.	544	679	124	303	679	114	375	838	141
LP Relax.	3.33	18.60	8.249	6,691.5	10,777.8	8,447	5246.7	7,086.7	5,120
Obj (h or $)	26	20.6	24.8	6,690	8,588	8,447	5,214	5,310	5,120
(Refined)			20.6			8,447			5,120
CPU (s)	0.296	2.19	0.037	2.13	3,600	0.032	1.00	3,600	0.015
Nodes	0	5,018	0	30,098	$> 10^6$	0	10,154	$> 10^6$	0
Gap (%)	-	-	-	-	1.25	-	-	11.8	-



(a) Objectives and error evaluation function

(b) Targeted η simulation

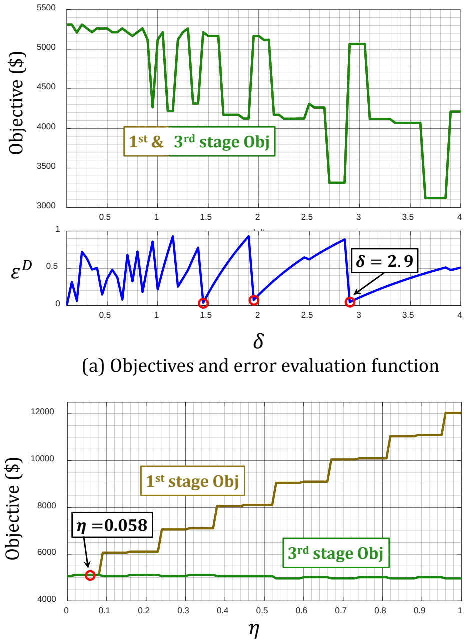
Figure 4.16: Selection of δ and η for motivating example 2 (profit maximization).

To obtain a better solution, we relax the horizon. Specifically, 6.2 h is obtained based on $\tilde{\varepsilon}_{\eta}^{BE}$, which for a 40 h horizon, corresponds to $\eta = 0.154$. This is very close to the targeted η (i.e. 0.16) as shown in Figure 4.16b, leading to a third stage objective value equal to the one obtained using the targeted η. With $\delta = 3.9$ h and $\eta = 0.154$, we obtain a third stage objective value ($8,447) that is less than 3% away from the best solution.

In terms of computational performance (see Table 4.2), CM obtains the best solution ($8,588), but fails to prove optimality within an hour. DCA returns a solution ($8,447) that is slightly inferior (∼1.5%) to CM's, within a fraction of a second. DM proves optimality within a few seconds, but provides a significantly inferior solution ($6,690).

Example 3

The network in Figure 4 is used again, but with slightly different task-unit compatibility. First, the δ's that lead to local minima of ε^D are identified (Figure 4.17a). Among them, $\delta = 2.9$ h is selected for computational efficiency, leading to a solution ($5,066) that is slightly inferior than the best solution ($5,310).



(a) Objectives and error evaluation function

(b) Targeted η simulation

Figure 4.17: Selection of δ and η for motivating example 3.

A better solution can be found by relaxing the horizon. We calculate $\tilde{\varepsilon}_{\eta}^{BE} = 2.3$ h, which leads to $\eta = 0.058$, or a 5.8% relaxation. Although the suggested η is smaller than the targeted η (i.e. 0.08), the obtained solution ($5,120) is the same as when the targeted η is used (Figure 4.17b).

In terms of solution quality, CM finds the best known solution ($5,310) with a 11.8% gap after 1 hour, while DM, with $\delta = 1$ h, finds a high quality solution ($5,214) in 1 second (see Table 4.2). Note that 1 h is also one of the δ's that yield local minima of ε^D (see Figure 4.17a). The DCA returns a high quality solution ($5,120) in a fraction of a second, more than two orders of magnitude faster than the other methods.

Note that the tested examples are relatively simple and small. The effectiveness of DCA with appropriately chosen user-defined parameters becomes even more significant in difficult instances as illustrated in the next subsection.

4.6.2 *Performance*

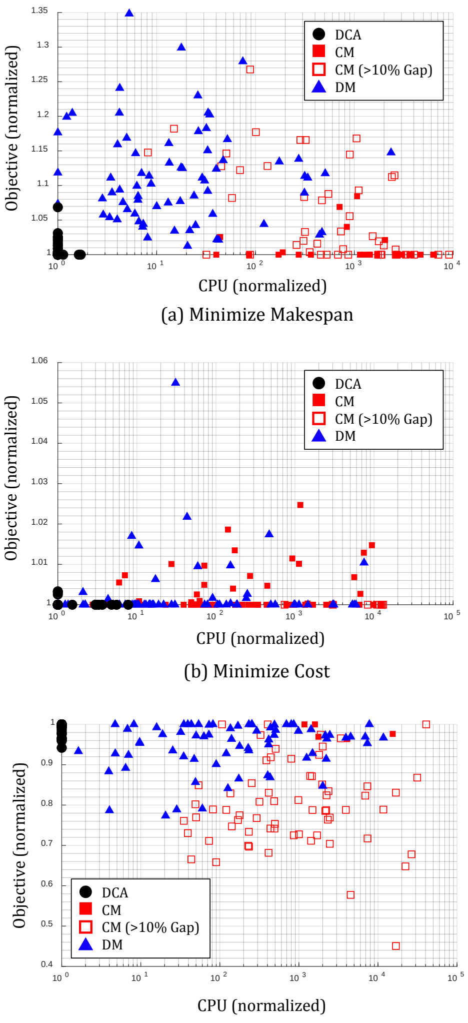
A total of 630 runs are performed in this study: we consider a combination of the three different approaches (i.e. DCA, DM, CM), three different objective functions, and 70 instances generated from 7 networks from the literature (Shah et al., 1993; Papageorgiou and Pantelides, 1996b; Maravelias and Papalamprou, 2009; Nie et al., 2015; Shi and You, 2016). For tractability, $\delta = 1$ h is used for DM. The number of time periods for CM was chosen by solving the instances iteratively, increasing the number of periods in each iteration until no improvement in both the LP relaxation and the integer solution are observed, thereby representing a good estimate of the number of periods necessary to represent the optimal solution. A resource limit of 5 hours is used.

The performance of the three approaches, in terms of computational speed and solution quality, is compared in Figure 4.18, where the x-axis represents the CPU time, normalized with respect to the CPU time of the fastest approach for a given instance, and the y-axis shows the objective values, normalized with respect to the best objective value found for the given instance.

The results show that the DCA (black circles) significantly outperforms CM (red squares) and DM (blue triangles) in terms of CPU time, with a speedup up to 10^4. In terms of solution quality, DCA yields the best objective values in up to 95% of instances, depending on the objective, while providing very good solutions (i.e. 0 to 6% inferior to the best) in all other cases. Figure 4.19 shows the proportion of instances in which DCA found better/same/worse solutions compared to the other approaches. It clearly shows that DCA finds same or better solutions in the majority of the instances, and even when it does not, the difference is small.

4.6.3 *Case studies*

In this section we discuss the implementation of DCA to hard instances based on larger networks and longer horizons. We consider three benchmark examples from the literature. In the



(a) Minimize Makespan

(b) Minimize Cost

(c) Maximize Profit

Figure 4.18: The performance of DCA, DM, CM compared in terms of CPU and objective value for (a) makepsan minimization, (b) cost minimization, and (c) profit maximization instances.

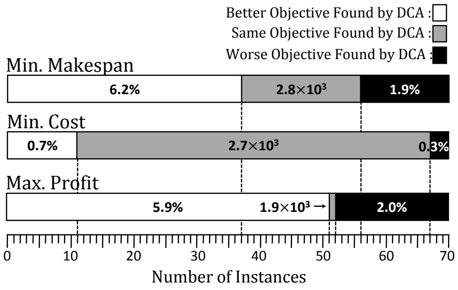


Figure 4.19: Proportion of instances in which DCA found better/same/worse objective value compared to DM and CM combined. Percentiles represent average difference in objective value; numbers represent average speedup.

first example, denoted as K, we consider a modified network from Kallrath (2002) with cost minimization as the objective. The second example considers a modified network from Papageorgiou and Pantelides (1996b) with makespan minimization objective, denoted as PP. Lastly, a profit maximization instance is solved for a network modified from Maravelias and Papalamprou (2009), denoted as MP. The scheduling horizon is 120 h and irregular processing times (i.e. greatest common factor of 0.1 h for K and PP, 0.05 h for MP) are considered. A resource limit of 5 h is used, and a relative gap of 2% is allowed for tractability. The networks and problem data of the instances can be found in Lee and Maravelias (2019a).

The discretization step length, δ, and the horizon relaxation, η, for each example are chosen based on the proposed methods. Specifically, for the first example, $\delta = 2.6$ h is selected based on ε^{B}, and $\eta = 0.133$ is selected using ε_{η}^{B1}. For the next two examples, we use ε^{D} to obtain δ's: 1.85 h and 2 h, respectively. For the third example, $\eta = 0.018$ is selected based on $\tilde{\varepsilon}_{\eta}^{BE}$. The computational results are given in Table 4.3.

In example K, the best solution ($3,365.8) is found by both CM and DCA, while a slightly inferior solution ($3,408.3) is found by DM. In terms of CPU time, CM is slower in finding the best solution (first found after 1,713.3 seconds) than DCA (68.2 seconds).

In example PP, only DCA is able to find the best makespan (56.4 h), and it did so in 3.8 seconds. On the other hand, although DM with $\delta = 1$ h terminates in around 40 seconds, it returns a solution (63 h) that is inferior. Interestingly, if δ's identified by the proposed methods, 0.95, 1.25 or 1.85 h, are used, DM would have been able to find a better solution (58.9 h). Regardless of the small model size, CM fails to close the gap within 5 hours, terminating with a suboptimal solution (57 h).

Finally, in example MP, DCA finds the best known solution ($1,169) within 3 minutes (see Figure 4.20 for the Gantt chart). The performance of DCA stands out even more when compared to the other methods. Specifically, CM finds significantly inferior solution ($810.5) with a large gap (42.4%) after 5 hours. Moreover, 40 time periods are not enough to represent the best solution, as the LP relaxation is smaller than the best known solution. CM fails to find a feasible solution within 5 hours if 132 time periods, the number of time periods necessary to represent the solution obtained by DCA (see Figure 4.20), is used. Next, DM with $\delta = 1$ h yields a reasonable solution ($1,147.5), because 1 h is one of the δ's identified by our methods, but fails to prove

Table 4.3: Computational results for the case studies

	K (min Cost)			PP (min Makespan)			MP (max Profit)		
	DM	CM	DCA	DM	CM	DCA	DM	CM	DCA
δ or #Slots	1.00	25	2.60	1.00	25	1.85	1.00	40	2.00
Constraints	9,586	6,211	4,215	8,550	5,732	4,598	14,084	14,798	7,243
Disc. Var.	3,050	841	1,350	2,156	742	1,166	4,514	1,948	2,331
Cont. Var.	5,344	4,893	2,352	4,782	4,829	2,577	7,894	11,443	4,059
LP Relax.	3,155.2	3,154.7	3,154.0	5.9	43.2	10.3	1,228.2	1,160.2	1,220.6
Obj (h or $)	3,408.3	3,365.8	3,365.8	63	57	59.2	1,147.5	810.5	1,169
(Refined)			3,365.8			56.4			1,169
CPU (s)	4.1	2,004.4	68.2	36.9	18,000.1	3.8	18,000.1	18,000.2	150.0
Nodes	0	$>10^5$	40,884	1,707	$>10^5$	2,291	9,366	35,966	30,794
Gap1 (%)	1.7	2.0	1.1	1.6	13.5	-	4.1	42.4	1.9
Gap2 (%)	1.3	-	-	11.7	1.1	-	1.8	44.2	-

**Gap1: MIP optimality gap, Gap2: gap from the best (or best known) solution*

optimality after 5 hours due to large model size.

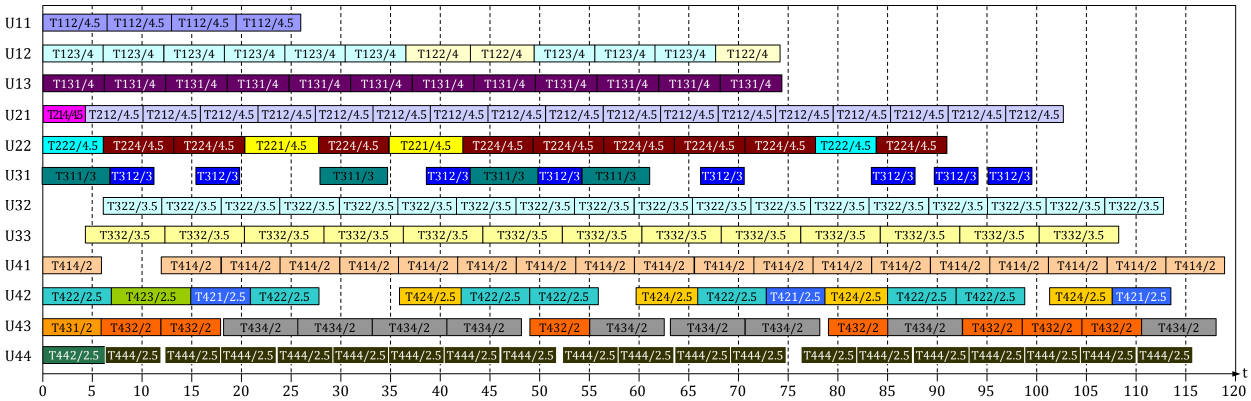


Figure 4.20: The Gantt chart for example MP solved with DCA. The numbers in each box represent Task/batch size.

4.7 CONCLUSIONS

Discrete- and continuous-time representations in mathematical programming scheduling formulations exhibit distinct advantages and disadvantages. In the last chapter, we proposed the Discrete-Continuous Algorithm, a novel framework that harnesses the strengths of both formulations and avoids the limitations. Importantly, DCA offers flexibility in selecting key user-defined parameters, discretization step length and horizon relaxation, which, if chosen carefully, lead to significant improvements both in solution quality and speed. However, choosing these parameters is nontrivial.

Accordingly, in this study, we proposed systematic methods to obtain sets of parameters that lead to superior computational performance and high-quality solutions. Specifically, the discretization step length, δ is selected based on an error evaluation function that takes into account various aspects of a scheduling instance, such as unit busyness and the dual variables of the clique constraints of the LP relaxed problem. The horizon relaxation, η, is calculated based on a cumulative error function that considers the cumulative approximation error in each unit, especially important for profit maximization and cost minimization problems. The proposed methods are not only applicable to DCA, but also to all discrete-time formulations: they can be used to select the best δ in large-scale instances, where larger δ's are preferred for tractability.

The performance of the proposed methods were extensively tested on multiple instances. The results showed that DCA, with systematically chosen δ and η, significantly outperforms traditional methods in terms of computational efficiency, showing up to 10^4 speedups, while obtaining identical or better solutions in the majority of instances. More importantly, the superiority of the proposed methods becomes more pronounced in difficult instances as showcased in the case studies.

4.8 NOMENCLATURE

Indices/sets

$i \in \mathbf{I}$	Tasks
$j \in \mathbf{J}$	Units
$k \in \mathbf{K}$	Materials
$t \in \mathbf{T}$	Time points or periods

Subsets

$\mathbf{I}_j$	Tasks that can be performed in unit j
$\mathbf{I}_k^+/\mathbf{I}_k^-$	Tasks that produce/consume material k
$\mathbf{J}_i$	Units that can perform task i

Parameters

$\alpha_{ij}^F/\alpha_{ij}^V$	Fixed/variable cost of performing task i in unit j
$\beta_j^{min}/\beta_j^{max}$	Minimum/maximum capacity of unit j
γ_k	Initial inventory of material k
δ	Discretization step length
E_{ij}	Approximation error of processing time of task i in unit j
H	Horizon length
η	Horizon length relaxation
ξ_k	Demand amount of material k at the end of the horizon
π_k	Price of material k
ρ_{ik}	Fraction of material k produced (>0) or consumed (<0) by task i
σ	Scaling factor for cumulative error function
τ_{ij}^F/τ_{ij}^V	Fixed/variable processing time of task i performed in unit j

Binary variables

X_{ijt}	=1 if task i begins in unit j at time point t

Nonnegative continuous variables

N_{ij} Number of task i executed in unit j

5

DISCRETE-CONTINUOUS ALGORITHM: GENERAL ALGORITHM

5.1 MOTIVATION

Due to the advances made in the field of optimization-based scheduling, as well as the improvements in computing power, it is now possible to solve medium scale instances in reasonable time using a personal desktop computer (Velez et al., 2015). However, the difficulties in applying optimization-based scheduling techniques to industrial problems come not only from the scale of these problems, but also from the various process characteristics that need to be taken into account. Although many studies have proposed different constraints (or models) to handle such characteristics, the resulting models can be computationally expensive. This is especially detrimental in industrial applications where schedules are revised and updated regularly to account for disturbances, necessitating fast solution of the models.

Although Discrete-Continuous Algorithm (DCA) can readily handle many characteristics including limited shared utilities, intermediate release and due dates and variable processing times, it relies on the assumption of unlimited intermediate storage (UIS), and it does not consider other characteristics, such as sequence-dependent changeovers and material storage in units.

Accordingly, the goal of this work is to propose a general DCA which accounts for various process characteristics commonly encountered in practice. The rest of the chapter is structured as follows. In the next section, we provide a short literature review of works that address industrial scheduling problems and present the problem statement. Next, we present the general DCA in

Sections 5.3-5.5, wherein each component of DCA is presented in detail. In Section 5.6, we present examples to illustrate the applicability and advantages of the algorithm. Finally, in Section 5.7, we show a case study inspired by a real-world process.

5.2 BACKGROUND

5.2.1 *Industrial applications*

We provide a short review of applications of optimization-based scheduling techniques to industrial problems, focusing on the modeling of different process characteristics. A more thorough list can be found in Harjunkoski et al. (2014) and Georgiadis et al. (2019).

Lin et al. (2002) studied production scheduling of specialty chemicals in a multiproduct chemical facility, which involved both batch and continuous processes, and featuring limited intermediate storage and unit-specific setup times for cleaning. Nie et al. (2014) and Velez et al. (2015) considered a drumming plant of Dow Chemical Company, which involved upstream batch and downstream continuous processes. Features they considered include multipurpose vessels with limited capacities for the storage of intermediate products, sequence-dependent changeovers between product families, and decrease of production capacity. Furthermore, Wassick and Ferrio (2011) proposed methods to model some constraints common in industrial settings, including intermediate deliveries with time windows and material transfer operations.

In terms of pharmaceutical industries, Stefansson et al. (2011) studied a secondary pharmaceutical facility, which involved sequence-dependent changeovers between product families and unit-dependent setup times in the packaging stage. Liu et al. (2014) considered production planning of a biopharmaceutical facility, which involved a two stage batch process with parallel units, featuring multipurpose storage vessels and units that are capable of temporarily storing intermediate materials.

In terms of applications in food and beverage industries, Baldo et al. (2014) studied scheduling of a beer production process which was modeled as a three stage process. Features they considered include temporary material storage in units in the first two stages, sequence-dependent changeovers between family of products in the third stage and periodic cleaning of the facility.

Georgiadis et al. (2018) studied a canned fish production process which involved series of batch processes and a continuous process. The intermediate products from the batch processes have limited dedicated storage, and the continuous process require sequence-dependent changeovers. Kopanos et al. (2011) studied scheduling of a yogurt production facility, which involved a batch process with parallel units that require cleaning after each task and a periodic facility-wide cleaning.

Finally, in terms of applications in paper and pulp industries, Giannelos and Georgiadis (2001) studied scheduling of paper cutting process in a paper mill, consisting of three parallel cutting units that require sequence-dependent changeovers. Castro et al. (2009b) considered scheduling of a tissue paper production process in a pulp mill plant. Features they considered include dedicated stockyard piles for some of the intermediate materials and specific mixing policies that have to be followed for the materials that are recycled.

5.2.2 *Problem Statement*

We study short-term chemical production scheduling in network environments. We consider processes with the following characteristics: (i) temporary material storage in processing units, (ii) multipurpose storage vessels, (iii) materials with handling restrictions, (iv) sequence-dependent changeovers, and (v) continuous processes. Note that the proposed algorithm can readily be extended to handle other process characteristics, such as periodic cleaning and filling policies, as will be illustrated in Section 5.7. We consider three objective functions, namely makespan minimization, cost minimization and profit maximization.

We use STN representation proposed by Kondili et al. (1993) to represent the production facility. We introduce the following indices and sets: tasks $i \in \mathbf{I}$, units and vessels $j \in \mathbf{J}$ and materials $k \in \mathbf{K}$. We introduce the following subsets: $\mathbf{I}_j$, tasks that can be performed in unit j; $\mathbf{I}_k^+/\mathbf{I}_k^-$, tasks that produce/consume material k; $\mathbf{J}_i$, units that can perform task i; $\mathbf{J}^{CO}$, units with changeovers; $\mathbf{J}^{CP}$, continuous processing units; $\mathbf{J}^H$, units that can store output materials; $\mathbf{J}^S$, storage vessels; $\mathbf{J}_k^S$, vessels that can store material k; $\mathbf{K}^{LS}$, materials with limited storage capacity; $\mathbf{K}^{NS}/\mathbf{K}^{NM}$, materials with no-splitting/no-mixing restriction; and $\mathbf{K}_j^S$, materials which can be stored in vessel j.

We represent the problem data as follows: η, horizon length; τ_{ij}^F/τ_{ij}^V, fixed/variable processing time of task i in unit j; $\alpha_{ij}^F/\alpha_{ij}^V$, fixed/variable processing cost of task i in unit j; ρ_{ik}, mass fraction of material k produced (> 0) or consumed (< 0) by a batch of task i; $\beta_j^{max}/\beta_j^{min}$, maximum/minimum capacity of unit or vessel j; and γ_k/π_k initial inventory/price of material k. Also, as a part of the problem data, we introduce $p \in \mathbf{P}$ to describe intermediate deliveries and demands: ξ_{kp}/ϕ_{kp} denote the amount/time of p^{th} delivery (< 0) or demand (> 0) of material k.

5.3 FIRST STAGE: DISCRETE-TIME MIP SCHEDULING MODEL

We adopt a common time grid that is discretized into intervals of equal length, given by δ. We introduce a set of time points $N = \{0, 1, 2, ..., \bar{\eta}\}$, and we denote the interval between time points $n - 1$ and n as time period n. The time-related problem data are converted into multiples of δ, rounded up or down to guarantee feasibility.

We introduce the following variables: $X_{ijn} \in \{0, 1\}$, 1 if a batch of task i starts in unit j at time point n; $B_{ijn} \in \mathbb{R}_+$, the batch size of task i in unit j starting at time point n; $S_{kn} \in \mathbb{R}_+$, the inventory level of material k during time period n; and $F_{jkn}^H \in \mathbb{R}_+$, the amount of material k discharged out from unit j at time point n.

Eq. 5.1 ensures that each unit can only execute one task at a time, and Eq. 5.2 includes the unit capacity constraints. Inventory balance constraints for materials without material handling restrictions are enforced by Eq. 5.3. Note that the balance for materials with handling restrictions (i.e. $\mathbf{K}^{NS} \cup \mathbf{K}^{NM}$) are modeled separately later in this section.

$$\sum_{i \in \mathbf{I}_j} \sum_{n' \geq n - \bar{\tau}_{ij} + 1}^{n} X_{ijn'} \leq 1 \quad \forall\, j \notin \mathbf{J}^H, n \tag{5.1}$$

$$\beta_j^{min} X_{ijn} \leq B_{ijn} \leq \beta_j^{max} X_{ijn} \quad \forall\, i, j \in \mathbf{J}_i, n \tag{5.2}$$

$$\begin{aligned} S_{k(n+1)} = S_{kn} + \sum_{i \in \mathbf{I}_k^+} \rho_{ik} \sum_{j \in \mathbf{J}_i \setminus \mathbf{J}^H} B_{ij(n-\bar{\tau}_{ij})} + \sum_{i \in \mathbf{I}_k^-} \rho_{ik} \sum_{j \in \mathbf{J}_i} B_{ijn} + \sum_{j \in \mathbf{J}_k^+ \cap \mathbf{J}^H} F_{jkn}^H - \bar{\xi}_{kn} \\ \forall\, k \notin \mathbf{K}^{NS} \cup \mathbf{K}^{NM}, n \end{aligned} \tag{5.3}$$

5.3.1 *Material storage in units*

We assume that the output materials stored in a unit can be discharged individually and that each material can be discharged across multiple time points. We introduce $X^H_{ijn} \in \{0, 1\}$, which is 1 if the output materials of task i are stored in unit j during time period n.

When a unit is executing a task or storing one or more materials, it cannot start a new task.

$$\sum_{i \in \mathbf{I}_j} \Big(\sum_{n' \geq n - \bar{\tau}_{ij} + 1}^{n} X_{ijn'} + X^H_{ij(n+1)} \Big) \leq 1 \quad \forall j \in \mathbf{J}^H, n \tag{5.4}$$

We introduce $S^S_{jkn} \in \mathbb{R}_+$, which is the amount of material k stored in unit or vessel j during time period n. Based upon Velez and Maravelias (2013b), we extend the material balance for the materials stored in unit j (Eq. 5.5), and we ensure that a unit can store materials only if the corresponding X^H_{ijn} is activated (Eq. 5.6).

$$\begin{aligned} S^S_{jk(n+1)} = S^S_{jkn} + \sum_{i \in \mathbf{I}_j \cap \mathbf{I}^+_k} \rho_{ik} B_{ij(n-\bar{\tau}_{ij})} - F^H_{jkn} \\ \forall k \in \mathbf{K}^{LS} \setminus (\mathbf{K}^{NS} \cup \mathbf{K}^{NM}), j \in \mathbf{J}^+_k \cap \mathbf{J}^H, n \end{aligned} \tag{5.5}$$

$$\sum_{k \in \mathbf{K}^+_i \cap \mathbf{K}^{LS}} S^S_{jkn} \leq \beta^{max}_j X^H_{ijn} \quad \forall i, j \in \mathbf{J}_i \cap \mathbf{J}^H, n \tag{5.6}$$

5.3.2 *Multipurpose storage vessels*

We introduce $X^S_{jkn} \in \{0, 1\}$, which is 1 if material k is stored in vessel j during time period n. At any given time period, each vessel can only store one material.

$$\sum_{k \in \mathbf{K}^S_j} X^S_{jkn} \leq 1 \quad \forall j \in \mathbf{J}^S, n \tag{5.7}$$

In the first stage model, we simply keep track of which vessels are used to store material.

$$S_{kn} \leq \sum_{j \in \mathbf{J}^S_k} \beta^{max}_j X^S_{jkn} \quad \forall k \in \mathbf{K}^{LS} \setminus (\mathbf{K}^{NS} \cup \mathbf{K}^{NM}), n \tag{5.8}$$

5.3.3 *Material handling restrictions*

We introduce $W_{jj'kn} \in \{0,1\}$, which is 1 if material k is transferred from unit or vessel j to $j^{'}$ at time point n; and $F_{jj'kn} \in \mathbb{R}_{+}$, the amount of material k transferred from j to $j^{'}$ at time point n.

First, material transfer can occur only if the corresponding binary variable is activated.

$$F_{jj'kn} \leq M \cdot W_{jj'kn} \;\; \forall k \in \mathbf{K}^{NS} \cup \mathbf{K}^{NM}, j \in \mathbf{J}_k^{+} \cup \mathbf{J}_k^{S}, j^{'} \in \mathbf{J}_k^{-} \cup \mathbf{J}_k^{S}, n \tag{5.9}$$

where M is a sufficiently large number.

A material with no-splitting restriction that is produced in a unit (or stored in a vessel) can only be transferred to at most one unit or vessel. Furthermore, if the material is transferred from a vessel or a unit, the amount transferred is equal to the amount that was stored in the vessel or the unit.

$$\sum_{j^{'} \in \mathbf{J}_k^{-} \cup \mathbf{J}_k^{S}} W_{jj'kn} \leq 1 \;\; \forall k \in \mathbf{K}^{NS}, j \in \mathbf{J}_k^{+} \cup \mathbf{J}_k^{S}, n \tag{5.10}$$

$$S_{jkn}^{S} - M(1 - \sum_{j^{'} \in \mathbf{J}_k^{-} \cup \mathbf{J}_k^{S}} W_{jj'kn}) \leq \sum_{j^{'} \in \mathbf{J}_k^{-} \cup \mathbf{J}_k^{S}} F_{jj'kn} \leq S_{jkn}^{S} \;\; \forall k \in \mathbf{K}^{NS}, j \in \mathbf{J}_k^{S} \cup (\mathbf{J}_k^{+} \cap \mathbf{J}^{H}), n \tag{5.11}$$

Similarly, a material with no-mixing restriction can only be transferred from at most one unit or vessel to a unit that can consume (or a vessel that can store) the material. Also, if the material is transferred to a vessel, the amount transferred is equal to the amount stored in the vessel after the transfer.

$$\sum_{j^{'} \in \mathbf{J}_k^{+} \cup \mathbf{J}_k^{S}} W_{j'jkn} \leq 1 \;\; \forall k \in \mathbf{K}^{NM}, j \in \mathbf{J}_k^{-} \cup \mathbf{J}_k^{S}, n \tag{5.12}$$

$$S_{jk(n+1)}^{S} - M(1 - \sum_{j^{'} \in \mathbf{J}_k^{+} \cup \mathbf{J}_k^{S}} W_{j'jkn}) \leq \sum_{j^{'} \in \mathbf{J}_k^{+} \cup \mathbf{J}_k^{S}} F_{j'jkn} \leq S_{jk(n+1)}^{S} \;\; \forall k \in \mathbf{K}^{NM}, j \in \mathbf{J}_k^{S}, n \tag{5.13}$$

The amount of material transferred into a unit must be equal to the amount consumed.

Likewise, the amount of material transferred out of a unit must be equal to the amount produced.

$$\sum_{i \in \mathbf{I}_j \cap \mathbf{I}_k^-} -\rho_{ik} B_{ijn} = \sum_{j' \in \mathbf{J}_k^+ \cup \mathbf{J}_k^S} F_{j'jkn} \quad \forall k \in \mathbf{K}^{NS} \cup \mathbf{K}^{NM}, j \in \mathbf{J}_k^-, n \tag{5.14}$$

$$\sum_{i \in \mathbf{I}_j \cap \mathbf{I}_k^+} \rho_{ik} B_{ij(n-\bar{\tau}_{ij})} = \sum_{j' \in \mathbf{J}_k^- \cup \mathbf{J}_k^S} F_{jj'kn} \quad \forall k \in \mathbf{K}^{NS} \cup \mathbf{K}^{NM}, j \in \mathbf{J}_k^+ \setminus \mathbf{J}^H, n \tag{5.15}$$

The material balance in vessels that store materials with handling restrictions is,

$$S^S_{jk(n+1)} = S^S_{jkn} + \sum_{j' \in \mathbf{J}_k^+ \cup \mathbf{J}_k^S} F_{j'jkn} - \sum_{j' \in \mathbf{J}_k^- \cup \mathbf{J}_k^S} F_{jj'kn} \leq \beta_j^{max} X^S_{jkn} \quad \forall k \in \mathbf{K}^{NS} \cup \mathbf{K}^{NM}, j \in \mathbf{J}_k^S, n \tag{5.16}$$

The material balance in units that store output materials with handling restrictions is,

$$S^S_{jk(n+1)} = S^S_{jkn} + \sum_{i \in \mathbf{I}_j \cap \mathbf{I}_k^+} \rho_{ik} B_{ij(n-\bar{\tau}_{ij})} - \sum_{j' \in \mathbf{J}_k^- \cup \mathbf{J}_k^S} F_{jj'kn} \quad \forall k \in \mathbf{K}^{NS} \cup \mathbf{K}^{NM}, j \in \mathbf{J}_k^+ \cap \mathbf{J}^H, n \tag{5.17}$$

5.3.4 *Sequence-dependent changeovers*

We approximate the sequence-dependent changeover times with sequence-independent changeover times (setups). Given the changeover time from task i' to i in unit j, $\bar{\sigma}_{i'ij} = \lceil \sigma_{i'ij}/\delta \rceil$, we obtain an approximate setup time, $\tilde{\sigma}_{ij} = \max_{i' \in \mathbf{I}_j} \bar{\sigma}_{i'ij}$, from any task to task i in unit j. Likewise, we approximate the changeover cost for switching from task i' to i in unit j, $\sigma^C_{i'ij}$, with $\tilde{\sigma}^C_{ij} = \max_{i' \in \mathbf{I}_j} \sigma^C_{i'ij}$.

Next, we model setups utilizing the concept of unit modes. For a unit to transition from one mode to another, the unit must first transition to the neutral mode, and then to the target mode. We introduce $R_{ijn} \in \{0,1\}$, which is 1 if unit j is in mode i at time point n (i.e. it can carry out task i); $R^N_{jn} \in \{0,1\}$, which is 1 if unit j is in neutral mode at time point n; and $X^C_{ijn}/X^N_{ijn} \in \{0,1\}$, which is 1 if the mode of unit j is shifted to/from mode i from/to neutral at time point n. We enforce mode balance constraints as follows.

$$R_{ijn} = R_{ij(n-1)} + X_{ij(n-\bar{\tau}_{ij})} - X_{ijn} + X^{C}_{ij(n-\bar{\sigma}_{ij})} - X^{N}_{ijn} \quad \forall i, j \in \mathbf{J}_i \cap \mathbf{J}^{CO}, n \tag{5.18}$$

$$R^{N}_{jn} = R^{N}_{j(n-1)} + \sum_{i \in \mathbf{I}_j} X^{N}_{ijn} - \sum_{i \in \mathbf{I}_j} X^{C}_{ijn} \quad \forall j \in \mathbf{J}^{CO}, n \tag{5.19}$$

$$\sum_{i \in \mathbf{I}_j} R_{ij(n=0)} = 1 \quad \forall j \in \mathbf{J}^{CO} \tag{5.20}$$

Eq. 5.20 ensures that every unit with changeover starts at a single mode.

5.3.5 *Continuous processes*

We model a continuous processing task as a consecutive series of batch tasks with processing time of δ (i.e. $\bar{\tau}_{ij} = 1$). Accordingly, the minimum and maximum processing rate of unit j, given by $\hat{\beta}^{min}_{j}$ and $\hat{\beta}^{max}_{j}$, are converted to minimum and maximum capacities of the unit per time period (i.e. $\beta^{min}_{j} = \hat{\beta}^{min}_{j} \cdot \delta$ and $\beta^{max}_{j} = \hat{\beta}^{max}_{j} \cdot \delta$). Likewise, we convert the per hour cost of running task i in unit j, given by $\hat{\alpha}_{ij}$, to per period cost (i.e. $\alpha^{V}_{ij} = \hat{\alpha}_{ij} \cdot \delta$).

We introduce $Y_{ijn} \in \{0, 1\}$, which is 1 if continuous processing task i starts in unit j at time point n.

$$Y_{ijn} \geq X_{ijn} - X_{ij(n-1)} \quad \forall j \in \mathbf{J}^{CP}, i \in \mathbf{I}_j, n \tag{5.21}$$

If continuous processing task i has to continue for at least ψ_i hours, we enforce,

$$X_{ijn} \geq \sum_{n'=n}^{n-\bar{\psi}_i+1} Y_{ijn'} \quad \forall j \in \mathbf{J}^{CP}, i \in \mathbf{I}_j, n \tag{5.22}$$

where $\bar{\psi}_i = \lceil \psi_i / \delta \rceil$.

5.3.6 *Objective functions*

We consider three objective functions: makespan minimization (Eq. 5.23), cost minimization (Eq. 5.24) and profit maximization (Eq. 5.25).

$$min\ MS:\ \ MS \geq \sum_{i} \sum_{j \in \mathbf{J}_i} (n + \bar{\tau}_{ij}) X_{ijn} \ \ \forall\ n \tag{5.23}$$

$$min\ \sum_{i} \sum_{j \in \mathbf{J}_i} \sum_{n} (\alpha^F_{ij} X_{ijn} + \alpha^V_{ij} B_{ijn}) + \sum_{i} \sum_{j \in \mathbf{J}_i} \sum_{n} \tilde{\sigma}^C_{ij} X^C_{ijn} \tag{5.24}$$

$$max\ \sum_{k} \pi_k (S_{k(\bar{H}+1)} + \sum_{n} \bar{\bar{\xi}}_{kn}) - \sum_{i} \sum_{j \in \mathbf{J}_i} \sum_{n} (\alpha^F_{ij} X_{ijn} + \alpha^V_{ij} B_{ijn}) - \sum_{i} \sum_{j \in \mathbf{J}_i} \sum_{n} \tilde{\sigma}^C_{ij} X^C_{ijn} \tag{5.25}$$

where $MS \in \mathbb{R}_+$ is a variable for makespan. The full first stage model consists of Eqs. 5.1-5.22 and an objective function.

5.4 SECOND STAGE: MAPPING ALGORITHM

5.4.1 *Overview*

The overview of the algorithm is presented in Figure 5.1. In brief, the algorithm scans through the first stage solution from the beginning of the horizon to the end and detects the occurrence of specific events: (i) material production (production event), (ii) material consumption (consumption event), and (iii) material storage or withdrawal (storage event). When an event is detected, the algorithm generates grid points on the appropriate grids, maps the event to the generated UGP and MGP, and stores necessary information in algorithmic sets and parameters.

The first stage solution is scanned by looping over set elements in sets, in the following hierarchy (from highest to lowest): set of (1) time points (n), (2) materials (k), (3) tasks (i), and (4) units (j). The blocks presented in Figure 5.1 either represent (i) a set in which the algorithm is looping over (dashed outline), or (ii) a conditional statement to detect the aforementioned events (solid outline). Different colors are used to distinguish the process characteristics addressed in each block. The full algorithm is presented in Appendix B.3.

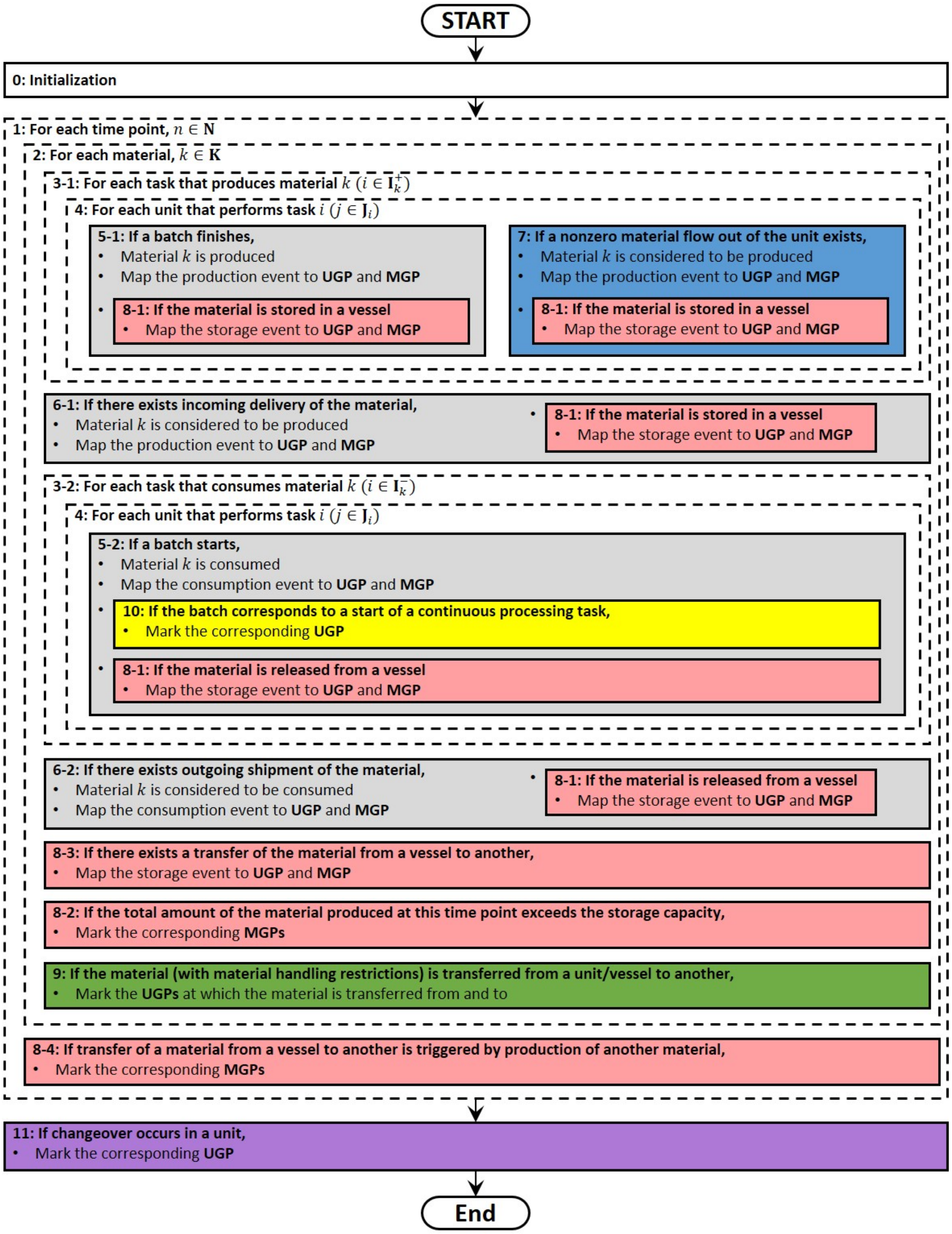

START
0: Initialization
1: For each time point, $n \in \mathbf{N}$
2: For each material, $k \in \mathbf{K}$
3-1: For each task that produces material k ($i \in \mathbf{I}_k^+$)
4: For each unit that performs task i ($j \in \mathbf{J}_i$)
5-1: If a batch finishes,
• Material k is produced
• Map the production event to UGP and MGP
• 8-1: If the material is stored in a vessel
• Map the storage event to UGP and MGP
7: If a nonzero material flow out of the unit exists,
• Material k is considered to be produced
• Map the production event to UGP and MGP
• 8-1: If the material is stored in a vessel
• Map the storage event to UGP and MGP
6-1: If there exists incoming delivery of the material,
• Material k is considered to be produced
• Map the production event to UGP and MGP
• 8-1: If the material is stored in a vessel
• Map the storage event to UGP and MGP
3-2: For each task that consumes material k ($i \in \mathbf{I}_k^-$)
4: For each unit that performs task i ($j \in \mathbf{J}_i$)
5-2: If a batch starts,
• Material k is consumed
• Map the consumption event to UGP and MGP
• 10: If the batch corresponds to a start of a continuous processing task,
• Mark the corresponding UGP
• 8-1: If the material is released from a vessel
• Map the storage event to UGP and MGP
6-2: If there exists outgoing shipment of the material,
• Material k is considered to be consumed
• Map the consumption event to UGP and MGP
• 8-1: If the material is released from a vessel
• Map the storage event to UGP and MGP
8-3: If there exists a transfer of the material from a vessel to another,
• Map the storage event to UGP and MGP
8-2: If the total amount of the material produced at this time point exceeds the storage capacity,
• Mark the corresponding MGPs
9: If the material (with material handling restrictions) is transferred from a unit/vessel to another,
• Mark the UGPs at which the material is transferred from and to
8-4: If transfer of a material from a vessel to another is triggered by production of another material,
• Mark the corresponding MGPs
11: If changeover occurs in a unit,
• Mark the corresponding UGP
End

Figure 5.1: Overview of the mapping algorithm.

5.4.2 *Production events*

A production event is an event in which a material is produced. Three types of production events are detected: (i) a material is produced at the end of a batch, (ii) a material is delivered, and (iii) a material that was held in a unit is discharged.

In block 5-1, the first type is detected by checking whether a batch that produces a given material finishes. If so, the algorithm adds a UGP and MGP for the event and maps the event to them. Such information is recorded in an algorithmic set, $\mathbf{M}_k^P$, which is the grid points at which material k is produced, and algorithmic parameters $f_{ikm}^1 \in \{0,1\}$, which is 1 if an event associated with a batch of task i is mapped to MGP (k,m); $f_{jkm}^2 \in \{0,1\}$, which is 1 if an event associated with a batch in unit j is mapped to MGP (k,m); and $f_{ijkmm'}^4 \in \{0,1\}$, which is 1 if an event associated with a batch of task i is mapped to UGP (j,m) and MGP $(k,m^{'})$.

The second type is detected in block 6-1, where it is checked if there exists material delivery. For each event detected, the algorithm adds a MGP for the event, maps the event to it, and stores the mapping information in $\mathbf{M}_k^P$ and $f_{kmp}^5 \in \{0,1\}$, which is 1 if p^{th} delivery or demand is at MGP (k,m).

The third type of production event is detected in block 7. A material that is stored in a unit is produced only at time points at which nonzero flow out of the unit exists. The algorithm detects nonzero material flows out of the unit, adds a UGP and MGP for the event, and maps the event to them. The algorithm utilizes algorithmic sets $\mathbf{M}_k^P$ and $\mathbf{M}_k^H$, and parameters f_{ikm}^1 and $g_{jkmm'}^1$, to record the mapping information. Set $\mathbf{M}_k^H$ is the grid points at which material k is held in a unit after being produced, and $g_{jkmm'}^1 \in \{0,1\}$ is 1 if the production event due to discharge of material k from unit j is mapped to UGP (j,m) and MGP $(k,m^{'})$

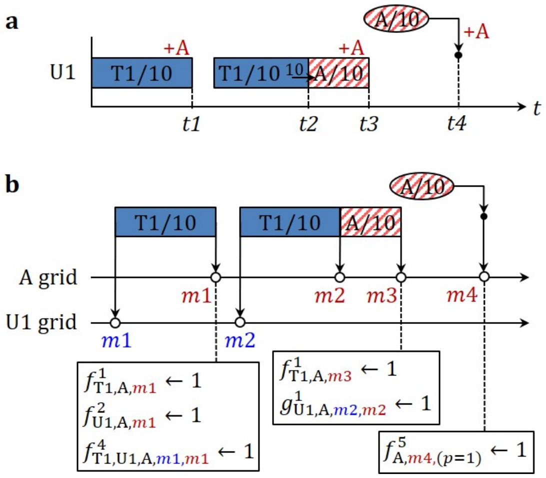

a
U1
T1/10
+A
T1/10
10
A/10
+A
A/10
+A
t
t1
t2
t3
t4
b
A/10
T1/10
T1/10
A/10
A grid
m1
m2
m3
m4
U1 grid
m1
m2
$f^1_{\mathrm{T1,A},m1} \leftarrow 1$
$f^2_{\mathrm{U1,A},m1} \leftarrow 1$
$f^4_{\mathrm{T1,U1,A},m1,m1} \leftarrow 1$
$f^1_{\mathrm{T1,A},m3} \leftarrow 1$
$g^1_{\mathrm{U1,A},m2,m2} \leftarrow 1$
$f^5_{\mathrm{A},m4,(p=1)} \leftarrow 1$

Figure 5.2: Illustration of production events: (a) a first stage solution, and (b) the mapping of the production events onto unit and material specific grids.

The mapping of the three types of events is illustrated in Figure 5.2. In Figure 5.2(a), two batches of T1 finish at $t1$ and $t2$, producing A. Material A produced by the second batch of T1 is first held in the unit and is discharged at $t3$. Finally, A is delivered at $t4$. The output of the algorithm is presented in Figure 5.2(b). At $t1$, the algorithm detects that a batch of T1, which produces A, is finished. Hence, the corresponding event is mapped to UGP $(\mathrm{U1}, m1)$ and MGP $(\mathrm{A}, m1)$. At $t3$, the algorithm detects the existence of nonzero flow out of U1 and maps the corresponding event to UGP $(\mathrm{U1}, m2)$ and MGP $(\mathrm{A}, m3)$. Next, at $t4$, the algorithm detects an delivery of A. The event is mapped to MGP $(\mathrm{A}, m4)$. Finally, since A is produced at MGPs $(\mathrm{A}, m1)$, $(\mathrm{A}, m3)$ and $(\mathrm{A}, m4)$, and it was held in MGP $(\mathrm{A}, m2)$, we have $\mathbf{M}_{\mathrm{A}}^{P} = \{m1, m3, m4\}$ and $\mathbf{M}_{\mathrm{A}}^{H} = \{m2\}$.

5.4.3 *Consumption events*

A consumption event is an event in which a material is consumed. Two types of consumption events are detected: (i) a material is consumed at the start of a batch, and (ii) a material is shipped (outgoing flow).

The first type of event is detected in block 5-2, where it is checked whether a batch of task that consumes a given material starts. If such an event is identified, the algorithm adds a UGP and MGP and maps the event to them, and the mapping information is recorded in an algorithmic set, $\mathbf{M}_{k}^{C}$, which is the grid points at which material k is consumed; and parameters f_{ikm}^{1}, f_{jkm}^{2}, $f_{kmm'}^{3}$ and $f_{ijkmm'}^{4}$. Parameter $f_{kmm'}^{3} \in \{0, 1\}$ is 1 if the timing of MGP (k, m) at which material k is produced is before $(k, m^{'})$ at which it is consumed.

The second type of consumption event is detected in block 6-2, where it is checked if there exists material shipment. A shipment is treated as material consumption and thereby the algorithm adds a MGP and maps the event to it. Finally, the mapping information is recorded in $\mathbf{M}_{k}^{C}$ and f^{5}.

To illustrate, consider the first stage solution shown in Figure 5.3(a), where 20 units of A are produced at $t1$ by a batch of T1, and 10 units of A are consumed at $t2$ by a batch of T2. At $t3$, the remaining 10 units of A are shipped. At $t2$, the algorithm detects the start of a batch of T2 that consumes A. As shown in Figure 5.3(b), the corresponding event is then mapped to UGP $(\mathrm{U2}, m1)$ and MGP $(\mathrm{A}, m2)$. Next, at $t3$, the algorithm detects the shipment of A, which

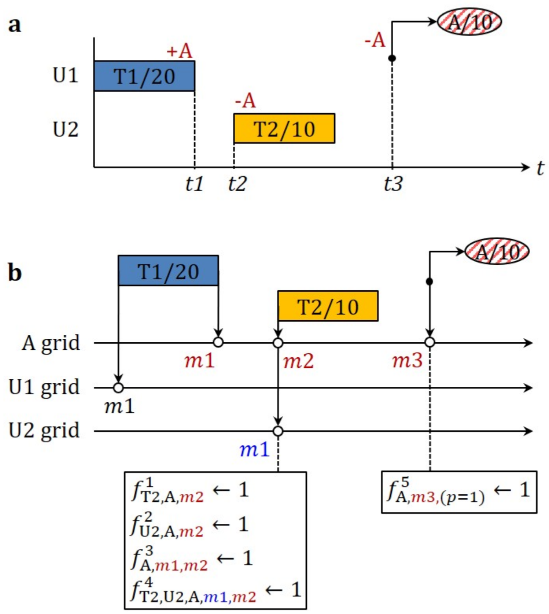

a
U1
U2
+A
T1/20
-A
T2/10
-A
A/10
t1
t2
t3
t
b
T1/20
T2/10
A/10
A grid
U1 grid
U2 grid
m1
m2
m3
m1
m1
$f^{1}_{\mathrm{T2,A},m2} \leftarrow 1$
$f^{2}_{\mathrm{U2,A},m2} \leftarrow 1$
$f^{3}_{\mathrm{A},m1,m2} \leftarrow 1$
$f^{4}_{\mathrm{T2,U2,A},m1,m2} \leftarrow 1$
$f^{5}_{\mathrm{A},m3,(p=1)} \leftarrow 1$

Figure 5.3: Illustration of consumption events: (a) a first stage solution, and (b) the mapping of the consumption events onto unit and material specific grids.

is mapped to $(\mathrm{A}, m3)$. Finally, since A is consumed at MGPs $(\mathrm{A}, m2)$ and $(\mathrm{A}, m3)$, we have $\mathbf{M}^{\mathrm{C}}_{\mathrm{A}} = \{m2, m3\}$.

5.4.4 *Storage events: type 1*

Naturally, storage events do not occur independently unlike production and consumption events; instead, they are triggered by another event. Depending on how the events are triggered, there are two types: storage events of a material that are triggered by (i) production or consumption events of the same material, and (ii) storage events of another material.

The first type of events are detected in block 8-1. Due to how these storage events are triggered, the algorithm executes block 8-1 whenever a production or consumption event is detected (i.e. blocks 5-1, 5-2, 6-1, 6-2 and 7) (see Figure 5.1). When a storage event is detected, the algorithm adds a UGP, maps the event to the UGP and the MGP to which the triggering event is mapped. We note that the storage event and the triggering event are mapped to the same MGP, since the timing of the two events are identical. The mapping information is then stored in $g^{3}_{jkmm'} \in \{0, 1\}$, which is 1 if storage event of material k in vessel j is mapped to UGP (j, m) and MGP $(k, m^{'})$. Next, the algorithm checks whether the material is stored in the vessel after the event, and if so, such information is stored in $g^2_{jkm} \in \{0,1\}$, which is 1 if material k is stored in vessel j between MGPs (k,m) and $(k,m+1)$.

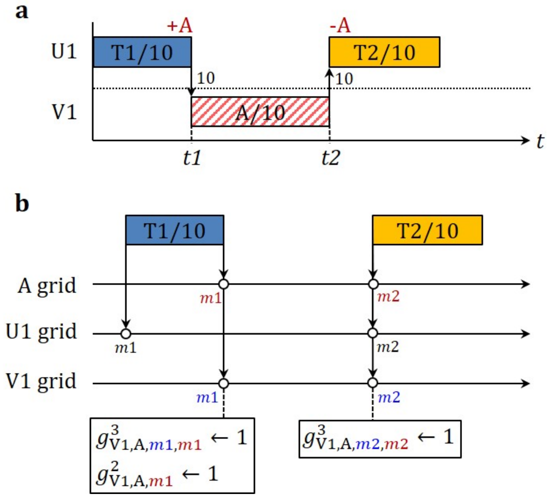

a
+A
-A
U1
T1/10
T2/10
10
10
V1
A/10
t
t1
t2
b
T1/10
T2/10
A grid
m1
m2
U1 grid
m1
m2
V1 grid
m1
m2
$g^3_{V1,A,m1,m1} \leftarrow 1$
$g^2_{V1,A,m1} \leftarrow 1$
$g^3_{V1,A,m2,m2} \leftarrow 1$

Figure 5.4: Illustration of the first type of storage events: (a) a first stage solution, and (b) the mapping of the storage events onto unit and material specific grids.

To illustrate, consider the first stage solution shown in Figure 5.4(a), where two storage events of A in V1 triggered by (i) a production event of A at $t1$ and (ii) a consumption event of A at $t2$ are observed. As shown in Figure 5.4(b), the algorithm detects and maps the first storage event to UGP $(\text{V1}, m1)$ and MGP $(\text{A}, m1)$, and the second storage event to UGP $(\text{V1}, m2)$ and MGP $(\text{A}, m2)$. Note that, because A is stored in V1 after the first event, $g^2_{\text{V1,A},m1}$ is set to 1.

However, a special case of this type of events requires further attention. For example, consider the first stage solution shown in Figure 5.5(a), where at $t1$, 10 units of A are produced, 5 of which are consumed by a batch of T2 and the remaining 5 are stored in V1. In Figure 5.5(b), at $t1$, the mapping algorithm detects two storage events that are triggered by (i) the production event of A at the end of batch of T1, and (ii) the consumption event of A at the beginning of batch of T2. Between these two events, the amount of A stored in V1 is 10, and after the second event, the amount drops to 5.

If the maximum capacity of V1 is greater than or equal to 10, any timing of the two events will provide a feasible solution as long as the first event precedes the second. However, if the maximum capacity of V1 is smaller than 10, a feasible solution can only be obtained when the timing of the two events are equal such that, at max, 5 units of A are stored in V1 at any time.

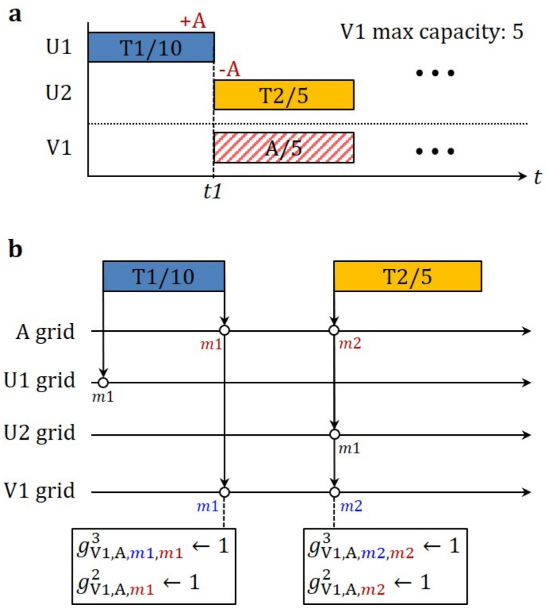

a
+A
V1 max capacity: 5
U1
T1/10
-A
U2
T2/5
V1
A/5
t1
t
b
T1/10
T2/5
A grid
m1
m2
U1 grid
m1
U2 grid
m1
V1 grid
m1
m2
$g^3_{\mathrm{V1,A},m1,m1} \leftarrow 1$
$g^2_{\mathrm{V1,A},m1} \leftarrow 1$
$g^3_{\mathrm{V1,A},m2,m2} \leftarrow 1$
$g^2_{\mathrm{V1,A},m2} \leftarrow 1$

Figure 5.5: Special case of the first type of storage events: (a) a first stage solution, and (b) the mapping of the storage events onto unit and material specific grids.

To enforce such a constraint in the third stage model, the MGPs mapped from the two events are paired using $g^4_{kmm'} \in \{0,1\}$, which is 1 if MGPs (k,m) and (k,m') have the same timing. Although we set the timings of the two MGPs to be identical, the calculated inventory at the first MGP (e.g. $(\mathrm{A},m1)$ in this example) would still exceed the vessel capacity in the third stage model. Thus, we do not enforce vessel inventory constraints (Eq. 5.52) at these MGPs, and they are stored in sets $\mathbf{M}_k^{P-}/\mathbf{M}_k^{C-}$, which is the grid points at which material k is produced/consumed but Eq. 5.52 is not enforced. The detection of such case of events, pairing of associated MGPs and update of sets $\mathbf{M}_k^{P-}$ and $\mathbf{M}_k^{C-}$ are done in block 8-2.

5.4.5 *Storage events: type 2*

The second type of storage events include events that are triggered by storage events of another material. For example, a storage event of A in a vessel can be triggered if another material B transfers into the vessel (i.e. storage event of B), which then forces A to be transferred to another vessel.

These events are detected in block 8-3 in the mapping algorithm, where it is checked if (i) no production or consumption event of a material has occurred, but (ii) the material is transferred from one vessel to another. In such cases, the algorithm adds a UGP and MGP, maps the event to them, and records the mapping information in g^2 and g^3.

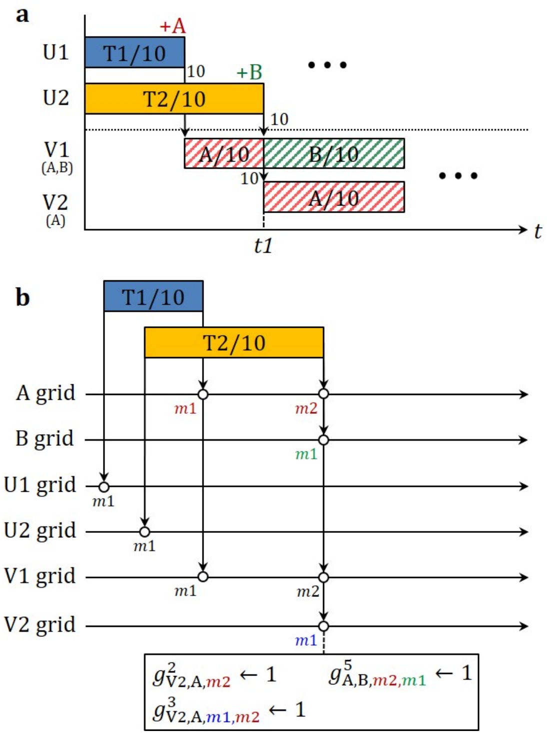


Figure 5.6: Illustration of the second type of storage events: (a) a first stage solution, and (b) the mapping of the storage events onto unit and material specific grids.

For example, consider the first stage solution shown in Figure 5.6(a). At $t1$, a batch of T2 finishes and produces B, which then gets stored in V1. As a result, A that was stored in V1 is transferred to V2. In Figure 5.6(b), when the batch of T2 finishes, the algorithm detects a storage event of B in V1 (i.e. B is transferred into V1), which triggers a storage event of A in V2 (i.e. A is transferred into V2). Hence, the algorithm maps the storage event of A in V2 to UGP $(\mathrm{V2}, m1)$ and MGP $(\mathrm{A}, m2)$.

However, the above mapping information is not sufficient to guarantee a feasible solution in the third stage model. Specifically, the timings of MGPs $(\mathrm{A}, m2)$ and $(\mathrm{B}, m1)$ must be equal in a feasible solution, since the storage event of A, mapped to the former MGP, is triggered by the storage event of B, mapped to the latter MGP. Such pairing of MGPs is done in block 8-4, and the pairing information is recorded in $g^5_{kk'mm'} \in \{0,1\}$, which is 1 if MGPs (k,m) and (k',m') have the same timing (e.g. $g^5_{\mathrm{A,B},m2,m1} = 1$ in the example).

5.4.6 *Extraction of other information*

In addition to mapping of production, consumption and storage events onto the unit and material specific grids, further extraction of information from the first stage solution is necessary to accommodate (i) material handling restrictions, (ii) continuous processes and (iii) sequence-dependent changeovers.

First, when materials with handling restrictions are present, the algorithm in block 9 detects material transfers of such materials by checking the values of the material transfer binaries (i.e. $W_{jj'kn}$) and stores the information in $g^{6}_{jj'kmm'} \in \{0,1\}$, which is 1 if material k is transferred between events mapped to UGPs (j,m) and $(j^{'},m^{'})$. For instance, consider the first stage solution shown in Figure 5.7(a), where A has no mixing restriction, and the curved arrows represent the transfer of A. In Figure 5.7(b), the algorithm detects that A is transferred between the events mapped to UGPs $(\mathrm{U1},m1)$ and $(\mathrm{U3},m1)$, $g^{6}_{\mathrm{U1,U3,A},m1,m1}$ is set to 1. Likewise, the other two material transfers of A are detected, and $g^{6}_{\mathrm{U2,V1,A},m1,m2}$ and $g^{6}_{\mathrm{V1,U3,A},m4,m2}$ are set to 1.

Second, we identify the batches that correspond to the start of continuous processing runs by checking the values of the corresponding binaries (i.e. Y_{ijn}). Such information is stored in $g^{7}_{ijm} \in \{0,1\}$, which is 1 if the start of a continuous processing task i is mapped to UGP (j,m). An illustration of this process is shown in Figure 5.8. At $t1$ in the first stage solution, a continuous processing task T1 starts. At the beginning of the first batch, the algorithm detects that there is no immediately preceding batch, and thus sets $g^{7}_{\mathrm{T1,U1}m1}$ to 1.

Finally, to extract information on sequence-dependent changeovers, block 11 identifies the sequence of tasks executed in each unit and stores the information in $g^{8}_{ii'jm} \in \{0,1\}$, which is 1 if a batch of task i of which events are mapped to UGP (j,m) is followed by a batch of task $i^{'}$. An illustration of this process is shown in Figure 5.9. At UGP $(\mathrm{U1},m1)$, the algorithm identifies that the UGP is mapped from the events associated with a batch of T1 and that the next UGP is mapped from the events associated with a batch of T2. Accordingly, $g^{8}_{\mathrm{T1,T2,U1},m1}$ is set to 1.

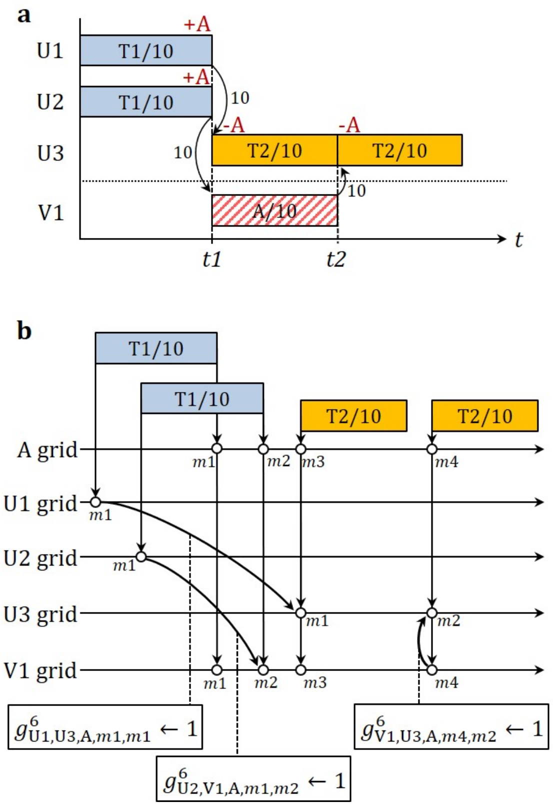

a
U1
+A
T1/10
U2
+A
T1/10
10
-A
-A
U3
10
T2/10
T2/10
10
V1
A/10
t
t1
t2
b
T1/10
T1/10
T2/10
T2/10
A grid
m1
m2
m3
m4
U1 grid
m1
U2 grid
m1
U3 grid
m1
m2
V1 grid
m1
m2
m3
m4
$g^{6}_{\mathrm{U1,U3,A},m1,m1} \leftarrow 1$
$g^{6}_{\mathrm{V1,U3,A},m4,m2} \leftarrow 1$
$g^{6}_{\mathrm{U2,V1,A},m1,m2} \leftarrow 1$

Figure 5.7: Detection of material transfers: (a) a first stage solution, and (b) the mapping of the solution onto unit and material specific grids.

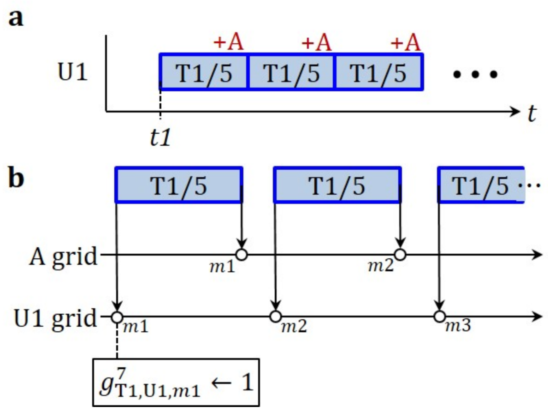


Figure 5.8: Detection of the start of continuous processing runs: (a) a first stage solution, and (b) the mapping of the solution onto unit and material specific grids.

5.5 THIRD STAGE: CONTINUOUS-TIME LP MODEL

Based on the information collected in the second stage mapping algorithm, we formulate an LP model to improve the solution accuracy and quality by re-optimizing the batch sizes, inventory

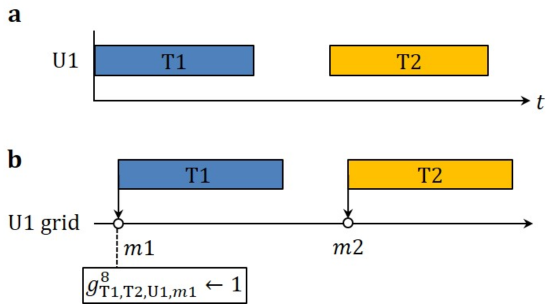


Figure 5.9: Detection of sequence-dependent changeovers: (a) a first stage solution, and (b) the mapping of the solution onto unit and material specific grids.

level, and timing of events, while accurately accounting for different process characteristics. We introduce the following variables (see Figure 5.10 for an illustration).

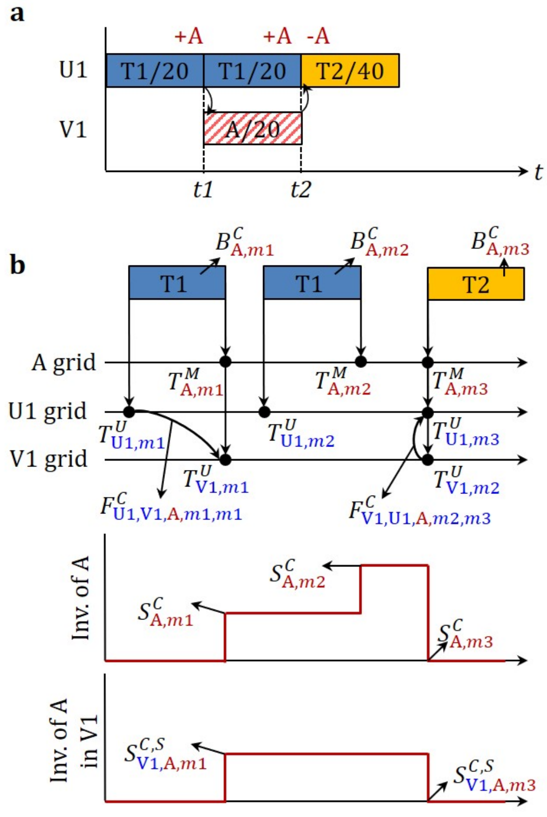


Figure 5.10: Illustration of the decision variables in the third stage model: (a) the first stage solution and (b) the third stage decision variables.

$B^{C}_{km} \in \mathbb{R}_{+}$ batch size of the production or consumption event mapped to MGP (k, m)

$F^{C}_{jj'kmm'} \in \mathbb{R}_{+}$ amount of material k transferred between the events mapped to UGPs (j, m) and $(j^{'}, m^{'})$

$S^{C}_{km} \in \mathbb{R}_{+}$ inventory level of material k at MGP (k, m)

$S^{C,S}_{jkm} \in \mathbb{R}_{+}$ amount of material k stored in vessel j at MGP (k, m)

$T^{U}_{jm} \in \mathbb{R}_{+}$ timing of UGP (j, m)

$T^{M}_{km} \in \mathbb{R}_{+}$ timing of MGP (k, m)

5.5.1 *Grid point sequencing and timing constraints*

The sequencing between the production and consumption events of each material is enforced as follows.

$$T^{M}_{km} \leq T^{M}_{km'} \quad \forall k, m, m^{'} : f^{3}_{kmm'} \tag{5.26}$$

Note that, for simplicity, we treat the algorithmic parameters as booleans.

In batch processing units, a new batch can only start after the finish time of the preceding batch plus the sequence-dependent changeover time.

$$T^{U}_{jm} + (\tau^{F}_{ij} + \tau^{V}_{ij} B^{C}_{km'} + \sum_{i^{'} \neq i : g^{8}_{ii^{'} jm}} \sigma_{ii^{'} j}) \leq T^{U}_{j(m+1)} \quad \forall i, j \notin \mathbf{J}^{CP}, k, m, m^{'} : f^{4}_{ijkmm'} \tag{5.27}$$

A continuous processing run can only start after the finish time of the preceding run plus the sequence-dependent changeover time.

$$T^{M}_{km'} + \sum_{i^{'} \neq i : g^{8}_{ii^{'} jm}} \sigma_{ii^{'} j} \leq T^{U}_{j(m+1)} \quad \forall i, j \in \mathbf{J}^{CP}, k, m, m^{'} \in \mathbf{M}^{P}_{k} : f^{4}_{ijkmm'} \tag{5.28}$$

When a unit stores one or more materials, we ensure that the materials can only be discharged after they are produced in the unit, and the unit can start the next batch only after all the materials stored in the unit are discharged.

$$T^{M}_{km'} \leq T^{M}_{km''} \quad \forall i, j, k, m, m^{'} \in \mathbf{M}^{H}_{k}, m^{''} : f^{4}_{ijkmm'} \wedge g^{1}_{jkmm''} \tag{5.29}$$

$$T^{M}_{km''} \leq T^{U}_{j(m+1)} \quad \forall j, k, m, m^{''} : g^{1}_{jkmm''} \tag{5.30}$$

Furthermore, we ensure no idle time in between batches that correspond to a single continuous processing run.

$$T^{M}_{km'} = T^{U}_{j(m+1)} \ \ \forall i, j \in \mathbf{J}^{CP}, k, m, m^{'} \in \mathbf{M}^{P}_{k} : f^{4}_{ijkmm'} \wedge g^{8}_{iijm} \wedge \neg g^{7}_{ij(m+1)} \tag{5.31}$$

Finally, we fix the intermediate release and due dates.

$$T^{M}_{km} = t^{\xi}_{kp} \ \ \forall \ k, m, p : f^{5}_{kmp} \tag{5.32}$$

5.5.2 *Time matching constraints*

Since (i) a single event can be mapped to multiple material- and unit-specific grids and (ii) some events must have identical timing in a feasible solution, we enforce the following.

$$T^{U}_{jm} = T^{M}_{km'} - (\tau^{F}_{ij} + \tau^{V}_{ij} B^{C}_{km'}) \ \ \forall i, j, k, m, m^{'} \in \mathbf{M}^{P}_{k} \cup \mathbf{M}^{H}_{k} : f^{4}_{ijkmm'} \tag{5.33}$$

$$T^{U}_{jm} = T^{M}_{km'} \ \ \forall i, j, k, m, m^{'} \in \mathbf{M}^{C}_{k} : f^{4}_{ijkmm'} \tag{5.34}$$

$$T^{U}_{jm} = T^{M}_{km'} \ \ \forall j, k, m, m^{'} : g^{3}_{jkmm'} \tag{5.35}$$

$$T^{M}_{km} = T^{M}_{km'} \ \ \forall k, m, m^{'} : g^{4}_{kmm'} \tag{5.36}$$

$$T^{M}_{km} = T^{M}_{k'm'} \ \ \forall k, k^{'}, m, m^{'} : g^{5}_{kk'mm'} \tag{5.37}$$

$$T^{M}_{km''} = T^{U}_{j'm'} \ \ \forall i, j \notin \mathbf{J}^{H}, j^{'}, k, m, m^{'}, m^{''} : f^{4}_{ijkmm''} \wedge g^{6}_{jj'kmm'} \tag{5.38}$$

$$T^{M}_{km''} = T^{U}_{j'm'} \ \ \forall j \in \mathbf{J}^{H}, j^{'}, k, m, m^{'}, m^{''} : g^{1}_{jkmm''} \wedge g^{6}_{jj'kmm'} \tag{5.39}$$

$$T^{M}_{km''} = T^{U}_{j'm'} \ \ \forall j, j^{'}, k, m, m^{'}, m^{''} : g^{3}_{jkmm''} \wedge g^{6}_{jj'kmm'} \tag{5.40}$$

The timings of UGPs and MGPs that are mapped from the same production, consumption and storage events are matched through Eqs. 5.33, 5.34 and 5.35, respectively; Eqs. 5.36 and 5.37 ensure that the MGPs paired by blocks 8-2 and 8-4 have the same timing; and Eqs. 5.38 and 5.39 ensure that, when a material with handling restrictions is transferred from a unit to another unit (or vessel), the timings at which the material is produced and consumed (or stored) are identical. Eq. 5.40 does the same for the material transfers from a vessel to a unit (or vessel).

5.5.3 *Material transfer and capacity constraints*

Whenever a material with handling restrictions transfers out of a unit, the amount transferred is equal to the amount produced in the unit. Likewise, whenever a material with handling restrictions transfers into a unit, the amount transferred is equal to the amount consumed in the unit.

$$\sum_{i \in \mathbf{I}_j \cap \mathbf{I}_k^+} \sum_{m'' \in \mathbf{M}_k^P : f^4_{ijkmm''}} \rho_{ik} B^C_{km''} = \sum_{j' \in \mathbf{J}_k^- \cup \mathbf{J}_k^S} \sum_{m' : g^6_{jj'kmm'}} F^C_{jj'kmm'} \quad \forall k \in \mathbf{K}^{NS} \cup \mathbf{K}^{NM}, j \in \mathbf{J}_k^+, m \tag{5.41}$$

$$\sum_{i \in \mathbf{I}_j \cap \mathbf{I}_k^-} \sum_{m'' \in \mathbf{M}_k^C : f^4_{ijkmm''}} -\rho_{ik} B^C_{km''} = \sum_{j' \in \mathbf{J}_k^+ \cup \mathbf{J}_k^S} \sum_{m' : g^6_{j'jkm'm}} F^C_{j'jkm'm} \quad \forall k \in \mathbf{K}^{NS} \cup \mathbf{K}^{NM}, j \in \mathbf{J}_k^-, m \tag{5.42}$$

For each batch of material produced and stored in a unit, the total amount of the material discharged from the unit over time is equal to the amount initially produced.

$$B^C_{km'} = \sum_{m'' : g^1_{jkmm''}} B^C_{km''} \quad \forall i, j, k, m < M_j^U, m' \in \mathbf{M}_k^H : f^4_{ijkmm'} \tag{5.43}$$

$$B^C_{km'} \geq \sum_{m'' : g^1_{jkmm''}} B^C_{km''} \quad \forall i, j, k, m = M_j^U, m' \in \mathbf{M}_k^H : f^4_{ijkmm'} \tag{5.44}$$

However, we allow some materials to be left over in the unit if they are output materials of the last batch assigned to the unit (Eq. 5.44).

The unit capacity constraints for batch processing units that can (Eq. 5.45) or cannot (Eq. 5.46) store materials are enforced as follows.

$$\beta_j^{min} \leq B^C_{km} \leq \beta_j^{max} \quad \forall j \in \mathbf{J}^H \setminus \mathbf{J}^{CP}, k, m \in \mathbf{M}_k^H : f^2_{jkm} \tag{5.45}$$

$$\beta_j^{min} \leq B^C_{km} \leq \beta_j^{max} \quad \forall j \in \mathbf{J} \setminus (\mathbf{J}^H \cup \mathbf{J}^{CP}), k, m : f^2_{jkm} \tag{5.46}$$

In a continuous processing unit, the amount of material produced in the unit by a given run is bounded by the minimum and maximum processing rate of the unit multiplied by the duration of the run.

$$\begin{aligned} \hat{\beta}_j^{min}(T^M_{km'} - T^M_{k'm''}) \leq & B^C_{km'} \leq \hat{\beta}_j^{max}(T^M_{km'} - T^M_{k'm''}) \\ & \forall i, j \in \mathbf{J}^{CP}, k, k', m, m' \in \mathbf{M}^P_k, m'' \in \mathbf{M}^C_{k'} : f^4_{ijkmm'} \wedge f^4_{ijk'mm''} \end{aligned} \tag{5.47}$$

Furthermore, the batch sizes of the events that are associated with the same batch should be identical.

$$B^C_{km'} = B^C_{k'm''} \quad \forall i, j, k, k', m, m', m'' : f^4_{ijkmm'} \wedge f^4_{ijk'mm''},\ k < k' \tag{5.48}$$

Finally, the inventory level of a material stored in a vessel cannot exceed its capacity.

$$S^{S,C}_{jkm} \leq \beta^{max}_j \quad \forall j, k, m : g^2_{jkm} \tag{5.49}$$

5.5.4 *Inventory constraints*

We model the inventory level of materials at MGPs at which they are consumed (i.e. $\mathbf{M}^C_k$) to prevent subzero inventory level.

$$\begin{aligned} S^C_{km} = \gamma_k + & \sum_{m > m' \in \mathbf{M}^P_k} \Big(\sum_{i \in \mathbf{I}^+_k : f^1_{ikm'}} \rho_{ik} B^C_{km'} - \sum_{p : f^5_{km'p}} \xi_{kp} \Big) \\ & + \sum_{m \geq m' \in \mathbf{M}^C_k} \Big(\sum_{i \in \mathbf{I}^-_k : f^1_{ikm'}} \rho_{ik} B^C_{km'} - \sum_{p : f^5_{km'p}} \xi_{kp} \Big) \quad \forall k, m \in \mathbf{M}^C_k \end{aligned} \tag{5.50}$$

For materials with limited storage (i.e. $\mathbf{K}^{LS}$), the inventory levels at MGPs at which the materials are produced (i.e. $\mathbf{M}^P_k$) are modeled to prevent the inventory level from exceeding the storage capacity.

$$\begin{aligned} S^C_{km} = \gamma_k + & \sum_{m \geq m' \in \mathbf{M}^P_k} \Big(\sum_{i \in \mathbf{I}^+_k : f^1_{ikm'}} \rho_{ik} B^C_{km'} - \sum_{p : f^5_{km'p}} \xi_{kp} \Big) \\ & + \sum_{m > m' \in \mathbf{M}^C_k} \Big(\sum_{i \in \mathbf{I}^-_k : f^1_{ikm'}} \rho_{ik} B^C_{km'} - \sum_{p : f^5_{km'p}} \xi_{kp} \Big) \\ & \forall k \in \mathbf{K}^{LS} \setminus (\mathbf{K}^{NS} \cup \mathbf{K}^{NM}), m \in \mathbf{M}^P_k \end{aligned} \tag{5.51}$$

The inventory level of a material with limited storage must be equal to the summation of the inventory level of the material in each vessel.

$$S^{C}_{km} = \sum_{j:g^{2}_{jkm}} S^{C,S}_{jkm} \ \ \forall k \in \mathbf{K}^{LS} \setminus (\mathbf{K}^{NS} \cup \mathbf{K}^{NM}), m \in \left(\mathbf{M}^{P}_{k} \cup \mathbf{M}^{C}_{k}\right) \setminus \left(\mathbf{M}^{P-}_{k} \cup \mathbf{M}^{C-}_{k}\right) \tag{5.52}$$

Notice that the inventory levels of the vessels at MGPs in $\mathbf{M}^{P-}_{k}$ or $\mathbf{M}^{C-}_{k}$ are not modeled for the reasons discussed in Subsection 5.4.4.

The inventory level of a material with handling restrictions stored in a vessel at the current MGP is equal to the inventory at the previous MGP plus amount transferred in, minus the amount transferred out.

$$\begin{aligned} S^{C,S}_{jkm''} = S^{C,S}_{jk(m''-1)} + \sum_{j' \in \mathbf{J}^{+}_{k} \cup \mathbf{J}^{S}_{k}} \ \sum_{m':g^{6}_{j'jkm'm}} F^{C}_{j'jkm'm} - \sum_{j' \in \mathbf{J}^{-}_{k} \cup \mathbf{J}^{S}_{k}} \ \sum_{m':g^{6}_{jj'kmm'}} F^{C}_{jj'kmm'} \\ \forall k \in \mathbf{K}^{NS} \cup \mathbf{K}^{NM}, j \in \mathbf{J}^{S}_{k}, m, m'' : g^{3}_{jkmm''} \end{aligned} \tag{5.53}$$

5.5.5 *Objective functions*

For makespan minimization, we add the following constraint to ensure that the makespan is greater than or equal to the time each material is produced.

$$minMS : MS \geq T^{M}_{km} \ \ \forall k, m \in \mathbf{M}^{P}_{k} \tag{5.54}$$

We minimize the total processing and changeover costs for cost minimization.

$$min \sum_{i,k} \sum_{j \in \mathbf{J}_i} \sum_{m \in \mathbf{M}^{P}_{k}:f^{1}_{ikm} \wedge f^{2}_{jkm}} \frac{1}{|\mathbf{K}^{+}_{i}|} (\alpha^{F}_{ij} + \alpha^{V}_{ij} B^{C}_{km}) + \sum_{i,i'} \sum_{j \in \mathbf{J}_i} \sum_{m:g^{8}_{ii'jm}} \sigma^{C}_{ii'j} \tag{5.55}$$

We negate the effect of double counting the cost of tasks that are mapped to multiple material grids by dividing the term by the number of the materials produced by each given task.

When maximizing profit, we allow excess sales.

$$max \sum_{k} \pi_k (S^{C}_{k,M^{M}_{k}} + \sum_{p} \xi_{kp}) - \sum_{i,k} \sum_{j \in \mathbf{J}_i} \sum_{m \in \mathbf{M}^{P}_{k}:f^{1}_{ikm} \wedge f^{2}_{jkm}} \frac{1}{|\mathbf{K}^{+}_{i}|} (\alpha^{F}_{ij} + \alpha^{V}_{ij} B^{C}_{km}) - \sum_{i,i'} \sum_{j \in \mathbf{J}_i} \sum_{m:g^{8}_{ii'jm}} \sigma^{C}_{ii'j} \tag{5.56}$$

5.6 ILLUSTRATIVE EXAMPLES

We showcase the applicability and advantages of the proposed algorithm through three illustrative examples. The process data for the examples can be found in Supplementary Information of Lee and Maravelias (2019b).

5.6.1 *Example 1*

Consider the network shown in Figure 5.11(a), which consists of seven tasks executed on five units, producing four different products (P1-P4) from four raw materials (R1-R4). Multipurpose vessels V1-V3 have maximum capacities of 100, 50 and 50 kg, respectively, and the materials that can be stored in each vessel are $\{A, P4\}$, $\{A, B, P4\}$ and $\{A, B, C, P4\}$, respectively. The objective is to minimize makespan.

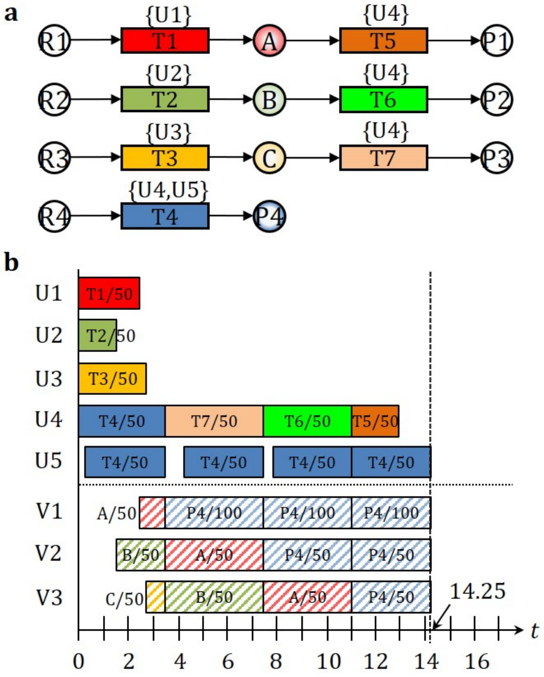


Figure 5.11: Example 1: (a) STN of example 1, and (b) DCA solution.

The solution obtained by DCA is shown in Figure 5.11(b), and we see that DCA correctly models multipurpose storage vessels. We observe both first and second types of storage events (e.g. at $t = 1.5$ h and $t = 3.5$ h, respectively) being detected and their timings optimized in the solution. In terms of the computational performance, DCA using $\delta = 1$ h finds the optimal solution of this example, 14.25 h, in a fraction of a second. In contrast, to obtain the same solution using a discrete-time model, $\delta = 0.25$ h is necessary, resulting in a model that is four times the size of the first stage model of DCA and a longer computational time (a few seconds).

5.6.2 *Example 2*

We consider an example using the network shown in Figure 5.12(a), consisting of 10 tasks performed on six units, where two raw materials (R1-R2) are converted into three products (P1-P3). Intermediate materials, S3 and S5, have handling restrictions and can be stored in two multipurpose vessels. Sequence-dependent changeover times and costs are present in U1, U4 and U5. At 24 h, 50, 100 and 100 kg of P1, P2 and P3 are demanded, respectively. The objective is to minimize cost while meeting demand.

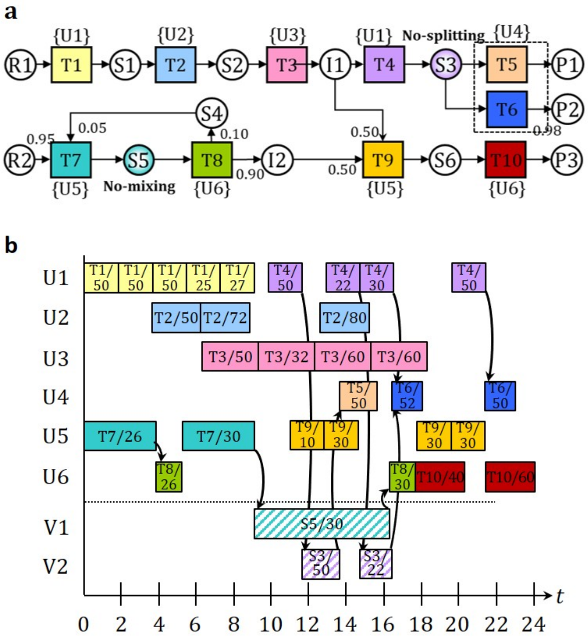


Figure 5.12: Example 2: (a) STN of example 2, and (b) DCA solution. The curved arrows represent material transfers of materials with handling restrictions.

The solution obtained by DCA is shown in Figure 5.12(b). We observe that the number of task transitions in U1, U4 and U5 is minimized to reduce changeover costs, and the material handling restrictions for S3 and S5 are satisfied. For instance, two batches of S5 produced by T7 (23 and 30 kg, respectively) are each consumed by a batch of T8 without getting mixed, even though the unit in which T8 is executed has maximum capacity of 60kg. In terms of computational performance, DCA using $\delta = 1$ h finds the optimal solution of this example, \$1083.93, in less than two minutes, while discrete-time model with $\delta = 0.1$ h takes more than two hours to find the same solution.

5.6.3 *Example 3*

We consider the network in Figure 5.13(a), where continuous processing tasks are represented as blocks with blue outline. Three continuous processing tasks, executed in U2, consume the same intermediate material A, which is produced by batch processing task T1. The output materials of these tasks are then further processed by another set of continuous processing tasks in U3, producing P1-P3. Unit U1 has a capacity of 0-200 kg, and the minimum/maximum processing rate of U2 and U3 are 30/50 kg/h. The continuous processing tasks executed in U2 and U3 require sequence-dependent changeovers. The demands for P1, P2 and P3 at the end of the horizon are 300, 200 and 200 kg, respectively. The objective is to maximize profit with excess sales allowed.

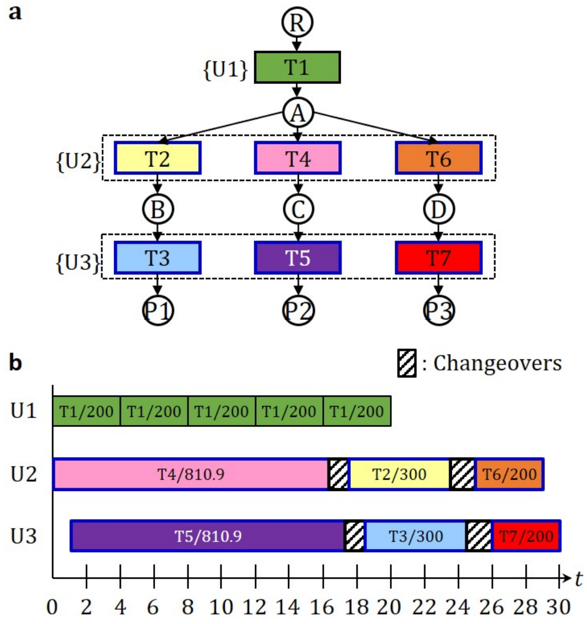


Figure 5.13: Example 3: (a) STN of example 3, and (b) DCA solution. Blocks with blue outline represent continuous processing tasks.

The solution obtained by DCA is shown in Figure 5.13(b). All the time and resources are allocated to produce the most profitable product, P2, except for the amounts needed to satisfy the demands of P1 and P3. The sequence of tasks executed in U2 (and U3) is the sequence in which the sequence-dependent changeover times and costs are minimized. Assuming the minimum residence time of the continuous processing tasks as 1 h, continuous-time model with 15 time slots finds the optimal solution for this example, $2760.00, in 10 minutes. Using DCA with $\delta = 1$ h a high quality solution of $2747.81 (less than 0.5% difference) is obtained in a few seconds.

5.7 Case study: Heineken brewing process

We present a case study inspired by a brewery from Heineken Mexico. For confidentiality reasons, the process data used in this case study cannot be explicitly revealed.

5.7.1 *Beer production process*

The facility produces a total of 10 different brand beers, through four steps (see Figure 5.14): (i) brewing, (ii) fermentation, (iii) maturation, (iv) filtering, carbonation and storage in bright beer tanks (BBTs). The products are distinguished by the raw materials (e.g. malt, hop, yeast, etc.) used and the required processing time in each step. The brew line, where the first step is performed, is capable of producing six different types of worts using two types of yeast, which, in turn, diversify to 10 different brand beers at the last step by means of standardization and ingredients adjustments (see Table 5.2).

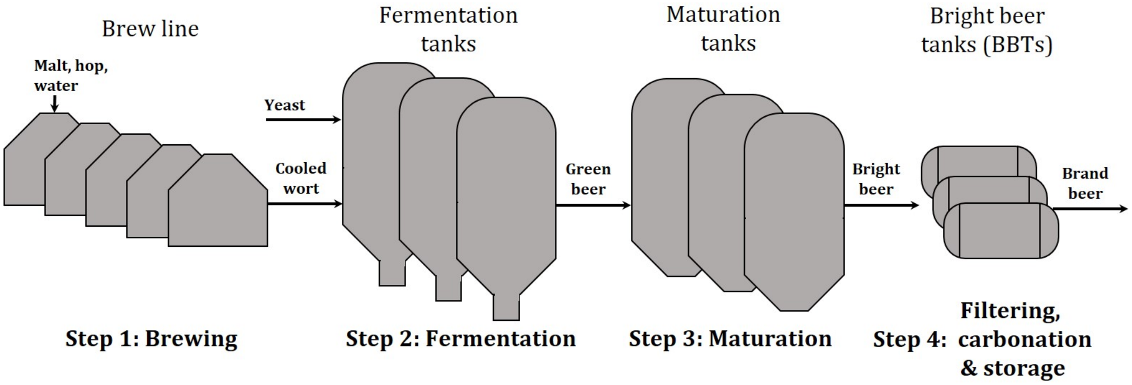


Figure 5.14: An illustration of brewing process in 4 steps.

Table 5.2: Brand beers based on wort and yeast types

Yeast types	Wort types	Brand beer
M	MW1	MB1 MB2 MB3 MB4
	MW2	MB5
C	CW1	CB1
	CW2	CB2 CB3
	CW3	CB4
	CW4	CB5

The first step consists of several batch tasks, namely mashing, lautering, boiling, whirlpooling and cooling. This step converts the aforementioned raw materials to different worts, which, at this stage, are referred to as cooled worts. The facility has one brew line, capable of producing a batch of wort every 2 h. The brew line has to be cleaned once every two days, which takes 2 h. Finally, to maximize recycling of yeast, worts that use the same type of yeast (see Table 5.2) are preferred to be produced consecutively, and the transition between the families of worts usually occurs twice per week.

Next, fermentation starts after filling cooled wort and yeast in a tank, a process which can take from 9 to 11 days depending on the wort type. The facility has eight large tanks that can hold up to six batches of cooled wort from the brew line, and a small tank that can hold up to three batches. The filling time (i.e. the difference between the arrivals of the first and the last batches) should not exceed 24 h. An exception to this policy is when filling CW3, where the second half of the tank should be filled after 24 h from when the first half is filled. After yeast is removed at the end of fermentation, beer of a given wort type is produced, referred to as green beer. Finally, the tank has to be cleaned (requires 4 h) before being used for another fermentation task.

Subsequently, in the maturation step, green beer is filled and matured in a tank, which can take from 5 to 10 days depending on wort type. The facility has a total of nine maturation tanks (four large and five small) which have the same capacity as the fermentation tanks of the respective sizes. After the maturation step is completed, beer of a given wort type is obtained, referred to as bright beer. The maturation tanks also have to be cleaned (4 h) before being used for another maturation task.

In the last step, bright beer is filtered, diluted, carbonated and cooled, and then stored in BBTs. The facility has a total of seven BBTs (six large and one small), where both large and small BBTs have 1/3 capacities of the large and small maturation tanks, respectively. The brand beers stored in BBTs are then quickly bottled, canned or kegged by the packaging lines.

In terms of storage, the facility does not have any vessels for storing intermediate materials, but both green and bright beers can be temporarily stored in fermentation and maturation tanks, respectively. However, to maintain quality, they can only be held for a limited time.

5.7.2 *Process modeling*

In Figure 5.15, an STN of a brand beer production process is shown. The STN of the whole facility can be obtained when this STN is duplicated for all brand beers leading to an STN with 23 states and 22 tasks.

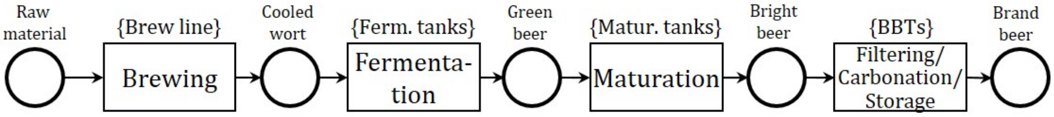


Figure 5.15: An STN of a process of producing a single brand beer.

Each step is modeled as a batch task with fixed processing times. We modify the DCA algorithm to allow for task-material interaction throughout the execution of each task, similar to how it is modeled in the RTN model proposed by Pantelides (1994). For instance, filling of the fermentation tanks with multiple batches of cold wort throughout the execution of the fermentation task is modeled by allowing cold wort to be consumed at predetermined times with respective to the start of the task (see Figure 5.16). Similarly, material transfer operations and smaller intermediate tasks in between steps (e.g. yeast filtering between steps 2 and 3, tank cleaning, etc.) can be taken into account by setting the timings of task-material interactions of each task, appropriately.

To account for temporary material storage in fermentation and maturation tanks, we model them as units that can store output materials, but with additional constraints to limit the duration

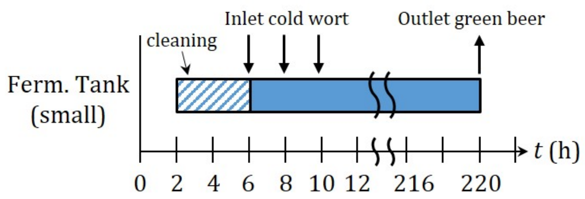


Figure 5.16: An example of task-material interaction of a fermentation task during its execution.

of storage.

Furthermore, to model periodic cleaning of the brew line (i.e. 2 h of cleaning every 2 days), we set $X_{ijn} = 0$ at times when the brew line has to be cleaned. To account for production of wort types in families in the brew line, we introduce changeovers with zero changeover time when switching between tasks that produce wort types that use different yeasts, and limit the number of switches to two within any given week.

Finally, we adopt a discretization step length of 6 h in the first stage model. This is because the brew line can produce up to three batches of cold wort in 6 h, and more importantly, three batches of cold wort is the minimum capacity of the fermentation and maturation tanks, while six is the maximum. Refer to Supplementary Information of Lee and Maravelias (2019b) for a tailored DCA for this case study.

5.7.3 *Results*

The algorithm is executed on a desktop with Intel i7-4770 processors at 3.40 GHz and 8GB of RAM running on Windows 8 and the models are solved using CPLEX 12.8.0 via GAMS 26.1.0. A resource limit of 1h and allowed optimality gap of 1% are used.

To simulate a monthly scheduling scenario, we assume that there exist several batches of fermentation and maturation tasks that are carried over from the previous scheduling horizon. The objective is to maximize production while meeting a given production target, and the horizon length considered is 4 weeks. We allow backlogs. The solution obtained by DCA is shown in Figure 5.17, and weekly production amount, total production target and excess production amount of each brand beer is given in Table 5.3.

In terms of accommodation of process characteristics, it can be seen that the cleaning operations are carried out as required (e.g. 4 h of cleaning of fermentation and maturation tanks before

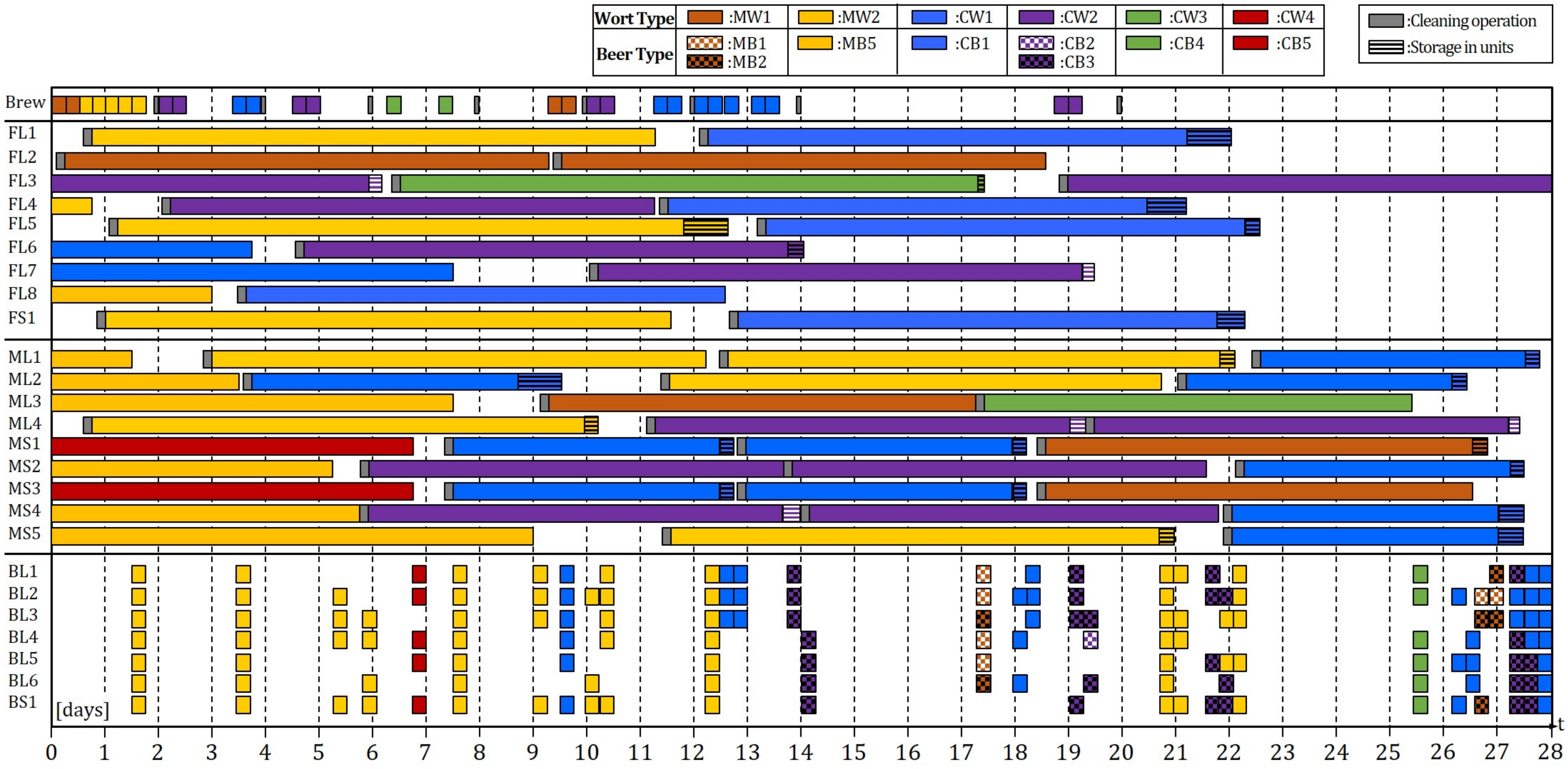


Figure 5.17: Gantt chart of the schedule obtained for Heineken case study. Brew, FL/FS, ML/MS and BL/BS represent the brew line, large/small fermentation tank, large/small maturation tank and large/small BBT, respectively.

Table 5.3: Weekly production amounts [hL]

Brand beer	Week 1	Week 2	Week 3	Week 4	Total	Target	Excess
MB1			8,800	4,400	13,200	9,054	4,146
MB2			3,322	7,696	11,017	10,080	937
MB5	42,600	49,700	14,200	21,300	127,800	105,810	21,990
CB1		22,152	11,076	38,766	71,994	56,170	15,824
CB2			37		37	0	37
CB3		6,600	17,641	24,282	48,523	37,252	11,271
CB4				12,070	12,070	10,225	1,845
CB5	9,869				9,869	9,013	856
Total	52,469	78,452	55,076	108,514	294,511	237,604	56,907

usage). Also, many batches of fermentation and maturation tasks store their output materials in the tanks temporarily until units in the subsequent steps become available. Also, all the filling policies were met in the schedule, including the 24 h of delay in filling CW3 into the tanks.

The obtained schedule satisfies the production targets of all brand beers. It produces a total of 294511 hL of brand beer, which is 24% more than the total production target. However, we note that the total production amount partly depends on the state of the brewery at the beginning of the horizon. Also, note that, due to the long processing times, the brand beers produced in the first and second week of the horizon mostly originate from the batches of maturation and fermentation tasks that have been carried over from the previous horizon, respectively.

Table 5.4: Computational statistics of Heineken case study

δ used	6 h
Constraints	80667
Discrete Variables	32697
Continuous Variables	50467
LP relaxation (hL)	296741.5
Objective (hL)	294510.6
CPU (s)	1136.4
Nodes	1720
Gap (%)	0.7

Finally, in terms of computational performance (see Table 5.4 for statistics), DCA found a high quality solution for this case study in less than 20 minutes. In comparison, if a standalone discrete-time model were to be used for this case study, δ smaller than 1 h would be necessary to obtain a relatively accurate schedule, which will lead to an intractable model. Similarly, a standalone continuous-time model would require more than 100 time slots to represent the obtained schedule, not to mention the additional binary variables that are required to model some of the characteristics (e.g. filling policy), thereby leading to an intractable model.

5.8 CONCLUSIONS

Industrial application of optimization-based scheduling techniques has attracted increasing attention, due to the advances made in the field as well as the increase in computational capabilities. Unfortunately, however, modeling of common process characteristics in different industrial sectors, such as sequence-dependent changeovers, various storage policies, etc., make the scheduling models significantly more difficult to solve.

Building upon DCA (Lee and Maravelias, 2018), we developed general discrete-continuous algorithm (DCA) that enables fast and accurate solution of industrial-scale instances while accommodating various process characteristics efficiently. The advantage that DCA brings is that these characteristics are modeled with less accurate but computationally less expensive constraints without the loss of solution accuracy. The process characteristics considered in the algorithm are (i) sequence-dependent changeovers and setups, (ii) multipurpose storage vessels, (iii) storage of output materials in units, (iv) material handling restrictions, and (v) continuous processes. Furthermore, we showed that the algorithm can be readily extended to accommodate other process characteristics as well.

We applied our algorithm to a case study inspired by a real-world brewing process and showed that, within a reasonable time, our method is capable of finding a high quality schedule while considering the various process characteristics accurately. As demonstrated, the proposed algorithm opens up the possibility of applying optimization-based scheduling techniques in industrial settings, which was thought to be computationally near impossible if a standalone discrete- or continuous-time models were to be used.

Finally, future research direction includes the development of a scheduling tool to facilitate the development of new method and to make the existing methods more widely available.

5.9 NOMENCLATURE

Indices/sets

$i \in \mathbf{I}$	Tasks
$j \in \mathbf{J}$	Units or vessels
$k \in \mathbf{K}$	Materials
$m \in \mathbf{M}$	Unit or material grid points
$n \in \mathbf{N}$	Time points or periods
$p \in \mathbf{P}$	Incident points

Subsets

$\mathbf{I}_j$	Tasks that can be performed in unit j
$\mathbf{I}_k^+/\mathbf{I}_k^-$	Tasks that produce/consume material k
$\mathbf{J}^{CO}$	Units with changeovers
$\mathbf{J}^{CP}$	Continuous processing units
$\mathbf{J}^{H}$	Units that can store output materials
$\mathbf{J}^{S}$	Storage vessels
$\mathbf{J}_i$	Units that can perform task i
$\mathbf{J}_k^+/\mathbf{J}_k^-$	Units that can produce/consume material k
$\mathbf{J}_k^S$	Vessels that can store material k
$\mathbf{K}^{LS}$	Materials with limited storage
$\mathbf{K}^{NM/NS}$	Materials with no-mixing/no-splitting restriction
$\mathbf{K}_j^S$	Materials that can be stored in vessel j
$\mathbf{M}_k^P/\mathbf{M}_k^C$	Grid points at which material k is produced/consumed
$\mathbf{M}_k^{P-}/\mathbf{M}_k^{C-}$	Grid points at which material k is produced/consumed but Eq. 5.52 is not enforced
$\mathbf{M}_k^H$	Grid points at which material k is held in a unit after being produced

Parameters

$\alpha^F_{ij}/\alpha^V_{ij}$	Fixed/variable cost of performing task i in unit j
$\hat{\alpha}_{ij}$	Cost of processing task i in unit $j \in \mathbf{J}^{CP}$ per unit production rate per hour
$\beta^{min}_j/\beta^{max}_j$	Minimum/maximum capacity of unit j
$\hat{\beta}^{min}_j/\hat{\beta}^{max}_j$	Minimum/maximum processing rate of unit $j \in \mathbf{J}^{CP}$
γ_k	Initial inventory of material k
η	Horizon length
ξ_{kp}/ϕ_{kp}	Amount/time p^{th} delivery (<0) or demand (>0) of material k
π_k	Price of material k
ρ_{ik}	Mass fraction of material k produced (>0) or consumed (<0) by task i
$\sigma_{ii'j}/\sigma^C_{ii'j}$	Changeover time/cost from task i to task i' in unit j
τ^F_{ij}/τ^V_{ij}	Fixed/variable processing time of task i performed in unit j
ψ_i	Minimum duration of continuous processing task i in hours

Parameters for discrete-time grid

δ	Step size for discretization of time
$\bar{\eta}$	Horizon length (i.e. $\bar{\eta} = \lfloor \eta/\delta \rfloor$)
$\bar{\xi}_{kn}$	Delivery (<0) or demand (>0) amount of material k at time point n (i.e. $\bar{\xi}_{kn} = \xi_{kp}\ \ \forall(n,p) : n = \lceil t^{\xi}_{kp}/\delta \rceil$ for delivery and $n = \lfloor t^{\xi}_{kp}/\delta \rfloor$ for demand)
$\bar{\sigma}_{ii'j}$	Changeover time from task i to i' in unit j
$\tilde{\sigma}_{ij}/\tilde{\sigma}^C_{ij}$	Approximated changeover time/cost from any task to task i in unit j
$\bar{\tau}_{ij}$	Estimated processing time of task i performed in unit j (i.e. $\bar{\tau}_{ij} = \lceil (\tau^F_{ij} + \beta^{max}_j \tau^V_{ij})/\delta \rceil$)
$\bar{\psi}_i$	Minimum duration of continuous processing task i in time periods

Algorithmic parameters

M^U_j	Number of tasks performed in unit j
M^M_k	Number of tasks that produce/consume material k performed
$f^1_{ikm} \in \{0,1\}$	=1 if an event associated with a batch of task i is mapped to material grid point (k,m)

$f^2_{jkm} \in \{0,1\}$ =1 if an event associated with a batch in unit j is mapped to material grid point (k,m)

$f^3_{kmm'} \in \{0,1\}$ =1 if the timing of material grid point (k,m) at which material k is produced is before $(k,m^{'})$ at which it is consumed

$f^4_{ijkmm'} \in \{0,1\}$ =1 if an event associated with a batch of task i is mapped to unit grid point (j,m) and material grid point $(k,m^{'})$

$f^5_{kmp} \in \{0,1\}$ =1 if p^{th} delivery or demand is at material grid point (k,m)

$g^1_{jkmm'} \in \{0,1\}$ =1 if the production event due to discharge of material k from unit j is mapped to unit grid point (j,m) and material grid point $(k,m^{'})$

$g^2_{jkm} \in \{0,1\}$ =1 if material k is stored in vessel j between material grid pont (k,m) and $(k,m+1)$

$g^3_{jkmm'} \in \{0,1\}$ =1 if storage event of material k in vessel j is mapped to unit grid point (j,m) and material grid point $(k,m^{'})$

$g^4_{kmm'} \in \{0,1\}$ =1 if material grid points (k,m) and $(k,m^{'})$ have the same timing

$g^5_{kk'mm'} \in \{0,1\}$ =1 if material grid points (k,m) and $(k^{'},m^{'})$ have the same timing

$g^6_{jj'kmm'} \in \{0,1\}$ =1 if material k is transferred between the events mapped to unit grid points (j,m) and $(j^{'},m^{'})$

$g^7_{ijm} \in \{0,1\}$ =1 if the start of a continuous processing task i is mapped to unit grid point (j,m)

$g^8_{ii'jm} \in \{0,1\}$ =1 if a batch of task i of which events are mapped to unit grid point (j,m) is followed by a batch of task $i^{'}$

Binary variable

R_{ijn} =1 if unit j is in mode i at time point n

R^{N}_{jn} =1 if unit j is in neutral mode at time point n

X_{ijn} =1 if task i begins in unit j at time point n

X^{H}_{ijn} =1 if the output materials of task i is stored in unit j during time period n

X^{S}_{jkn} =1 if material k is stored in vessel j during time period n

X^{C}_{ijn}/X^{N}_{ijn} =1 if mode of unit j is shifted to/from mode i from/to neutral

Y_{ijn} =1 if continuous processing task i starts in unit j at time point n

$W_{jj'kn}$ =1 if material k is transferred from unit or vessel j to $j^{'}$ at time point n

Non-negative continuous variables

B_{ijn} Batch size of task i in unit j starting at time point n

B^{C}_{km} Batch size of the production or consumption event mapped to material point (k, m)

$F_{jj'kn}$ Amount of material transferred from unit or vessel j to $j^{'}$ at time point n

$F^{C}_{jj'kmm'}$ Amount of material transferred from unit grid point (j, m) to $(j^{'}, m^{'})$

F^{H}_{jkn} Amount of material k discharged from unit j at time point n

MS Makespan

S_{kn} Inventory of material k during time period n

S^{S}_{jkn} Amount of material k stored in unit or vessel j during time period n

S^{C}_{km} Inventory level of material k at material grid point (k, m)

$S^{C,S}_{km}$ Amount of material k stored in vessel j at material grid point (k, m)

T^{M}_{km} Timing of material grid point (k, m)

T^{U}_{jm} Timing of unit grid point (j, m)

6

SYSTEMATIC GENERATION OF ALTERNATIVE SCHEDULES

6.1 MOTIVATION

Over the last decades, production scheduling has been the subject of active research in the process systems engineering (PSE) community, resulting in significant advances in several directions, e.g. modeling, solution methods, and online (re)scheduling.

Despite the fact that the advances in solution methods, combined with increased computing power, allow for faster solution of industrial scale instances, other critical challenges remain when it comes to industrial applications (Henning, 2009; Harjunkoski, 2016). First, an important and fundamental challenge is the limited scope of the models, i.e. the introduction of modeling simplifications to reduce complexity. Second, as schedules are revised, it is important to find schedules with low *shop-floor nervousness* (hereinafter referred to as nervousness), i.e. the negative impact on the shop floor due to major and/or frequent changes in the schedule. This is typically not considered in the current scheduling models.

From a practical standpoint, one effective and common method to address the two challenges is to obtain multiple alternative schedules, instead of one, for the production scheduler to compare, select and deploy (Framinan and Ruiz, 2012). Accordingly, the goal of this work is to develop systematic methods for generating alternative schedules.

6.2 BACKGROUND

6.2.1 *Online rescheduling and plant nervousness*

Advances in the development of efficient models and solution methods for chemical production scheduling problems have opened up the possibility of obtaining multiple alternative schedules within reasonable amount of time, even for industrial scale instances.

Another field that has made significant advances is rescheduling, which, in turn, gave rise to the importance of considering nervousness in scheduling problems. Earlier contributions in rescheduling include methods based on heuristics and repair algorithms (Sanmarti et al., 1996; Mendez and Cerda, 2003; Janak et al., 2006; Novas and Henning, 2010, 2012). More recently, state-space models for online (re)scheduling have been proposed and generalized to account for various uncertainties (Subramanian et al., 2012; Nie et al., 2014; Gupta and Maravelias, 2017; Risbeck et al., 2019). Finally, the quality of the implemented (closed-loop) schedule and strategies to improve it have been investigated (Zhuge and Ierapetritou, 2012; Gupta et al., 2016; Gupta and Maravelias, 2016).

6.2.2 *Generation of alternative schedules*

While most studies on production scheduling within the PSE community have focused on generating one, ideally optimal, schedule, there are significant advantages in generating multiple schedules. First, the issues that arise from modeling simplifications can be mitigated. For example, it is common that the scheduling models neglect some details such as material transfer times, which may lead to infeasible or suboptimal schedule when deployed. Such cases can be avoided by allowing the production schedulers to choose from alternative schedules while considering the neglected details. Second, when revising schedules, generation of alternative schedules with different degrees of nervousness allows for the schedulers to selectively deploy the schedule that best suits their current situation, by carefully weighing the benefit of the revised schedule and the potential increase in nervousness.

An obvious method to generate multiple solutions from a MIP model is to solve the model iteratively while adding integer cuts in each iteration to exclude previously obtained solutions.

This method, however, is highly ineffective, as it typically results in multiple very similar schedules (e.g. essentially identical schedules with slightly different start time of one or two batches) rather than schedules that are substantially different in more essential aspects, e.g., the number and sequencing of batches.

6.2.3 *Problem statement*

We study the problem of short-term production scheduling in network environments. Given resource availability, production recipe and demand information, the goal is to obtain schedules that satisfy the demand while optimizing a given objective.

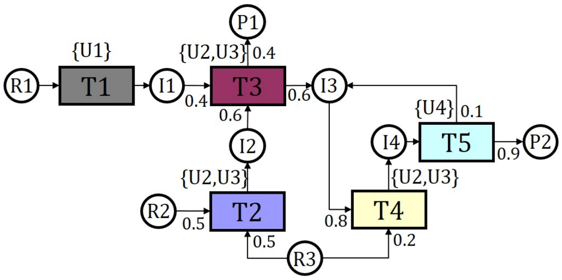


Figure 6.1: STN representation of a batch process (Kondili et al., 1993).

As the STN representation can be used to address problems in general production environments (Sundaramoorthy and Maravelias, 2011a), the methods we develop can be easily extended to address, potentially, all production scheduling problems. In an STN, circle nodes represent materials and rectangular nodes represent tasks that convert one or more input materials to one or more output materials (see Figure 6.1). The following indices/sets are introduced: $i \in \mathbf{I}$ to denote tasks, $j \in \mathbf{J}$ to denote units, and $k \in \mathbf{K}$ to denote materials. We also define the following compatibility subsets: $\mathbf{I}_j$ to denote tasks that can be processed in unit j, $\mathbf{I}^+_k/\mathbf{I}^-_k$ to denote tasks that produce/consume material k, and $\mathbf{J}_i$ to denote units that can process task i. We represent fixed/variable processing times of task i performed in unit j by $\bar{\tau}^F_{ij}/\bar{\tau}^V_{ij}$. The fixed/variable processing costs of task i performed in unit j are represented by $\alpha^F_{ij}/\alpha^V_{ij}$. The conversion coefficient of material k produced (>0) or consumed (<0) by task i is given by ρ_{ik}. The units have maximum/minimum capacities represented by $\beta^{max}_j/\beta^{min}_j$. The initial inventory of material k and its demand amount at time t are represented by γ_k and ξ_{kt}, respectively. The price of material k is given by π_k.

We use the discrete-time STN model (Shah et al., 1993) as the base scheduling model, though the proposed methods (with small modifications) are applicable to any discrete-time models. The horizon, $\bar{H}$, is discretized into intervals of length δ. The resulting set of time points is denoted by $\mathbf{T} = \{0, 1, ..., H\}$, where $H = \lfloor \bar{H}/\delta \rfloor$. We define time period t as the time interval between points $t-1$ and t. The processing times are rounded up to multiples of δ: $\tau_{ij} = \lceil (\bar{\tau}_{ij}^F + \beta_j^{max} \bar{\tau}_{ij}^V)/\delta \rceil$. The model consists of the following constraints:

$$\sum_{i \in \mathbf{I}_j} \sum_{t' \geq t - \tau_{ij} + 1}^{t} X_{ijt'} \leq 1 \;\; \forall\, j, t \tag{6.1}$$

$$\beta_j^{min} X_{ijt} \leq B_{ijt} \leq \beta_j^{max} X_{ijt} \;\; \forall\, i, j \in \mathbf{J}_i, t \tag{6.2}$$

$$S_{k(t+1)} = S_{kt} + \sum_{i \in \mathbf{I}_k^+} \rho_{ik} \sum_{j \in \mathbf{J}_i} B_{ij(t-\tau_{ij})} + \sum_{i \in \mathbf{I}_k^-} \rho_{ik} \sum_{j \in \mathbf{J}_i} B_{ijt} - \xi_{kt} \;\; \forall\, k, t \tag{6.3}$$

where $X_{ijt} \in \{0, 1\}$ is 1 if task i starts in unit j at time point t, $B_{ijt} \in \mathbb{R}_+$ denotes the batch size of task i executed in unit j at time point t, and $S_{kt} \in \mathbb{R}_+$ denotes the inventory level of material k during time period t. The inventory level of material k at the beginning of the horizon is set as its initial inventory level (i.e. $S_{k(t=0)} = \gamma_k \; \forall k$).

The objective functions for profit maximization, cost minimization and makespan minimization are as follows,

$$max \; \sum_k \pi_k (S_{k(H+1)} + \sum_t \xi_{kt}) - \sum_i \sum_{j \in \mathbf{J}_i} \sum_t (\alpha_{ij}^F X_{ijt} + \alpha_{ij}^V B_{ijt}) \tag{6.4}$$

$$min \; \sum_i \sum_{j \in \mathbf{J}_i} \sum_t (\alpha_{ij}^F X_{ijt} + \alpha_{ij}^V B_{ijt}) \tag{6.5}$$

$$min \; MS : \;\; MS \geq \delta \sum_{i \in \mathbf{I}_j} (t + \tau_{ij}) X_{ijt} \;\; \forall j, t \tag{6.6}$$

where $MS \in \mathbb{R}_+$ represents the makespan. The scheduling model, hereinafter referred to as the base model, consists of Eqs. 6.1-6.3 and an objective function.

6.3 METHODS TO ADDRESS MODELING SIMPLIFICATIONS

While finding multiple alternative schedules can effectively address the introduced modeling simplifications, not all alternative schedules are useful. For example, schedules with the same number of batches having identical unit assignments and sequencing, but with different batch start times are technically different schedules in terms of decision variables, but from a practical standpoint, they are not providing alternative options. Hence, we focus on generating alternative schedules that are different in three major characteristics: (1) the number of batches assigned to each unit, (2) the sequencing of batches within units, and (3) the presence of idle time in between batches.

First, we develop constraints that enable us to quantify the aforementioned characteristics by introducing appropriate metrics. Next, we favor such characteristics by penalizing the objective function using these metrics. We illustrate the effectiveness of the proposed approach through an instance considering the network shown in Figure 6.1, referred to as instance 1. Other instance data, including processing times (and costs), initial inventory and demand, can be found in Lee et al. (2019).

6.3.1 *Number of batches assigned to each unit*

We introduce $N_{ij} \in \mathbb{Z}_+$ to denote the number of batches of task i assigned to unit j,

$$N_{ij} = \sum_t X_{ijt} \quad \forall i, j \in \mathbf{J}_i \tag{6.7}$$

where N_{ij} can be modeled as a non-negative continuous variable, instead of an integer variable, since it is a sum of binary variables.

Prior to the optimization, the expected number of batches of task i assigned to unit j, denoted by N_{ij}^E, is obtained by solving the base model with the integrality constraints relaxed.

$$N_{ij}^E = \sum_t X_{ijt}^* \quad \forall i, j \in \mathbf{J}_i \tag{6.8}$$

where X_{ijt}^* is the value of X_{ijt} variable in the solution.

Next, we introduce $\hat{N}_{ij} \in \mathbb{R}_+$ to denote the difference between the expected and the actual number of batches of task i assigned to unit j (i.e. $\hat{N}_{ij} = \left|N_{ij} - N^E_{ij}\right|$). We model $\hat{N}_{ij}$ linearly through Eqs. 6.9-6.12, employing a binary variable, W^N_{ij}, which is 1 if N_{ij} is greater than or equal to N^E_{ij}, otherwise 0.

$$N_{ij} - N^E_{ij} = \hat{N}^+_{ij} - \hat{N}^-_{ij} \quad \forall j \in \mathbf{J}^N, i \in \mathbf{I}_j \tag{6.9}$$

$$\hat{N}_{ij} = \hat{N}^+_{ij} + \hat{N}^-_{ij} \quad \forall j \in \mathbf{J}^N, i \in \mathbf{I}_j \tag{6.10}$$

$$\hat{N}^+_{ij} \le M \cdot W^N_{ij} \quad \forall j \in \mathbf{J}^N, i \in \mathbf{I}_j \tag{6.11}$$

$$\hat{N}^-_{ij} \le M(1 - W^N_{ij}) \quad \forall j \in \mathbf{J}^N, i \in \mathbf{I}_j \tag{6.12}$$

where $\hat{N}^+_{ij} \in \mathbb{R}_+$ and $\hat{N}^-_{ij} \in \mathbb{R}_+$ are auxiliary variables, M is a sufficiently large number (e.g. $M_{ij} = H/\tau_{ij}\ \forall i, j$), and $\mathbf{J}^N$ is the set of units in which the number of batches of tasks assigned is quantified.

We define metric P^N as follows, since we are incentivizing $\hat{N}_{ij} > 0$.

$$P^N = -\sum_{j \in \mathbf{J}^N} \sum_{i \in \mathbf{I}_j} \hat{N}_{ij} \tag{6.13}$$

The objective functions (Eqs. 6.4-6.6) are modified to include a penalty term, $\omega^N P^N$, where ω^N is the penalty weight. For example, the modified objective function for profit maximization becomes,

$$max \sum_k \pi_k (S_{k(\bar{H}+1)} + \sum_t \xi_{kt}) - \sum_i \sum_{j \in \mathbf{J}_i} \sum_n (\alpha^F_{ij} X_{ijn} + \alpha^V_{ij} B_{ijn}) - \omega^N \cdot P^N \tag{6.14}$$

The model consists of Eqs. 6.1-6.3, 6.7, 6.9-6.12, and a modified objective function.

Figure 6.2 shows the multiple alternative schedules obtained for instance 1 with $\omega^N \in \{0, 5, 10, 15\}$. The profits shown in the figure do not include the contribution from $\omega^N P^N$. It can be observed that different ω^N's lead to schedules with different number of batches assigned to each unit. For instance, while the schedule obtained with $\omega^N = 0$ (Figure 6.2(a)) has two batches of T4 (light yellow) assigned to U2 and five assigned to U3, the schedule obtained with $\omega^N = 15$ (Figure 6.2(d)) has five assigned to U2 and none assigned to U3.

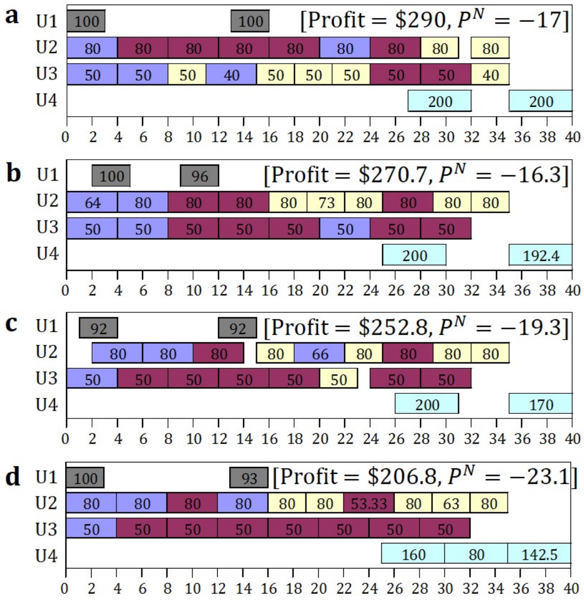


Figure 6.2: Schedules obtained for instance 1 with different ω^N's: (a) $\omega^N = 0$, (b) $\omega^N = 5$, (c) $\omega^N = 10$, and (d) $\omega^N = 15$. The numbers within colored blocks represent batch sizes [kg].

6.3.2 *Sequencing of batches*

The sequencing of batches within units is modeled by introducing the concept of unit *modes*, i.e. the status of unit that determines the task the unit is ready to process. We introduce $R_{ijt} \in \{0,1\}$, which is 1 if unit j is in mode for processing task i at time point t, and $R^S_{jt} \in \{0,1\}$, which is 1 if unit j is in neutral mode at time point t. For a unit to transition from mode i to mode $i^{'}$, it has to transition to neutral mode, and then to $i^{'}$. The transition between modes are modeled through the mode resource balance constraints as follows:

$$R_{ijt} = R_{ij(t-1)} + X_{ij(t-\tau_{ij})} - X_{ijt} + W^S_{ijt} - W^T_{ijt} \quad \forall j \in \mathbf{J}^S, i \in \mathbf{I}_j, t \tag{6.15}$$

$$R^S_{jt} = R^S_{j(t-1)} + \sum_{i \in \mathbf{I}_j} W^T_{ijt} - \sum_{i \in \mathbf{I}_j} W^S_{ijt} \quad \forall j \in \mathbf{J}^S, t \tag{6.16}$$

$$\sum_{i \in \mathbf{I}_j} R_{ij(t=0)} = 1 \quad \forall j \in \mathbf{J}^S \tag{6.17}$$

where $W^S_{ijt} \in \{0,1\}$ is 1 if unit j transitions from the neutral mode to task i mode at time point t; $W^T_{ijt} \in \{0,1\}$ is 1 if unit j transitions from task i mode to neutral mode at time point t; and $\mathbf{J}^S$ denotes the set of units in which the number of mode transitions is quantified.

Based on the W^S_{ijt} binaries, the number of mode transitions from neutral to any task is quantified and penalized in the objective function, using a metric, P^S:

$$P^S = \sum_{j \in \mathbf{J}^S} \sum_{i \in \mathbf{I}_j} \sum_{t} W^S_{ijt} \tag{6.18}$$

The objective function of choice is then modified to include the penalty term, $\omega^S P^S$, where ω^S is the penalty weight. The model consists of Eqs. 6.1-6.3, 6.15-6.17 and a modified objective function.

Figure 6.3 shows the schedules obtained for instance 1 with $\omega^S \in \{0, 1, 7, 30\}$. The schedules shown in Figures 6.3(a) and 6.3(b) are very similar, except for the sequencing of the 3rd and 4th batches assigned to unit U3. When larger ω^S's are used, the differences in the schedule become more apparent: the schedules obtained with larger ω^S's have fewer transition between unit modes, and as a result, exhibit different batch sequencing, although some of them achieve this with decreased profit.

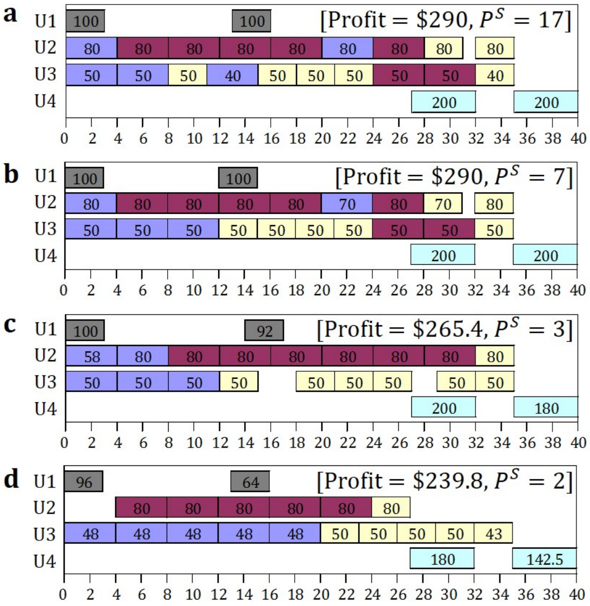


Figure 6.3: Schedules obtained for instance 1 with different ω^S's: (a) $\omega^S = 0$, (b) $\omega^S = 1$, (c) $\omega^S = 7$, and (d) $\omega^S = 30$. The numbers within colored blocks represent batch sizes [kg].

6.3.3 *Idle time in between batches*

We introduce a parameter θ_{ij} to denote the amount of idle time the production scheduler desires after a batch of task i in unit j (in multiples of δ), and calculate the modified processing time, τ^I_{ij}, as follows.

$$\tau^I_{ij} = \tau_{ij} + \theta_{ij} \;\; \forall i, j \in \mathbf{J}_i \tag{6.19}$$

Based on τ^I_{ij}, we enforce a task-unit assignment constraint (Eq. 6.20), in addition to the one already in the base model (Eq. 6.1) with a continuous slack variable, $W^I_{jt} \in [0, 1]$, to allow constraint violation,

$$\sum_{i \in \mathbf{I}_{ij}} \sum_{t' \geq t - \tau^I_{ij} + 1}^{t} X_{ijt'} \leq 1 + W^I_{jt} \;\; \forall j \in \mathbf{J}^I, t \tag{6.20}$$

where $\mathbf{J}^I$ denotes the set of units in which idle times are desired by the production schedulers.

The proposed constraint, when satisfied with zero slack, ensures idle time of θ_{ij} after completion of task i in unit j. Therefore, the slack variable can be used to quantify the existence of idle time. We introduce metric P^I,

$$P^I = \sum_{j \in \mathbf{J}^I} \sum_t W^I_{jt} \tag{6.21}$$

The objective function of choice is then modified to include the penalty term, $\omega^I P^I$, where ω^I is the penalty weight. The model consists of Eqs. 6.1-6.3, 6.20, and one of the modified objective functions.

Figure 6.4 shows the schedules obtained by solving instance 1 with $\omega^I \in \{0, 3, 10, 12\}$ and $\theta_{ij} = 1$ h $\forall i, j$. Note that larger ω^I's lead to schedules with more idle time between batches with a trade-off of decreased profit.

Finally, we note that the characteristics of a schedule discussed in this section can be considered simultaneously. The full model, referred to as M1, consists of Eqs. 6.1-6.3, 6.7-6.12, 6.15-6.17, 6.20 and a modified objective function that includes all penalty terms.

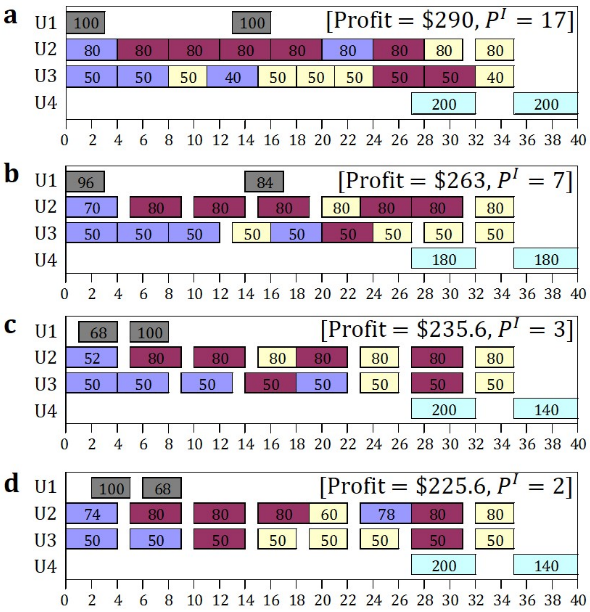


Figure 6.4: Schedules obtained for instance 1 with different ω^I's: (a) $\omega^I = 0$, (b) $\omega^I = 3$, (c) $\omega^I = 10$, and (d) $\omega^I = 12$. The numbers within colored blocks represent batch sizes [kg].

6.4 Methods to address shop-floor nervousness

When schedules are revised, it is important for the production schedulers to account for nervousness. Accordingly, we propose constraints that allow us to quantify how different the revised schedule is from the current schedule, considering addition, removal, reassignment of batches and changes in start time and batch sizes.

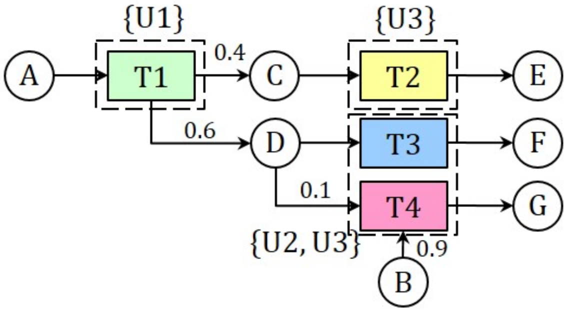


Figure 6.5: STN representation of network 2.

The network shown in Figure 6.5 is considered for demonstration. The schedules are computed in a rolling horizon fashion with a horizon of 48 h and re-optimization performed every 12 h. Each optimization is referred to as an open-loop iteration. Here, we focus on a particular open-loop iteration, which solves an instance (referred to as instance 2). The process data and demand information can be found in Lee et al. (2019). We assume that the schedule from the previous open-loop iteration, referred to as the base schedule, is known, which is used for comparison. The schedule in the current open-loop iteration is referred to as the revised schedule (see Figure 6.6).

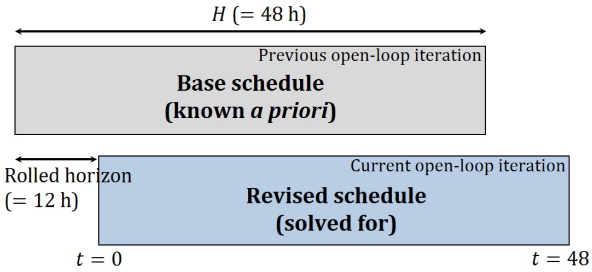


Figure 6.6: Illustration of the rolling horizon approach used in instance 2.

Naturally, changes in schedule that occur at the beginning of the horizon have higher impact in terms of nervousness than the ones occurring at the end of the horizon. To reflect this, we divide the horizon into three intervals, in which the schedule (i) is fixed to be the base schedule (i.e. no change allowed), (ii) can be revised subject to penalty, and (iii) can be revised without penalty. Accordingly, we introduce subsets of time points that correspond to each interval, namely $\mathbf{T}^1$, $\mathbf{T}^2$ and $\mathbf{T}^3$ (see Figure 6.7(a) for illustration). The length of each interval can be set by the user. Note that the sets are defined for the model for the revised schedule (e.g. $\mathbf{T} = \mathbf{T}^1 \cup \mathbf{T}^2 \cup \mathbf{T}^3$).

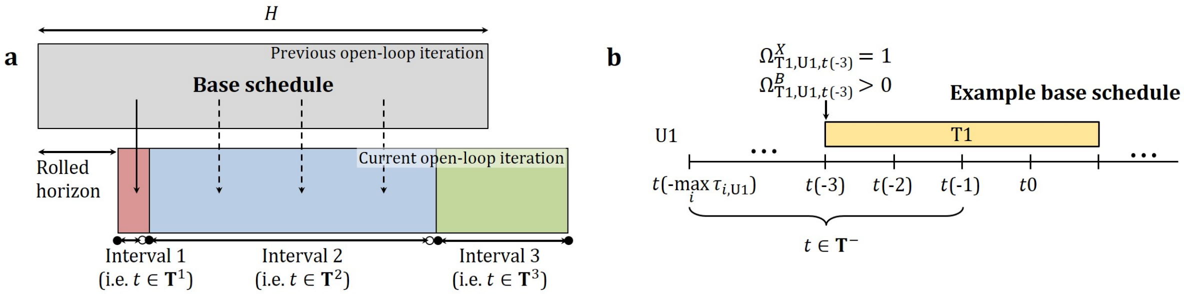

a
H
Previous open-loop iteration
Base schedule
Rolled horizon
Current open-loop iteration
Interval 1
(i.e. $t \in \mathbf{T}^1$)
Interval 2
(i.e. $t \in \mathbf{T}^2$)
Interval 3
(i.e. $t \in \mathbf{T}^3$)
b
$\Omega^X_{\text{T1,U1},t(-3)} = 1$
$\Omega^B_{\text{T1,U1},t(-3)} > 0$
Example base schedule
U1
T1
...
$t(-\max_i \tau_{i,\text{U1}})$
$t(-3)$
$t(-2)$
$t(-1)$
$t0$
$t \in \mathbf{T}^-$

Figure 6.7: Illustration of concepts: (a) the divided horizon and the three intervals, and (b) the set of negative time points.

The base schedule at time points before $t = 0$ is modeled via set $\mathbf{T}^- = \{-\max_{i,j}(\tau_{ij}), \ldots -2, -1\}$ (see Figure 6.7(b) for an illustration). Accordingly, we define a superset, $\mathbf{T}^A$ ($:= \mathbf{T}^- \cup \mathbf{T}$), for the set of all time points. The following parameters are introduced to represent the base schedule: $\Omega^X_{ijt} \in \{0,1\}$, equal to 1 if a batch of task i is scheduled in unit j at time point t in the base schedule; $\Omega^B_{ijt} \in \mathbb{R}_+$, the batch size of task i scheduled in unit j at time point t in the base schedule; $\Omega^S_k \in \mathbb{R}_+$, the inventory level of material k during time period $t = 0$ in the base schedule.

6.4.1 *Addition and removal of batches*

We introduce the following binary variables to model addition and removal of batches: Y_{ijt} is equal to 1 if a new batch of task i is scheduled in unit j at time point t; Z_{ijt} is equal to 1 if the batch of task i in unit j at time point t in the base schedule (i.e. $\Omega^X_{ijt} = 1$) is executed in the revised schedule, though not necessarily in unit j and at time point t. In other words, the batch is removed if $Z_{ijt} = 0$ (for $\Omega^X_{ijt} = 1$).

The proposed method involves a few modifications to the base model: (i) the set of time points/periods in which the constraints are defined is modified to $\mathbf{T}^A$ (see Eqs. 6.22-6.24), and (ii) the initial inventory level of material k is now given by Ω^S_k (see Eq. 6.25).

$$\sum_{i \in \mathbf{I}_j} \sum_{t' \geq t - \tau_{ij} + 1}^{t} X_{ijt'} \leq 1 \quad \forall\, j, t \in \mathbf{T}^A \tag{6.22}$$

$$\beta^{min}_j X_{ijt} \leq B_{ijt} \leq \beta^{max}_j X_{ijt} \quad \forall\, i, j \in \mathbf{J}_i, t \in \mathbf{T}^A \tag{6.23}$$

$$S_{k(t+1)} = S_{kt} + \sum_{i \in \mathbf{I}^+_k} \rho_{ik} \sum_{j \in \mathbf{J}_i} B_{ij(t-\tau_{ij})} + \sum_{i \in \mathbf{I}^-_k} \rho_{ik} \sum_{j \in \mathbf{J}_i} B_{ijt} - \xi_{kt} \quad \forall\, k, t \in \mathbf{T}^A \tag{6.24}$$

$$S_{k(t=0)} = \Omega^S_k \quad \forall k \tag{6.25}$$

The following constraints ensure the schedule is identical to the base schedule in interval 1:

$$X_{ijt} = \Omega^X_{ijt} \quad \forall i, j \in \mathbf{J}_i, t \in \mathbf{T}^- \cup \mathbf{T}^1 \tag{6.26}$$

$$B_{ijt} = \Omega^B_{ijt} \quad \forall i, j \in \mathbf{J}_i, t \in \mathbf{T}^- \cup \mathbf{T}^1 \tag{6.27}$$

In intervals 2 and 3, we allow addition and removal of batches through Eqs. 6.28-6.29. The batches already in the base schedule and the new batches activate the binary variables (i.e. X_{ijt}).

$$Z_{ijt}|_{\Omega^X_{ijt}=1} + Y_{ijt} = X_{ijt} \quad \forall i, j \in \mathbf{J}_i, t \in \mathbf{T}^2 \tag{6.28}$$

$$Z_{ijt} \leq \Omega^X_{ijt} \quad \forall i, j \in \mathbf{J}_i, t \in \mathbf{T}^2 \tag{6.29}$$

The metrics for addition and removal of batches, denoted by P^A and P^R, are defined as follows.

$$P^A = \sum_i \sum_{j \in \mathbf{J}_i} \sum_{t \in \mathbf{T}^2} \mu^A_t Y_{ijt} \tag{6.30}$$

$$P^R = \sum_i \sum_{j \in \mathbf{J}_i} \sum_{t \in \mathbf{T}^2 : \Omega^X_{ijt}=1} \mu^R_t (1 - Z_{ijt}) \tag{6.31}$$

where μ^A_t and μ^R_t are the corresponding, time-scaled weights. The selection of appropriate values for these parameters is discussed at the end of the section. The objective functions are modified to include penalty terms $\omega^{AR} P^A$ and $\omega^{AR} P^R$, where ω^{AR} is the penalty weight. The model consists of Eqs. 6.22-6.29, and a modified objective function.

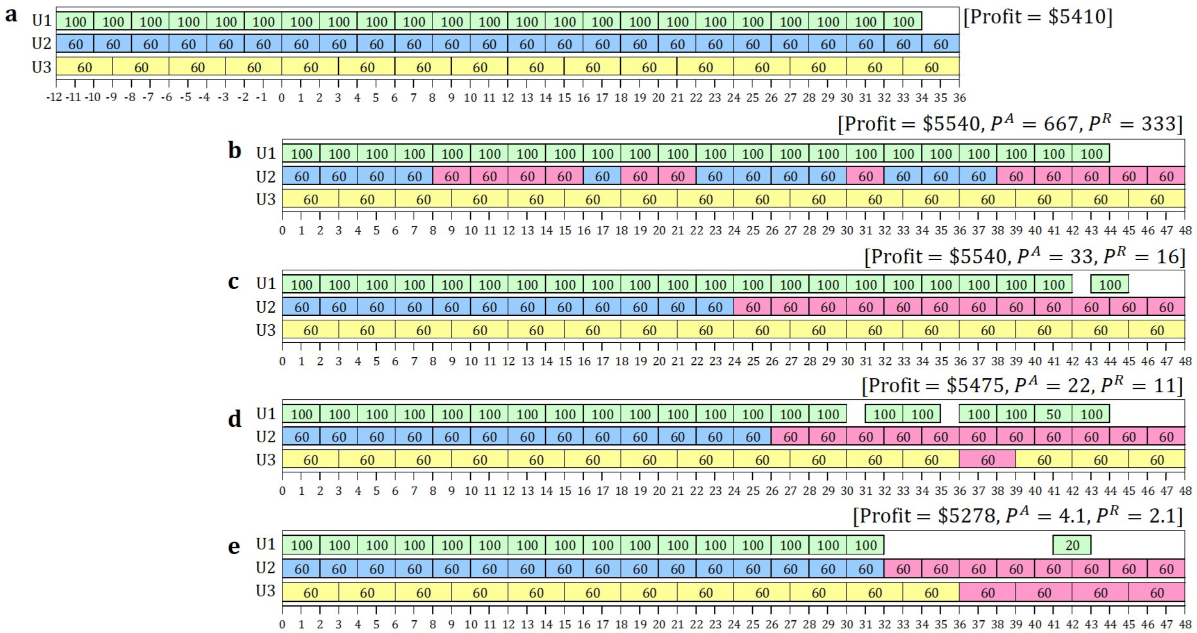


Figure 6.8: Schedules obtained for instance 2 with different ω^{AR}'s: (a) base schedule, (b) $\omega^{AR} = 0$, (c) $\omega^{AR} = 1$, (d) $\omega^{AR} = 5$, and (e) $\omega^{AR} = 15$. The numbers within colored blocks represent batch sizes [kg].

To illustrate, we solve instance 2, maximizing profit. We set interval 1 as 0-5 h, interval 2 as 6-35 h, and interval 3 as 36-48 h. We use μ_t^A and μ_t^R that decrease with time to model the decreased impact of schedule changes on nervousness when they are made in the later part of the horizon. The base schedule is given in Figure 6.8(a), and the schedules obtained using $\omega^{AR} \in \{0, 1, 5, 15\}$ are shown in Figure 6.8(b)-(e). At 48 h in the revised schedule a new demand for material G is observed, leading to addition of batches of T4 (red) and removal of several batches of T3 (blue). When $\omega^{AR} = 0$ (Figure 6.8(b)), the batches are removed and added at any time, and when $\omega^{AR} = 1$ (Figure 6.8(c)), the batches are removed and added as late as possible to reduce nervousness. When larger ω^{AR}'s are used (Figure 6.8(d),(e)), more changes occur in interval 3.

6.4.2 *Reassignment and time shift of batches*

We introduce a binary variable, $A_{ijj'tt'}$, to model reassignment and time shift of batches; $A_{ijj'tt'}$ is equal to 1 if a batch of task i scheduled in unit j at time t in the base schedule (i.e. $\Omega^X_{ijt} = 1$) is reassigned to unit j' at time point t' in the revised schedule.

Eq. 6.32 ensures that if a batch of task i in unit j at time point t in the base schedule is executed in the revised schedule (i.e. $\Omega^X_{ijt} = 1$ and $Z_{ijt} = 1$), it can be executed in any compatible unit (j') and time (t'). Through Eq. 6.33, the binary variable in the revised schedule (i.e. $X_{ij't'}$) is activated. An illustration of the values of Y_{ijt}, Z_{ijt} and $A_{ijj'tt'}$ obtained when a batch is added, removed or reassigned/time shifted is shown in Figure 6.9.

$$\sum_{t' \in \mathbf{T} \setminus \mathbf{T}^1} \sum_{j' \in \mathbf{J}_i} A_{ijj'tt'} = Z_{ijt} \quad \forall i, j \in \mathbf{J}_i, t \in \mathbf{T} \setminus \mathbf{T}^1 : \Omega^X_{ijt} = 1 \tag{6.32}$$

$$\sum_{t \in \mathbf{T} \setminus \mathbf{T}^1} \sum_{j \in \mathbf{J}_i : \Omega^X_{ijt} = 1} A_{ijj'tt'} + Y_{ij't'} = X_{ij't'} \quad \forall i, j' \in \mathbf{J}_i, t' \in \mathbf{T} \setminus \mathbf{T}^1 \tag{6.33}$$

Batch reassignment and time shift are penalized using metrics P^U and P^T, respectively:

$$P^U = \sum_i \sum_{j, j' \in \mathbf{J}_i : j \neq j'} \sum_{t, t' \in \mathbf{T} \setminus \mathbf{T}^1} \mu^U_{tt'} A_{ijj'tt'} \tag{6.34}$$

$$P^T = \sum_i \sum_{j, j' \in \mathbf{J}_i} \sum_{t, t' \in \mathbf{T} \setminus \mathbf{T}^1 : t \neq t'} \mu^T_{tt'} A_{ijj'tt'} \tag{6.35}$$

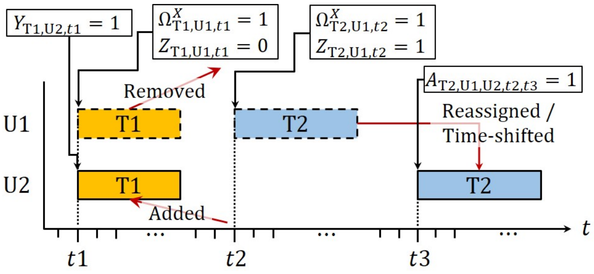


Figure 6.9: Illustration of binary variables, Y_{ijt}, Z_{ijt} and $A_{ijj'tt'}$, when a batch is added, removed or reassigned/time shifted. Batches with dashed outline are the batches scheduled in the base schedule (i.e. $\Omega^X_{ijt} = 1$).

where $\mu^U_{tt'}$ and $\mu^T_{tt'}$ are weights for reassignment and time shift of batches, respectively. The objective function of choice is then modified to include the penalty terms, $\omega^{UT} P^U$ and $\omega^{UT} P^T$, where ω^{UT} is the penalty weights. The model consists of Eqs. 6.22-6.27, 6.29, 6.32-6.33, and a modified objective function.

To illustrate, instance 2 with increased demand of material G (i.e. 720 kg → 1200 kg) at 48 h is considered. The length of the intervals and the base schedule are the same as in the previous subsection. We use $\mu^U_{tt'}$ and $\mu^T_{tt'}$ as given in Lee et al. (2019). In addition to $\omega^{AR} = 1$, $\omega^{UT} \in \{0, 1, 3, 8\}$ are used to generate multiple alternative schedules. In general, due to higher demand of material G observed at 48 h, many batches of T4 (red) are executed instead of batches of T2 (yellow) and T3 (blue). When reassignment and time shift of batches are not penalized (Figure 6.10(b)), three batches of T3 are reassigned to U3, and several batches of T1 (green) are shifted in time freely. However, when $\omega^{UT} = 1$ is used (Figrue 6.10(c)), only two batches of T3 are reassigned, and the batches of T1 do not shift in time. When $\omega^{UT} = 8$ is used (Figrue 6.10(e)), no batch is reassigned nor shifted.

6.4.3 *Change of batch sizes*

Batch size changes in the revised schedule are only quantified for the batches that are reassigned and/or time shifted from the base schedule. We introduce $Q_{ijt} \in \mathbb{R}_+$ to denote the absolute difference between the batch sizes of the batch of task i in unit j at time point t in the revised

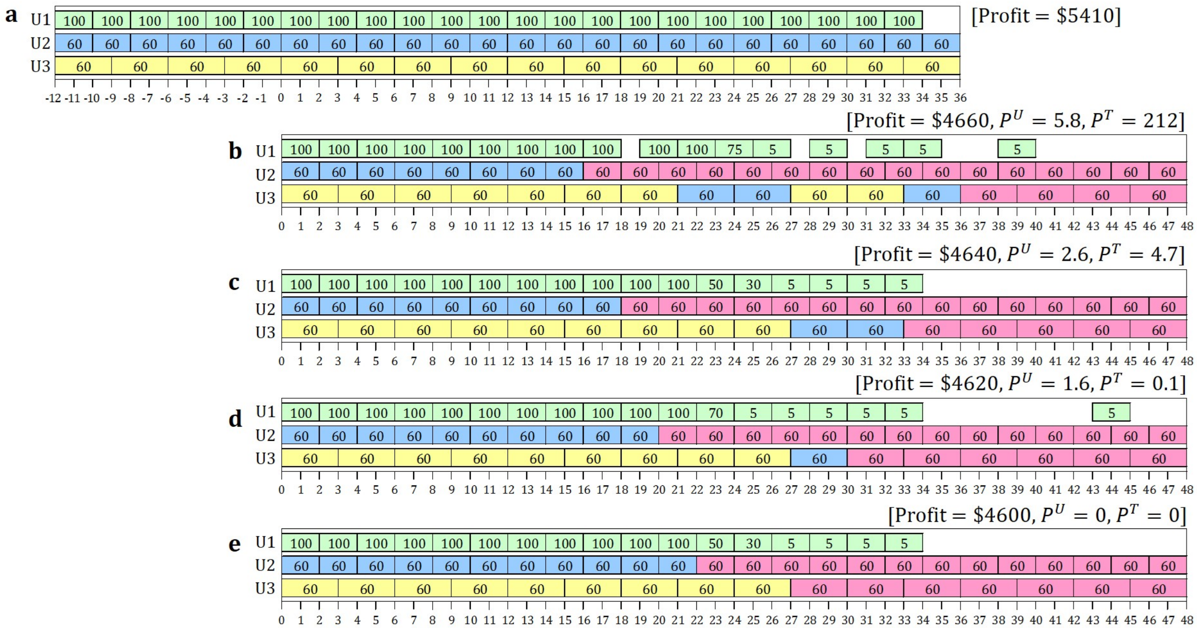


Figure 6.10: Schedules obtained for instance 2 with $\omega^{AR} = 1$ and different ω^{UT}'s: (a) base schedule, (b) $\omega^{UT} = 0$, (c) $\omega^{UT} = 1$, (d) $\omega^{UT} = 3$, and (e) $\omega^{UT} = 8$. The numbers within colored blocks represent batch sizes [kg].

schedule and the corresponding batch in the base schedule.

$$Q_{ij't'} \geq B_{ij't'} - \sum_{t \in \mathbf{T} \setminus \mathbf{T}^1} \sum_{j \in \mathbf{J}_i : \Omega^X_{ijt} = 1} \Omega^B_{ijt} A_{ijj'tt'} - M \cdot Y_{ij't'} \quad \forall i, j' \in \mathbf{J}_i, t' \in \mathbf{T} \setminus \mathbf{T}^1 \tag{6.36}$$

$$Q_{ij't'} \geq \sum_{t \in \mathbf{T} \setminus \mathbf{T}^1} \sum_{j \in \mathbf{J}_i : \Omega^X_{ijt} = 1} \Omega^B_{ijt} A_{ijj'tt'} - B_{ij't'} - M \cdot Y_{ij't'} \quad \forall i, j' \in \mathbf{J}_i, t' \in \mathbf{T} \setminus \mathbf{T}^1 \tag{6.37}$$

where M is a sufficiently large number (e.g. $M = \max_j \beta_j^{max}$). Note that Eqs. 6.36-6.37 become trivial if the batch of task i in unit j' at time point t' is a newly added batch (i.e. $Y_{ij't'} = 1$).

The metric for change in batch sizes is defined as follows.

$$P^B = \sum_i \sum_{j \in \mathbf{J}_i} \sum_{t \in \mathbf{T}^2} \mu_t^B Q_{ijt} \tag{6.38}$$

where μ_t^B is the weight for change in batch size at time point t. The objective function of choice is then modified to include the penalty term, $\omega^B P^B$. The full model, referred to as M2, consists of Eqs. 6.22-6.27, 6.29, 6.32-6.33, 6.36-6.37, and a modified objective function.

To illustrate, we consider instance 2 where a large demand of material G is observed at 48 h.

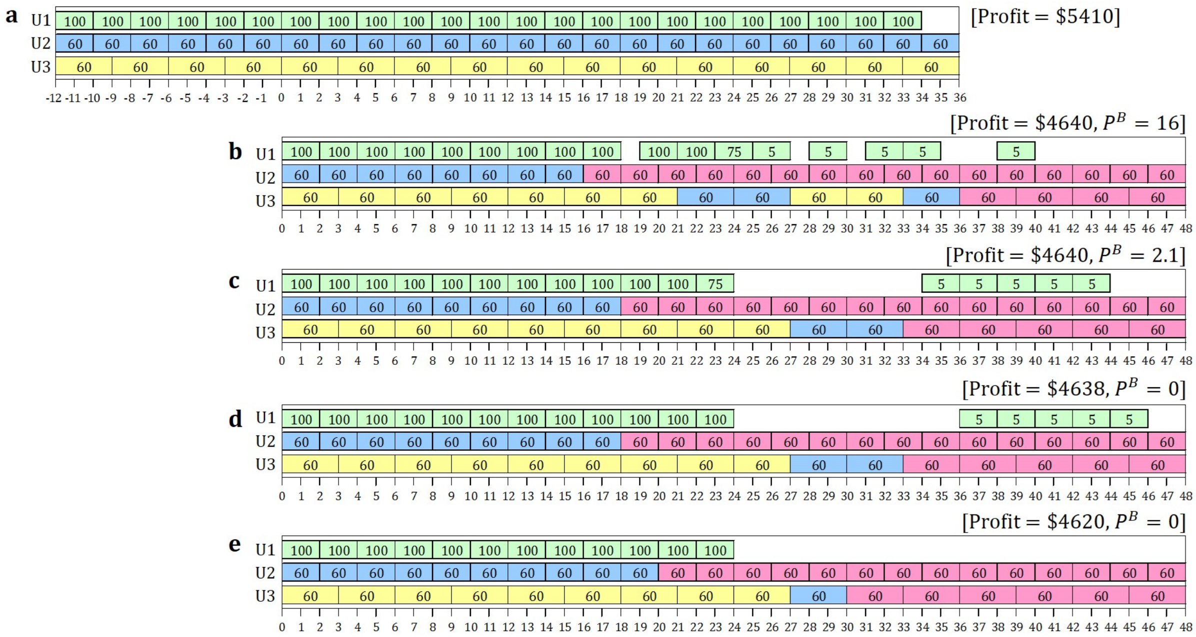


Figure 6.11: Schedules obtained for instance 2 with $\omega^{AR} = \omega^{UT} = 1$, and different ω^B's: (a) base schedule, (b) $\omega^B = 0$, (c) $\omega^B = 1$, (d) $\omega^B = 5$, and (e) $\omega^{UT} = \omega^B = 5$. The numbers within colored blocks represent batch sizes [kg].

The same interval lengths and base schedule as in the previous subsection is used. We use μ_t^B as given in Lee et al. (2019). We show the schedules obtained with different ω^B's in Figure 6.11. In Figure 6.11(b) ($\omega^B = 0$), many batches of T4 (red) are scheduled, which require less amount of intermediate material D. This is reflected in the decreased batch sizes of T1 (green). In Figure 6.11(c), when the batch size changes are penalized ($\omega^B = 1$), the batches with changed batch sizes are shifted in time towards interval 3 to reduce the penalty. In Figure 6.11(d) ($\omega^B = 5$), all batches whose batch size is changed are completely pushed to interval 3, leading to zero penalty. When time shifts are penalized more heavily, as in Figure 6.11(e) ($\omega^{UT} = 5$), the batches of T1 are removed, rather than having different start time and batch size.

6.4.4 *Selection of penalty weights*

Selecting penalty weights for a given process depends on how nervousness is measured. It is difficult to propose a general set of weights (i.e. μ^{AR}, μ^{UT} and μ^B) that is applicable to any process. Instead, we provide some insights on the selection of such weights and discuss several desirable properties that they should satisfy.

First, the penalty for adding a batch at time point t should be greater than or equal to the penalty for shifting a batch of the same task from interval 3 to time point t (see Figure 6.12(a)). Likewise, the penalty for removing a batch at time point t should be greater than or equal to the penalty for shifting a batch of the same task from time point t to interval 3 (see Figure 6.12(b)). Finally, shifting a batch closer to the beginning of the horizon should be penalized more heavily than pushing it towards the end of the horizon (see Figure 6.12(c)).

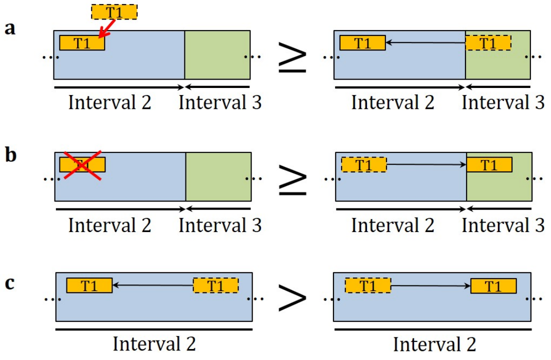


Figure 6.12: Properties that penalty weights should satisfy: (a) addition vs time shift, (b) removal vs time shift, and (c) shift closer vs farther.

In this study, we use the following weights.

$$\mu_t^A = g(t) \cdot \mu A \tag{6.39}$$

$$\mu_t^R = g(t) \cdot \mu R \tag{6.40}$$

$$\mu_{tt'}^U = \begin{cases} g(t') \cdot \mu U & (t < t') \\ g(t) \cdot \mu U & (t \geq t') \end{cases} \tag{6.41}$$

$$\mu_{tt'}^T = \begin{cases} g(t') \cdot \delta \cdot \left| t - t' \right| \cdot \mu T & (t \geq t') \\ g(t) \cdot \delta \cdot \left| t - t' \right| \cdot \mu T' & (t < t') \end{cases} \tag{6.42}$$

$$\mu_t^B = g(t) \cdot \mu B \tag{6.43}$$

where μA, μR, μU, μT and μB are constants that determine the overall magnitude of the weights. We use a logarithmically decaying function, $g(t)$ (see Eq. 6.44), which is 1 at the beginning of interval 2 and decays to 0 at the beginning of interval 3 (see Figure 6.13). The proposed weights satisfy all the aforementioned properties if the values of the constants are set appropriately.

$$g(t) = \begin{cases} \frac{log(|\mathbf{T}^1|+|\mathbf{T}^2|)}{log(|\mathbf{T}^1|+|\mathbf{T}^2|)-log(|\mathbf{T}^1|)} - \frac{1}{log(|\mathbf{T}^1|+|\mathbf{T}^2|)-log(|\mathbf{T}^1|)} log(t) & (|\mathbf{T}^1| \leq t \leq |\mathbf{T}^1| + |\mathbf{T}^2|) \\ 0 & (t > |\mathbf{T}^1| + |\mathbf{T}^2|) \end{cases} \tag{6.44}$$

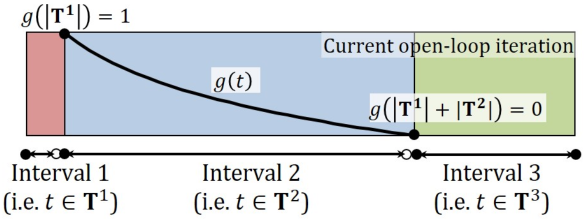

$g(|\mathbf{T}^1|) = 1$
Current open-loop iteration
$g(t)$
$g(|\mathbf{T}^1| + |\mathbf{T}^2|) = 0$
Interval 1
(i.e. $t \in \mathbf{T}^1$)
Interval 2
(i.e. $t \in \mathbf{T}^2$)
Interval 3
(i.e. $t \in \mathbf{T}^3$)

Figure 6.13: Illustration of the proposed $g(t)$.

We end this section noting that measuring nervousness is challenging, and is, by itself, a research topic. In this work, we provide a potential method for measuring nervousness, mainly accounting for addition, removal, reassignment and time shifting of batches. If needed, the above changes can be weighted by the batch sizes, B_{ijt}. Furthermore, changes in batch sequencing can also be accounted for by utilizing the modes of units, similar to the ones used in metric P^S.

6.5 APPLICATIONS

6.5.1 *Example 1: modeling simplifications*

We consider an illustrative instance based on the network shown in Figure 6.14(a), referred to as instance 3. We obtain four alternative schedules by solving M1 with makespan minimization objective function using $\boldsymbol{\omega}$'s: $(\omega^N, \omega^S, \omega^I)$ = (0,0,0), (0,0,1), (0,1,1) and (0.05,1,1). The obtained schedules are shown in Figure 6.15, and the model statistics along with the objective function and metric values are shown in Table 6.1.

The best makespan for this instance is 37 h (see Figure 6.15(a)). In this schedule, many batches are executed back-to-back without idle time in between. In Figure 6.15(b), an alternative schedule with idle times in between batches is found using $\omega^I = 1$. The presence of idle times leads to an increased makespan of 43 h. Another alternative schedule is obtained with $\omega^S = 1$ (Figure 6.15(c)), where the sequencing of batches in all units has changed significantly. Finally, using

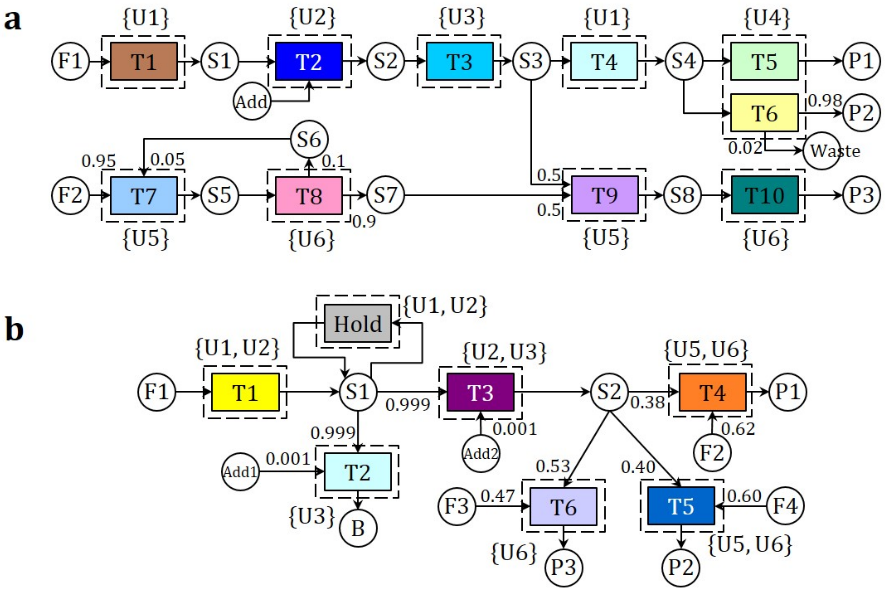


Figure 6.14: STN of illustrative instances (a) 3 and (b) 4.

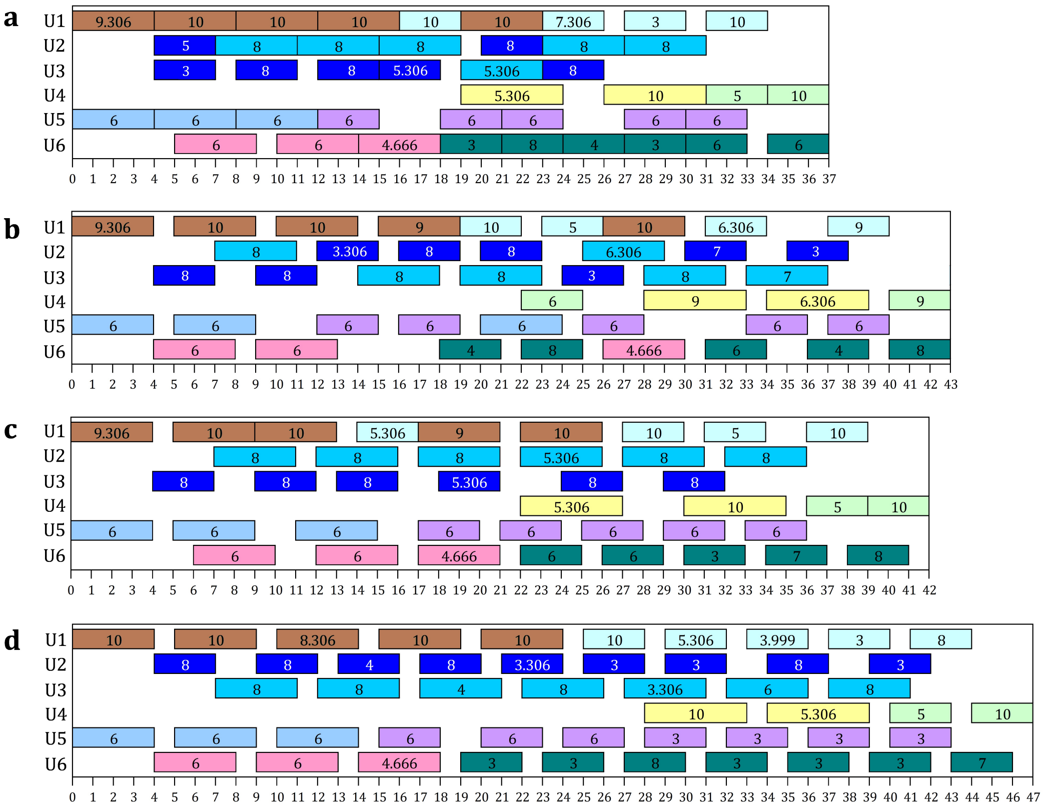


Figure 6.15: Schedules obtained for instance 3 with $(\omega^N, \omega^S, \omega^I)$'s: (a) $(0, 0, 0)$, (b) $(0, 0, 1)$, (c) $(0, 1, 1)$, and (d) $(0.05, 1, 1)$. The numbers within colored blocks represent batch sizes.

Table 6.1: Model statistics and results for instances 3 and 4

	Instance 3				Instance 4			
ω	(0,0,0)	(0,0,1)	(0,1,1)	(0.05,1,1)	(0,0,0)	(1,1,0)	(1,5,0)	(1,5,20)
Constraints	3383				5031			
Disc. Var.	1188				4076			
Cont. Var.	2173				1955			
Obj. w/ Pen.	37.0	45.0	51.0	44.6	1545.1	1588.4	1604.4	1634.8
Obj. (h or $)	37	43	42	47	1545.1	1553.2	1557.1	1562.1
P^N	-11.3	-10.3	-16.3	-127.3	-	-	-	-
P^S	37.0	65.0	6.0	4.0	-	-	-	-
P^I	26.0	2.0	3.0	0.0	-	-	-	-
P^A	-	-	-	-	645.5	25.6	35.1	58.0
P^R	-	-	-	-	257.4	0.0	7.3	7.3
P^U	-	-	-	-	0.0	7.5	0.5	0.5
P^T	-	-	-	-	0.0	2.1	0.5	1.0
P^B	-	-	-	-	0.0	86.2	78.8	0.0

$\omega^N = 0.05$, in addition to $\omega^I = 1$ and $\omega^S = 1$, we obtain a schedule with different number of batches assigned to units as shown in Figure 6.15(d). For example, we observe that the batches assigned to U2 and the ones assigned to U3 have switched their assignments compared to the schedule shown in Figure 6.15(c). Also, several larger batches are split into multiple smaller ones, increasing the makespan to 47 h.

6.5.2 *Example 2: shop-floor nervousness*

Next, we consider cost minimization problem for an instance based on network shown in Figure 6.14(b), referred to as instance 4, focusing on nervousness. We assume that the schedules are obtained in a rolling horizon fashion with a horizon of 48 h and re-optimization performed every 12 h. The base schedule is given in Figure 6.16(a). We set interval 1 as 0-3 h and interval 2 as 4-48 h (i.e. no interval 3). New demands for P2 and B are observed at 24 and 36 h. Other details of the instance can be found in Lee et al. (2019). Figure 6.16(b)-(e) shows the alternative schedules obtained when using $(\omega^{AR}, \omega^{UT}, \omega^{B})$ = (0,0,0), (1,1,0), (1,5,0) and (1,5,20). The model statistics along with the objective function and penalty function values are shown in Table 6.1.

Without any penalty (Figure 6.16(b)), additional batches of T5 (blue) are scheduled in U5, and several batches of T4 (orange) are reassigned to U6. When addition, removal, reassignment and time shift of batches are penalized (Figure 6.16(c)), we obtain a schedule that is closer to

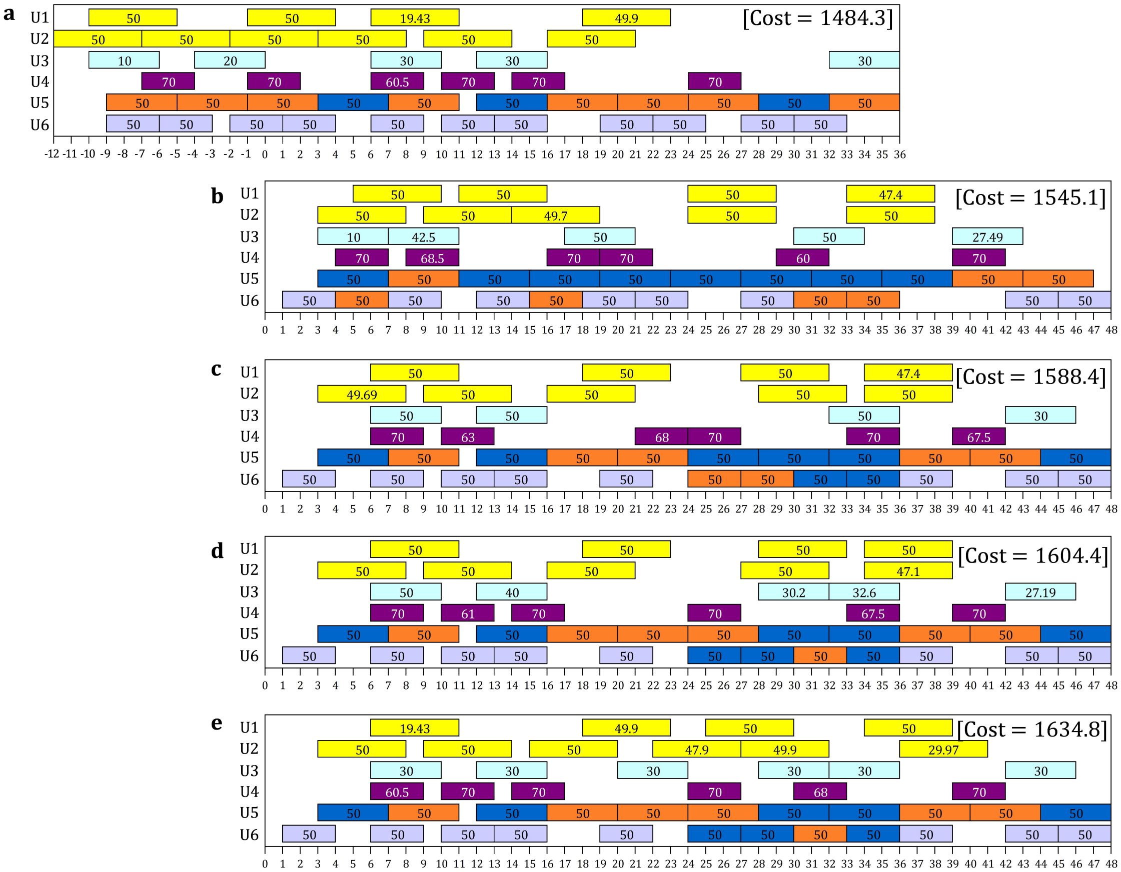


Figure 6.16: Schedules obtained for instance 4 with different $(\omega^{AR}, \omega^{UT}, \omega^{B})$'s: (a) base schedule, (b) $(0,0,0)$, (c) $(1,1,0)$, (d) $(1,5,0)$, and (e) $(1,5,20)$. The numbers within colored blocks represent batch sizes.

the base schedule, but we still observe several changes, including reassignment of two batches of T4 to U6 and increased batch sizes of T2 (light blue) to meet the increased demand for B. The increased batch size, in turn, causes the third and fourth batches of T3 (purple) in U4 to be delayed, due to the limited shared intermediate, S1. If reassignment and time shift of batches are undesired, a larger ω^{UT} can be used (Figure 6.16(d)). In this schedule, more batches of T5 are assigned to U6 to avoid reassigning batches of T4 as much as possible. Also, it can be observed that the third batch of T3 is no longer delayed, leading to decreased batch size of T2 at 12 h and an additional batch scheduled at 28 h to compensate for the decreased batch size. Finally, if the changes in batch size should be avoided unless absolutely necessary, one can use a large ω^{B} (Figure 6.16(e)). The obtained schedule has no changes in batch sizes, but multiple additional batches (e.g. batches of T1 (yellow) and T2) are scheduled instead.

6.6 CONCLUSIONS

Despite significant advances in the development of mathematical models and solution methods for chemical production scheduling, practical challenges still exist when it comes to industrial applications: (i) limited scope of the models due to introduced modeling simplifications, and (ii) consideration of nervousness when revising schedules. One practical and efficient method to address these challenges is to generate multiple alternative schedules for schedulers to evaluate, select and deploy.

Accordingly, in this work, we developed a systematic approach for generating multiple alternative schedules. Specifically, by introducing metrics, we quantify specific characteristics of a schedule, which are favored at different degrees by penalizing the objective function using varying penalty weights. The proposed methods allow for the generation of schedules with different (i) number of batches assigned to each unit, (ii) sequencing of batches, and (iii) idle times in between batches. In addition, the methods can be used to generate schedules with different degrees of nervousness, which is measured by quantifying (i) addition/removal, (ii) reassignment and time shift of batches and (iii) change in batch sizes. We showed that, by leveraging penalty weights, schedules with desirable properties can be readily found.

Future research directions include the development of algorithms to facilitate the process of obtaining multiple solutions, taking advantage of the fact that the models being solved in parallel are identical except for the penalty weights in the objective functions.

6.7 nomenclature

Indices/sets

$i \in \mathbf{I}$	tasks
$j \in \mathbf{J}$	units
$k \in \mathbf{K}$	materials
$t \in \mathbf{T}^A$	time points (or periods)
$\mathbf{I}_j$	tasks that can be processed in unit j
$\mathbf{I}_k^+/\mathbf{I}_k^-$	tasks that produce/consume material k
$\mathbf{J}_i$	units that can process task i
$\mathbf{J}^I$	units in which idle times are added
$\mathbf{J}^N$	units in which the number of mode transitions is quantified
$\mathbf{J}^S$	units in which the batch sequencing is quantified
$\mathbf{T}$	time points (or periods) in the based model
$\mathbf{T}^-$	time points (or periods) before $t = 0$ in model M2
$\mathbf{T}^1/\mathbf{T}^2/\mathbf{T}^3$	time points (or periods) in interval 1/2/3

Parameters

$\alpha_{ij}^F/\alpha_{ij}^V$	fixed/variable processing cost of task i in unit j
$\beta_j^{max}/\beta_j^{min}$	maximum/minimum capacity of unit j
γ_k	initial inventory of material k
δ	discretization step length
H	horizon length
θ_{ij}	length of desired idle time after the completion task i in unit j
$\mu^{[char]}$	weights, $[char] \in \{A, R, U, T, B\}$
N_{ij}^E	expected number of batch of task i assigned to unit j
π_k	price of material k
ρ_{ik}	conversion coefficient of material k of task i (produced (>0) or consumed (<0))
τ_{ij}	processing time of task i in unit j

τ^I_{ij} processing time of task i in unit j plus θ_{ij} (i.e. $\tau^I_{ij} = \tau_{ij} + \theta_{ij}$)

ξ_{kt} demand amount of material k due at time point t

$\omega^{[char]}$ penalty weights, $[char] \in \{N, S, I, AR, UT, B\}$

Ω^B_{ijt} batch size of task i scheduled in unit j at time point t in the base schedule

Ω^S_k inventory level of material k during time period $t = 0$ in the base schedule

Ω^X_{ijt} binary parameter equal to 1 if a batch of task i is scheduled in unit j at time point t in the base schedule

Variables

$A_{ijj'tt'} \in \{0, 1\}$ =1 if the batch of task i scheduled in unit j at time point t in the base schedule is reassigned to unit $j^{'}$ at time point $t^{'}$ in the revised schedule

$R_{ijt} \in \{0, 1\}$ =1 if unit j is in mode of processing task i at time t

$R^S_{jt} \in \{0, 1\}$ =1 if unit j is in neutral mode at time t

$W^N_{ij} \in \{0, 1\}$ =1 if the number of batches of task i assigned to unit j (N_{ij}) is greater than or equal to the expected (N^E_{ij})

$W^S_{ijt} \in \{0, 1\}$ =1 if mode of unit j is shifted from neutral to task i at time point t

$W^T_{ijt} \in \{0, 1\}$ =1 if mode of unit j is shifted from task i to neutral at time point t

$X_{ijt} \in \{0, 1\}$ =1 if task i starts in unit j at time point t

$Y_{ijt} \in \{0, 1\}$ =1 if a new batch of task i is added to unit j at time point t

$Z_{ijt} \in \{0, 1\}$ =1 if the batch of task i scheduled in unit j at time point t in the base schedule is executed in the revised schedule

$B_{ijt} \in \mathbb{R}_+$ batch size of task i in unit j at time point t

$\hat{N}_{ij} \in \mathbb{R}_+$ absolute difference between the expected (N^E_{ij}) and the scheduled (N_{ij}) number of batches of task i assigned to unit j

$MS \in \mathbb{R}_+$ makespan

$Q_{ijt} \in \mathbb{R}_+$ absolute difference between the batch sizes of the batch of task i in unit j at time point t in the revised schedule and the corresponding batch in the base schedule

$S_{kt} \in \mathbb{R}_+$ inventory level of material k during time period t

$N_{ij} \in \mathbb{Z}_+$ number of batches of task i assigned to unit j

$P^{[char]} \in \mathbb{R}$	metrics, $[char] \in \{N, S, I, A, R, U, T, B\}$
$W^{I}_{jt} \in [0, 1]$	slack variable in Eq. 6.20

7

CONCLUSIONS AND FUTURE DIRECTIONS

7.1 CONCLUSIONS

This thesis deals with the development of optimization models and solution methods for chemical production scheduling with an aim to bring optimization-based scheduling techniques closer to industrial applications. In particular, we addressed the following challenges in implementing such techniques in industrial settings: (i) scheduling models need to be general and should be able to handle different production environments as well as process-specific characteristics, (ii) the solution of such models should be fast to allow for frequent updates to the schedules, and (iii) the models should be capable of providing multiple alternative schedules for the practitioners to compare and implement.

In Chapter 2, we proposed two models for simultaneous batching and scheduling in general multipurpose batch plants, which is a production environment that has not been extensively studied. The models were based on novel modeling approaches, namely (i) explicit labeling of batches, and (ii) identification of feasible batch size intervals for each unit routing. Due to the different modeling approaches that were employed, the models allow for exploitation of instance characteristics, thus leading to solution of large-scale instances.

In Chapter 3, we proposed a framework for a solution algorithm, i.e. Discrete-Continuous Algorithm (DCA), that allows us to harness the advantages of both time representations while overcoming their limitations. We showed that, due to the flexible nature of the framework, any discrete-time model and solution methods can be employed in the algorithm.

In Chapter 4, we discussed the impact of the model parameters used in DCA, namely discretization step length and horizon relaxation, on the computational performance and solution quality. However, since finding a *good* set of model parameters for a given instance is a non-trivial task, we proposed systematic methods to find them through the introduction of error evaluation and cumulative error functions. We showed that DCA with the systematically determined model parameters brings up to four orders of magnitude improvement in computational speed, compared to traditional scheduling models.

In Chapter 5, we generalized DCA to account for a wide range of process characteristics that are commonly observed in industrial settings. Although accounting for such characteristics in a model is computationally demanding, it is done much more efficiently in DCA by modeling them with potentially less accurate but computationally less expensive constraints without the loss of solution accuracy. Through a real-world industrial case study, we showed that generalized DCA can readily account for various characteristics, including the ones that are not explicitly discussed in this work, with remarkable computational efficiency.

Next, in Chapter 6, we proposed systematic methods to generate multiple alternative schedules. The proposed methods can effectively address some of the challenges of implementing scheduling techniques to industries, namely (i) addressing the modeling simplifications introduced in the scheduling models, and (ii) consideration of nervousness when revising schedules. The methods employed metrics, in which specific characteristics of a schedule are quantified, and multiple schedules were generated by favoring the metrics at different degrees in the objective function.

7.2 FUTURE DIRECTIONS

The contributions from this work open up the possibility of industrial application of optimization-based scheduling techniques. To facilitate the process, as well as to nurture new methods, future research direction includes the development of scheduling tools that enable the users to freely experiment with the existing and newly developed methods.

Although DCA is capable of finding high quality solutions fast even for industrial-scale instances, its limitation is that it may provide suboptimal solutions in some cases. One effective method to overcome this is to generate pool of solutions in the first stage of DCA, instead of one, which will greatly enhance the chances of finding the optimal solutions in such cases. One should investigate different strategies of generating the solution pool, which may involve utilizing different combinations of key model parameters that are identified with the methods we proposed in Chapter 4.

Another method to overcome the limitation of DCA is to develop a branch-and-bound algorithm that employs a pruning strategy that considers the potential improvements in the objective function value achieved at the third stage of DCA. Specifically, when an integer solution for the first stage discrete-time model is found at a node of the branch-and-bound tree, second and third stages of DCA is called to calculate the improved objective function value, which then is used as the basis for pruning. This method can also be extended to calculate improvements in the objective function value not only at the integer feasible nodes, but also at fractional nodes, allowing for a more efficient pruning.

In Chapter 6, we discussed how plant nervousness can be measured and be taken into account when generating schedules. Another future research direction would be to formalize the concept of plant nervousness and standardize the method of quantifying nervousness of a given schedule. In addition to the elements that we considered when measuring nervousness, a critical element that should be taken into account is the changes in batch sequencing.

Furthermore, our systematic method for generating alternative schedules can potentially benefit from a parallel computing algorithm that takes advantage of the fact that the models being solved in parallel are identical except for the penalty weights in the objective functions. Specifically, throughout the solution process, the bounds obtained in one of the models can be propagated in real-time to the other models, potentially leading to better bound improvements.

A

EXTENDED BATCH SIZE INTERVAL ALGORITHMS

A.1 BATCH SIZE INTERVAL ALGORITHM WITH LIMITED VESSEL CAPACITY

Vessel capacities are taken into account when generating the intervals, reflected in $\mathbf{J}_i^{max}$ and $\mathbf{J}_i^{min}$.

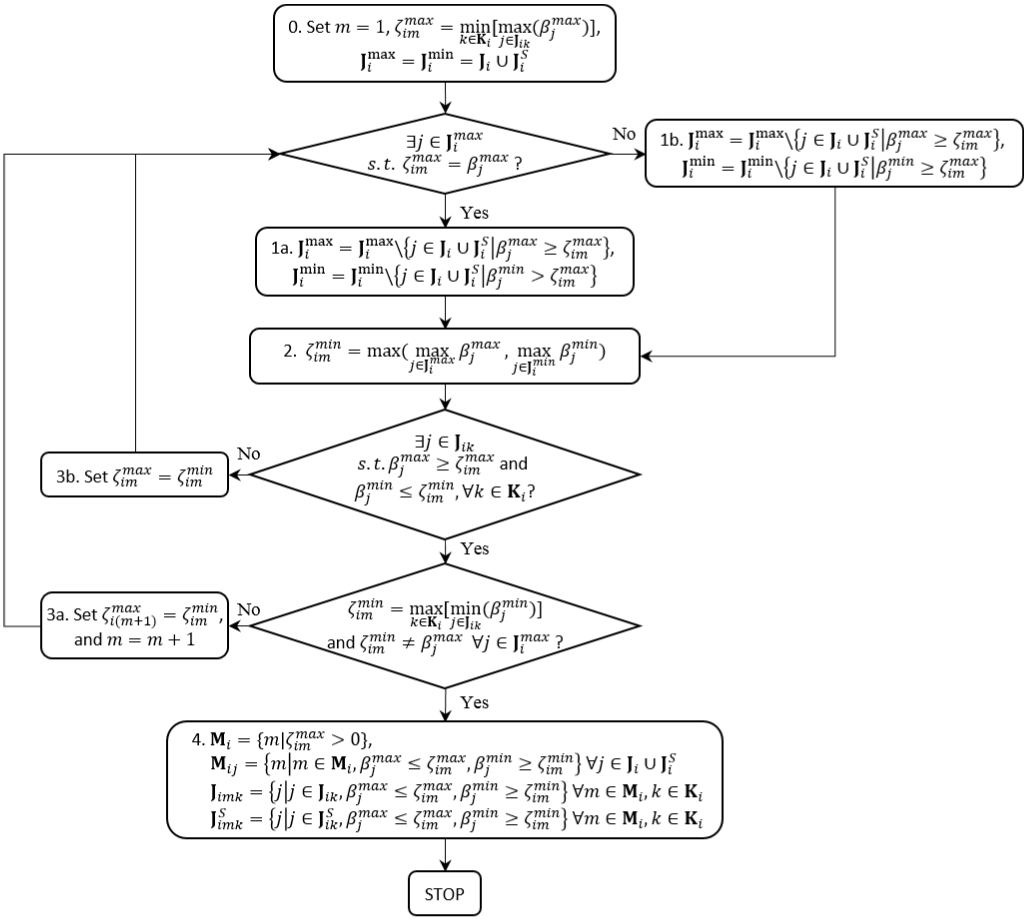

0. Set $m = 1$, $\zeta_{im}^{max} = \min_{k\in\mathbf{K}_i}[\max_{j\in\mathbf{J}_{ik}}(\beta_j^{max})]$, $\mathbf{J}_i^{\mathbf{max}} = \mathbf{J}_i^{\mathbf{min}} = \mathbf{J}_i \cup \mathbf{J}_i^S$
$\exists j \in \mathbf{J}_i^{max}$ $s.t.\ \zeta_{im}^{max} = \beta_j^{max}$?
No
1b. $\mathbf{J}_i^{\mathbf{max}} = \mathbf{J}_i^{\mathbf{max}} \backslash \{j \in \mathbf{J}_i \cup \mathbf{J}_i^S | \beta_j^{max} \geq \zeta_{im}^{max}\}$, $\mathbf{J}_i^{\mathbf{min}} = \mathbf{J}_i^{\mathbf{min}} \backslash \{j \in \mathbf{J}_i \cup \mathbf{J}_i^S | \beta_j^{min} \geq \zeta_{im}^{max}\}$
Yes
1a. $\mathbf{J}_i^{\mathbf{max}} = \mathbf{J}_i^{\mathbf{max}} \backslash \{j \in \mathbf{J}_i \cup \mathbf{J}_i^S | \beta_j^{max} \geq \zeta_{im}^{max}\}$, $\mathbf{J}_i^{\mathbf{min}} = \mathbf{J}_i^{\mathbf{min}} \backslash \{j \in \mathbf{J}_i \cup \mathbf{J}_i^S | \beta_j^{min} > \zeta_{im}^{max}\}$
2. $\zeta_{im}^{min} = \max(\max_{j\in\mathbf{J}_i^{max}} \beta_j^{max}, \max_{j\in\mathbf{J}_i^{min}} \beta_j^{min})$
$\exists j \in \mathbf{J}_{ik}$ $s.t.\ \beta_j^{max} \geq \zeta_{im}^{max}$ and $\beta_j^{min} \leq \zeta_{im}^{min}, \forall k \in \mathbf{K}_i$?
No
3b. Set $\zeta_{im}^{max} = \zeta_{im}^{min}$
Yes
$\zeta_{im}^{min} = \max_{k\in\mathbf{K}_i}[\min_{j\in\mathbf{J}_{ik}}(\beta_j^{min})]$ and $\zeta_{im}^{min} \neq \beta_j^{max}\ \forall j \in \mathbf{J}_i^{max}$?
No
3a. Set $\zeta_{i(m+1)}^{max} = \zeta_{im}^{min}$, and $m = m + 1$
Yes
4. $\mathbf{M}_i = \{m|\zeta_{im}^{max} > 0\}$,
$\mathbf{M}_{ij} = \{m|m \in \mathbf{M}_i, \beta_j^{max} \leq \zeta_{im}^{max}, \beta_j^{min} \geq \zeta_{im}^{min}\}\ \forall j \in \mathbf{J}_i \cup \mathbf{J}_i^S$
$\mathbf{J}_{imk} = \{j|j \in \mathbf{J}_{ik}, \beta_j^{max} \leq \zeta_{im}^{max}, \beta_j^{min} \geq \zeta_{im}^{min}\}\ \forall m \in \mathbf{M}_i, k \in \mathbf{K}_i$
$\mathbf{J}_{imk}^S = \{j|j \in \mathbf{J}_{ik}^S, \beta_j^{max} \leq \zeta_{im}^{max}, \beta_j^{min} \geq \zeta_{im}^{min}\}\ \forall m \in \mathbf{M}_i, k \in \mathbf{K}_i$
STOP

Figure A.1: Algorithm when vessel capacity is considered.

A.2 Batch size interval algorithm with limited vessel capacity and stage-dependent batch sizes

This is extended based on the algorithm that only considers stage-dependent batch size (Figure 2.9), taking the vessel capacity into consideration. Since $\mathbf{J}_{ik}^{S}$ denote the set of vessels that can store batches that *have been* processed on stage k, the batch that is being stored in $j \in \mathbf{J}_{ik}^{S}$ has the batch size of its subsequent stage (i.e. $k+1$). This results in index shifts for $\mathbf{J}_{ik}^{S}$ and χ_{ik} when initializing $\mathbf{P}_i^{max}/\mathbf{P}_i^{min}$ (Step 0) and generating $\mathbf{M}_{ijk}^{S}$ (Step 4), respectively.

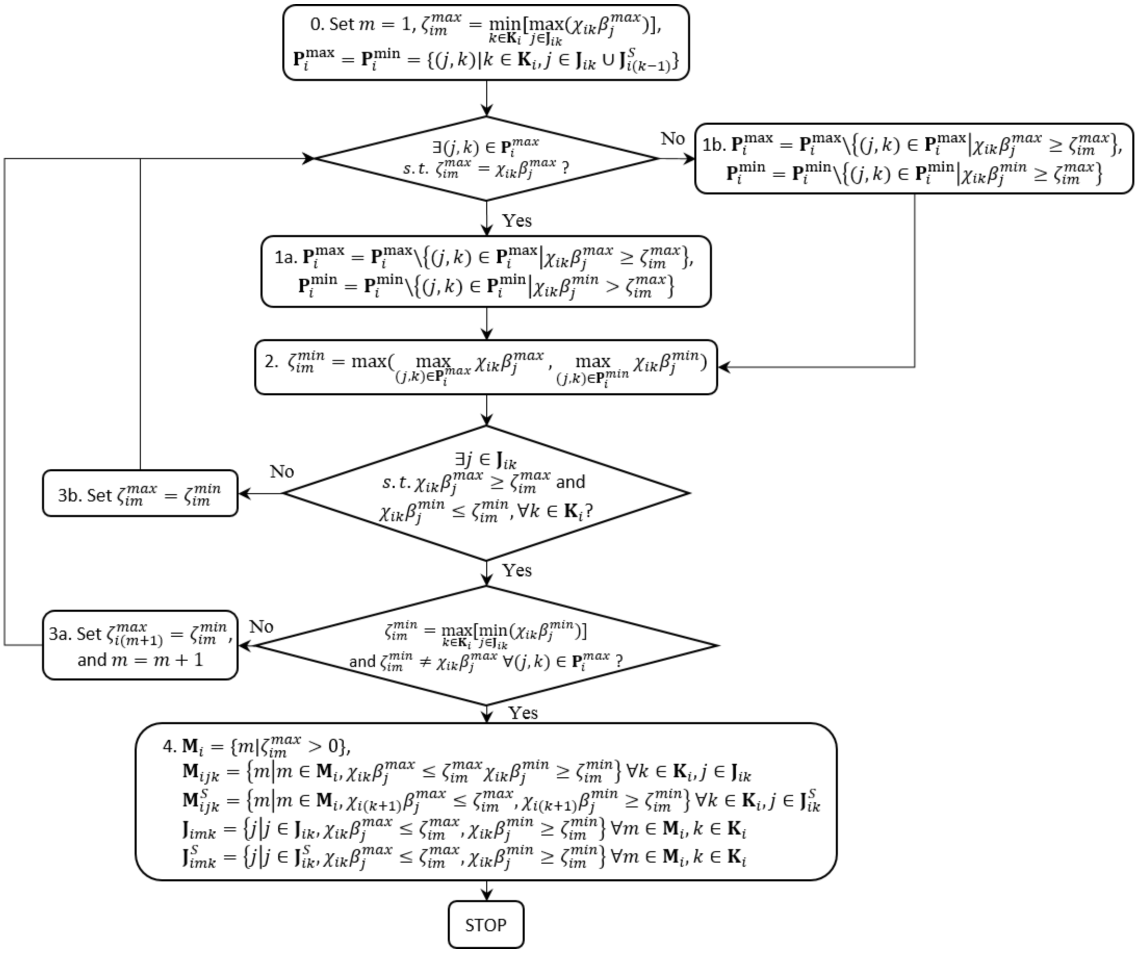

0. Set $m = 1$, $\zeta_{im}^{max} = \min_{k\in\mathbf{K}_i}[\max_{j\in\mathbf{J}_{ik}}(\chi_{ik}\beta_j^{max})]$,
$\mathbf{P}_i^{\mathrm{max}} = \mathbf{P}_i^{\mathrm{min}} = \{(j,k)|k \in \mathbf{K}_i, j \in \mathbf{J}_{ik} \cup \mathbf{J}_{i(k-1)}^{S}\}$
$\exists(j,k) \in \mathbf{P}_i^{max}$
$s.t.\ \zeta_{im}^{max} = \chi_{ik}\beta_j^{max}$?
No
1b. $\mathbf{P}_i^{\mathrm{max}} = \mathbf{P}_i^{\mathrm{max}} \backslash \{(j,k) \in \mathbf{P}_i^{\mathrm{max}} | \chi_{ik}\beta_j^{max} \geq \zeta_{im}^{max}\}$,
$\mathbf{P}_i^{\mathrm{min}} = \mathbf{P}_i^{\mathrm{min}} \backslash \{(j,k) \in \mathbf{P}_i^{\mathrm{min}} | \chi_{ik}\beta_j^{min} \geq \zeta_{im}^{max}\}$
Yes
1a. $\mathbf{P}_i^{\mathrm{max}} = \mathbf{P}_i^{\mathrm{max}} \backslash \{(j,k) \in \mathbf{P}_i^{\mathrm{max}} | \chi_{ik}\beta_j^{max} \geq \zeta_{im}^{max}\}$,
$\mathbf{P}_i^{\mathrm{min}} = \mathbf{P}_i^{\mathrm{min}} \backslash \{(j,k) \in \mathbf{P}_i^{\mathrm{min}} | \chi_{ik}\beta_j^{min} > \zeta_{im}^{max}\}$
2. $\zeta_{im}^{min} = \max(\max_{(j,k)\in\mathbf{P}_i^{max}} \chi_{ik}\beta_j^{max}, \max_{(j,k)\in\mathbf{P}_i^{min}} \chi_{ik}\beta_j^{min})$
$\exists j \in \mathbf{J}_{ik}$
$s.t.\ \chi_{ik}\beta_j^{max} \geq \zeta_{im}^{max}$ and
$\chi_{ik}\beta_j^{min} \leq \zeta_{im}^{min}, \forall k \in \mathbf{K}_i$?
No
3b. Set $\zeta_{im}^{max} = \zeta_{im}^{min}$
Yes
$\zeta_{im}^{min} = \max_{k\in\mathbf{K}_i}[\min_{j\in\mathbf{J}_{ik}}(\chi_{ik}\beta_j^{min})]$
and $\zeta_{im}^{min} \neq \chi_{ik}\beta_j^{max}\ \forall(j,k) \in \mathbf{P}_i^{max}$?
No
3a. Set $\zeta_{i(m+1)}^{max} = \zeta_{im}^{min}$,
and $m = m + 1$
Yes
4. $\mathbf{M}_i = \{m|\zeta_{im}^{max} > 0\}$,
$\mathbf{M}_{ijk} = \{m|m \in \mathbf{M}_i, \chi_{ik}\beta_j^{max} \leq \zeta_{im}^{max} \chi_{ik}\beta_j^{min} \geq \zeta_{im}^{min}\}\ \forall k \in \mathbf{K}_i, j \in \mathbf{J}_{ik}$
$\mathbf{M}_{ijk}^{S} = \{m|m \in \mathbf{M}_i, \chi_{i(k+1)}\beta_j^{max} \leq \zeta_{im}^{max}, \chi_{i(k+1)}\beta_j^{min} \geq \zeta_{im}^{min}\}\ \forall k \in \mathbf{K}_i, j \in \mathbf{J}_{ik}^{S}$
$\mathbf{J}_{imk} = \{j|j \in \mathbf{J}_{ik}, \chi_{ik}\beta_j^{max} \leq \zeta_{im}^{max}, \chi_{ik}\beta_j^{min} \geq \zeta_{im}^{min}\}\ \forall m \in \mathbf{M}_i, k \in \mathbf{K}_i$
$\mathbf{J}_{imk}^{S} = \{j|j \in \mathbf{J}_{ik}^{S}, \chi_{ik}\beta_j^{max} \leq \zeta_{im}^{max}, \chi_{ik}\beta_j^{min} \geq \zeta_{im}^{min}\}\ \forall m \in \mathbf{M}_i, k \in \mathbf{K}_i$
STOP

Figure A.2: Algorithm when vessel capacity and stage-dependent batch sizes are considered.

B

MAPPING ALGORITHMS

B.1 MAPPING ALGORITHM WITH INTERMEDIATE DELIVERIES AND DEMANDS

We make an addition (i.e. lines 16-21 and 35-42) to the algorithm to detect whether there exist intermediate deliveries and demands.

Table B.1: Mapping algorithm with intermediate deliveries and demand

```
Initialize: $M_j^U = 0 \ \forall j,\ M_k^M = 0 \ \forall k,\ f^{1\sim 4} = 0,\ \mathbf{M}_k^P = \mathbf{M}_k^C = \emptyset \ \forall k$
Loop $n \in \mathbf{N}$
  Loop $k \in \mathbf{K}$
    Loop $i \in \mathbf{I}_k^+$
      Loop $j \in \mathbf{J}_i$
        If $X_{ij(n-\bar{\tau}_{ij})} = 1$
          $M_k^M \leftarrow M_k^M + 1$
          $M_j^U \leftarrow \sum_{n'<n} \sum_{i' \in \mathbf{I}_j} X_{i'jn'}$
          $\mathbf{M}_k^P \leftarrow \mathbf{M}_k^P \cup \{M_k^M\}$
          $f^1_{ik,m(=M_k^M)} \leftarrow 1$
          $f^2_{jk,m(=M_k^M)} \leftarrow 1$
          $f^4_{ijk,m(=M_j^U),m'(=M_k^M)} \leftarrow 1$
        EndIf
    EndLoop x2
    Loop $p \in \mathbf{P}$
      If $\xi_{kp} < 0$ and $n = \lceil t^{\xi}_{kp}/\delta \rceil$
        $M_k^M \leftarrow M_k^M + 1$
        $\mathbf{M}_k^P \leftarrow \mathbf{M}_k^P \cup \{M_k^M\}$
        $f^5_{kmp} \leftarrow 1$
      EndIf
    EndLoop
    Loop $i \in \mathbf{I}_k^-$
      Loop $j \in \mathbf{J}_i$
        If $X_{ijn} = 1$
          $M_k^M \leftarrow M_k^M + 1$
          $M_j^N \leftarrow \sum_{n' \leq n} \sum_{i' \in \mathbf{I}_j} X_{i'jn'}$
          $\mathbf{M}_k^C \leftarrow \mathbf{M}_k^C \cup \{M_k^M\}$
          $f^1_{ik,m(=M_k^M)} \leftarrow 1$
          $f^2_{jk,m(=M_k^M)} \leftarrow 1$
          $f^3_{kmm'} \leftarrow 1 \ \ \forall m < M_k^M,\ m \in \mathbf{M}_k^P,\ m' = M_k^M$
          $f^4_{ijk,m(=M_j^U),m'(=M_k^M)} \leftarrow 1$
        EndIf
    EndLoop x2
    Loop $p \in \mathbf{P}$
      If $\xi_{kp} > 0$ and $n = \lfloor t^{\xi}_{kp}/\delta \rfloor$
        $M_k^M \leftarrow M_k^M + 1$
        $\mathbf{M}_k^C \leftarrow \mathbf{M}_k^C \cup \{M_k^M\}$
        $f^3_{kmm'} \leftarrow 1 \ \ \forall m < M_k^M,\ m \in \mathbf{M}_k^P,\ m' = M_k^M$
        $f^5_{kmp} \leftarrow 1$
      EndIf
    EndLoop
EndLoop x2
```

B.2 Mapping algorithm with limited shared utilities with time-varying availability

Instead of detecting and recording consumption and production of materials as in Table B.1, we detect and record consumption and release of utilities in the algorithm.

Table B.2: Mapping algorithm with limited shared utilities

```
Initialize: $M^U_j = 0 \ \forall j,\ M^R_r = 0 \ \forall r,\ f^{6\sim 9} = 0,\ \mathbf{M}^{R,P}_r = \mathbf{M}^{R,C}_r = \emptyset \ \forall r$
Loop $n \in \mathbf{N}$
  Loop $r \in \mathbf{R}$
    Loop $i \in \mathbf{I}_r$
      Loop $j \in \mathbf{J}_i$
        If $X_{ij(n-\bar{\tau}_{ij})} = 1$
          $M^R_r \leftarrow M^R_r + 1$
          $M^U_j \leftarrow \sum_{n'<n} \sum_{i' \in \mathbf{I}_j} X_{i'jn'}$
          $\mathbf{M}^{R,P}_r \leftarrow \mathbf{M}^{R,P}_r \cup \{M^R_r\}$
          $f^6_{ir,m(=M^R_r)} \leftarrow 1$
          $f^8_{ijr,m(=M^U_j),m'(=M^R_r)} \leftarrow 1$
        EndIf
    EndLoop x2
    Loop $p \in \mathbf{P}$
      If $\psi^{Var}_{pr} > 0$ and $n = \lceil t^{\psi}_{pr}/\delta \rceil$
        $M^R_r \leftarrow M^R_r + 1$
        $\mathbf{M}^{R,P}_r \leftarrow \mathbf{M}^{R,P}_r \cup \{M^R_r\}$
        $f^9_{rmp} \leftarrow 1$
      EndIf
    EndLoop
    Loop $i \in \mathbf{I}_r$
      Loop $j \in \mathbf{J}_i$
        If $X_{ijn} = 1$
          $M^R_r \leftarrow M^R_r + 1$
          $M^U_j \leftarrow \sum_{n' \le n} \sum_{i' \in \mathbf{I}_j} X_{i'jn'}$
          $\mathbf{M}^{R,C}_r \leftarrow \mathbf{M}^{R,C}_r \cup \{M^R_r\}$
          $f^6_{ir,m(=M^R_r)} \leftarrow 1$
          $f^7_{rmm'} \leftarrow 1 \ \forall m < M^R_r,\ m \in \mathbf{M}^{R,P}_r,\ m' = M^R_r$
          $f^8_{ijr,m(=M^U_j),m'(=M^R_r)} \leftarrow 1$
        EndIf
    EndLoop x2
    Loop $p \in \mathbf{P}$
      If $\psi^{Var}_{pr} < 0$ and $n = \lfloor t^{\psi}_{pr}/\delta \rfloor$
        $M^R_r \leftarrow M^R_r + 1$
        $\mathbf{M}^{R,C}_r \leftarrow \mathbf{M}^{R,C}_r \cup \{M^R_r\}$
        $f^7_{rmm'} \leftarrow 1 \ \forall m < M^R_r,\ m \in \mathbf{M}^{R,P}_r,\ m' = M^R_r$
        $f^9_{rmp} \leftarrow 1$
      EndIf
    EndLoop
EndLoop x2
```

B.3 GENERAL MAPPING ALGORITHM

We present the mapping algorithm in blocks. The full algorithm is presented in Table B.3, and individual blocks are presented in Tables B.4-B.15.

Table B.3: Mapping algorithm

```
Initialize: $M_j^U = 0 \ \forall j,\ M_k^M = 0 \ \forall k,$
            $f^{1\sim 4} = g^{0\sim 7} = 0,$
            $\mathbf{M}_k^P = \mathbf{M}_k^C = \mathbf{M}_k^{P-} = \mathbf{M}_k^{C-} = \emptyset \ \forall k$
Loop $n \in \mathbf{N}$
  Loop $k \in \mathbf{K}$
    Loop $i \in \mathbf{I}_k^+$
      Loop $j \in \mathbf{J}_i$
        Go to Block 5-1
        Go to Block 7
    EndLoop x2
    Go to Block 6-1
    Loop $i \in \mathbf{I}_k^-$
      Loop $j \in \mathbf{J}_i$
        Go to Block 5-2
    EndLoop x2
    Go to Block 6-2
    Go to Block 8-3
    Go to Block 8-2
    Go to Block 9
  EndLoop
  Go to Block 8-4
EndLoop
Go to Block 11
```

Table B.4: Block 5-1

```
If $X_{ij(n-\bar{\tau}_{ij})} = 1$
  $M_k^M \leftarrow M_k^M + 1$
  $M_j^U \leftarrow \sum_{n'<n} \sum_{i' \in \mathbf{I}_j} X_{i'jn'}$
  If $j \in \mathbf{J}^H$
    $\mathbf{M}_k^H \leftarrow \mathbf{M}_k^H \cup \{M_k^M\}$
  Else
    $\mathbf{M}_k^P \leftarrow \mathbf{M}_k^P \cup \{M_k^M\}$
  EndIf
  $f^1_{ik,m(=M_k^M)} \leftarrow 1$
  $f^2_{jk,m(=M_k^M)} \leftarrow 1$
  $f^4_{ijk,m(=M_j^U),m'(=M_k^M)} \leftarrow 1$
  Go to Block 8-1
  $\mathbf{M}_{kn}^{P,N} \leftarrow \mathbf{M}_{kn}^{P,N} \cup \{M_k^M\}$
EndIf
```

Table B.5: Block 5-2

```
If $X_{ijn} = 1$
  $M_k^M \leftarrow M_k^M + 1$
  $M_j^N \leftarrow \sum_{n' \leq n} \sum_{i' \in \mathbf{I}_j} X_{i'jn'}$
  $\mathbf{M}_k^C \leftarrow \mathbf{M}_k^C \cup \{M_k^M\}$
  $f^1_{ik,m(=M_k^M)} \leftarrow 1$
  $f^2_{jk,m(=M_k^M)} \leftarrow 1$
  $f^3_{kmm'} \leftarrow 1 \ \forall m < M_k^M,\ m \in \mathbf{M}_k^P,\ m' = M_k^M$
  $f^4_{ijk,m(=M_j^U),m'(=M_k^M)} \leftarrow 1$
  Go to Block 10
  Go to Block 8-1
  $\mathbf{M}_{kn}^{C,N} \leftarrow \mathbf{M}_{kn}^{C,N} \cup \{M_k^M\}$
EndIf
```

Table B.6: Block 6-1

```
Loop $p \in \mathbf{P}$
    If $\xi_{kp} < 0$ and $n = \lceil t^{\xi}_{kp}/\delta \rceil$
        $M^M_k \leftarrow M^M_k + 1$
        $\mathbf{M}^P_k \leftarrow \mathbf{M}^P_k \cup \{M^M_k\}$
        $f^5_{kmp} \leftarrow 1$
        Go to Block 8-1
        $\mathbf{M}^{P,N}_{kn} \leftarrow \mathbf{M}^{P,N}_{kn} \cup \{M^M_k\}$
    EndIf
EndLoop
```

Table B.7: Block 6-2

```
Loop $p \in \mathbf{P}$
    If $\xi_{kp} > 0$ and $n = \lceil t^{\xi}_{kp}/\delta \rceil$
        $M^M_k \leftarrow M^M_k + 1$
        $\mathbf{M}^P_k \leftarrow \mathbf{M}^C_k \cup \{M^M_k\}$
        $f^3_{kmm'} \leftarrow 1 \ \ \forall m < M^M_k,\ m \in \mathbf{M}^P_k,\ m' = M^M_k$
        $f^5_{kmp} \leftarrow 1$
        Go to Block 8-1
    $\mathbf{M}^{C,N}_{kn} \leftarrow \mathbf{M}^{C,N}_{kn} \cup \{M^M_k\}$
    EndIf
EndLoop
```

Table B.8: Block 7

```
If $F_{jkn} > 0$
    $M^U_j \leftarrow \sum_{n'<n} \sum_{i' \in \mathbf{I}_j} X_{i'jn'}$
    $M^M_k \leftarrow M^M_k + 1$
    $\mathbf{M}^P_k \leftarrow \mathbf{M}^P_k \cup \{M^M_k\}$
    $f^1_{ik(m=M^M_k)} \leftarrow 1$
    $g^1_{jk(m=M^U_j)(m'=M^M_k)} \leftarrow 1$
    Go to Block 8-1
    $\mathbf{M}^{P,N}_{kn} \leftarrow \mathbf{M}^{P,N}_{kn} \cup \{M^M_k\}$
EndIf
```

Table B.9: Block 8-1

```
Loop $j' \in \mathbf{J}^S_k$
    If $X^S_{j'kn} = 1$ or $X^S_{j'k(n+1)} = 1$
        $M^U_{j'} \leftarrow M^U_{j'} + 1$
        $g^3_{j'k(m=M^U_{j'})(m'=M^M_k)} \leftarrow 1$
    EndIf
    If $X^S_{j'k(n+1)} = 1$
        $g^2_{j'k(m=M^M_k)} \leftarrow 1$
    EndIf
EndLoop
```

Table B.10: Block 8-2

```
If $k \in \mathbf{K}^{LS}$ and $\sum_{i \in \mathbf{I}^+_k} \sum_{j \in \mathbf{J}_i \setminus \mathbf{J}^H} \rho_{ik} B_{ij(n-\bar{\tau}_{ij})}$
        $+ \sum_{j \in \mathbf{J}^H \cap \mathbf{J}^+_k} F_{jkn} + \bar{\xi}_{kn} > \sum_{j \in \mathbf{J}^S_k} \beta^{max}_j X^S_{jk(n+1)}$
    $\mathbf{M}^{P-}_k \leftarrow \mathbf{M}^{P-}_k \cup \mathbf{M}^{P,N}_{kn}$
    $\mathbf{M}^{C-}_k \leftarrow \mathbf{M}^{C-}_k \cup \mathbf{M}^{C,N}_{kn} \setminus \{M^M_k\}$
    $g^4_{kmm'} \leftarrow 1 \ \ \forall m \in \mathbf{M}^{P,N}_{kn}, m' \in \mathbf{M}^{C,N}_{kn}$
EndIf
```

Table B.11: Block 8-3

```
If $\sum_{i \in \mathbf{I}^+_k} \sum_{j \in \mathbf{J}_i} X_{ij(n-\bar{\tau}_{ij})} = 0$
        and $\sum_{i \in \mathbf{I}^-_k} \sum_{j \in \mathbf{J}_i} X_{ijn} = 0$
    If $\exists j \in \{j | X^S_{jkn} - X^S_{jk(n+1)} = 1\}$
        $M^M_k \leftarrow M^M_k + 1$
        $\mathbf{M}^P_k \leftarrow \mathbf{M}^P_k \cup \{M^M_k\}$
        $\mathbf{M}^{P,N}_{kn} \leftarrow \mathbf{M}^{P,N}_{kn} \cup \{M^M_k\}$
        Loop $j \in \{j | X^S_{jk(n+1)} = 1\}$
            $M^U_j \leftarrow M^U_j + 1$
            $g^2_{jk(m=M^M_k)} \leftarrow 1$
            $g^3_{jk(m=M^U_j)(m'=M^M_k)} \leftarrow 1$
        EndLoop
EndIf x2
```

Table B.12: Block 8-4

```
Loop $k \in \mathbf{K}^{LS}$
    Loop $j \in \{j | X^S_{jkn} - X^S_{jk(n+1)} = 1\}$
        Loop $k' \in \{k' | X^S_{jk'(n+1)} = 1\}$
            $g^5_{kk'(m=M^M_k)m'} \leftarrow 1 \ \ \forall m' \in \mathbf{M}^{P,N}_{k'n}$
EndLoop x3
```

Table B.13: Block 9

```
Loop $j \in \mathbf{J}^+_k \cup \mathbf{J}^S_k$
    Loop $j' \in \mathbf{J}^-_k \cup \mathbf{J}^S_k$
        If $W_{jj'kn} = 1$
            $g^6_{jj'(m=M^U_j)(m'=M^U_{j'})} \leftarrow 1$
        EndIf
EndLoop x2
```

Table B.14: Block 10

```
If $j \in \mathbf{J}^{CP}$ and $X_{ij(n-1)} = 0$
    $g^7_{ij(m=M^U_j)} \leftarrow 1$
EndIf
```

Table B.15: Block 11

```
Loop $j \in \mathbf{J}^{CO}$
    Loop $m \leq M_j^U$
        Loop $i \in \{i| \sum_{k,m'} f^4_{ijkmm'} > 0\}$
            Loop $i^{'} \in \mathbf{I}_j$
                If $\sum_{k,m'} f^4_{i'jk(m+1)m'} > 0$
                    $g^8_{ii'jm} \leftarrow 1$
                EndIf
EndLoop x4
```

C

TASK VALUE CALCULATION ALGORITHM

We define a set, $\mathbf{I}_i \in \{i^{'} \in \mathbf{I} | \exists k \in \mathbf{K}_i^P \cap \mathbf{K}_{i'}^N\}$, to denote the set of tasks that consume at least one material produced by task i. We also introduce $\omega_{ii'}$ to denote the value weight of task $i^{'}$ on task i (see Eq. C.1), where $\beta_j^{max,avg}$ is the average of the maximum capacities of the units that can process task i.

$$\omega_{ii'} = \sum_{k \in \mathbf{K}_i^P \cap \mathbf{K}_{i'}^N} |\rho_{ik}\rho_{i'k}| \frac{\beta_i^{max,avg}}{\beta_{i'}^{max,avg}} \frac{1}{|\mathbf{I}_k^N|} \quad \forall i, i^{'} \in \mathbf{I}_i \tag{C.1}$$

The algorithm back propagates the value of the succeeding tasks to the proceeding tasks until all Val_i converge. It utilizes algorithmic set, $\mathbf{I}^E$, to keep track of Val_i's that have not converged.

Table C.1: Value Propagation Algorithm

```
Initialize: $Val_i = 0 \ \ \forall i$ and $\mathbf{I}^E = \mathbf{I}$
While $\mathbf{I}^E \neq \emptyset$
    $Val_i \leftarrow \sum_{i' \in \mathbf{I}_i} \omega_{ii'} Val_{i'} + \sum_{k \in \mathbf{K}^F \cap \mathbf{K}_{i'}^P} \rho_{ik} \beta_i^{max,avg} \pi_k \ \ \forall i \in \mathbf{I}^E$
    $err_i \leftarrow \left| Val_i - \sum_{i' \in \mathbf{I}_i} \omega_{ii'} Val_{i'} - \sum_{k \in \mathbf{K}^F \cap \mathbf{K}_{i'}^P} \rho_{ik} \beta_i^{max,avg} \pi_k \right| \ \ \forall i$
    $\mathbf{I}^E \leftarrow \{i \in \mathbf{I} | err_i > 10^{-5}\}$
EndWhile
```

BIBLIOGRAPHY

Baldo, T.A., Santos, M.O., Almada-Lobo, B., Morabito, R., 2014. An optimization approach for the lot sizing and scheduling problem in the brewery industry. Computers & Industrial Engineering 72, 58 – 71. doi:10.1016/j.cie.2014.02.008.

Bassett, M.H., Pekny, J.F., Reklaitis, G.V., 1996. Decomposition techniques for the solution of large-scale scheduling problems. AIChE Journal 42, 3373–3387. doi:10.1002/aic.690421209.

Calfa, B.A., Agarwal, A., Grossmann, I.E., Wassick, J.M., 2013. Hybrid bilevel-lagrangean decomposition scheme for the integration of planning and scheduling of a network of batch plants. Industrial & Engineering Chemistry Research 52, 2152–2167. doi:10.1021/ie302788g.

Capon-Garcia, E., Ferrer-Nadal, S., Graells, M., Puigjaner, L., 2009. An extended formulation for the flexible short-term scheduling of multiproduct semicontinuous plants. Industrial & Engineering Chemistry Research 48, 2009–2019. doi:10.1021/ie800539f.

Castro, P., Barbosa-Povoa, A., Matos, H., 2001. An improved RTN continuous-time formulation for the short-term scheduling of multipurpose batch plants. Industrial & Engineering Chemistry Research 40, 2059–2068. doi:10.1021/ie000683r.

Castro, P.M., Barbosa-po, A.P., Matos, H.a., Novais, A.Q., 2004. Simple continuous-time formulation for short-term scheduling of batch and continuous processes. Industrial & Engineering Chemistry Research 43, 105–118. doi:10.1021/ie0302995.

Castro, P.M., Erdirik-Dogan, M., Grossmann, I.E., 2008. Simultaneous batching and scheduling of single stage batch plants with parallel units. Aiche Journal 54, 183–193. doi:10.1002/aic.11364.

Castro, P.M., Grossmann, I.E., 2005. New continuous-time MILP model for the short-term scheduling of multistage batch plants. Industrial & Engineering Chemistry Research 44, 9175–9190. doi:10.1021/ie0507301.

Castro, P.M., Hariunkoski, I., Grossmann, I.E., 2009a. Optimal short-term scheduling of large-scale multistage batch plants. Industrial & Engineering Chemistry Research 48, 11002–11016. doi:10.1021/ie900734x.

Castro, P.M., Westerlund, J., Forssell, S., 2009b. Scheduling of a continuous plant with recycling of byproducts: A case study from a tissue paper mill. Computers & Chemical Engineering 33, 347 – 358. doi:10.1016/j.compchemeng.2008.10.004.

Cerda, J., Henning, G.P., Grossmann, I.E., 1997. A mixed-integer linear programming model for short-term scheduling of single-stage multiproduct batch plants with parallel lines. Industrial & Engineering Chemistry Research 36, 1695–1707. doi:10.1021/ie9605490.

Chen, C.L., Liu, C.L., Feng, X.D., Shao, H.H., 2002. Optimal short-term scheduling of multiproduct single-stage batch plants with parallel lines. Industrial & Engineering Chemistry Research 41, 1249–1260. doi:10.1021/ie010465d.

Dessouky, M.M., Kijowski, B.A., 1997. Production scheduling of single-stage multi-product batch chemical processes with fixed batch sizes. Iie Transactions 29, 399–408.

Ferrer-Nadal, S., Capon-Garcia, E., Mendez, C.A., Puigjaner, L., 2008. Material Transfer Operations in Batch Scheduling. A Critical Modeling Issue. Industrial & Engineering Chemistry Research 47, 7721–7732. doi:10.1021/ie800075u.

Ferris, M.C., Maravelias, C.T., Sundaramoorthy, A., 2009. Simultaneous batching and scheduling using dynamic decomposition on a grid. Informs Journal on Computing 21, 398–410. doi:10.1287/ijoc.1090.0339.

Floudas, C.A., Lin, X.X., 2004. Continuous-time versus discrete-time approaches for scheduling of chemical processes: a review. Computers & Chemical Engineering 28, 2109–2129. doi:10.1016/j.compchemeng.2004.05.002.

Framinan, J.M., Ruiz, R., 2012. Guidelines for the deployment and implementation of manufacturing scheduling systems. International Journal of Production Research 50, 1799.

Fumero, Y., Corsano, G., Montagna, J.M., 2014. Simultaneous Batching and Scheduling of Batch Plants That Operate in a Campaign-Mode, Considering Nonidentical Parallel Units and Sequence-Dependent Changeovers. Industrial & Engineering Chemistry Research 53, 17059–17074. doi:10.1021/ie500454x.

Georgiadis, G.P., Elekidis, A.P., Georgiadis, M.C., 2019. Optimization-based scheduling for the process industries: From theory to real-life industrial applications. Processes 7, 438. doi:10.3390/pr7070438.

Georgiadis, G.P., Ziogou, C., Kopanos, G., Garcia, M., Cabo, D., Lopez, M., Georgiadis, M.C., 2018. Production scheduling of multi-stage, multi-product food process industries, in: Friedl, A., Klemeš, J.J., Radl, S., Varbanov, P.S., Wallek, T. (Eds.), 28th European Symposium on Computer Aided Process Engineering. Elsevier. volume 43 of *Computer Aided Chemical Engineering*, pp. 1075 – 1080. doi:10.1016/B978-0-444-64235-6.50188-1.

Giannelos, N., Georgiadis, M., 2001. Scheduling of cutting-stock processes on multiple parallel machines. Chemical Engineering Research and Design 79, 747 – 753. doi:10.1205/026387601753192064. distillation and Absorption.

Giannelos, N.F., Georgiadis, M.C., 2002. A simple new continuous-time formulation for short-term scheduling of multipurpose batch processes. Industrial & Engineering Chemistry Research 41, 2178–2184. doi:10.1021/ie010399f.

Gimenez, D.M., Henning, G.P., Maravelias, C.T., 2009. A novel network-based continuous-time representation for process scheduling: Part I. Main concepts and mathematical formulation. Computers & Chemical Engineering 33, 1511–1528. doi:10.1016/j.compchemeng.2009.03.007.

Gupta, D., Maravelias, C.T., 2016. On deterministic online scheduling: Major considerations, paradoxes and remedies. Computers & Chemical Engineering 94, 312–330. doi:10.1016/j.compchemeng.2016.08.006.

Gupta, D., Maravelias, C.T., 2017. A general state-space formulation for online scheduling. Processes 5. doi:10.3390/pr5040069.

Gupta, D., Maravelias, C.T., Wassick, J.M., 2016. From rescheduling to online scheduling. Chemical Engineering Research & Design 116, 83–97. doi:10.1016/j.cherd.2016.10.035.

Gupta, S., Karimi, I.A., 2003. An improved MILP formulation for scheduling multiproduct, multistage batch plants. Industrial & Engineering Chemistry Research 42, 2365–2380. doi:10.1021/ie020180g.

Harjunkoski, I., 2016. Deploying scheduling solutions in an industrial environment. Computers & Chemical Engineering 91, 127 – 135. doi:10.1016/j.compchemeng.2016.03.029.

Harjunkoski, I., Grossmann, I.E., 2002. Decomposition techniques for multistage scheduling problems using mixed-integer and constraint programming methods. Computers & Chemical Engineering 26, 1533–1552. doi:10.1016/s0098-1354(02)00100-x.

Harjunkoski, I., Maravelias, C.T., Bongers, P., Castro, P.M., Engell, S., Grossmann, I.E., Hooker, J., Mendez, C., Sand, G., Wassick, J., 2014. Scope for industrial applications of production scheduling models and solution methods. Computers & Chemical Engineering 62, 161–193. doi:10.1016/j.compchemeng.2013.12.001.

Henning, G.P., 2009. Production scheduling in the process industries: Current trends, emerging challenges and opportunities. Computer Aided Chemical Engineering 27, 23 – 28. doi:10.1016/S1570-7946(09)70224-X.

Hui, C.W., Gupta, A., Van Der Meulen, H.A.J., 2000. A novel MILP formulation for short-term scheduling of multi-stage multi-product batch plants with sequence-dependent constraints. Computers and Chemical Engineering 24, 2705–2717. doi:10.1016/S0098-1354(00)00623-2.

Ierapetritou, M.G., Floudas, C.A., 1998. Effective continuous-time formulation for short-term scheduling. 1. Multipurpose batch processes. Industrial & Engineering Chemistry Research 37, 4341–4359. doi:10.1021/ie970927g.

Jain, V., Grossmann, I.E., 2001. Algorithms for hybrid milp/cp models for a class of optimization problems. Informs Journal on Computing 13, 258–276. doi:10.1287/ijoc.13.4.258.9733.

Janak, S.L., Floudas, C.A., Kallrath, J., Vormbrock, N., 2006. Production scheduling of a large-scale industrial batch plant. I. Short-term and medium-term scheduling. Industrial & Engineering Chemistry Research 45, 8234–8252. doi:10.1021/ie0600588.

Kallrath, J., 2002. Planning and scheduling in the process industry. Or Spectrum 24, 219–250.

Kelly, J.D., Mann, J.L., 2003. Crude oil blend scheduling optimization: an application with multimillion dollar benefits - Part 1 - The ability to schedule the crude oil blendshop more effectively provides substantial downstream benefits. Hydrocarbon Processing 82, 47–+.

Kelly, J.D., Zyngier, D., 2008. Hierarchical decomposition heuristic for scheduling: Coordinated reasoning for decentralized and distributed decision-making problems. Computers & Chemical Engineering 32, 2684–2705. doi:10.1016/j.compchemeng.2007.08.007.

Kim, M., Jung, J.H., Lee, I.B., 1996. Optimal scheduling of multiproduct batch processes for various intermediate storage policies. Industrial & Engineering Chemistry Research 35, 4058–4066. doi:10.1021/ie9601817.

Kim, S.B., Lee, H.K., Lee, I.B., Lee, E.S., Lee, B., 2000. Scheduling of non-sequential multipurpose batch processes under finite intermediate storage policy. Computers & Chemical Engineering 24, 1603–1610. doi:10.1016/s0098-1354(00)00548-2.

Kim, Y.J., Kim, M.H., Jung, J.H., 2002. Optimal scheduling of the single line multi-purpose batch process with re-circulation products. Journal of Chemical Engineering of Japan 35, 117–130. doi:10.1252/jcej.35.117.

Kondili, E., Pantelides, C.C., Sargent, R.W.H., 1993. A general algorithm for short-term scheduling of batch-operations – I. MILP formulation. Computers & Chemical Engineering 17, 211–227. doi:10.1016/0098-1354(93)80015-f.

Kopanos, G.M., Lainez, J.M., Puigjaner, L., 2009. An efficient mixed-integer linear programming scheduling framework for addressing sequence-dependent setup issues in batch plants. Industrial & Engineering Chemistry Research 48, 6346–6357. doi:10.1021/ie801127t.

Kopanos, G.M., Puigjaner, L., Georgiadis, M.C., 2011. Resource-constrained production planning in semicontinuous food industries. Computers & Chemical Engineering 35, 2929 – 2944. doi:10.1016/j.compchemeng.2011.04.012.

Ku, H.M., Karimi, I.A., 1988. Scheduling in Serial Multiproduct Batch Processes with Finite Interstage Storage - A Mixed Integer Linear Program Formulation. Industrial & Engineering Chemistry Research 27, 1840–1848. doi:10.1021/ie00082a017.

Lee, H., Gupta, D., Maravelias, C.T., 2019. Systematic generation of alternative production schedules to account for model simplifications and plant nervousness. AIChE Journal, submitted .

Lee, H., Maravelias, C.T., 2017a. Discrete-time mixed-integer programming models for short-term scheduling in multipurpose environments. Computers & Chemical Engineering 107, 171 – 183. doi:10.1016/j.compchemeng.2017.06.013. in honor of Professor Rafiqul Gani.

Lee, H., Maravelias, C.T., 2017b. Mixed-integer programming models for simultaneous batching and scheduling in multipurpose batch plants. Computers & Chemical Engineering 106, 621 – 644. doi:10.1016/j.compchemeng.2017.07.007.

Lee, H., Maravelias, C.T., 2018. Combining the advantages of discrete- and continuous-time scheduling models: Part 1. framework and mathematical formulations. Computers & Chemical Engineering 116, 176 – 190. doi:10.1016/j.compchemeng.2017.12.003.

Lee, H., Maravelias, C.T., 2019a. Combining the advantages of discrete- and continuous-time scheduling models: Part 2. systematic methods for determining model parameters. Computers & Chemical Engineering 128, 557 – 573. doi:10.1016/j.compchemeng.2018.10.020.

Lee, H., Maravelias, C.T., 2019b. Combining the advantages of discrete- and continuous-time scheduling models: Part 3. general algorithm. Computers & Chemical Engineering, submitted .

Lee, K.H., Park, H.I., Lee, I.B., 2001. A novel nonuniform discrete time formulation for short-term scheduling of batch and continuous processes. Industrial & Engineering Chemistry Research 40, 4902–4911. doi:10.1021/ie000513e.

Lin, X., Floudas, C.A., Modi, S., Juhasz, N.M., 2002. Continuous-time optimization approach for medium-range production scheduling of a multiproduct batch plant. Industrial & Engineering Chemistry Research 41, 3884–3906. doi:10.1021/ie011002a.

Liu, S., Yahia, A., Papageorgiou, L.G., 2014. Optimal production and maintenance planning of biopharmaceutical manufacturing under performance decay. Industrial & Engineering Chemistry Research 53, 17075–17091. doi:10.1021/ie5008807.

Liu, S.S., Pinto, J.M., Papageorgiou, L.G., 2010. Single-stage scheduling of multiproduct batch plants: An edible-oil deodorizer case study. Industrial & Engineering Chemistry Research 49, 8657–8669. doi:10.1021/ie1002137.

Liu, Y., Karimi, I.A., 2007. Scheduling multistage, multiproduct batch plants with nonidentical parallel units and unlimited intermediate storage. Chemical Engineering Science 62, 1549–1566. doi:10.1016/j.ces.2006.11.053.

Liu, Y., Karimi, I.A., 2008. Scheduling multistage batch plants with parallel units and no interstage storage. Computers & Chemical Engineering 32, 671–693. doi:10.1016/j.compchemeng.2007.02.002.

Maravelias, C.T., 2005. Mixed-time representation for state-task network models. Industrial & Engineering Chemistry Research 44, 9129–9145. doi:10.1021/ie0500117.

Maravelias, C.T., 2006. A decomposition framework for the scheduling of single- and multi-stage processes. Computers & Chemical Engineering 30, 407–420. doi:10.1016/j.compchemeng.2005.09.011.

Maravelias, C.T., 2012. General framework and modeling approach classification for chemical production scheduling. Aiche Journal 58, 1812–1828. doi:10.1002/aic.13801.

Maravelias, C.T., Grossmann, I.E., 2003a. Minimization of the makespan with a discrete-time state-task network formulation. Industrial & Engineering Chemistry Research 42, 6252–6257. doi:10.1021/ie034053b.

Maravelias, C.T., Grossmann, I.E., 2003b. New general continuous-time state-task network formulation for short-term scheduling of multipurpose batch plants. Industrial & Engineering Chemistry Research 42, 3056–3074. doi:10.1021/ie020923y.

Maravelias, C.T., Grossmann, I.E., 2004. A hybrid milp/cp decomposition approach for the continuous time scheduling of multipurpose batch plants. Computers & Chemical Engineering 28, 1921–1949. doi:10.1016/j.compchemeng.2004.03.016.

Maravelias, C.T., Papalamprou, K., 2009. Polyhedral results for discrete-time production planning mip formulations for continuous processes. Computers & Chemical Engineering 33, 1890–1904. doi:10.1016/j.compchemeng.2009.05.015.

Mendez, C., Cerda, J., 2003. Dynamic scheduling in multiproduct batch plants. Computers & Chemical Engineering 27, 1247–1259. doi:10.1016/S0098-1354(03)00050-4.

Mendez, C.A., Cerda, J., 2002. An efficient MILP continuous-time formulation for short-term scheduling of multiproduct continuous facilities. Computers & Chemical Engineering 26, 687–695. doi:10.1016/s0098-1354(01)00789-x.

Mendez, C.A., Cerda, 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. doi:10.1023/a:1021856229236.

Mendez, C.A., Cerda, 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. doi:10.1016/j.compchemeng.2006.02.008.

Mendez, C.A., Henning, G.P., Cerda, J., 2000. Optimal scheduling of batch plants satisfying multiple product orders with different due-dates. Computers & Chemical Engineering 24, 2223–2245. doi:10.1016/s0098-1354(00)00584-6.

Mendez, C.A., Henning, G.P., Cerda, J., 2001. An MILP continuous-time approach to short-term scheduling of resource-constrained multistage flowshop batch facilities. Computers & Chemical Engineering 25, 701–711. doi:10.1016/s0098-1354(01)00671-8.

Merchan, A.F., Lee, H., Maravelias, C.T., 2016. Discrete-time mixed-integer programming models and solution methods for production scheduling in multistage facilities. Computers & Chemical Engineering 94, 387 – 410. doi:10.1016/j.compchemeng.2016.04.034.

Merchan, A.F., Maravelias, C.T., 2016. Preprocessing and tightening methods for time-indexed MIP chemical production scheduling models. Computers & Chemical Engineering 84, 516–535. doi:10.1016/j.compchemeng.2015.10.003.

Merchan, A.F., Velez, S., Maravelias, C.T., 2013. Tightening methods for continuous-time mixed-integer programming models for chemical production scheduling. AIChE Journal 59, 4461–4467. doi:10.1002/aic.14249.

Mockus, L., Reklaitis, G.V., 1999. Continuous time representation approach to batch and continuous process scheduling. 1. MINLP formulation. Industrial & Engineering Chemistry Research 38, 197–203. doi:10.1021/ie970311r.

Nie, Y., Biegler, L.T., Villa, C.M., Wassick, J.M., 2015. Discrete time formulation for the integration of scheduling and dynamic optimization. Industrial & Engineering Chemistry Research 54, 4303–4315. doi:10.1021/ie502960p.

Nie, Y., Biegler, L.T., Wassick, J.M., Villa, C.M., 2014. Extended discrete-time resource task network formulation for the reactive scheduling of a mixed batch/continuous process. Industrial & Engineering Chemistry Research 53, 17112–17123. doi:10.1021/ie500363p.

Novas, J.M., Henning, G.P., 2010. Reactive scheduling framework based on domain knowledge and constraint programming. Computers & Chemical Engineering 34, 2129–2148. doi:10.1016/j.compchemeng.2010.07.011.

Novas, J.M., Henning, G.P., 2012. A comprehensive constraint programming approach for the rolling horizon-based scheduling of automated wet-etch stations. Computers & Chemical Engineering 42, 189–205. doi:10.1016/j.compchemeng.2012.01.005.

Panek, S., Engell, S., Subbiah, S., Stursberg, O., 2008. Scheduling of multi-product batch plants based upon timed automata models. Computers & Chemical Engineering 32, 275–291. doi:10.1016/j.compchemeng.2007.07.009.

Pantelides, C.C., 1994. Unified frameworks for optimal process planning and scheduling, in: Proceedings on the second conference on foundations of computer aided operations, Cache Publications New York. pp. 253–274.

Papageorgiou, L.G., Pantelides, C.C., 1996a. Optimal campaign planning scheduling of multipurpose batch semicontinuous plants .1. mathematical formulation. Industrial & Engineering Chemistry Research 35, 488–509. doi:10.1021/ie950081l.

Papageorgiou, L.G., Pantelides, C.C., 1996b. Optimal campaign planning scheduling of multipurpose batch semicontinuous plants .2. A mathematical decomposition approach. Industrial & Engineering Chemistry Research 35, 510–529. doi:10.1021/ie950082d.

Pinedo, M.L., 2012. Scheduling: theory, algorithms, and systems. Springer Science & Business Media.

Pinto, J.M., Grossmann, I.E., 1995. A continous-time mixed-integer linear programming model for short-term scheduling of multistage batch plants. Industrial & Engineering Chemistry Research 34, 3037–3051. doi:10.1021/ie00048a015.

Pinto, J.M., Grossmann, I.E., 1998. Assignment and sequencing models for the scheduling of process systems. Annals of Operations Research 81, 433–466.

Prasad, P., Maravelias, C.T., 2008. Batch selection, assignment and sequencing in multi-stage multi-product processes. Computers & Chemical Engineering 32, 1106–1119. doi:10.1016/j.compchemeng.2007.06.012.

Reklaitis, G.V., 1996. Overview of scheduling and planning of batch process operations, in: Batch processing systems engineering. Springer, pp. 660–705.

Risbeck, M.J., Maravelias, C.T., Rawlings, J.B., 2019. Unification of closed-loop scheduling and control: state-space formulations, terminal constraints, and nominal theoretical properties. Computers & Chemical Engineering doi:10.1016/j.compchemeng.2019.06.021.

Roe, B., Papageorgiou, L.G., Shah, N., 2005. A hybrid milp/clp algorithm for multipurpose batch process scheduling. Computers & Chemical Engineering 29, 1277–1291. doi:10.1016/j.compchemeng.2005.02.024.

Sahinidis, N.V., Grossmann, I.E., 1991. Reformulation of multiperiod MILP models for planning and scheduling of chemical processes. Computers & Chemical Engineering 15, 255–272. doi:10.1016/0098-1354(91)85012-j.

Sanmarti, E., Huercio, A., Espuna, A., Puigjaner, L., 1996. A combined scheduling reactive scheduling strategy to minimize the effect of process operations uncertainty in batch plants. Computers & Chemical Engineering 20, S1263–S1268. doi:10.1016/0098-1354(96)00218-9.

Schilling, G., Pantelides, C.C., 1996. A simple continuous-time process scheduling formulation and a novel solution algorithm. Computers & Chemical Engineering 20, S1221–S1226.

Shah, N., Pantelides, C.C., Sargent, R.W.H., 1993. A general algorithm for short-term scheduling of batch-operations – II. Computational issues. Computers & Chemical Engineering 17, 229–244. doi:10.1016/0098-1354(93)80016-g.

Shaik, M.A., Floudas, C.A., 2008. Unit-specific event-based continuous-time approach for short-term scheduling of batch plants using RTN framework. Computers and Chemical Engineering 32, 260–274. doi:10.1016/j.compchemeng.2007.05.007.

Shi, H., You, F., 2016. A computational framework and solution algorithms for two-stage adaptive robust scheduling of batch manufacturing processes under uncertainty. AIChE Journal 62, 687–703. doi:10.1002/aic.15067.

Stefansson, H., Sigmarsdottir, S., Jensson, P., Shah, N., 2011. Discrete and continuous time representations and mathematical models for large production scheduling problems: A case study from the pharmaceutical industry. European Journal of Operational Research 215, 383 – 392. doi:10.1016/j.ejor.2011.06.021.

Subrahmanyam, S., Kudva, G.K., Bassett, H.H., Pekny, J.F., 1996. Application of distributed computing to batch plant design and scheduling. AIChE Journal 42, 1648–1661. doi:10.1002/aic.690420617.

Subramanian, K., Maravelias, C.T., Rawlings, J.B., 2012. A state-space model for chemical production scheduling. Computers & Chemical Engineering 47, 97–110. doi:10.1016/j.compchemeng.2012.06.025.

Sundaramoorthy, A., Karimi, I.A., 2005. A simpler better slot-based continuous-time formulation for short-term scheduling in multipurpose batch plants. Chemical Engineering Science 60, 2679–2702. doi:10.1016/j.ces.2004.12.023.

Sundaramoorthy, A., Maravelias, C.T., 2008a. Modeling of storage in batching and scheduling of multistage processes. Industrial & Engineering Chemistry Research 47, 6648–6660. doi:10.1021/ie701737a.

Sundaramoorthy, A., Maravelias, C.T., 2008b. Simultaneous batching and scheduling in multistage multiproduct processes. Industrial & Engineering Chemistry Research 47, 1546–1555. doi:10.1021/ie070944y.

Sundaramoorthy, A., Maravelias, C.T., 2011a. A general framework for process scheduling. AIChE Journal 57, 695–710. doi:10.1002/aic.12300.

Sundaramoorthy, A., Maravelias, C.T., 2011b. Computational study of network-based mixed-integer programming approaches for chemical production scheduling. Industrial & Engineering Chemistry Research 50, 5023–5040. doi:10.1021/ie101419z.

Sundaramoorthy, A., Maravelias, C.T., Prasad, P., 2009. Scheduling of multistage batch processes under utility constraints. Industrial & Engineering Chemistry Research 48, 6050–6058. doi:10.1021/ie801386m.

Susarla, N., Li, J., Karimi, I.A., 2010. A novel approach to scheduling multipurpose batch plants using unit-slots. AIChE Journal 56, 1859–1879. doi:10.1002/aic.12120.

Velez, S., Maravelias, C.T., 2013a. A branch-and-bound algorithm for the solution of chemical production scheduling MIP models using parallel computing. Computers & Chemical Engineering 55, 28–39. doi:10.1016/j.compchemeng.2013.03.030.

Velez, S., Maravelias, C.T., 2013b. Mixed-Integer Programming Model and Tightening Methods for Scheduling in General Chemical Production Environments. Industrial & Engineering Chemistry Research 52, 3407–3423. doi:10.1021/ie302741b.

Velez, S., Maravelias, C.T., 2013c. Multiple and nonuniform time grids in discrete-time MIP models for chemical production scheduling. Computers & Chemical Engineering 53, 70–85. doi:10.1016/j.compchemeng.2013.01.014.

Velez, S., Maravelias, C.T., 2013d. Reformulations and branching methods for mixed-integer programming chemical production scheduling models. Industrial & Engineering Chemistry Research 52, 3832–3841. doi:10.1021/ie303421h.

Velez, S., Maravelias, C.T., 2014. Advances in mixed-integer programming methods for chemical production scheduling. Annual Review of Chemical and Biomolecular Engineering 5, 97–121. doi:10.1146/annurev-chembioeng-060713-035859.

Velez, S., Maravelias, C.T., 2015. Theoretical framework for formulating MIP scheduling models with multiple and non-uniform discrete-time grids. Computers & Chemical Engineering 72, 233–254. doi:10.1016/j.compchemeng.2014.03.003.

Velez, S., Merchan, A.F., Maravelias, C.T., 2015. On the solution of large-scale mixed integer programming scheduling models. Chemical Engineering Science 136, 139–157. doi:10.1016/j.ces.2015.05.021.

Velez, S., Sundaramoorthy, A., Maravelias, C.T., 2013. Valid inequalities based on demand propagation for chemical production scheduling MIP models. AIChE Journal 59, 872–887. doi:10.1002/aic.14021.

Wassick, J.M., 2009. Enterprise-wide optimization in an integrated chemical complex. Computers & Chemical Engineering 33, 1950–1963. doi:10.1016/j.compchemeng.2009.06.002.

Wassick, J.M., Ferrio, J., 2011. Extending the resource task network for industrial applications. Computers & Chemical Engineering 35, 2124 – 2140. doi:10.1016/j.compchemeng.2011.01.010.

Westerlund, J., Hastbacka, M., Forssell, S., Westerlund, T., 2007. Mixed-time mixed-integer linear programming scheduling model. Industrial & Engineering Chemistry Research 46, 2781–2796. doi:10.1021/ie060991a.

Wu, D., Ierapetritou, M., 2004. Cyclic short-term scheduling of multiproduct batch plants using continuous-time representation. Computers & Chemical Engineering 28, 2271–2286. doi:10.1016/j.compchemeng.2004.04.002.

Wu, D., Ierapetritou, M.G., 2003. Decomposition approaches for the efficient solution of short-term scheduling problems. Computers & Chemical Engineering 27, 1261–1276. doi:10.1016/s0098-1354(03)00051-6.

Yee, K.L., Shah, N., 1998. Improving the efficiency of discrete time scheduling formulation. Computers & Chemical Engineering 22, S403–S410. doi:10.1016/s0098-1354(98)00081-7.

Zhuge, J., Ierapetritou, M.G., 2012. Integration of scheduling and control with closed loop implementation. Industrial & Engineering Chemistry Research 51, 8550–8565. doi:10.1021/ie3002364.

Zyngier, D., Kelly, J.D., 2009. Multiproduct Inventory Logistics Modeling in the Process Industries, in: Optimization and Logistics Challenges in the Enterprise. Springer. volume 30 of *Springer Series in Optimization and Its Applications*, pp. 61–95. doi:10.1007/978-0-387-88617-6_2.