Solution Methods for Chemical Production Scheduling

By

Andres F. Merchan Galindo

A dissertation submitted in partial fulfillment of

the requirements for the degree of

Doctor of Philosophy

(Chemical Engineering)

at the

UNIVERSITY OF WISCONSIN-MADISON

2015

Date of final oral examination: 10/26/2015

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

Christos T. Maravelias, Professor, Chemical and Biological Engineering

James B. Rawlings, Professor, Chemical and Biological Engineering

Ross E. Swaney, Associate Professor, Chemical and Biological Engineering

Victor M. Zavala, Assistant Professor, Chemical and Biological Engineering

Jeffrey T. Linderoth, Professor, Industrial and Systems Engineering

Abstract

Applications of scheduling to the chemical processing industry are ubiquitous. Detailed planning and programming of facilities constitute fundamental functions in the operation of chemical plants and therefore have been the subject of extensive research during the last decades. In particular, systematic methods that are applicable to a broad range of processing characteristics have been the focus of several works. Among these methods, optimization-based strategies have become increasingly popular due to their generality, flexibility and potential to produce the best solutions in terms of operating costs, customer satisfaction, and efficient use of resources.

Traditionally, most efforts in the scheduling literature have been directed towards the development of mathematical models that capture most of the common characteristics that appear in real-life problems, while remaining computationally tractable. However, the effective solution of large-scale scheduling models remains nontrivial. This is precisely the main objective of the present work: the development of solution methods for the main types of processing environments found in chemical plants. Our aim is to address existing opportunities to complete the spectrum of models and solution strategies available for general scheduling problems.

Our discussion is largely motivated by a new approach to the analysis of timing restrictions in scheduling problems. In particular, we extend the role that time has played as either a decision variable or an index in optimization models, and focus on its dimension as a limited resource that interacts with all the elements of the scheduling problem. Using this approach we can reinterpret existing concepts such as earliest start and latest finishing times, in order to derive new parameters and equations that improve the solution performance of existing models. Moreover, we fill in some modeling blanks in particular production environments.

First, we propose a family of algorithms that are suitable for maximization problems in network environments, whose goal is to preprocess the original data and derive parameters that can be used

to develop constraints that tighten the optimization models. In addition, we introduce the concept of variable start and finish times and derive expressions to relate them and connect them with original decision variables. By means of computational experiments we show the effectiveness of these methods in improving the solution process of optimization-based models for scheduling.

Second, we develop a new family of mathematical models for sequential environments, whose main characteristic is a discrete modeling of time. Almost all the existing models in the literature use a continuous representation of time. We discuss the advantages of discrete-time models and propose different solution methods to improve their computational performance. A computational study is included to test the improvements and compare with existing approaches.

Third, we extend methods based on reformulations and tightening constraints from discrete-time to continuous time models in network environments. We use specific characteristics of the latter to improve computational performance, testing our methods on several benchmark instances.

Finally we test the proposed methods on large-scale instances for which optimal solutions had not been found before or whose computational performances demanded long solution times. This way we show that our formulations and methods improve the tractability of industrial-scale instances. Optimal or near-optimal solutions are now accessible in reasonable time for many cases for which only suboptimal solutions from heuristics procedures or empirical methods were available.

Acknowledgements

The opportunity to work towards my doctorate degree at the University of Wisconsin-Madison has been an amazing life-changing experience. Working in this world-class nurturing academic environment has forever reshaped my intellectual and personal dimensions. This journey could have not been possible without the company of great people that have provided me with support and encouragement during the last six years.

I would like to express my gratitude to my advisor, Professor Christos T. Maravelias, for allowing me to be part of his research group. His patience, dedication, mentorship and support have been the key to a successful completion of this work.

I also want to thank the members of my Final Oral Examination Committee, Professors James B. Rawlings, Ross E. Swaney, Victor M. Zavala, and Jeffrey Linderoth, for taking the time to review this thesis and attend my defense.

My gratitude also goes to all the professors with whom I had the pleasure to work as a Teaching Assistant. Their commitment to teaching has been a great example for someone who has always loved the art of educating young students. Special mention to Professors James B. Rawlings, Daniel J. Klingenberg, and Rafael Chávez.

Thanks to my research group colleagues, with whom I had the pleasure to work and interact on a daily basis. In particular, I would like to express my appreciation to Carlos A. Henao, who has encouraged and guided me for several years, ever since our days at UPB. His valuable advice has been fundamental to my academic and personal success.

This experience has also rewarded me with countless colleagues, most of whom I have the honor to call friends; thanks for your company and support. Special mention to Kushal Sinha, Suyash Singh, Amritava Das, Fulya Akpinar, Paulina Rincón, Gretchen González, Murat Sen, Dennis Yang, and Josh

Hamilton. Thanks to my “family away from home,” Ana Porras, Laura Cuesta, Ángela Puerta, Jacqueline Mejía, Erick Jiménez, Francisco Moya, Pamela Camejo, Verónica Kramm, Jaime Yáñez, Diego Yáñez, Carol Díaz, and Carlos Coriano.

I would like to thank the Department of Chemical and Biological Engineering and its staff for their commitment to our success and permanent support. My gratitude also goes to the institutions that sponsor my studies: Wisconsin Alumni Research Foundation, National Science Foundation, and American Chemical Society (Petroleum Research Fund).

At the center of my life, God and family. The encouragement and lifelong example of my mother Emperatriz, the unconditional love and abnegation of my wife Catalina, and the innocence and playfulness of my son Benjamin constitute the fundamental pillars on which I support my existence. My perennial love and gratitude goes to them for their support throughout this and all my adventures.

Andres F. Merchan G.

October, 2015

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1. Scheduling

Production scheduling is one of the main planning functions within the scope of process operations. It can be defined as a decision-making process in which a set of limited shared resources is allocated to competing production tasks over time[1]. Although originally regarded as a feasibility problem, production scheduling is most often associated with the optimization of one or multiple objectives that are concerned with the adequate use of installed resources, customer satisfaction, and operating costs. The output of the scheduling function is the *production schedule*, a timetable that contains the list of start and end times as well as the main resources utilized by all tasks. Most often, this schedule is presented in the form of a Gantt chart, where the vertical axis lists the shared resources, the horizontal axis represents time and the horizontal bars illustrate assignment to resources and timing of tasks. Figure 1.1 shows the main elements of the scheduling problem.

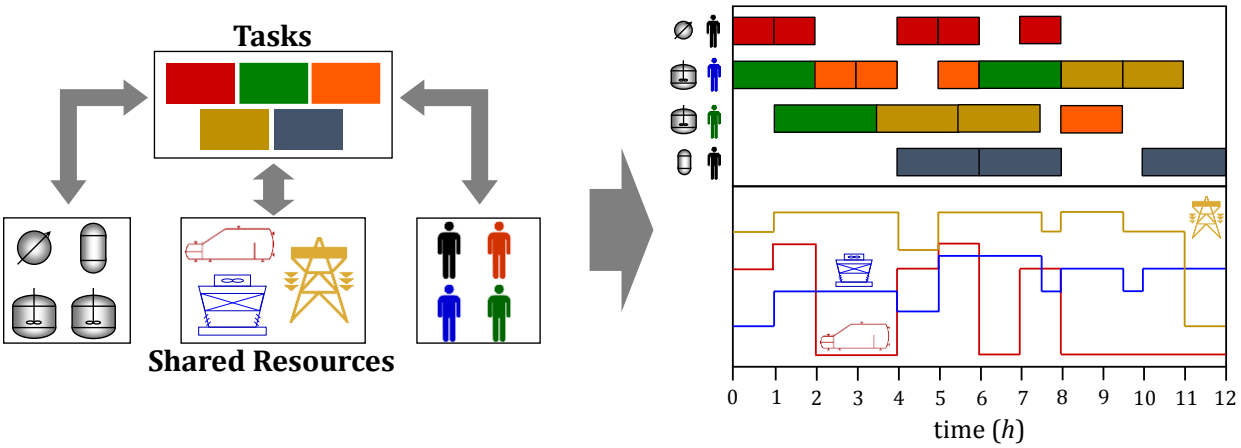


Figure 1.1. Scheduling problem
Assignment to limited resources and timing of different production tasks. The Gantt chart on the bottom represents the production schedule.

Studies on scheduling of projects and manufacturing processes appeared as early as the first decade of the twentieth century. However, the first systematic approach to find the problem solution

was first developed in the early 1950s[2]. Later in that decade, two of the most famous procedures to solve scheduling problems in project management, the critical path method, CPM[3], and the program evaluation and review technique, PERT[4], were developed. Since then, scheduling concepts and methods have been used in fields as different as health management, transportation, sports and construction. Due to this diverse range of applications, different areas have contributed to the development of scheduling theory and applications. Initially it was considered a subarea of operations research, but nowadays it is part of disciplines as computer science, statistics, mathematics, finances, and engineering. The focus in this work is on applications and developments in terms of modeling and solution procedures that have appeared within the context of Process Systems Engineering (PSE) as a subarea of Chemical Engineering.

1.2. Chemical production scheduling

The scheduling function encounters applications in practically every type of process industry, ranging from large-scale oil operations, to medium-scale commodities production, to small-scale high-value pharmaceuticals manufacturing. It has been therefore an active research area within the PSE community for the last decades[5-11]. A diverse range of exact and approximate solution approaches have been proposed. Exact methods, in which we are interested in this study, are mostly based on optimization techniques from the general area of mathematical programming.

Early works on the application of mathematical programming to chemical production scheduling followed from the related literature on discrete manufacturing systems, by considering facilities that resembled the structure of jobs and machines in series or parallel studied in the manufacturing sector[12-14]. Later on, Pantelides and coworkers[15-18] proposed different frameworks and models to address problems in facilities with complex topology and various types of resources as commonly encountered in the chemical process industry. Since then, chemical facilities have been broadly classified according to process structure and restrictions on material handling into *sequential* and

network environments[6, 9, 10]. The next section provides a detailed discussion on production environments and the different modeling approaches that have driven research efforts in the area of optimization-based chemical production scheduling.

1.3. Production environments and modeling approaches

Maravelias[10] introduced a unifying notation and nomenclature for chemical process production. Under this paradigm, chemical processes can be classified into *production environments* that in turn are characterized by topological and material-handling restrictions. The main types of environments that can be encountered in chemical facilities are *sequential* and *network*. Sequential environments are characterized by the flow of batches throughout well-defined stages. The number and size of all batches is defined by properly dividing the customer orders, in what constitutes the *batching* problem, which might be solved before or simultaneously with the scheduling problem. Every batch is required to preserve its identity, be trackable and follow a specific route; mixing of multiple batches and splitting of a single batch into multiple batches are not allowed. In network environments, more complex flow patterns are allowed. Stages cannot be defined, batch mixing and splitting are permitted, and recycling of materials may be present. Maravelias[10] also includes in this classification the *hybrid* environments, which includes different network and sequential subsystems, as different restrictions for material handling in each subsystem are enforced.

Several models have been proposed for both environments in the last decades in order to address common features such as changeovers, inventory cost and restrictions, utility requirements, and maintenance tasks, among others. In terms of modeling approaches, sequential environments typically use *batch-based* formulations, whereas network processes employ *material-based* models[10].

Batch-based formulations for sequential environments have attracted the interest of several researchers in the PSE community since the late 1970s, and several surveys are available[5, 6, 8, 9, 19]. The vast majority of these works have been devoted to develop continuous-time models and improve

their computational efficiency. Two major approaches have been adopted: (i) precedence-based modeling in which time is only a variable and sequencing is enforced through explicit binary variables[20-23], and (ii) time-grid-based model in which time is both a variable and an index, and the sequencing is enforced by assigning one batch to a single time interval with no extra variables required[24-26]. In turn, each approach can be subdivided in terms of global or local precedence for the former, and global or unit-specific time grid for the latter. The main advantage of the precedence-based approach is that no extra assumptions are required to model time. Its main disadvantage is that it can lead to relatively large model sizes, since the sequencing variables and constraints increase quadratically with the number batches. On the other hand, the main benefit of the time-grid-based approach is a relatively small model size, but its main drawback is the iterative procedure it requires to determine the number of time points to include in the grid.

Material-based formulations for network production environments were first introduced in the early 1990s by Pantelides and coworkers[15-17] to model chemical production facilities in which the mixing or splitting of fluid materials occurs naturally. Network processing can be distinguished from sequential processing because (i) the identity of the batches does not need to be preserved, (ii) the concept of stage is not used, since products can be produced following multiple routes, (iii) tasks are defined in terms of single/multiple input and output and conversion coefficients which define the production recipe, and (iv) recycle streams might be present.

In terms of mathematical modeling, network environments are represented in terms of the tasks that define the production recipe. Two abstract representations have extensively been used in the PSE literature, the State-Task Network (STN)[15], and the Resource-Task Network (RTN)[17]. Although both the STN and RTN represent the production facility as a network, the main difference between them is that the former only consider materials as resource nodes (units and other resources are implicitly mapped), while the latter also includes processing units and other limited resources as nodes of the network. Both the STN and RTN representations allow for either continuous or discrete

modeling of time. The term material-based derives from the fact that, in general, network processing relies on material balances to enforce sequencing constraints.

Several models have been proposed using the STN and RTN representations. Similarly to models for sequential environments, most of the existing models for network processing use continuous modeling of time and can be classified according to grid type. Some formulations use a common time grid across all the units[18, 27-32], whereas others employ unit-specific grids[33-36].

Although formulations are relatively small when a continuous-time representation is used, these models are not as tight as their discrete-time counterparts. In addition, they are less intuitive to model common features and require extra variables and constraints to account for common characteristics. In the worst case, continuous-time models might introduce nonlinearities in the model and therefore the linear-based methodologies are no longer valid or very limited.

1.4. An overview on solution methods

The main focus of optimization-based scheduling in the PSE literature has been on the development of mixed-integer programming (MIP) models to address different classes of problems, rather than the development of solution methods[9]. Historically, the development of methods has closely followed the modeling approach classification based on either batches or materials. Extensions to include commonly found features have been developed to account for storage policies[37, 38], utility constraints[21], simultaneous batching and scheduling[39-41], including problems with storage[42], utility[43], and other constraints[44].

Recently, research in PSE has focused on the solvability of the chemical production scheduling models, including (1) the study of the structure of MIP modes[45-47]; (2) decomposition algorithms[22, 48, 49]; (3) tightening methods[35, 50]; (4) reformulations[51, 52]; (5) algorithms that harness parallel computational resources[53-55]; and hybrid methods[56-58]. Of particular interest for the methods discussed in this thesis are the tightening methods of Maravelias and co-workers[59, 60], where

instance-specific demand information is used to generate tightening constraints for makespan or cost minimization problems subject to demand satisfaction constraints.

Solution methods have been also developed for specific environments. In particular, various solution methods are proposed in the literature to reduce the computational complexity of batch-based models in sequential environments. Pinto and Grossmann[24] introduced preordering constraints and applied Benders decomposition to their slot-based model to solve large-scale instances. Harjunkoski and Grossmann[22] decomposed the problem into two levels: higher level MIP subproblem for the assignment decisions, and lower level constraint programming (CP) subproblem for sequencing. These subproblems are solved iteratively by adding integer cuts whenever infeasibilities occur. Another decomposition method proposed by Maravelias[58] also decomposes the problem into assignment and sequencing subproblems, but special-purpose sequencing algorithms enabled more efficient sequencing in the second level. Castro et al.[61] introduced a decomposition algorithm for large-scale problems, in which only one or two orders are scheduled iteratively at a time, until the schedule of the complete set of orders is obtained. For the objective of makespan minimization specifically, Castro and Grossmann[26] extended the iterative approach originally proposed by Maravelias and Grossmann[62]. In this approach, the initial time horizon of the problem is estimated and then increased by a fixed value at each iteration, until a feasible solution is obtained.

1.5. Thesis scope

The general goal of this thesis is to develop solution methodologies that improve computational efficiency and results for scheduling problems in different chemical production environments. Accordingly, we set the following specific objectives.

(i) To develop a preprocessing/tightening methodology that addresses maximization problems in network environments.

(ii) To derive a new family of discrete-time models for sequential environments.

(iii) To propose solution methods based on tightening constraints for sequential environments.

(iv) To evaluate specific methods based on reformulation and tightening constraints for continuous-time formulations in network environments.

The remainder of this thesis is organized in five chapters. Chapter 2 presents a new family of preprocessing algorithms that use information on time and inventory restrictions to derive tightening constraints for maximization problems in network environments. Chapter 3 introduces new discrete-time models and tightening methods for sequential environments. In chapter 4, reformulations and tightening constraints are applied to continuous-time models in network environments. Chapter 5 presents applications of the proposed solution methods to large-scale instances. Finally, chapter 6 closes the thesis with concluding remarks and future directions for research. Appendices are included to present additional equations and figures. Graphic representations and complete data, as well as results and statistics are available online as Supporting Information of the articles on which each chapter is based.

Each chapter includes its own notation at the end. We use uppercase italic letters for variables, uppercase bold letters for sets, lowercase italic letters for indices, and lowercase Greek letters for parameters.

Chapter 2

Preprocessing and tightening methods for maximization problems in network environments[1]

The goal of this chapter is to develop methods for the generation of strong valid inequalities for profit or production maximization problems for which the tightening methods developed for minimization problems[59] cannot be used or are ineffective. Thus, rather than generating constraints that enforce lower bounds on the number of times tasks are executed (or the total production of materials and tasks), the methods discussed in this chapter lead to the generation of constraints that set upper bounds on the number of batches. To this end, we use restrictions imposed by the availability of processing units and the initial inventories rather than restrictions imposed by demand.

The remaining of the chapter is organized as follows. Section 2.1 presents background material, including the models we use to test our methods and several motivating examples to introduce the algorithms we propose. Section 2.2 contains the details of each of the algorithms that comprise our methods, with application to small-scale instances for illustration. In section 2.3, we extend our ideas from parameter calculation to constraint propagation. In section 2.4 we discuss how the computational implementation was carried out and section 2.5 presents the results of a comprehensive computational study we close in section 2.6, with concluding remarks and future directions.

[1] This chapter is modified from Merchan and Maravelias[63]

2.1. Background

2.1.1. Problem definition

In its most general definition, a chemical production facility is a collection of units that transform a set of input materials into a set of added-value products. From the operating point of view, each unit executes a set of processing tasks that generate intermediate materials or final products by the chemical or physical transformation of raw or other intermediate materials. The fact that different tasks might share different resources requires the introduction of a time grid to keep track of the unit utilization and inventory levels for all the materials involved in the process. The concept of batch is then introduced to define a single execution of a task in a compatible unit. We study production environments in which the identity of a single batch does not need to be tracked throughout the process because batch splitting and mixing are allowed (i.e. network environments). For these facilities, the chemical production scheduling problem can be defined as the problem of determining the number, size, assignment to units, and timing of individual batches for particular tasks.

Since the scheduling problem focuses on tasks rather than unit operations, a traditional process flow diagram representation of the plant is not useful, and an abstract representation of the production facility is required. As mentioned in Chapter 1, two commonly used representations are the STN[15] and the RTN[17]. The former represents the production plant as a network in which the nodes define tasks and materials, the arcs represent streams for the flow of materials between tasks, and the units are implicitly mapped to compatible tasks. The latter explicitly includes the processing units as nodes in the network, and unify them with the materials under a common family of resources that can be used by tasks at different moments. Our methods are independent of the abstract representation and can be applied to either STN- or RTN-based optimization models. The present work uses the STN representation, for which the following sets are required to define the structure of the plant:

$\mathbf{I} = \{i: i \text{ is a task}\}$

$\mathbf{J} = \{j: j \text{ is a processing unit}\}$

$\mathbf{J}_i = \{j \in \mathbf{J}: \text{unit } j \text{ can process task } i\}$

$\mathbf{I}_j = \{i \in \mathbf{I}: \text{task } i \text{ can be processed in unit } j\}$

$\mathbf{K} = \{k: k \text{ is a material}\}$

$\mathbf{I}_k^+/\mathbf{I}_k^- = \{i \in \mathbf{I}: \text{task } i \text{ produces/consumes material } k\}$

$\mathbf{K}_i^+/\mathbf{K}_i^- = \{k \in \mathbf{K}: \text{material } k \text{ is produced/consumed by task } i\}$

The main assumptions we make for our STN-based formulations are (a) no preemption is allowed and (b) the problem data are deterministic. The derivation of our methods does not depend on most of the commonly found features such as additional resources (utilities, labor), changeovers, shared storage vessels and variable processing times. Therefore, our methods are applicable to problems that include these features. In particular, they can be directly applied to problems with additional resources, while minor changes are required to include formulations of changeovers, shared storage and variable processing time.

A specific instance of a process network is defined using the following parameters:

$\beta_j^{max}/\beta_j^{min}$: Maximum/minimum batch size for unit $j \in \mathbf{J}$

$\bar{\tau}_{ij}$: Processing time for task $i \in \mathbf{I}$ in unit $j \in \mathbf{J}_i$

γ_k: Storage capacity for material $k \in \mathbf{K}$

ξ_{k0}: Initial inventory of material $k \in \mathbf{K}$

ρ_{ik}: Conversion coefficient for material $k \in \mathbf{K}$ produced ($\rho_{ik} > 0$) or consumed ($\rho_{ik} < 0$) by task $i \in \mathbf{I}$

The final element required for the abstract definition of the scheduling problem in network environments is the definition of a feasible schedule through three main requirements: (1) a unit can perform at most one task at a given time (i.e. unit utilization); (2) the batch sizes satisfy the processing unit capacities; and (3) material inventory satisfy nonnegativity and storage vessel capacity constraints[10]. The next section discusses how these requirements are mathematically modeled using MIP formulations.

2.1.2. Mathematical models

The development of a mathematical model for scheduling requires that we make a decision about the type of time representation to be used. In discrete-time models, the scheduling horizon, η, is divided into a set of equal time periods of length (step size) δ (such that $\eta/\delta \in \mathbb{Z}$); we also use a set of points $\mathbf{T} = \{t \in \mathbb{Z}: 0 \leq t \leq \eta/\delta\}$. In continuous-time formulations, the horizon is divided into N time intervals of unknown length that in turn define the set of time points $\mathbf{N} = \{n \in \mathbb{Z}: 0 \leq n \leq N\}$. Interval n runs between points $n-1$ and n, and since the position of the time point is not known a priori, it has to be included as a decision variable.

The decision variables that are common to both time representations are:

X_{ijt}/X_{ijn}: Binary. It is equal to one if task $i \in \mathbf{I}$ starts in unit $j \in \mathbf{J}_i$ at time $t \in \mathbf{T}/\ n \in \mathbf{N}$

BS_{ijt}/BS_{ijn}: Continuous nonnegative. Batch size of task $i \in \mathbf{I}$ that starts in unit $j \in \mathbf{J}_i$ at time $t \in \mathbf{T}/\ n \in \mathbf{N}$

S_{kt}/S_{kn}: Continuous nonnegative. Inventory level of material $k \in \mathbf{K}$ at time point $t \in \mathbf{T}/\ n \in \mathbf{N}$ with $S_{k0} = \xi_{k0}$

The main difference between discrete- and continuous-time models appears when the unit utilization constraint is enforced. In discrete-time models, it is typically modeled using the single-machine clique constraint[64], while continuous-time formulations require multiple types of

constraints to map the tasks onto the variable time grid. The batch size and material balance constraints remain essentially the same for both types of time representations. Next, we present the models that we use to test our methods. However, we note that the methods are applicable to all time-indexed material-based MIP models

Model SP&S

The discrete-time model we consider was developed by Shah et al.[16] and in this work it is referred to as model SP&S. It consists of single equations to model each of the three scheduling constraints discussed in section 2.1.1. Equation (2.1) is the clique constraint that represents unit utilization, equation (2.2) restricts the batch size, and equation (2.3) expresses the material balance and enforces the storage vessel capacity.

$$\sum_{i\in\mathbf{I}_j}\sum_{t'=t-\tau_{ij}+1}^{t} X_{ijt'} \leq 1 \quad \forall j \in \mathbf{J}, t \in \mathbf{T} \tag{2.1}$$

$$\beta_j^{min} X_{ijt} \leq BS_{ijt} \leq \beta_j^{max} X_{ijt} \quad \forall i \in \mathbf{I}, j \in \mathbf{J}_i, t \in \mathbf{T} \tag{2.2}$$

$$S_{kt} = S_{k(t-1)} + \sum_{i\in\mathbf{I}_k^+}\sum_{j\in\mathbf{J}_i} \rho_{ik} BS_{ij(t-\tau_{ij})} + \sum_{i\in\mathbf{I}_k^-}\sum_{j\in\mathbf{J}_i} \rho_{ik} BS_{ijt} + \xi_{kt} \leq \gamma_k \quad \forall k \in \mathbf{K}, t \in \mathbf{T} \tag{2.3}$$

Note that in equations (2.1) and (2.3), the processing time is expressed in terms of the number of time points, by using $\tau_{ij} = \lceil \bar{\tau}_{ij}/\delta \rceil$. Equation (2.3) also includes the parameter ξ_{kt} that represents the delivery ($\xi_{kt} > 0$) or demand ($\xi_{kt} < 0$) of material $k \in \mathbf{K}$ at time $t \in \mathbf{T}$.

Model S&K

The first continuous-time model we use was originally developed by Sundaramoorthy and Karimi[31] and we refer to it as model S&K. It relies on four separate *balances* to model the three scheduling constraints: resource and processing time balance to enforce unit utilization, material residing in process units to satisfy batch sizes, and material inventory in storage vessels to enforce

material balances and storage capacities. They also modified the definition of the set of tasks, by introducing an *idle task* to occupy a time slot when no task is assigned to a particular unit, although no batch sizing variables are explicitly assigned to such a task. As a result we redefine $\mathbf{I} = \{i: i \text{ is a task}\} \cup \{idle\}$. Full details of model S&K are given in Appendix A.

Model GH&M

The second continuous-time model we study was proposed by Gimenez et al.[32] It addresses most of the features that appear in chemical production scheduling, but we only use those parts of the model that make it comparable to model S&K. Specifically, we do not consider explicit modeling of material transfer and shared vessel utilization. This formulation is based on the *current state* of a processing unit: execution, storage or idle. Time balances are used to model unit utilization, while batch size and material balance constraints are analogous to those in model S&K. We refer to this formulation as model GH&M and present the complete list of equations in Appendix A.

2.1.3. Motivating examples

In this section we introduce the basic concepts that will be used to develop the new methods we propose by means of small-scale illustrations. Each instance highlights the benefit of specific algorithms that are then fully developed in section 2.2.

Time availability and inventory restrictions

The first concept we use is the *time window* to define the effective time a task could actually be carried out within the scheduling horizon. Although in principle the entire horizon is available for the execution of the tasks, limitations in the inventory of input and output materials of a given task shorten the time available for its execution. To calculate the time window of a specific task we use the concepts of *earliest start time* (EST) and *latest finish time* (LFT), which have been used before in the scheduling literature[61, 65]. The novelty of our approach is that we consider not only processing times, but also required amounts of materials associated with a task to calculate EST. For

convenience, we define the *shortest tail* (ST) of a task as the difference between the horizon and the LFT. Thus, tasks that exclusively produce final products could finish at the very end of the scheduling horizon (i.e. a zero shortest tail), whereas the rest of the tasks should finish earlier so the intermediates they produce have enough time to be converted into final products. The time window for a specific task is then calculated by subtracting the EST and ST values for that task from the total horizon.

Consider the network depicted in Figure 2.1. We present next the necessary conceptual steps to calculate the time window for a particular task. A formal discussion and generalization is presented in section 3.1. Let us calculate the EST for task T3, which is nonzero since the initial inventory for material S3 is zero. Note that at least 25 kg of S3 are required to start running T3. The earliest start time for T1 is zero, since S1 has enough inventory. We use data on processing time and unit capacity to calculate maximum amounts that could be produced at different times by T1. Figure 2.2 shows these calculations and establishes that the EST for T3 in U3 is 4 hours, which is the time required to produce at least 25 kg of S3. Note that if only data for processing time were considered, an incorrect value of 2 hours would be calculated. The shortest tail for T3 is calculated by considering the executions that lead to production of final product S6, and not those that only increase the inventory of intermediate material S5. Since S5 has to be consumed by task T4 in unit U4, for which the processing time is 2 hours, we conclude that, for a horizon $\eta = 12h$, the LFT for task T3 is 10 hours and therefore its ST is 2 hours. Consequently, the effective time window for task T3 is 6 hours ($= \eta - EST - ST = 12 - 4 - 2$). Figure 2.3 presents the results for time windows of all tasks depicted in Figure 2.1, with and without considering inventories of intermediates.

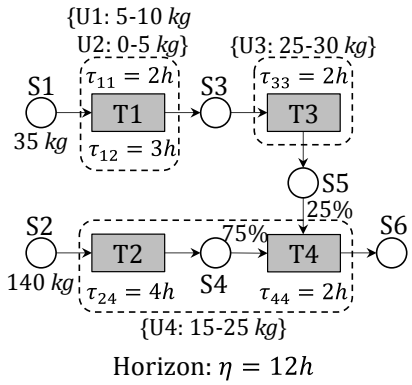


Figure 2.1. STN representation for the first illustrative example.
Unit-task compatibility and data for processing times, unit capacities, initial inventory, and conversion coefficients are provided.

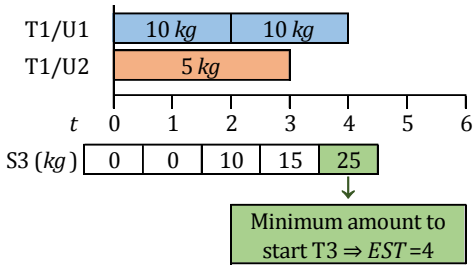


Figure 2.2. EST calculation for task T3.
The earliest start time for task T3 of the illustrative example is 4 h, which is the time it takes to produce 25 kg of material S3 to satisfy its minimum batch size.

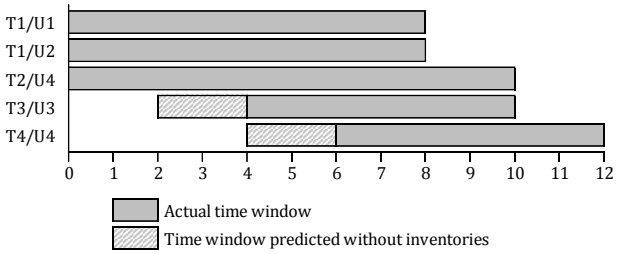


Figure 2.3. Time windows for all tasks in the first illustrative example.
If inventory of intermediates is not taken into account, longer and incorrect time windows are predicted.

Once the time windows have been calculated for each task, we propagate the information regarding initial inventories for raw materials and intermediates following the direction of the material flow (i.e. forward propagation). This propagation allows the calculation of the *maximum cumulative production of each material*. Once this value has been determined for all input materials of a task, the maximum cumulative amount that the task can process based on material inventories is calculated. This value can then be compared with the maximum amount the task could process

within its time window, based on the maximum capacity of the units in which it can be processed. The minimum of the last two values defines the *maximum cumulative production* for the specific task.

To illustrate these concepts we consider task T3 of the network in Figure 2.1. We fix the initial inventory of raw material S1 to 35 kg and begin the forward propagation. Based on its time window, T1 could process 50 kg of S1 ($maximum\ capacity \cdot \left\lfloor \frac{time\ window}{processing\ time} \right\rfloor$; from U1: $40 = 10 \cdot \lfloor 8/2 \rfloor$, and from U2: $10 = 5 \cdot \lfloor 8/3 \rfloor$) but there are only 35 kg of S1 available, so T1 is limited by inventory and its maximum cumulative production is 35 kg. This value can in turn be propagated to material S3, to define its maximum cumulative production. Based on the time window for T3, a maximum of 90 kg ($= 30 \cdot \lfloor 6/2 \rfloor$) can be processed, so we conclude that T3 is also inventory-limited and its maximum cumulative production is 35 kg.

An additional refinement can be made by analyzing the feasible batch sizes derived from the unit capacities. Task T3 is processed in unit U3, whose capacity is $[25, 30]$ kg. This means that one batch of T3 in U3 can produce a maximum of 30 kg and two batches produce at least 50 kg. There is no way to produce exactly 35 kg, so the maximum cumulative production of T3 is adjusted to 30 kg.

When we include information on available time and maximum cumulative production in the MIP formulations, the corresponding LP relaxations are improved. For instance, if the production of product S6 is maximized, the LP relaxations of the three models we consider (SP&S, S&K, GH&M) predict 60 kg of S6. However, when time and inventory restrictions are included in the model, the LP relaxations improve for models SP&S and GH&M. The former predicts a value of 50 kg, whereas the latter calculates a value of 57.14 kg. Note that the integer solution obtained when integrality is enforced is 50 kg, which coincides with the LP relaxation for model SP&S. Although in more complex networks the improved results do not match the integer solution, the introduction of these methods greatly enhances the efficiency of the solution procedures as shown in section 2.5.

Task dependence and feasible number of batches

Given a specific network, it is possible to identify groups of *dependent* tasks for which their number of batches are correlated. Being able to identify these groups and generate equations to represent the dependence between the numbers of batches of different tasks within a group is one of the key components of our methods.

Consider the process network shown in Figure 2.4. We can identify three different groups of tasks based on different types of interactions that render their number of batches dependent. First, tasks T1 and T2 can be grouped together because they *share unit* U1. Although we can identify separate time windows for each task, the dependence appears when the unique time window for unit U1 is considered. If we assume that S1 and S2 have large inventories at the beginning of the horizon, then the earliest start time for both T1 and T2 is zero. From the processing time data we calculate shortest tail values of $3h$ and $2h$ for T1 and T2 respectively. Consequently, the time window for T1 is $7h$ and for T2 is $8h$ and since the inventory of S1 and S2 is large enough, both tasks are time-limited. The maximum number of batches of T1 and T2 that can be processed independently ($\left\lfloor \frac{time\ window}{processing\ time} \right\rfloor$) are $N_{T1}^{max} = 3\ (= \lfloor 7/2 \rfloor)$ and $N_{T2}^{max} = 4\ (= \lfloor 8/2 \rfloor)$. However, since they share a common unit with a time window of $8h$ (using minimum values of earliest start time and shortest tail), only some combinations of number of batches are feasible, as shown in Figure 2.5.

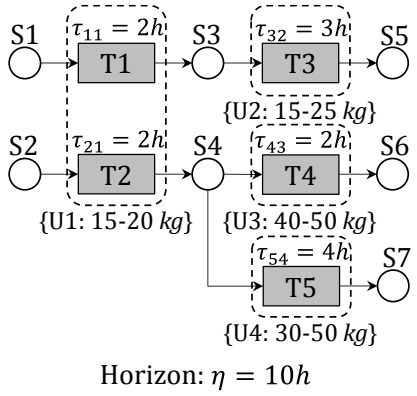


Figure 2.4. STN representation for second illustrative example.
Unit-task compatibility and data for processing times and unit capacities are provided.

Second, tasks T4 and T5 form a new group because they *share input material* S4. Let us assume that the maximum cumulative amount of S4 is 100 kg. Individual estimates for the maximum number of batches of T4 and T5 ($\left\lfloor \frac{Cumulative\ input}{Conversion \cdot Min\ Capacity} \right\rfloor$) result in $N_{T4}^{max} = 2\ (= \lfloor 100/(1 \cdot 40) \rfloor)$ and $N_{T5}^{max} = 3\ (= \lfloor 100/(1 \cdot 30) \rfloor)$. Once more, the shared resource results in reduced feasible combinations of batches, as presented in Figure 2.6.

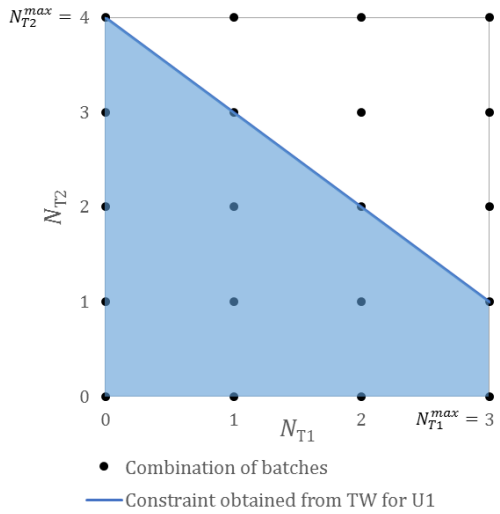


Figure 2.5. Group of dependent tasks that share a common processing unit.
If tasks T1 and T2 are considered independent then $(N_{T1}, N_{T2}) \in [0,3] \times [0,4]$. Since they share unit U1 we can write $2N_{T1} + 2N_{T2} \leq 8$ and $N_{T1} \leq 3$, based on the time window for the unit. Here $N_{T1} = \sum_t X_{T1,U1,t}$ and $N_{T2} = \sum_t X_{T2,U1,t}$

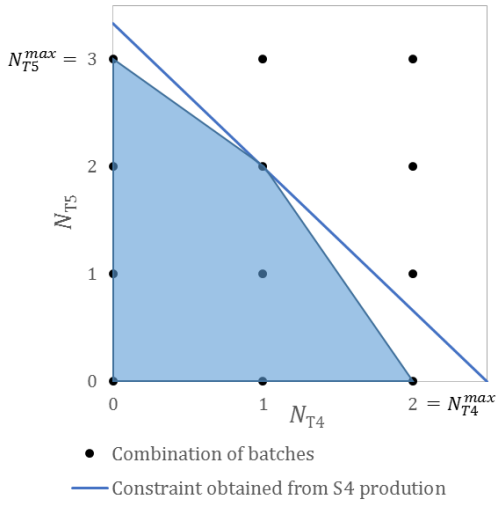


Figure 2.6. Group of dependent tasks that share a common input material.
If tasks T4 and T5 are considered independent then $(N_{T4}, N_{T5}) \in [0,2] \times [0,3]$. Since they share input material S4 we can first write $40N_{T4} + 30N_{T5} \leq 100$ based on cumulative production of the material. Moreover, we can

determine the convex hull of the feasible integer points to obtain $N_{T4} + N_{T5} \leq 3$ and $2N_{T4} + N_{T5} \leq 4$. Here $N_{T4} = \sum_t X_{T4,U3,t}$ and $N_{T5} = \sum_t X_{T5,U4,t}$

Third, tasks T3 and T4 constitute a less intuitive group, since they consume *materials produced by the same unit*. Intermediates S3 and S4 are both produced by unit U1, which induces a dependence between T3 and T4. The simple forward-propagation algorithm would suggest that the number of batches of T3 and T4 are not related, but since the cumulative production of S3 and S4 is affected by the production in unit U1, we have to propagate the available information on U1 to determine if the required amounts of input materials constitute a feasible production scenario. In this case, there is not a single closed-form constraint that can be used to determine the feasible set of points. As we discuss in section 3.2, a more sophisticated procedure is required to identify such a set. Figure 2.7 shows the feasible combinations of batches obtained when this dependence is taken into account.

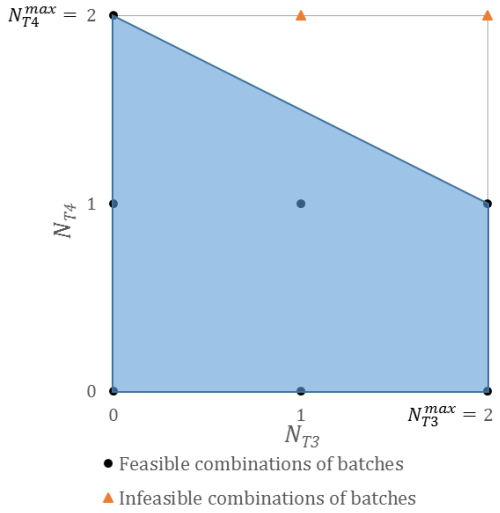


Figure 2.7. Group of dependent tasks that consume materials produced by a common unit.
If tasks T3 and T4 are considered independent then $(N_{T3}, N_{T4}) \in [0,2] \times [0,2]$. Since they consume materials S3 and S4 both produced by unit U1, there is an implicit dependence, that renders the combinations $(N_{T3}, N_{T4}) \in \{(1,2), (2,2)\}$ infeasible. The convex hull of the feasible integer points can be used to obtain $N_{T3} + 2N_{T4} \leq 4$ and $N_{T3} \leq 2$. Here $N_{T3} = \sum_t X_{T3,U2,t}$ and $N_{T4} = \sum_t X_{T4,U3,t}$

The analysis of the three types of groups described above leads to the generation of constraints describing the convex hull of feasible combinations of number of batches for the tasks within the group. The tightening effect is shown in Figures Figure 2.5 through Figure 2.7, in which some portions of the search space of the LP-relaxation are removed. For more complex instances, in which

combinations of time- and inventory-limited tasks are present, this tightening can be greatly beneficial as discussed in section 2.5.

2.2. Proposed Methods

The methods we propose are based on preprocessing instance data (processing time, initial inventory, conversion coefficients, and unit capacities) to obtain parameters that are used to generate tightening constraints. We discuss a sequence of algorithms that identify the initial grouping of tasks and calculate parameters that are propagated following the flow of materials in the network. These parameters are then used to write tightening constraints. In order to execute the proposed algorithms, we need to define new sets and parameters as follows.

$d \in \mathbf{D}$	Groups of dependent tasks
$\mathbf{I}_d$	Tasks that belong to group d
$\mathbf{J}_d$	Units that belong to group d
$h \in \mathbf{H}_d$	Equations derived from group d
ε_{ij}	Earliest start time for task $i \in \mathbf{I}$ in unit $j \in \mathbf{J}_i$
σ_{ij}	Shortest tail for task $i \in \mathbf{I}$ in unit $j \in \mathbf{J}_i$
θ_{ij}	Time window for task $i \in \mathbf{I}$ in unit $j \in \mathbf{J}_i$
μ_i	Maximum cumulative production of task $i \in \mathbf{I}$ within the given horizon
ω_k	Maximum cumulative production of material $k \in \mathbf{K}$ within the given horizon
φ_{idh}	Coefficient for task $i \in \mathbf{I}_d$ in equation $h \in \mathbf{H}_d$
φ_{dh}	Right-hand-side coefficient in equation $h \in \mathbf{H}_d$

The definition of time window is direct, based on the discussion presented before:

$$\theta_{ij} = \eta - \varepsilon_{ij} - \sigma_{ij} \quad \forall i \in \mathbf{I}, \forall j \in \mathbf{J}_i \tag{2.4}$$

We define four basic algorithms: (1) time window (TW) algorithm, (2) forward-propagation (FP) algorithm, (3) group identification (GI) algorithm, and (4) feasible region's convex hull (FR) algorithm. Each of these algorithms defines and calculates some of the parameters that we just

introduced. The flowchart in Figure 2.8 shows the sequence of calculations and how the different algorithms interact with each other. Subsequent subsections explain each algorithm in detail.

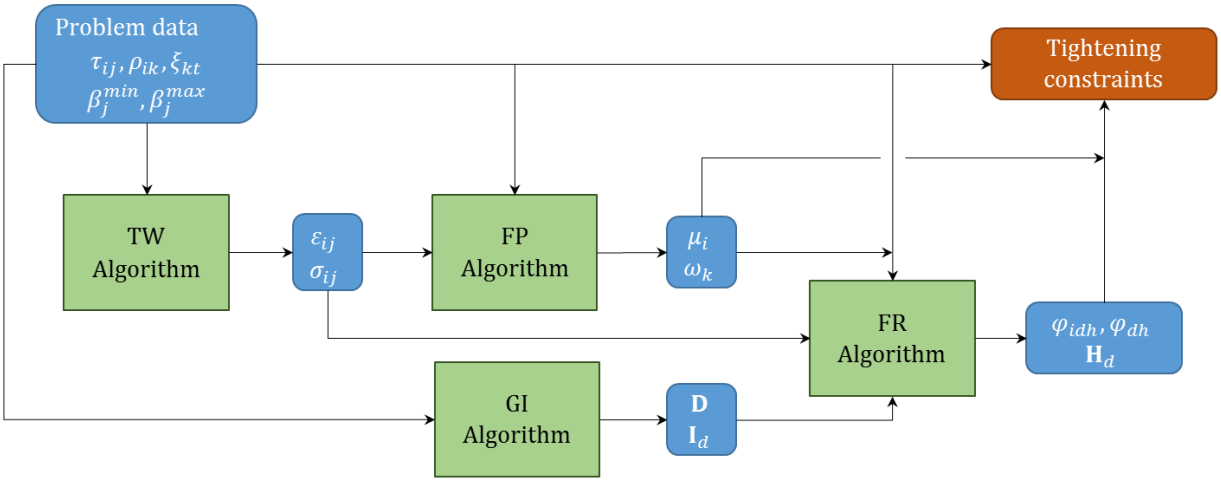
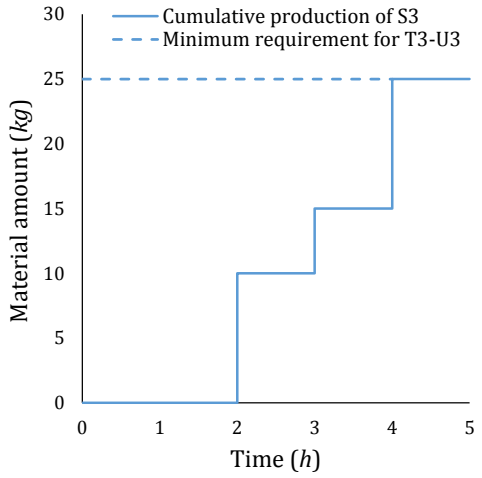


Figure 2.8. Flowchart for the proposed methods.
Flow of information, results of each algorithm and interactions among them are shown.

2.2.1. Time Window (TW) Algorithm

The first algorithm is divided into two steps; for every task in the network one step calculates the earliest start time (TW_EST) and the other computes the shortest tail (TW_ST). The TW_EST step calculates ε_{ij} by finding the time value for which the maximum amount of a specific input material, $k \in \mathbf{K}_i^-$, that can be produced up to that time exceeds the minimum amount required to execute one batch of task i. This is illustrated for task T3 of the motivating example presented before using Figure 2.9. The stair-shaped line in Figure 2.9a represents the production profile for material S3 if the tasks producing it used all available units at their maximum capacities. The dotted line is the minimum amount of material required for a single batch of task T3. The time at which these two lines intersect represents the earliest start time for T3 in unit U3.



(a)

$i = T3, j = U3, \mathbf{K}^{-}_{T3} = S3$
$\mathbf{I}^{+}_{S3} = \{T1\}, \mathbf{K}^{-}_{T1} = \{S1\}, \rho_{T1,S3} = 1, \rho_{T1,S1} = -1$

$\bar{t}$	$\psi^{E}_{S1,\bar{t}}$	$\psi^{E}_{S3,\bar{t}}$	$\chi^{E}_{T1,U1,\bar{t}}$	$\chi^{E}_{T1,U2,\bar{t}}$	$\phi^{E}_{T1,\bar{t}}$	$\chi^{E}_{T3,U3,\bar{t}}$
0	35	0	10	5	15	0
1	35	0	10	5	15	0
2	35	10	20	5	25	0
3	35	15	20	10	30	0
4	35	25	30	10	40	25

$\Rightarrow \varepsilon_{T3,U3} = \min\{\bar{t}: \chi^{E}_{T3,U3,\bar{t}} > 0\} = 4$

(b)

Figure 2.9. Calculations for the TW_EST algorithm.
(a) Cumulative production profile for material S3 and minimum input material requirement for task T3 in unit U3, for the illustrative example in Figure 1. (b) Calculation steps for the TW_EST step.

The TW_EST step uses the following parameters in order to keep track of the two separate quantities:

$\psi^{E}_{k\bar{t}}$ Maximum amount of material $k \in \mathbf{K}$ available at time $\bar{t}$

$\chi^{E}_{ij\bar{t}}$ Cumulative production for task $i \in \mathbf{I}$ in unit $j \in \mathbf{J}_i$ starting at or before time $\bar{t}$

$\phi^{E}_{i\bar{t}}$ Cumulative production for task $i \in \mathbf{I}$ in *any unit* starting at or before time $\bar{t}$

The TW_EST step is based on a predefined time grid $\bar{t} \in \overline{\mathbf{T}}$ over which it loops to determine the earliest time a task can start. In discrete-time models this grid coincides with the discretization introduced by the step size δ, i.e. $\overline{\mathbf{T}} = \mathbf{T}$. In continuous-time formulations, an auxiliary grid is defined using two steps: (1) all the processing times are rounded to a fixed number of decimal places, and (2) a step size equal to the greatest common factor of the processing times is introduced. Equation (2.5) defines $\psi^{E}_{k\bar{t}}$ as the total supply of material k plus the maximum amount that can be produced up to time $\bar{t}$. The second term is calculated as the minimum between two quantities: (1) the amount of material k that can be produced in all units that process tasks producing k up to time $\bar{t}$, and (2) the amount of material k produced by all tasks producing k up to time $\bar{t}$.

$$\psi^E_{k\bar{t}} = \sum_{0 \le \bar{t}' \le \bar{t}} \xi_{k\bar{t}'} + \sum_{i \in \mathbf{I}^+_k} \left[\rho_{ik} \min \left\{ \sum_{j \in \mathbf{J}_i} \chi^E_{ij(\bar{t}-\tau_{ij})} \, , \phi^E_{i(\bar{t}-\min_{j \in \mathbf{J}_i} \tau_{ij})} \right\} \right] \tag{2.5}$$

Equation (2.6) defines $\chi^E_{ij\bar{t}}$, which is initialized based on the amount of input material k available for task i (first term), but cannot increase by more than β^{max}_j every τ_{ij} time periods (second term). Equation (2.6) also expresses that if $\chi^E_{ij\bar{t}}$ is less than the minimum unit capacity, β^{min}_j, then no material can be processed by task i in unit j and therefore $\chi^E_{ij\bar{t}} = 0$.

$$\chi^E_{ij\bar{t}} = \min \left\{ \min_{k \in \mathbf{K}_i} \left(-\psi^E_{k\bar{t}} / \rho_{ik} \right), \chi^E_{ij(\bar{t}-\tau_{ij})} + \beta^{max}_j \right\}; \text{ if } \chi^E_{ij\bar{t}} < \beta^{min}_j \text{ then set } \chi^E_{ij\bar{t}} \leftarrow 0 \tag{2.6}$$

Next, equation (2.7) provides the value of $\phi^E_{i\bar{t}}$ based on the total amount of input material available. This value is at most the sum of the amounts that can be produced in all units that process task i.

$$\phi^E_{i\bar{t}} = \min \left\{ \min_{k \in \mathbf{K}_i} \left\{ -\psi^E_{k\bar{t}} / \rho_{ik} \right\}, \sum_{j \in \mathbf{J}_i} \chi^E_{ij\bar{t}} \right\} \tag{2.7}$$

Finally, the earliest start time is the first time point for which $\chi^E_{ij\bar{t}}$ is nonzero as expressed through equation (2.8).

$$\varepsilon_{ij} = \min \left\{ \bar{t} : \chi^E_{ij\bar{t}} > 0 \right\} \tag{2.8}$$

Figure 2.10 presents the outline of the TW_EST step. We use the set $\mathbf{T}^{Exp}$ to define the set of time points that need to be explored, according to the time grid used. Figure 2.11 shows the procedure of the TW_EST step when applied to the motivating example introduced before.

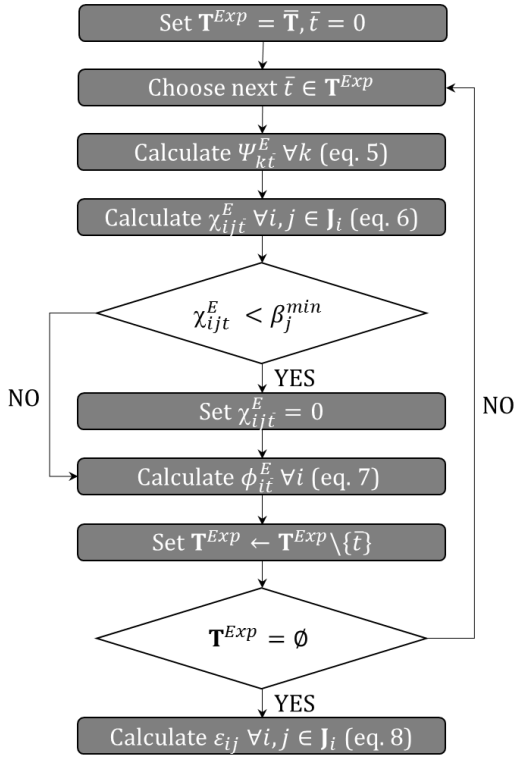


Figure 2.10. Flowchart for the TW_EST step in TW algorithm.

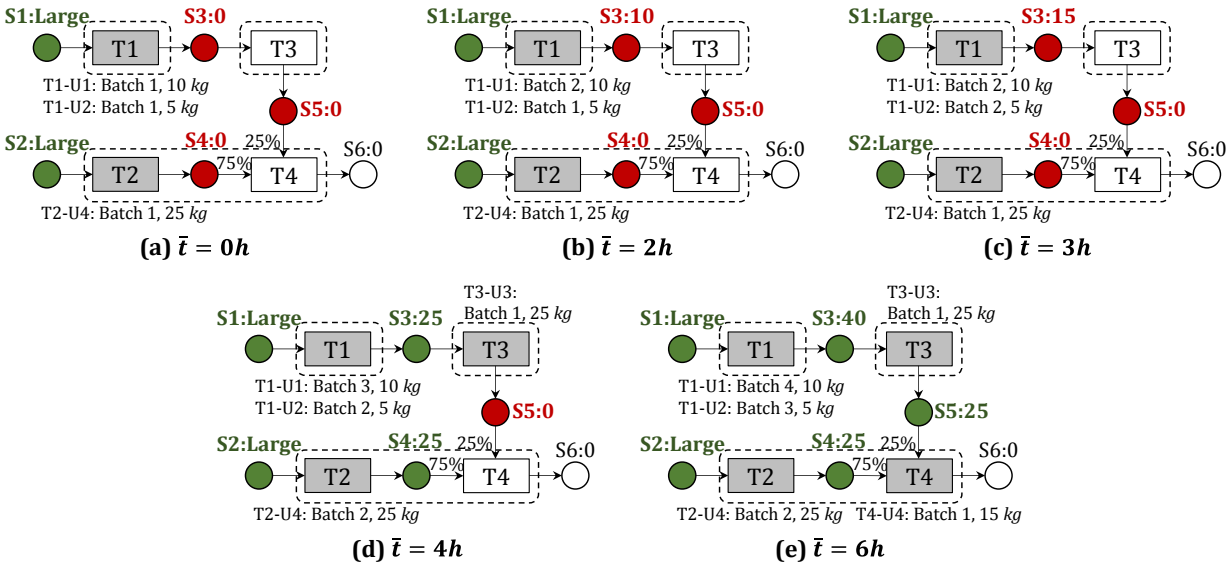


Figure 2.11. Calculation procedure for the TW_EST step for the process network in Figure 1.
Evolution of cumulative production at times when batches of different tasks start/finish. A green material indicates its cumulative amount is enough to start tasks that consume it. A red material indicates its cumulative amount is below the minimum required to start tasks consuming it. A gray task indicates it is ready to process its first batch. A white task is still waiting for its input materials to reach minimum levels to start. The value of ε_{ij} corresponds to the first $\bar{t}$ where the task is gray. Minimum requirements for tasks are T3: 25 kg of S3, T4: 11.25 kg of S4 and 3.75 kg of S5. From Figure (a) we obtain $\varepsilon_{T1,U1} = \varepsilon_{T1,U2} = \varepsilon_{T2,U4} = 0$. From Figures (d) an (e) respectively we get $\varepsilon_{T3,U3} = 4$ and $\varepsilon_{T4,U4} = 6$

The TW_ST step is simpler than the TW_EST step and uses a modification of the shortest path algorithm[66] as described by Velez and Maravelias[60]. It uses the following sets and parameters:

Sets

$\mathbf{K}^F = \{k \in \mathbf{K}: k \text{ is a raw material}\}$

$\mathbf{K}^P = \{k \in \mathbf{K}: k \text{ is a final product}\}$

Parameters

ϕ_k^S Minimum time required to process material $k \in \mathbf{K}$ so that it leads to production of final products.

λ_{ij}^S Latest finish time for task $i \in \mathbf{I}$ in unit $j \in \mathbf{J}_i$ so that the intermediates it produces have enough time to be transformed into final products.

The TW_ST step starts by assigning large values of ϕ_k^S to raw and intermediate materials and zero values to materials in $\mathbf{K}^P$. It also initializes λ_{ij}^S with large values. The set $\mathbf{K}^{Exp}$ is used to keep track of the materials that need to be evaluated. Equation (2.9) is used to calculate λ_{ij}^Sin terms of ϕ_k^S, and equation (2.10) to calculate ϕ_k^S based on tasks that consume material k. Finally, equation (2.11) is used to calculate the shortest tail σ_{ij}.

$$\lambda_{ij}^S = \min_{k \in \mathbf{K}_i^+} \phi_k^S \quad \forall i \in \mathbf{I}_k^+, j \in \mathbf{J}_i \tag{2.9}$$

$$\phi_k^S = \min_{i \in \mathbf{I}_j} \{\phi_k^S, \lambda_{ij}^S + \bar{\tau}_{ij}\} \quad \forall k \in \mathbf{K}_i^- \tag{2.10}$$

$$\sigma_{ij} = \eta - \lambda_{ij}^S \quad \forall i \in \mathbf{I}_k^+, j \in \mathbf{J}_i \tag{2.11}$$

Figure 12 presents the calculations of the TW_ST step. Figure 13 shows the application of the TW_ST step to the motivating example in section 2.3.1.

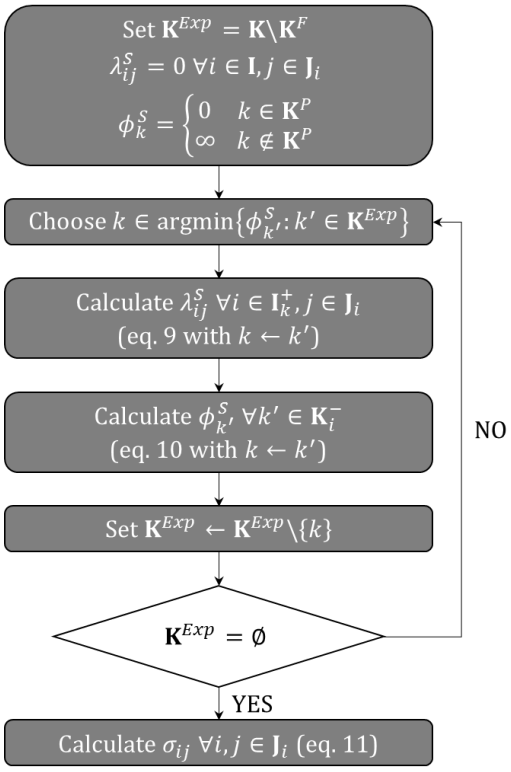


Figure 2.12. Flowchart for the TW_ST step in TW algorithm

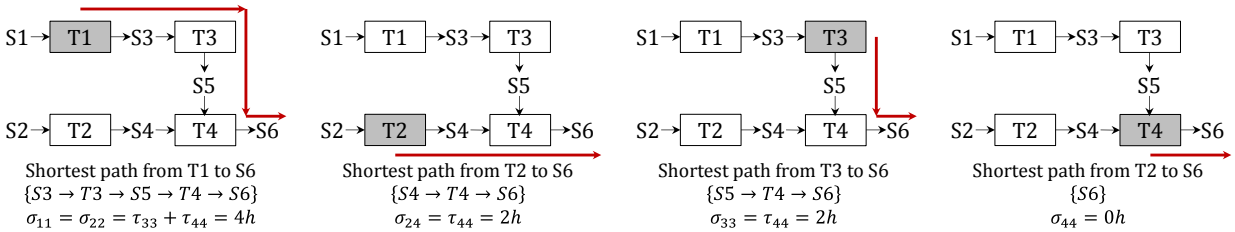


Figure 2.13. Calculation procedure for the TW_EST step

As previously mentioned, by extension it is possible to calculate the earliest start time and shortest tail of individual units, as well as the corresponding time windows as shown in equations (2.12)-(2.14). These parameters are important in developing the FR algorithm in section 2.2.4 and additional tightening constraints in section 2.2.5.

$$\varepsilon_j = \min_{i \in \mathbf{I}_j} \varepsilon_{ij} \quad \forall j \in \mathbf{J} \tag{2.12}$$

$$\sigma_j = \min_{i \in \mathbf{I}_j} \sigma_{ij} \quad \forall j \in \mathbf{J} \tag{2.13}$$

$$\theta_j = \eta - \varepsilon_j - \sigma_j \quad \forall j \in \mathbf{J} \tag{2.14}$$

2.2.2. Forward-Propagation (FP) Algorithm

The maximum cumulative production for task i, μ_i, and material k, ω_k, are calculated through an algorithm that propagates the inventory and time availability information in the direction of the material flow within the network. This forward-propagation algorithm generalizes the ideas presented for the illustrative example and is based on the comparison between values predicted for the maximum amount that a task can process when time and inventory availability are considered separately. Based on time, this maximum amount can be calculated once the time window for the task is known using equation (2.15).

$$\mu_i^{TW} = \sum_{j \in \mathbf{J}_i} \beta_j^{max} \left\lfloor \frac{\theta_{ij}}{\tau_{ij}} \right\rfloor \quad \forall i \in \mathbf{I} \tag{2.15}$$

To consider inventory availability we have to solve the linear program defined in equation (2.16), in which we maximize the amount produced by task i, represented by the variable Q_i. The constraints in this LP enforce that (i) the amount produced of each material plus any initial inventory is greater than the amount consumed, and that (ii) for each material the total consumption by downstream tasks does not exceed its maximum feasible production within the given horizon. Note that the LP in equation (2.16) implicitly uses the structure of the network, so it is valid for networks with or without recycled materials.

$$\mu_i^{LP} = \max\left\{ Q_i: \begin{array}{c} \xi_{k0} + \sum_{i' \in \mathbf{I}_k^+} \rho_{i'k} Q_{i'} + \sum_{i' \in \mathbf{I}_k^-} \rho_{i'k} Q_{i'} \geq 0 \ \ \forall k \\ \sum_{i' \in \mathbf{I}_k^+} \rho_{i'k} Q_{i'} \leq \omega_k \ \ \forall k; \ Q_i \geq 0 \end{array} \right\} \ \forall i \in \mathbf{I}: \omega_k \ is \ known \ \forall k \in \mathbf{K}_i^- \quad (2.16)$$

Equation (2.17) sets the maximum production of the task to the minimum predicted using time window and inventory restrictions. Finally, equation (2.18) allows the propagation of information to the materials produced by tasks whose μ_i is known.

$$\mu_i = \min\left\{\mu_i^{TW}, \mu_i^{LP}\right\} \quad \forall j \in \mathbf{J} \tag{2.17}$$

$$\omega_k = \xi_{k0} + \sum_{i \in \mathbf{I}_k^+} \rho_{ik} \mu_i^1 \quad \forall k \in \mathbf{K}: \mu_i \ is \ known \ \forall i \in \mathbf{I}_k^+ \tag{2.18}$$

Equation (2.18) introduces a new parameter, μ_i^1, based on the ideas discussed by Velez et al.[59] A correction in the original value of μ_i predicted through equation (2.17) is required, since the capacities of the units might prevent this exact amount to be processed. Next, we describe how this correction is made.

For a given task i, let us define the set $m \in \mathbf{M}_i$ to index the production intervals that result when different combinations of units processing task i are considered. Let us also define ϵ_j^m as the number of batches of task i processed in unit $j \in \mathbf{J}_i$ for interval m. The maximum value ϵ_j^m takes is obtained by rounding up the ratio of the cumulative production of task i to the maximum capacity of unit $j \in \mathbf{J}_i$, i.e. $\epsilon_j^{max} = \left\lceil \mu_i / \beta_j^{max} \right\rceil$. Then, equation (2.19) provides the number of elements in the set $\mathbf{M}_i$, where the first term represents combinations with $\epsilon_j^m \in \left[0, \epsilon_j^{max} - 1\right]$ and the second term is the number of intervals with $\epsilon_j^m = \epsilon_j^{max}$ and $\epsilon_{j'}^m = 0 \ \forall j' \neq j$, i.e. the combinations that contain only one unit.

$$|\mathbf{M}_i| = \prod_{j \in \mathbf{J}_i} \epsilon_j^{max} + |\mathbf{J}_i| \tag{2.19}$$

Equation (2.20) defines the corrected value for the feasible cumulative production of task i.

$$\mu_i^1 = \begin{cases} \mu_i & if\ \exists m: \sum_{j\in \mathbf{J}_i} \epsilon_j^m \beta_j^{min} \le \mu_i \le \sum_{j\in \mathbf{J}_i} \epsilon_j^m \beta_j^{max} \\ \max_{m\in \mathbf{M}_i} \left\{ \sum_{j\in \mathbf{J}_i} \epsilon_j^m \beta_j^{max} : \sum_{j\in \mathbf{J}_i} \epsilon_j^m \beta_j^{max} \le \mu_i \right\} & otherwise \end{cases} \quad (2.20)$$

As an illustration let us consider a task T1 that has a maximum cumulative production of 70 kg and is compatible with units U1 and U2, with capacities $\beta_{U1}^{min} = 25\ kg, \beta_{U1}^{max} = 30\ kg, \beta_{U2}^{min} = 55\ kg, \beta_{U2}^{max} = 60\ kg$. First, we calculate $\epsilon_{U1}^{max} = \lceil 70/30 \rceil = 3$ and $\epsilon_{U2}^{max} = \lceil 70/60 \rceil = 2$. Then, using equation B1 we obtain the total number of intervals $|\mathbf{M}_{T1}| = 8$. Figure 2.14 presents a complete analysis of the feasible intervals, based on combinations of U1 and U2 and the expressions included in equation 20. Since $\mu_{T1} = 70$ does not fall in any of the intervals, equation B2 uses the upper bound of the closest interval to obtain $\mu_{T1}^1 = 60$.

m	ϵ_{U1}^m	ϵ_{U2}^m	$\sum_{j\in\{U1,U2\}} \epsilon_j^m \beta_j^{min}$	$\sum_{j\in\{U1,U2\}} \epsilon_j^m \beta_j^{max}$
1	0	0	0	0
2	0	1	55	60
3	0	2	110	120
4	1	0	25	30
5	1	1	80	90
6	2	0	50	60
7	2	1	105	120
8	3	0	75	90

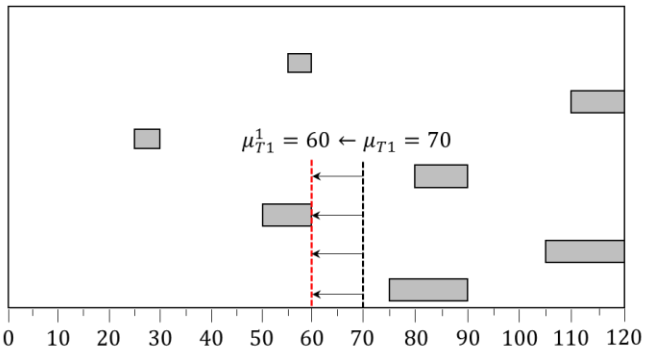


Figure 2.14. Production intervals for combinations of two units.
U1 and U2 have capacities [25,30] and [55,60]. Task T1 compatible with U1, U2 cannot have a maximum cumulative production of $\mu_{T1} = 70$. Feasible production intervals generate a corrected value $\mu_{T1}^1 = 60$.

Figure 2.15 summarizes the complete forward-propagation algorithm. The sets $\mathbf{I}^{Exp}$ and $\mathbf{K}^{Exp}$ respectively contain the tasks and materials that have been explored. Figure 2.16 presents the results of the application of the FP algorithm to the motivating example introduced in Figure 2.1.

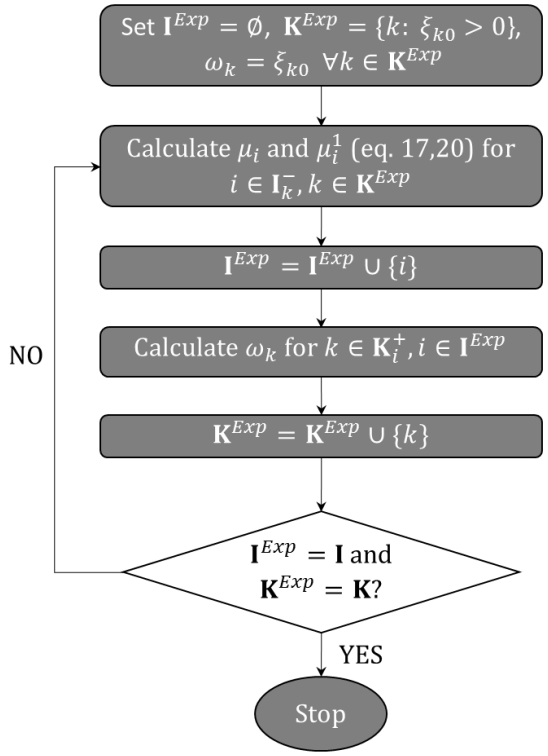


Figure 2.15. Flowchart for the FP Algorithm.
Calculates the maximum feasible production amounts for each task, μ_i, and material, ω_k, within the network

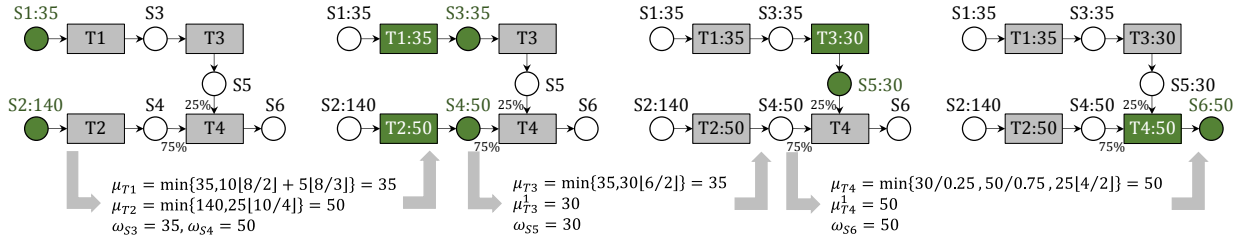


Figure 2.16. Calculation steps and results for FP algorithm applied to network in Figure 1.
Materials and tasks in green indicate their parameters, ω_k and μ_i respectively, are calculated in a specific step

2.2.3. Group Identification (GI) Algorithm

After time availability has been calculated for each task in the network through the time window θ_{ij}, and the forward-propagation has been executed to predict maximum feasible productions for both tasks (μ_i) and materials (ω_k), a new procedure is introduced to define the feasible combinations of the number of batches in subgroups of dependent tasks; i.e., tasks that are interrelated and whose production is limited by one another. There are three different types of groups: (1) Type I are tasks that share a processing unit; (2)Type II are tasks that share an upstream material; and (3) Type III

are tasks that consume materials produced by a common upstream unit. The first two types are simpler to analyze, since both the time window for the common unit and the maximum production for the shared material bound the maximum combined amount the tasks within the group can process. The third type also defines upper bounds on the combined processing amounts; however, the identification of a group and the feasible combination of batches requires the propagation of Type I constraints. The output of this algorithm is the set of dependent groups $d \in \mathbf{D}$. Figure 2.17 shows the basic structure of the GI Algorithm, where the following additional sets are defined:

$\mathbf{J}_k^+(\mathbf{J}_k^-) = \{j \in \mathbf{J}_i : i \in \mathbf{I}_k^+(\mathbf{I}_k^-)\}$ Units that produce (consume) material k

$\mathbf{K}_j^+(\mathbf{K}_j^-) = \{k \in \mathbf{I}_k^+(\mathbf{I}_k^-) : i \in \mathbf{I}_j\}$ Materials produced (consumed) by task j

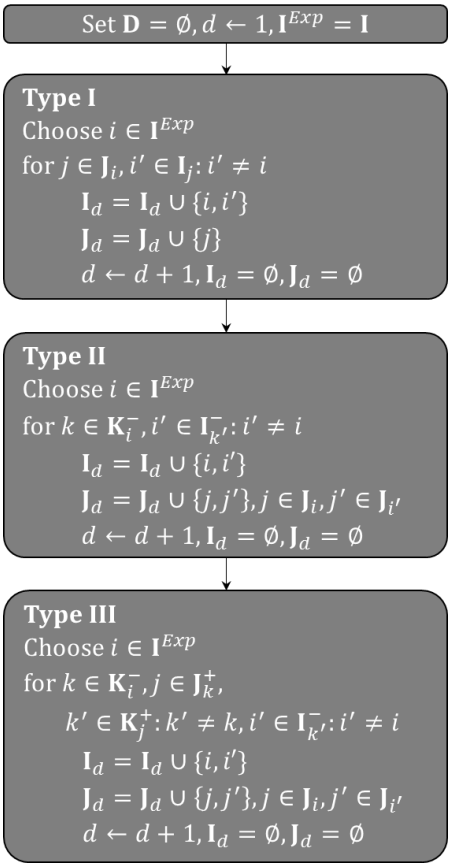


Figure 2.17. Flowchart for the GI algorithm.
It is used to find the groups of interdependent tasks. Type I corresponds to tasks sharing a unit, Type II appears when tasks share a common input material, and Type III groups tasks that use materials produced by the same unit.

As an illustration, we revisit the second motivating example presented before and focus on the Type II group. We follow the algorithm in Figure 2.17 to confirm the qualitative analysis we performed before where we identified that tasks T4 and T5 are dependent because they consume a common material. We select $i = T4$, so the relevant sets are $\mathbf{K}_{T4}^{-} = \{S4\}$ and $\mathbf{I}_{S4}^{-} \backslash \{T4\} = \{T5\}$. Without loss of generality, let us assume that the counting index is set at $d \leftarrow 1$. Then, the new group is defined as $\mathbf{I}_1 = \{T4, T5\}$, the counting index is increased by one (i.e. $d \leftarrow 2$), and $\mathbf{I}_2$ is initialized as an empty set. The algorithm continues until all tasks have been visited and grouped.

2.2.4. Feasible Region (FR) Algorithm

For each of the groups identified through the GI algorithm and indexed by d, the maximum number of batches of each task in the group, ζ_{id}^{max}, is calculated using equation (2.21) that considers restrictions in both available time and inventory. The former involves exploring all the time windows in compatible units for task i, whereas the latter finds the most restrictive input material for task i over all the compatible units.

$$\zeta_{id}^{max} = \min\left\{\sum_{j\in\mathbf{J}_i}\left\lfloor\frac{\theta_{ij}}{\tau_{ij}}\right\rfloor, \max_{j\in\mathbf{J}_i}\left(\min_{k\in\mathbf{K}_i^{-}}\left\lfloor\frac{\omega_k}{\rho_{ik}\beta_j^{min}}\right\rfloor\right)\right\} \quad \forall d \in \mathbf{D}, i \in \mathbf{I}_d \tag{2.21}$$

Subsequently, the set of points P for the feasible combinations of the number of batches of each task in the group is determined, depending on the type of dependence. For Type I, the time representation is important since it is based on the time window for a unit. Equations (2.22) and (2.23) are respectively used in continuous-time and discrete-time formulations to limit the possible combinations of batches for tasks within the group.

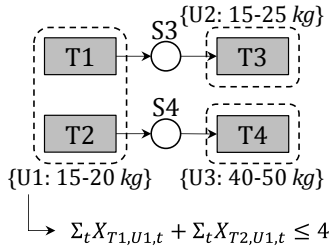
$$\sum_{i\in\mathbf{I}_j}\sum_{t\in\mathbf{T}}\bar{\tau}_{ij}X_{ijt} \le \theta_j \tag{2.22}$$

$$\sum_{i\in\boldsymbol{I}_j}\sum_{t}\tau_{ij}X_{ijt} \le \lfloor\theta_j/\delta\rfloor \quad \forall j \in \mathbf{J} \tag{2.23}$$

For Type II groups, equation 2.22 is used to determine the set P, based on the available amount of the shared material.

$$\sum_{i\in\mathbf{I}_k^-}\sum_{j\in\mathbf{J}_i}\sum_{t\in\mathbf{T}}\beta_j^{min}\rho_{ik}X_{ijt} \le \omega_k \tag{2.24}$$

Tasks in Type III groups consume materials produced by a common upstream unit. This processing unit necessarily defines a Type I group, for which a set of constraints can be found based on equations (2.22) or (2.23). These constraints can be then forward-propagated to determine feasible combination of batches for tasks in the Type III group. As an illustration, let us revisit the second motivating example. Figure 2.18 presents the portion of the network that corresponds to the Type III group with the relevant parameters associated to its tasks and the Type I constraint derived from unit U1.



i	j	τ_{ij}	θ_{ij}	$k \in \mathbf{K}_i^-$	ω_k	ζ_i^{max}
T3	U2	3	8	S3	60	2
T4	U3	2	6	S4	80	2

$$\zeta_i^{max} = \min\{\lfloor\theta_{ij}/\tau_{ij}\rfloor, \lfloor\omega_k/(\rho_{ik}\beta_j^{min})\rfloor\}$$

Figure 2.18. Analysis of Type III group.
Application to example in Figure 2.4 (partial STN is shown)

Using the Type I constraint in terms of the assignment variable and the unit capacity constraint from the original model, we can derive an equation in terms of batch sizes for tasks in unit U1: $\hat{B}_{T1} + \hat{B}_{T2} \leq 80$, where $\hat{B}_i = \sum_t BS_{i,U1,t}$ is the cumulative production of task i. If we define W_k to be the cumulative production of material $k \in \mathbf{K}_i^+, i \in \mathbf{I}_{U1}$, we can upper-bound it using $\hat{B}_i$: $W_{S3} \leq \rho_{T1,S3}\hat{B}_{T1}; W_{S4} \leq \rho_{T2,S4}\hat{B}_{T2}$. Therefore we can evaluate if a combination of batches of tasks T3 and T4 is feasible, by calculating the required values of S3 and S4, and evaluating feasibility using the constraints we just generated. For instance, in Figure 2.7 we show that the combination of one batch of T3 and two batches of T4 is infeasible. This combination requires $W_{S3} \geq 1 \cdot \rho_{T3,S3}\beta_{U2}^{min}; W_{S4} \geq 2 \cdot \rho_{T4,S4}\beta_{U3}^{min}$. But we can directly see that $\left\{\left(\hat{B}_{T1}, \hat{B}_{T2}, W_{S3}, W_{S4}\right): \hat{B}_{T1} + \hat{B}_{T2} \leq 80, 30 \leq W_{S3} \leq \hat{B}_{T1}; 80 \leq W_{S4} \leq \hat{B}_{T2}\right\} = \emptyset$ and therefore the combination is infeasible.

We now generalize this procedure by formally defining two new variables: (1) $\hat{B}_{ij} = \sum_t BS_{ijt}$, the cumulative production of task i in unit $j \in \mathbf{J}_i$ and (2) W_k, the cumulative production of material $k \in \mathbf{K}_i^+, i \in \mathbf{I}_j$ *is a Type I group*. We also define the required amount of input materials for $\upsilon_{i'j'}$ batches of task $i' \in \mathbf{I}_k^-$ in unit $j' \in \mathbf{J}_{i'}$ as $\omega_k^R = \upsilon_{i'j'}\rho_{i'k}\beta_{j'}^{min}$. Then, the LP feasibility problem in equation (2.25) is solved for *each combination* of batches up to the maximum value for each task.

$$\left\{\begin{array}{l}(\hat{B}_{ij}, W_k): \sum_{i \in \mathbf{I}_j} \varphi_{ijh}\hat{B}_{ij} \leq \varphi_{jh}\,\beta_j^{max}: \mathbf{I}_j \text{ is a Type I group}, h \in \mathbf{H}_j \\ \qquad\qquad \omega_k^R \leq W_k \leq \sum_{i \in \mathbf{I}_k^+}\sum_{j \in \mathbf{J}_i} \rho_{ik}\hat{B}_{ij} \quad \forall k\end{array}\right\} \tag{2.25}$$

Finally, once the set of points has been identified for each type of group, its convex hull is obtained by using the *qhull* algorithm[67]. The inequalities that describe $conv(P)$ can directly be added to the discrete- or continuous-time formulations to further tighten the original models. Figure 2.19 summarizes the basic steps in the FR algorithm, where the set $\mathbf{D}^{Exp}$ contain the groups of dependent tasks that still need to be analyzed

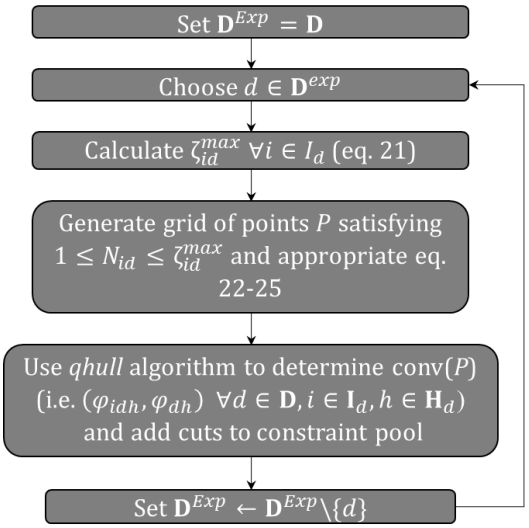


Figure 2.19. Flowchart for the FR algorithm.
This algorithm provides the coefficients for the linear combinations defining feasible combinations of batches within a given dependent group.

Let us consider again the second motivating example, and focus on the Type II group previously identified by the GI algorithm, i.e. $\{T4, T5\}$. Equation (2.24) in this case becomes $\sum_{i\in\mathbf{I}_{S4}^{-}}\sum_{j\in\mathbf{J}_i}\beta_j^{min}\left|\rho_{i,S4}\right|\sum_{t\in\mathbf{T}}X_{ijt} \leq \omega_{S4}$, the relevant sets are $\mathbf{I}_{S4}^{-} = \{T4, T5\}, \mathbf{J}_{T4} = \{U3\}, \mathbf{J}_{T5} = \{U4\}$ and the important parameters $\rho_{T4,S4} = \rho_{T5,S4} = -1, \beta_{U3}^{min} = 40, \beta_{U4}^{min} = 30, \omega_{S4} = 100$. After simplification, we obtain the constraint $4\sum_{t\in\mathbf{T}}X_{T4,U3,t} + 3\sum_{t\in\mathbf{T}}X_{T5,U4,t} \leq 10$, which is depicted in Figure 2.6. The set of possible combinations of batches of T4 and T5 is then

$$P = \{(0,0), (1,0), (2,0), (0,1), (1,1), (0,2), (1,2), (0,3)\}$$

And after applying the *qhull* algorithm we obtain:

$$conv(P) = \left\{ \left(\sum_{t\in\mathbf{T}} X_{T4,U3,t}, \sum_{t\in\mathbf{T}} X_{T5,U4,t} \right) : \begin{array}{l} \sum_{t\in\mathbf{T}} X_{T4,U3,t} \geq 0, \sum_{t\in\mathbf{T}} X_{T5,U4,t} \geq 0, \\ \sum_{t\in\mathbf{T}} X_{T4,U3,t} + \sum_{t\in\mathbf{T}} X_{T5,U4,t} \leq 3, \\ 2\sum_{t\in\mathbf{T}} X_{T4,U3,t} + \sum_{t\in\mathbf{T}} X_{T5,U4,t} \leq 4 \end{array} \right\}$$

This set is represented by the shadowed area in Figure 2.6.

2.2.5. Constraints

In this section we explain how the parameters calculated through the four algorithms are used to define different sets of tightening constraints that can be added to any discrete- or continuous-time formulation. For continuous-time models we assume that the processing time is fixed and does not depend on the batch size.

In continuous-time models, equation (2.22) can be directly used to bound the linear combination that defines the total time spent by a task in a single unit by using the time window defined through the TW Algorithm. Equation (2.26) is the equivalent constraint for a specific task.

$$\sum_{t} \bar{\tau}_{ij} X_{ijt} \leq \theta_{ij} \quad \forall i, j \in \boldsymbol{J}_i \tag{2.26}$$

The corresponding equations for discrete-time formulations are given by (2.23) and (2.27).

$$\sum_{t} \tau_{ij} X_{ijt} \leq \left\lfloor \theta_{ij}/\delta \right\rfloor \quad \forall i, j \in \boldsymbol{J}_i \tag{2.27}$$

The total cumulative production for a given task can be bounded above by μ_i^1 calculated in the FP Algorithm, regardless of time representation.

$$\sum_{j \in \mathbf{J}_i} \sum_{t} BS_{ijt} \leq \mu_i^1 \quad \forall i \in \mathbf{J} \tag{2.28}$$

Finally, the parameters calculated using the FR Algorithm are used to constrain a linear combination of batches of dependent tasks belonging to a group:

$$\sum_{i \in \mathbf{I}_d, j \in \mathbf{J}_i, t} \varphi_{idh} X_{ijt} \leq \varphi_{dh} \quad \forall d, h \in \mathbf{H}_d \tag{2.29}$$

2.3. Extension to variable EST and ST

A generalization of the concepts of earliest start time and shortest time to end of the horizon is proposed in order to obtain even tighter formulations, when the information from the algorithms

just described is propagated. If we allow these amounts to vary instead of fixing them through parameters ε_{ij} and σ_{ij}, new sets of constraints can be formulated. We define variable E_{ij} as the time at which task i can start in unit $j \in \mathbf{J}_i$ and ST_{ij} as the shortest time to the end of the horizon for task i in unit $j \in \mathbf{J}_i$. We can readily write equation (2.30) to lower-bound these variables.

$$E_{ij} \geq \varepsilon_{ij} \; ; \;\; ST_{ij} \geq \sigma_{ij} \;\; \forall i, j \in \mathbf{J}_i \tag{2.30}$$

We can also generalize equation 20 to introduce these two new variables through equation (2.31).

$$\sum_t \bar{\tau}_{ij} X_{ijt} \leq \eta - E_{ij} - ST_{ij} \quad \forall i, j \in \boldsymbol{J}_i \tag{2.31}$$

Consider the example shown in Figure 2.20. If we apply the TW algorithm to this network, we obtain $\varepsilon_{T2,U2} = \varepsilon_{T3,U2} = 4$, since two batches of task T1 are required to start either T2 or T3. But, since T2 and T3 are performed in a common unit, only one of them can have an earliest start time of $4h$, whereas the other will have an earliest start time of 6 or $7h$. However, we do not know *a priori* which task will start first, especially if this network is a subsystem of a bigger network. We can then define variables $E_{T2,U2}$ and $E_{T3,U2}$, for which we know that feasible values are $\left(E_{T2,U2}, E_{T3,U2}\right) = (4,6)$ or $\left(E_{T2,U2}, E_{T3,U2}\right) = (7,4)$. We conclude that a valid inequality for these variables is given by $E_{T2,U2} + E_{T3,U2} \geq 10$, which is stronger than the set $\left\{E_{T2,U2} \geq \varepsilon_{T2,U2} = 4, E_{T3,U2} \geq \varepsilon_{T3,U2} = 4\right\}$.

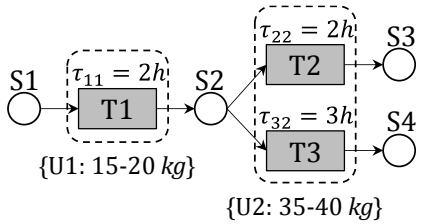


Figure 2.20. Small network to illustrate differences between constant and variable earliest start time.

We define three different families of constraints based on the relative position of tasks within the network. Our goal is to generalize these concepts to more complex structures to obtain valid

inequalities that can be used independently or coupled with the constraints derived in the previous section.

2.3.1. Subsequent tasks

Let us consider the case in which material k is produced by task i' and consumed by task i. Then the earliest start time of task i can be lower-bounded according to equation (2.32). We consider that this value has to be greater or equal than the sum of earliest start time for task i' and the time task i' takes to produce enough material to process the first batch of i.

$$E_{ij} \geq E_{i'j'} + \bar{\tau}_{i'j'} \left\lceil \frac{\rho_{ik}\beta_j^{min} - \xi_{k0}}{\rho_{i'k}\beta_{j'}^{max}} \right\rceil \quad \begin{matrix} \forall i, i' \\ \forall j \in \mathbf{J}_i, j' \in \mathbf{J}_{i'} : |\mathbf{J}_{i'}| = 1 \\ \forall k \in \mathbf{K}_i^- \cap \mathbf{K}_{i'}^+ : |\mathbf{I}_k^+| = 1, \xi_{k0} < \rho_{ik}\beta_j^{min} \end{matrix} \tag{2.32}$$

Similarly, equation (2.33) bounds the shortest tail of task i', using the processing time of task i.

$$ST_{i'j'} \geq ST_{ij} + \bar{\tau}_{ij} \quad \begin{matrix} \forall i, i' \\ \forall j \in \mathbf{J}_i, j' \in \mathbf{J}_{i'} : |\mathbf{J}_{i'}| = 1 \\ \forall k \in \mathbf{K}_i^- \cap \mathbf{K}_{i'}^+ : |\mathbf{I}_k^+| = 1 \end{matrix} \tag{2.33}$$

2.3.2. Tasks consuming the same material

If two tasks i and i' share a common input material k, its cumulative production limits the combinations of values of variable EST for both tasks. Let us consider the simple subsystem shown in Figure 2.21. If the capacity of unit U0 is such that one batch of task T0 is enough to start the first batch of both tasks T1 and T2 (e.g. U0: 60 kg), then simple relations between the variable EST of T0, T1 and T2 are given by $E_{T1} \geq E_{T0} + \bar{\tau}_{T0}$ and $E_{T2} \geq E_{T0} + \bar{\tau}_{T0}$, which can be combined to write $E_{T1} + E_{T2} \geq 2(E_{T0} + \bar{\tau}_{T0})$. On the other hand, when multiple batches of T0 are required to run the first batches of T1 and T2, then a careful analysis of the sequence and the required amounts has to be performed. For each sequence we can use equation 32 to find a valid inequality relating the EST variables. For the (T1→T2) sequence we can write $E_{T1} \geq E_{T0} + \bar{\tau}_{T0}\left\lceil \rho_{T1,S1}\beta_{T1}^{min} / \rho_{T0,S1}\beta_{T0}^{max} \right\rceil = E_{T0} + 2\bar{\tau}_{T0}$ and $E_{T2} \geq E_{T1} + \bar{\tau}_{T0,U0}\left\lceil \rho_{T2,S1}\beta_{T2}^{min} / \rho_{T0,S1}\beta_{T0}^{max} \right\rceil = E_{T1} + 4\bar{\tau}_{T0} \geq E_{T0} + 6\bar{\tau}_{T0}$. After combining

we obtain $E_{T1} + E_{T2} \geq E_{T0} + 8\bar{\tau}_{T0}$. Similarly, the (T2→T1) sequence results in $E_{T1} + E_{T2} \geq E_{T0} + 10\bar{\tau}_{T0}$. In order to avoid cutting feasible integer solutions we choose the former (i.e. less restrictive) inequality as the valid constraint for the group {T1,T2}.

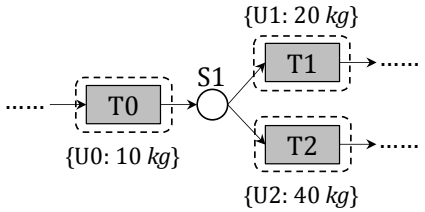


Figure 2.21. Simple subsystem with two tasks sharing an input material S1.

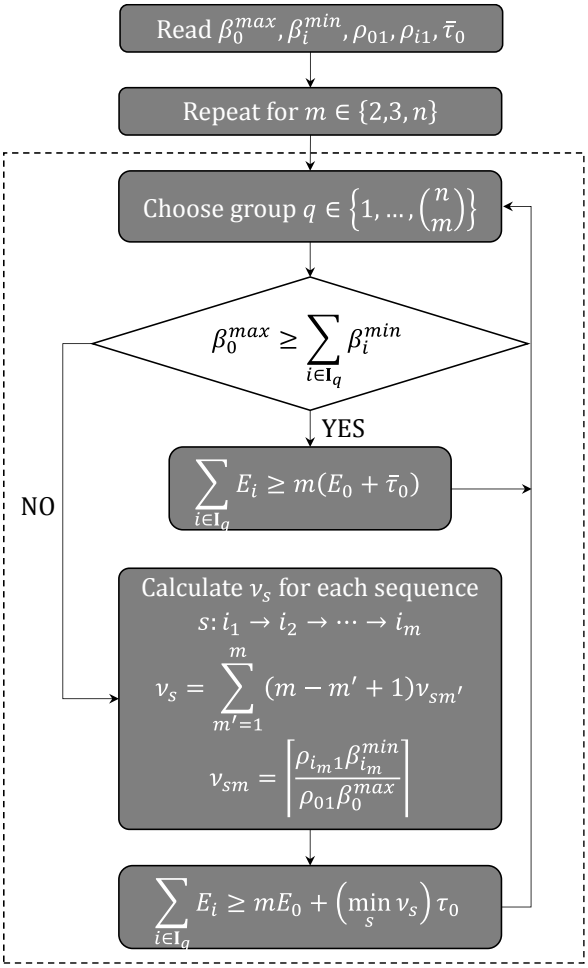


Figure 2.22. Valid inequalities for *n* tasks sharing an input material.

Figure 2.22 shows the flowchart used to generalize this procedure to an arbitrary number (n) of tasks sharing an input material. Note that we also choose to analyze subgroups of two and three tasks in order to add more valid inequalities to the constraint pool.

2.3.3. Tasks sharing a common unit

If $q \geq 2$ tasks share a common unit j, we consider the $q!$ points obtained from considering the possible permutations of tasks and their respective processing times. For instance, when two tasks i and i' are compatible with unit j, we need to consider the points $(\varepsilon_{ij}, \varepsilon_{ij} + \bar{\tau}_{ij})$ and $(\varepsilon_{i'j} + \bar{\tau}_{i'j}, \varepsilon_{i'j})$ for EST, and $(\sigma_{ij}, \sigma_{ij} + \bar{\tau}_{ij})$ and $(\sigma_{i'j} + \bar{\tau}_{i'j}, \sigma_{i'j})$ for ST. We can then find the equation of the hyperplane on which these $q!$ points lie in order to define the feasible half space whose inequality can be added to the pool of constraints. Equations (2.34) and (2.35) give the general form of such half spaces for variable EST and ST.

$$\sum_{i \in \mathbf{I}_j} v_{ij} E_{ij} \geq v_j \quad \forall j: \mathbf{I}_j \; is \; a \; type \; I \; group \tag{2.34}$$

$$\sum_{i \in \mathbf{I}_j} \breve{v}_{ij} ST_{ij} \geq \breve{v}_j \quad \forall j: \mathbf{I}_j \; is \; a \; type \; I \; group \tag{2.35}$$

Where v_{ij}/v_j ($\breve{v}_{ij}/\breve{v}_j$) are the coefficients of the EST (ST) hyperplane.

2.4. Implementation

The computational implementation of the algorithms described in section 2.2 and the extensions presented in section 2.3 requires a dual interface using a numerical computing environment in addition to an optimization engine. The numerical computing environment we use is Matlab R2014b and we primarily employ it to (1) efficiently generate the sets of points required for the FR algorithm and general EST hyperplanes, and (2) act as the master environment to make calls to the *qhull* package[67] as part of the FR algorithm. The optimization engine we use is Cplex 12.6 on GAMS 24.2.

Figure 2.23 shows the interaction and the flow of information between the different computational engines required to implement the algorithms. The original network data and mathematical models are directly written in GAMS. In addition, the first three algorithms (TW, FP and GI) are also coded in GAMS to take advantage of its powerful set/index manipulation. Then two separate scripts in Matlab are used to execute the FR algorithm and the generation of hyperplanes for the variable EST case. First, the data on dependent sets of tasks is used as input to Matlab to generate the set of points P satisfying equations (2.22)-(2.25). This set is then used as the input to *qhull* to generate an explicit set of inequalities –equation (2.31)– that defines $conv(P)$, which is sent back to Matlab to write the final output file containing the coefficients of each inequality in GAMS format. Second, the data on earliest start time and shortest tail is passed to Matlab, which calculates the coefficients for the equations described in sections 2.3.2 and 2.3.3. Then it uses its built-in singular value decomposition (SVD) to calculate the equation of the hyperplane that contains all the points. The collinearity of the points is accepted without a formal proof, but it is verified by checking that the smallest singular value vanishes.

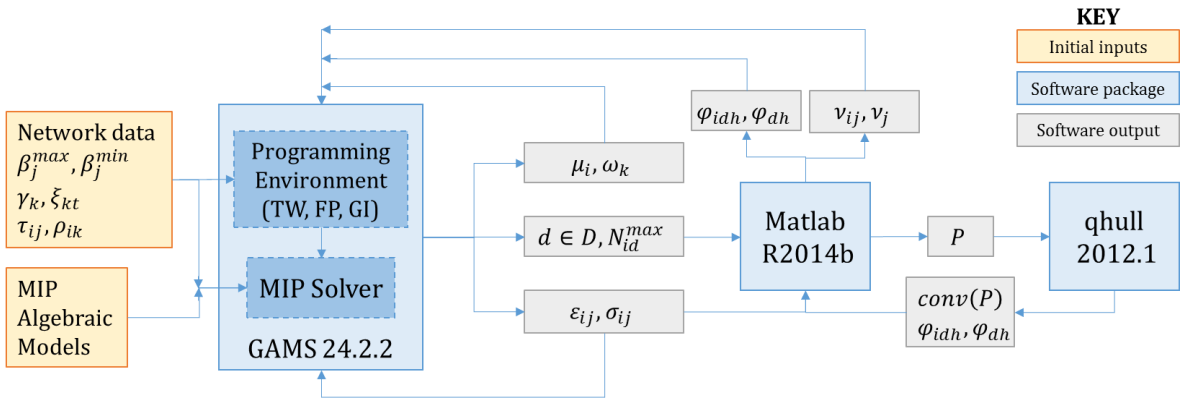


Figure 2.23. Flowchart for computational implementation.
Flow of information between the optimization engine (GAMS) and the numerical computing environment (Matlab) necessary to implement the proposed algorithms and the extension to variable EST.

The exchange of information between different packages is made using binary text files written by the software acting as master in each procedure. In general, other than the solution of the MIP itself, the most expensive step is the generation and communication of the set of points P that satisfy

the criteria associated with a particular type of dependence in the FR algorithm. This is due to the combinatorial nature of the problem of generating all the possible permutations of number of batches for tasks within a group. Moreover, the computational effort increases as the number of tasks in the group and/or the scheduling horizon increase. Also, note that one of the most time-consuming tasks in this process is the writing of the binary files to exchange information among the different pieces of software, particularly between Matlab and GAMS. That is why any internal data manipulation within a given software should be done using native variables instead of writing extra binary files to avoid artificial extra time in the preprocessing stage. On average, when the FR algorithm is used, 70% of the preprocessing time is spent in GAMS and 30% in Matlab/*qhull*. A possible way to improve computational efficiency is to solve the LP problems that appear in the preprocessing using Matlab.

2.5. Computational Study

To determine the effectiveness of the proposed methods, we test them on 27 instances of three different networks with different scheduling horizons and using the models presented in section 2.1.2. Networks 1 and 2 (PN1 and PN2) are modified from Papageorgiou and Pantelides[68] and network 3 (PN3) is from Maravelias and Papalamprou[46]. Table 2.1 gives a summary of all the instances and contains the step size δ for the discrete-time model SP&S, as well as the number of points, N^*, used to represent the best solution, which was determined through an iterative procedure. For this calculation, we used the solution obtained by the discrete-model as lower bound and iteratively increased the number of points in the continuous-time formulation until the optimal solution was found. The STN representations and corresponding data for all networks are available in the Supporting Information.

Table 2.1. Assignment of network, horizon and model for each computational run. A total of 27 instances are studied.

		Network											
		PN1				PN2				PN3			
	$\boldsymbol{\eta}$ ($\boldsymbol{h}$)	30	36	48	60	18	24	30	36	24	30	36	60
Model	SP&S, δ		0.25	0.25	0.25		0.5	0.5	0.5	0.05		0.05	0.05
	S&K, N^*	9	12	16		12	12	14		9	12	14	
	GH&M, N^*	14	16	19		15	19	22		11	14	16	

Note that the instances used for the discrete-time model are more difficult than those for the continuous-time models, since they are defined for longer horizon values.

The objective function we consider is profit maximization, equation (2.36). For the revenue term we use the selling price for each final product, π_k, while for the cost term we only consider a marginal production cost associated with the execution of task i in unit $j \in \mathbf{J}_i$, $\alpha_{ij} = 1 \times 10^{-4}$. This small value penalizes unnecessary task executions that do not lead to production of final products.

$$Profit = \sum_{k \in \mathbf{K}} \sum_{t \in \mathbf{T}} \pi_k S_{kt} - \sum_{i \in \mathbf{I}} \sum_{j \in \mathbf{J}_i} \sum_{t \in \mathbf{T}} \alpha_{ij} X_{ijt} \tag{2.36}$$

We analyze the effects of the various algorithms and the extension to variable EST and ST by defining four different formulations as shown in Table 2.2. Formulation F0 is the original model. Formulation F1 introduces the tightening from the four algorithms, using the results for the convex hull of the set of feasible number of task occurrences. Formulation F2 adds the tightening constraints from the variable earliest start time and shortest tail as discussed in section 2.3. Formulation F3 is a combination of both strategies. Each formulation F0-F3 is applied to each of the 27 instances defined in Table 2.1, for a total of 108 runs

Table 2.2. Definition of the four proposed formulations.

Formulation	**Equations**
F0	SP&S: 1-3 S&K: A1-A14 GH&M: A15-A29
F1	Discrete-time: F0+26-29 Continuous-time: F0+22,25,28,29
F2	F0+30-37
F3	F1+F2

We used GAMS 24.2.2/CPLEX 12.6 and Matlab R2014b on a computer with 8 GB of RAM and a 2.67 GHz Intel Core (i7-920) processor running on Windows 7. Default CPLEX settings, including cuts, are used. A time limit of one hour is enforced for all the runs. Complete model and solution statistics are given in the Supporting Information.

2.5.1. Results

The first measure we use to analyze the effectiveness of our methods is the number of runs solved to optimality. Approximately 52% of the total 108 runs found a solution and prove optimality within the time limit. In order to investigate the effect of the specific MIP model, Figure 2.24 presents the fraction of runs solved to optimality by each of the three models we consider.

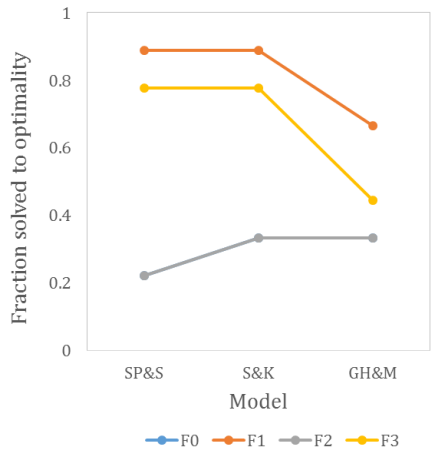


Figure 2.24. Fraction of runs solved to optimality using models SP&S, S&K, and GH&M.

Formulation F1 appears to solve more instances to optimality than any other formulation, regardless of the MIP model. Moreover, the discrete-time model proves to be more effective in finding optimal solutions with 90% of the runs solved to optimality when formulation F1 is used. Note that network PN2 represents a medium-scale process with 19 tasks and 27 materials and network PN3 uses a small step size, which significantly increases the number of binary variables and constraints in the SP&S model. On the other hand, formulation F2 has no practical improvement with respect to the original models, in fact the lines for F0 and F2 completely overlap in Figure 2.24.

In industrial applications it is often sufficient to obtain a “good enough” feasible solution, rather than a provably optimal solution. In such cases, the effectiveness of a model can be assessed based on the optimality gap. Figure 2.25 contains results for all three models in terms of optimality gap. Instances are classified into three categories: (1) instances solved to optimality by all formulations, (2) instances solved to optimality by one or more of the new formulations, and (3) instances that reach the time limit with nonzero optimality gap with all formulations. Figure 2.25 shows that introducing any of the three proposed formulations brings advantages in terms of both computational time and optimality gap. Figure 2.25(a) shows that formulations F1 and F3 coupled with the discrete-time model produce the best results, in accordance with Figure 2.24. Figure 2.25 (b) shows that F1 and F3 are the most effective for model S&K; they solve to optimality five more instances than F0 and F2, and lead to the lowest optimality gap for the unsolved instance. Figure 2.25 (c) shows the same pattern for formulations F1 and F3 with model GH&M, but in addition, reveals a significant reduction in optimality gap, much more pronounced than model S&K. Figure 2.25 (b) and (c) also show that model S&K outperforms model GH&M in all the instances. Model S&K with formulations F1 and F3 solves more instances to optimality than its model GH&M counterpart, reducing their average solution times. Instances not solved to optimality also exhibit a significant reduction in optimality gap with model S&K. Clearly, all proposed formulations have a significant impact on reducing optimality gap for instances that reach the time limit. From the analysis presented in Figure 2.24 and Figure 2.25, we establish that formulation F1 applied to model SP&S leads to the best results in terms of consistency and decreasing both computational time and gap.

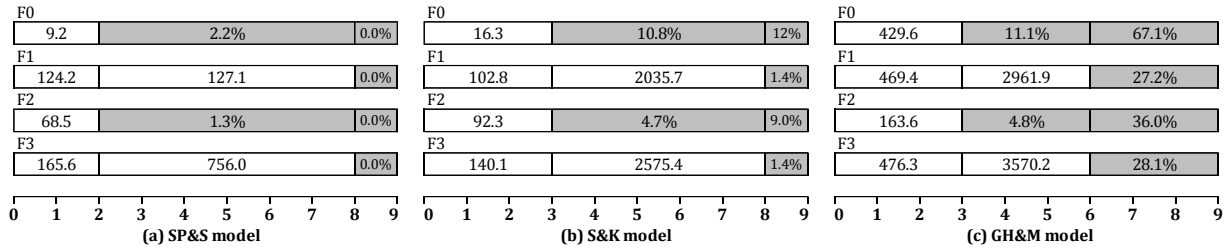


Figure 2.25. Average solution time or optimality gap.
Instances are solved using (a) discrete-time model SP&S, and continuous-time models (b) S&K, and (c) GH&M

Another measure to show the effectiveness of a given method is the integrality gap defined as the relative difference between the optimal objective function values of the LP relaxation and the MIP model. Table 2.3 presents the results for each model and different formulations. Interestingly, the addition of the tightening constraints does not reduce the optimality gap of model SP&S, although as discussed earlier, the reduction in computational time and optimality gap is significant. For the continuous-time models S&K and GH&M we observe once again that formulations F1 and F3 provide the greatest tightening when compared to their respective original formulations

Table 2.3. Average integrality gap (%) for models SP&S, S&K, GH&M and formulations F0-F3.

Formulation	SP&S	S&K	GH&M
F0	9.0	37.5	51.6
F1	9.0	18.6	22.6
F2	9.0	27.8	39.9
F3	9.0	18.6	22.6

In Figure 2.26, we present a performance chart for model SP&S which has been found to be the most effective. A general discussion on this type of charts is given in Appendix B. The chart shows that formulation F1 is the most effective – 80% of the instances are solved within 1.5 times the solution time of the fastest instance. The untightened model solves only 33% of the instances within one order of magnitude of the fastest instance, whereas formulations F2 and F3 solve 33% and 67% respectively within the same time.

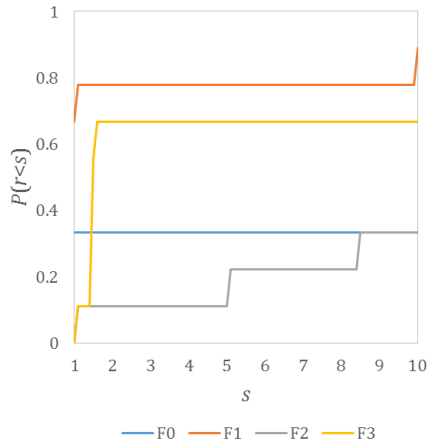


Figure 2.26. Performance chart for model SP&S and proposed methods.
The abscissa represents the ratio (CPU time)/(CPU fastest instance) for each instance. The ordinate is the fraction of instances solved faster than the relative time in the abscissa.

Finally we calculate an *improvement factor* (IF) which is the ratio of the computational times of the original model and the best proposed formulation. The overall value (i.e., the average over all networks, horizons and models) is 15.2. The biggest improvement comes from the discrete-time model (IF = 32.4), which is much higher than the factors of 2.7 and 7.0 for models S&K and GH&M respectively.

2.6. Conclusions

We developed new methods for the enhancement of the solution of Mixed-Integer Programming models for chemical production scheduling. These methods consist of four preprocessing algorithms based on instance-specific information (network structure, recipe, processing times, initial inventory, and unit capacities) to calculate parameters that are then used to generate tightening constraints; as well as some new constraints based on variable *earliest start times* and *shortest tails*.

The Time Window (TW) algorithm calculates the effective time window that is available for each task to run, based on time and inventory restrictions. The Forward-Propagation (FP) algorithm calculates parameters to define the maximum feasible production for each task and material, and identifies whether a task is limited by its time window or the availability of its input materials. The

Group Identification (GI) algorithm classifies tasks that share a common feature into groups, which in turn can be used to predict sets of feasible combinations of batches of these tasks. Finally, the Feasible Region (FR) algorithm uses the groups derived above to generate valid inequalities from the convex hull of the aforementioned set of points.

The proposed methods are applicable to all material-based, time-indexed, MIP scheduling models, discrete- and continuous-time. We show that the constraints derived from the sequence of preprocessing algorithms (formulation F1) produce the best results for both discrete- and continuous-time models. In many case, up to two orders of magnitude decrease in computational time is achieved when these constraints are applied to the discrete-time model.

2.7. Notation

Sets/indices

$d \in \mathbf{D}$ Groups of dependent tasks

$i \in \mathbf{I}$ Tasks

$j \in \mathbf{J}$ Units

$h \in \mathbf{H}_d$ Equations that define the convex hull of points that characterize group d

$k \in \mathbf{K}$ Materials

$n \in \mathbf{N}$ Time points in continuous-time models

q Number of tasks that share a common resource in variable EST/ST calculations

$t \in \mathbf{T}$ Time points in discrete-time models

$\bar{t} \in \overline{\mathbf{T}}$ Time points in TW_EST step

$\mathbf{I}_d$ Tasks that belong to group d

$\mathbf{I}^{Exp}$ Tasks that remain to be explored in FP and GI algorithms

$\mathbf{I}_j$ Tasks that can be executed in unit j

$\mathbf{I}_q$ q-tuples of tasks that share a common resource in variable EST/ST calculations

$\mathbf{I}_k^+/\mathbf{I}_k^-$ Tasks that produce/consume material k

$\mathbf{J}_d$ Units that belong to group d

$\mathbf{J}_i$ Units that can process task i

$\mathbf{J}_k^+/\mathbf{J}_k^-$ Units that produce/consume material k

$\mathbf{K}^{Exp}$ Materials that remain to be explored in TW and FP algorithms

$\mathbf{K}^F$ Set of raw materials

$\mathbf{K}_i^+/\mathbf{K}_i^-$ Materials produced/consumed by task i

$\mathbf{K}_j^+/\mathbf{K}_j^-$ Materials produced/consumed by unit j

$\mathbf{K}^P$ Set of final products

$\mathbf{T}^{Exp}$ Time points that remain to be explored in TW algorithm

Parameters

α_{ij} Production cost for task $i \in \mathbf{I}$ in unit $j \in \mathbf{J}_i$

$\beta_j^{max}/\beta_j^{min}$ Maximum/minimum batch size for unit $j \in \mathbf{J}$

γ_k Storage capacity for material $k \in \mathbf{K}$

δ Length (step size) of intervals in discrete-time models

ε_{ij} Earliest start time for task $i \in \mathbf{I}$ in unit $j \in \mathbf{J}_i$

ε_j Earliest start time for task $i \in \mathbf{I}$ in unit $j \in \mathbf{J}_i$

η Scheduling horizon

ζ_{id}^{max} Maximum number of batches of task $i \in \mathbf{I}$ in group d

θ_{ij} Time window for task $i \in \mathbf{I}$ in unit $j \in \mathbf{J}_i$

θ_j Time window for unit $j \in \mathbf{J}$

λ_{ij}^{S} Latest finish time for task $i \in \mathbf{I}$ in unit $j \in \mathbf{J}_i$ so that the intermediates it produces have enough time to be transformed into final products.

μ_i Maximum cumulative production of task $i \in \mathbf{I}$ within the given horizon

μ_i^1 Corrected value of μ_i due to production intervals

μ_i^{LP} Value of μ_i based on inventory restrictions modeled as an LP

μ_i^{TW} Value of μ_i based on available time window

$\nu_{ij}/\check{\nu}_{ij}$ Coefficient for EST/ST constraints of task $i \in \mathbf{I}_q$ in unit $j \in \mathbf{J}_i$

$\nu_j/\check{\nu}_j$ Right-hand-side coefficient for EST/ST constraints of unit $j \in \mathbf{J}$

ξ_{kt} Delivery of material $k \in \mathbf{K}$ at time t

π_k Unit price of material $k \in \mathbf{K}$

ρ_{ik} Conversion coefficient for material $k \in \mathbf{K}$ produced ($\rho_{ik} > 0$) or consumed ($\rho_{ik} < 0$) by task $i \in \mathbf{I}$

σ_{ij} Shortest tail for task $i \in \mathbf{I}$ in unit $j \in \mathbf{J}_i$

σ_j Shortest tail for unit $j \in \mathbf{J}$

$\bar{\tau}_{ij}$ Processing time for task $i \in \mathbf{I}$ in unit $j \in \mathbf{J}_i$

τ_{ij} Processing time for task $i \in \mathbf{I}$ in unit $j \in \mathbf{J}_i$ as a multiple of the step size δ for discrete-time representation

v_{ij} Batches of task $i \in \mathbf{I}_k^-$ in unit $j \in \mathbf{J}_i$ required to calculate ω_k^R

φ_{idh} Linear combination coefficient for task $i \in \mathbf{I}_d$ of group d in equation $h \in \mathbf{H}_d$

φ_{dh} Right-hand-side coefficient for group d in equation $h \in \mathbf{H}_d$

ϕ_{it}^E Cumulative production for task $i \in \mathbf{I}$ in *any unit* starting at or before time t

ϕ_k^S Minimum time required to process material $k \in \mathbf{K}$ so it leads to final products

χ_{ijt}^E Cumulative production for task $i \in \mathbf{I}$ in unit $j \in \mathbf{J}_i$ starting at or before time t

ψ_{kt}^E Maximum amount of material $k \in \mathbf{K}$ available at time t

ω_k Maximum cumulative production of material $k \in \mathbf{K}$ within the given horizon

$\omega_{k\breve{n}}^R$ Required amount of input materials for $\breve{n}_{ij}$ batches of task $i \in \mathbf{I}_k^-$ in unit $j \in \mathbf{J}_i$

Binary variables

X_{ijt}/X_{ijn} It is equal to one if task $i \in \mathbf{I}$ starts in unit $j \in \mathbf{J}_i$ at time $t \in \mathbf{T}/\ n \in \mathbf{N}$

Y_{ijn} It is equal to one if task $i \in \mathbf{I}$ finishes in unit $j \in \mathbf{J}_i$ at time $n \in \mathbf{N}$

Continuous nonnegative variables

$\hat{B}_{ij}$ Cumulative production of task $i \in \mathbf{I}$ in unit $j \in \mathbf{J}_i$

BF_{ijn}/BP_{ijn} Batch size of task $i \in \mathbf{I}$ that starts/continues to be processed in unit $j \in \mathbf{J}_i$ at time $n \in \mathbf{N}$

BS_{ijt}/BS_{ijn} Batch size of task $i \in \mathbf{I}$ that starts in unit $j \in \mathbf{J}_i$ at time $t \in \mathbf{T}/\ n \in \mathbf{N}$

E_{ij} Variable earliest start time for task $i \in \mathbf{I}$ in unit $j \in \mathbf{J}_i$

N_i Number of batches of task $i \in \mathbf{I}$

Q_i Amount produced by task $i \in \mathbf{I}$

ST_{ij}	Variable shortest tail for task $i \in \mathbf{I}$ in unit $j \in \mathbf{J}_i$
S_{kt}/S_{kn}	Inventory level of material $k \in \mathbf{K}$ at time point $t \in \mathbf{T}/\ n \in \mathbf{N}$ with $S_{k0} = \xi_k$
T_n	Actual value of time point $n \in \mathbf{N}$
W_k	Cumulative production of material $k \in \mathbf{K}$

Chapter 3

Discrete-Time Models and Solution Methods for Production Scheduling in Multistage Facilities[2]

The main goal of this chapter is to propose, test, and compare global-grid discrete-time models for sequential environments. We choose this time representation, because it is conceptually simpler than its continuous counterpart[8]. It does not require time-matching constraints and it leads to models that can be readily extended to include different features such as utilities, time-varying parameters, maintenance operations, and resource availability, among others[52, 70]. Discrete-time formulations are also known to have tighter Linear Programming (LP) relaxations, allow linear modeling of inventory and utility costs and seamlessly include features such as setups and material production at different times during the execution of a single task[15, 52]. In addition, a recent computational study[70] showed that discrete-time models are more effective not only in reducing computational time, but also in finding better solutions and decreasing the optimality gap within an arbitrary time limit. Moreover, recent solution methods tested on both discrete- and continuous-time formulations[51, 59] have shown better performances when applied to the former.

We also propose solution methods based on specific characteristics of sequential facilities. In addition, we consider utility constraints which are difficult to model using continuous-time models and show how they can naturally be modeled with discrete-time formulations. Our aim is to extend the improvements in computational performance obtained in network processing into sequential environments.

[2] This chapter is modified from Merchan et al.[69]

The chapter is structured as follows: Section 3.1 formally defines the problem under study. Section3.2 provides an extensive review of the relevant literature and concepts upon which our models and methods are developed. In section3.3, we define four discrete-time models for scheduling in sequential environments. In Section3.4, several solution methods are proposed. Section3.5 presents small-scale examples that include utility constraints. Section3.6 contains an extensive computational study with medium- and large-scale problems to test the proposed models and solution methods.

3.1. Problem statement

We study the problem of short-term scheduling of multistage batch chemical plants subject to utility constraints. In these facilities, batches of different products are processed in a series of stages, each of which consists of one or more unrelated parallel units and each unit belongs to a single stage. Batches are processed as discrete entities so their identity is preserved, thus mixing of different batches and splitting of a single batch into smaller quantities are not allowed. All the products follow the same sequence throughout the stages, although some products might skip one or more stages. This basic problem is hereinafter referred to as the Multistage Batch Plant Scheduling Problem, MBPSP. In addition, we consider the case where limited utilities are required by batches to complete their processing in a given unit. Next, we formally define the MBPSP by introducing the basic sets, indices and parameters that describe a multistage batch plant. Extensions to include utilities are deferred to section 3.3.8. Figure 3.1 depicts the type of facility we study.

Sets and indices

$i \in \mathbf{I}$	Batches
$j \in \mathbf{J}$	Processing units (denoted by $\mathbb{U}_j$)
$k \in \mathbf{K}$	Stages
$\mathbf{I}_j$	Batches that can be processed in unit $j \in \mathbf{J}$

$\mathbf{I}_k$ Batches that are processed on stage $k \in \mathbf{K}$

$\mathbf{J}_i$ Units that can process batch $i \in \mathbf{I}$

$\mathbf{J}_k$ Units that belong to stage $k \in \mathbf{K}$

$\mathbf{J}_{ik}$ Units that can process batch $i \in \mathbf{I}$ on stage $k \in \mathbf{K}_i$, i.e. $\mathbf{J}_{ik} \equiv \mathbf{J}_i \cap \mathbf{J}_k$

$\mathbf{K}_i$ Stages on which batch $i \in \mathbf{I}$ is processed

Parameters

$\bar{\tau}_{ij}$ Processing time of batch $i \in \mathbf{I}$ in unit $j \in \mathbf{J}_i$

$\bar{\mu}_i$ Release time for batch $i \in \mathbf{I}$

$\bar{\phi}_i$ Due date for batch $i \in \mathbf{I}$

$\kappa_i^{FS}/\kappa_i^{LS}$ First/last stage on which batch $i \in \mathbf{I}$ can be processed

ν_{ik} Defines the next stage on which batch $i \in \mathbf{I}$ can be processed after stage $k \in \mathbf{K}_i$, i.e. batch i is processed on stage $k + \nu_{ik}$ immediately after stage k

α_{ij} Fixed cost of processing batch $i \in \mathbf{I}$ in $j \in \mathbf{J}_i$

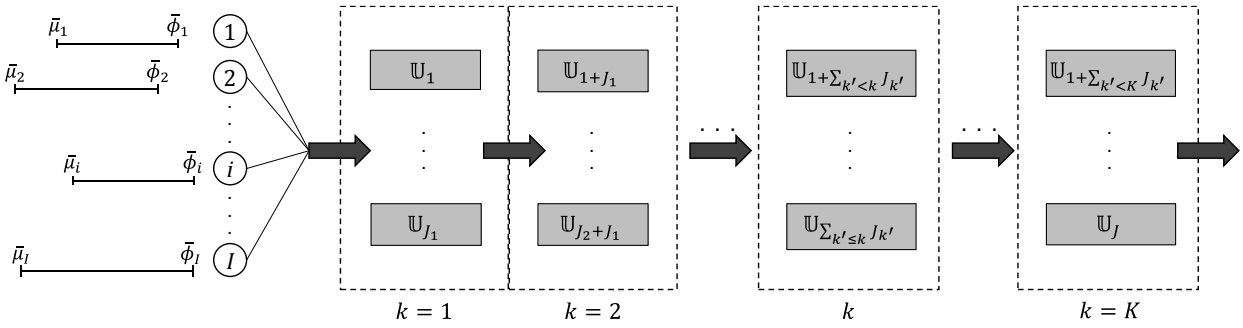


Figure 3.1. General multistage batch production plant.
Circles represent batches (time windows are given), gray boxes denote processing units, and dotted-line boxes define stages. $I \equiv |\mathbf{I}|$ is the total number of batches, $J \equiv |\mathbf{J}|$ denotes the total number of units, $K \equiv |\mathbf{K}|$ defines the number of stages, and $J_k \equiv |\mathbf{J}_k|$ is the total number of units on stage k

From this problem definition, we identify three major types of constraints that must be satisfied: (i) routing of a batch among different stages, henceforth referred to as *batch precedence*, (ii) unique assignment of a batch to every stage in which it is processed, hereinafter called *batch-stage assignment*, and (iii) unit utilization and sequencing of batches assigned to the same unit, henceforth

referred to as *unit assignment*. The requirement that every batch has to be processed between its release and due dates can explicitly or implicitly be enforced through additional constraints or sets of time points. All these constraints can be enforced using different Mixed-Integer Programming (MIP) models, as discussed in sections 3.2.1 and 3.3.

3.2. Background

3.2.1. MIP modeling in sequential environments

The first efforts to develop systematic methods to describe and solve the scheduling problem in sequential environments were made in the context of discrete manufacturing, which can be defined as the production of single items which are independent and enumerable (e.g. vehicles, mechanical parts, furniture, electronics, etc.). A vast literature in the operations research (OR) community is dedicated to exact, heuristic and metaheuristic methods to solve the scheduling problem in this type of industrial facilities[1, 71-74].

In the context of discrete manufacturing, the problem we are studying is known as the Flexible (hybrid, compound, multiprocessor) Flow Shop (FFS) scheduling problem and has also been the subject of extensive research[75-77]. However the MBPSP can be much more general since material handling restrictions can be relaxed when fluid materials are present and features such as utilities are included.

The MBPSP has also attracted the interest of several researchers in the PSE community, and several surveys are available[5, 6, 8, 9, 19]. As mentioned in Chapter 1, the majority of these works have been devoted to develop continuous-time models. For comparison purposes, we chose for this work the model presented independently by Mendez et al.[21] and Harjunkoski and Grossmann[22] as a generalization of the model developed by Brah[78] for identical machines in each stage. We refer to it as model MH&C. Next, we define the basic decision variables model MH&C employs.

$C_{ik} \geq 0$ Completion time for batch $i \in \mathbf{I}$ on stage $k \in \mathbf{K}_i$

$X_{ij}^{C} \in \{0,1\}$ One if batch $i \in \mathbf{I}$ is assigned to unit $j \in \mathbf{J}_i$

$Y_{ii'k} \in \{0,1\}$ One if batch $i \in \mathbf{I}$ is processed before batch $i' \in \mathbf{I}$ on stage $k \in \mathbf{K}_i \cap \mathbf{K}_{i'}$

The main constraints for the MBPSP are enforced as follows. Batch precedence is guaranteed using equation (3.1) that relates completion times of batch i in consecutive stages. Batch-stage assignment is enforced through equation (3.2). Unit assignment and sequencing is modeled via equations (3.3) and (3.4) which use a big-M reformulation to guarantee that if two batches are assigned to the same unit they are properly sequenced using variable $Y_{ii'k}$. Finally, satisfaction of the time window for every batch is achieved by enforcing equations (3.5) and (3.6).

$$C_{i(k+\nu_{ik})} \geq C_{ik} + \sum_{j \in \mathbf{J}_{i(k+\nu_{ik})}} \bar{\tau}_{ij} X_{ij}^{C} \quad \forall i \in \mathbf{I}, k \in \mathbf{K}_i \backslash \{\kappa_i^{LS}\} \tag{3.1}$$

$$\sum_{j \in \mathbf{J}_{ik}} X_{ij}^{C} = 1 \quad \forall i \in \mathbf{I}, k \in \mathbf{K}_i \tag{3.2}$$

$$C_{i'k} - \bar{\tau}_{i'j} \geq C_{ik} - M\left[(1 - Y_{ii'k}) + \left(2 - X_{ij}^{C} - X_{i'j}^{C}\right)\right] \quad \forall (i,i') \in \mathbf{I}^2 : i' > i, k \in \mathbf{K}_i \cap \mathbf{K}_{i'}, j \in \mathbf{J}_{ik} \cap \mathbf{J}_{i'k} \tag{3.3}$$

$$C_{ik} - \bar{\tau}_{ij} \geq C_{i'k} - M\left[Y_{ii'k} + \left(2 - X_{ij}^{C} - X_{i'j}^{C}\right)\right] \quad \forall (i,i') \in \mathbf{I}^2 : i' > i, k \in \mathbf{K}_i \cap \mathbf{K}_{i'}, j \in \mathbf{J}_{ik} \cap \mathbf{J}_{i'k} \tag{3.4}$$

$$C_{i\kappa_i^{FS}} \geq \bar{\mu}_i + \sum_{j \in \mathbf{J}_{i\kappa_i^{FS}}} \bar{\tau}_{ij} X_{ij}^{C} \quad \forall i \in \mathbf{I} \tag{3.5}$$

$$C_{i\kappa_i^{LS}} \leq \bar{\phi}_i \quad \forall i \in \mathbf{I} \tag{3.6}$$

For the large value included in equations (3.3) and (3.4) we use the length of the scheduling horizon, i.e. $M = \max_{i \in \mathbf{I}} \bar{\phi}_i - \min_{i \in \mathbf{I}} \bar{\mu}_i$, although other values can be used without a significant change in computational performance.

Finally, equations (3.7), (3.8), and (3.9) provide definitions for the minimization objectives of cost, earliness and makespan respectively. Equation (3.9) introduces an additional decision variable to represent the makespan, $MS \in \mathbb{R}_+$.

$$\min \sum_{i \in \mathbf{I}, j \in \mathbf{J}_i} \alpha_{ij} X_{ij}^C \tag{3.7}$$

$$\min \sum_{i \in \mathbf{I}} \left(\bar{\phi}_i - C_{i\kappa_i^{LS}} \right) \tag{3.8}$$

$$\min MS : MS \geq C_{i\kappa_i^{LS}} \quad \forall i \in \mathbf{I} \tag{3.9}$$

Although we mainly use model MH&C to compare its performance versus the discrete-time models introduced in section 3.3, we also use it in section3.4.3 as a tool to refine solutions obtained by discrete-time models.

3.2.2. Discrete-time modeling in network environments

In Chapter 1 we discussed two different abstract representations that can be used as the basis to derive mathematical models for production scheduling in network environments: STN and RTN. Next, we provide MIP formulations for both representations.

State-Task Network

We consider the STN formulation proposed by Kondili et al.[15] and refined by Shah et al.[16], which henceforth is referred to as model SP&S. Next, we define the indices, sets, parameters and variables used by model SP&S. Superscript N indicates that a definition is general for network representations, while superscript STN indicates a definition is only valid for STN.

Sets and indices

$i \in \mathbf{I}^N$ Tasks

$j \in \mathbf{J}^{STN}$ Processing units

$m \in \mathbf{M}^{STN}$ Materials

$t \in \mathbf{T}^N$ Time points

$\mathbf{I}_j^{STN}$ Tasks that can be processed in unit $j \in \mathbf{J}^{STN}$

$\mathbf{I}_m^+/\mathbf{I}_m^-$ Tasks that produce/consume material $m \in \mathbf{M}^N$

$\mathbf{J}_i^{STN}$ Processing units that can process task $i \in \mathbf{I}^N$

Parameters

δ^N Step size for time discretization

$\bar{\eta}^N/\eta^N$ Scheduling horizon in real time units/in terms of time grid, $\eta^N = \lceil \bar{\eta}^N/\delta^N \rceil$

$\bar{\tau}_{ij}^{STN}/\tau_{ij}^{STN}$ Processing time of task $i \in \mathbf{I}^N$ in unit $j \in \mathbf{J}_i^{STN}$ in real time units/in terms of time grid, $\tau_{ij}^{STN} = \lceil \bar{\tau}_{ij}^{STN}/\delta^N \rceil$

ρ_{im}^{STN} Conversion coefficient for material $m \in \mathbf{M}^{STN}$ produced ($\rho_{im}^{STN} > 0$) or consumed ($\rho_{im}^{STN} < 0$) by task $i \in \mathbf{I}^N$

γ_m^{STN} Storage capacity for material $m \in \mathbf{M}^{STN}$

ξ_{mt}^{STN} Delivery ($\xi_{mt}^{STN} > 0$) or demand ($\xi_{mt}^{STN} < 0$) for material $m \in \mathbf{M}^{STN}$ at time $t \in \mathbf{T}^N$

$\beta_{ij}^{m,STN}/\beta_{ij}^{M,STN}$ Minimum/maximum batch size of task $i \in \mathbf{I}^N$ in unit $j \in \mathbf{J}_i^{STN}$

Decision variables

$X_{ijt}^{STN} \in \{0,1\}$ One if task $i \in \mathbf{I}^N$ starts in unit $j \in \mathbf{J}_i^{STN}$ at time $t \in \mathbf{T}^N$

$B_{ijt}^{STN} \geq 0$ Batch size of task $i \in \mathbf{I}^N$ that starts in unit $j \in \mathbf{J}_i^{STN}$ at time $t \in \mathbf{T}^N$

$S_{mt}^{STN} \geq 0$ Inventory level of material $m \in \mathbf{M}^{STN}$ during interval $t \in \mathbf{T}^N \backslash \{0\}$

The set of time points is defined as $\mathbf{T}^N = \{t \in \mathbb{Z}: 0 \leq t \leq \eta^N\}$. Model SP&S is defined through equations (3.10)-(3.12). Equation (3.10) is a clique constraint that ensures a unit can only perform one task at a given time, equation (3.11) guarantees the batch sizes satisfy unit capacities, and equation (3.12) enforces material balances and storage capacities.

$$\sum_{i\in\mathbf{I}_j^{STN}}\sum_{t'=t-\tau_{ij}^{STN}+1}^{t} X_{ijt'}^{STN} \le 1 \quad \forall j \in \mathbf{J}^{STN}, t \in \mathbf{T}^N \tag{3.10}$$

$$\beta_{ij}^{m,STN} X_{ijt}^{STN} \le B_{ijt}^{STN} \le \beta_{ij}^{M,STN} X_{ijt}^{STN} \quad \forall i \in \mathbf{I}^N, j \in \mathbf{J}_i^{STN}, t \in \mathbf{T}^N \tag{3.11}$$

$$S_{m(t+1)}^{STN} = S_{mt}^{STN} + \sum_{i\in\mathbf{I}_m^+, j\in\mathbf{J}_i^{STN}} \rho_{ik}^{STN} B_{ij\left(t-\tau_{ij}^{STN}\right)}^{STN} + \sum_{i\in\mathbf{I}_m^-, j\in\mathbf{J}_i^{STN}} \rho_{ik}^{STN} B_{ijt}^{STN} + \xi_{mt}^{STN} \le \gamma_m^{STN} \quad \forall m \in \mathbf{M}^{STN}, t \in \mathbf{T}^N\backslash\{\eta^N\} \tag{3.12}$$

Resource-Task Network

The RTN-based model we present here was first introduced by Pantelides[17], and is referred to as model PRTN. Next, we define additional/modified sets, parameters and variables used in this formulation. Note that, although we use the same index/set for RTN tasks, their definition is broader than STN tasks. Under the RTN representation, a task is an abstract operation that interacts (uses, consumes, produces) with a specific set of resources.

Sets and indices

$r \in \mathbf{R}^{RTN}$	General resources (materials, units, utilities, manpower, etc.)
$\mathbf{I}_r^{RTN}$	Tasks that interact with resource $r \in \mathbf{R}^{RTN}$

Parameters

$\bar{\tau}_i^{RTN}/\tau_i^{RTN}$	Duration of task $i \in \mathbf{I}^N$ in real time units/in terms of time grid, $\tau_i^{RTN} = \left\lceil \bar{\tau}_{ij}^{RTN}/\delta^N \right\rceil$
$\chi_{irt}^{RTN}/\rho_{irt}^{RTN}$	Number of discrete/normalized continuous units of resource $r \in \mathbf{R}^{RTN}$ that interact with task $i \in \mathbf{I}_r^{RTN}$ during interval $t \in \mathbf{T}^N\backslash\{0\}$
γ_{rt}^{RTN}	Maximum storage allowed for resource $r \in \mathbf{R}^{RTN}$ during interval $t \in \mathbf{T}^N\backslash\{0\}$
ξ_{rt}^{RTN}	Amount of resource $r \in \mathbf{R}^{RTN}$ made available ($\xi_{rt}^{RTN} > 0$) or unavailable ($\xi_{rt}^{RTN} < 0$) at time $t \in \mathbf{T}^N$
$\beta_{ir}^{m,RTN}/\beta_{ir}^{M,RTN}$	Minimum/maximum capacity utilization of resource $r \in \mathbf{R}^{RTN}$ by task $i \in \mathbf{I}_r^{RTN}$

Decision variables

$X_{it}^{RTN} \in \{0,1\}$ Discrete extent. It is equal to one if task $i \in \mathbf{I}^N$ starts at time $t \in \mathbf{T}^N$

$B_{it}^{RTN} \geq 0$ Continuous extent. $\rho_{irt}^{RTN} B_{it}^{RTN}$ represents the total number of continuous units of resource $r \in \mathbf{R}^{RTN}$ that interact with task $i \in \mathbf{I}_r^{RTN}$ during interval $t \in \mathbf{T}^N \backslash \{0\}$

$R_{rt}^{RTN} \geq 0$ Excess resource $r \in \mathbf{R}^{RTN}$ during interval $t \in \mathbf{T}^N \backslash \{0\}$

Model PRTN is defined through equations (3.13) and (3.14). The former is an excess resource balance and the latter enforces resource capacity satisfaction. Note that equations (3.10) and (3.12) are particular cases of equation (3.13).

$$R_{r(t+1)}^{RTN} = R_{rt}^{RTN} + \sum_{i \in \mathbf{I}_r^{RTN}} \sum_{t'=0}^{\tau_i^{RTN}} \left(\chi_{irt'}^{RTN} X_{i(t-t')}^{RTN} + \rho_{irt'}^{RTN} B_{i(t-t')}^{RTN} \right) + \xi_{rt}^{RTN} \leq \gamma_{rt}^{RTN} \quad \forall r \in \mathbf{R}^{RTN}, t \in \mathbf{T}^N \backslash \{\eta^N\} \tag{3.13}$$

$$\beta_{ir}^{m,RTN} X_{it}^{RTN} \leq B_{it}^{RTN} \leq \beta_{ir}^{M,RTN} X_{it}^{RTN} \quad \forall r \in \mathbf{R}^{RTN}, i \in \mathbf{I}_r^{RTN}, t \in \mathbf{T}^N \tag{3.14}$$

3.2.3. Resource-Constrained Project Scheduling

The Resource-Constrained Project Scheduling Problem (RCPSP) is a classic scheduling problem in the area of project management. It consists of a set of activities with fixed duration that must be scheduled subject to precedence and resource constraints such that a project is completed in the shortest possible time. A vast literature has been dedicated to the RCPSP, in order to study different variants in its three major components, namely, activities, precedence relations and resources. Several reviews on the RCPSP problem are available to summarize different research directions[79-81].

The RCPSP has also attracted the attention of researchers in PSE, because of its applications in process-related projects such as plant retrofitting and product development. In particular, several works have applied modifications of the RCPSP to the problem of scheduling testing tasks in the development of new agricultural and pharmaceutical products[82-85]. A different line of work has been devoted to derive new discrete- and continuous-time formulations for the RCPSP by extending concepts originally developed for the process scheduling problem[86, 87].

A generalization of the RCPSP is the so-called Multimode RCPSP (MRCPSP), in which each activity is allowed to be executed using different *modes*[88]. Each activity mode is characterized by a different duration depending on the resources it utilizes. Several research efforts have been dedicated to the MRCPSP both in the OR[89] and the PSE communities[86, 90]. The MRCPSP is closely related to the MBPSP, since modes can be assimilated to units. However, our derivations are based on the original RCPSP, so the concepts and extensions we present can be clearly illustrated.

In this chapter, we are interested in using the basic ideas behind the precedence relations of the RCPSP to extend them and derive batch precedence constraints for the MBPSP. Next, we define the relevant sets and parameters that are included in the formal definition of the RCPSP.

Sets and indices

$a \in \mathbf{A}$	Activities
$r \in \mathbf{R}^P$	Resources
$\mathbf{E}$	Precedence relations, $\mathbf{E} = \{(a, a') \in \mathbf{A}^2 : a \text{ is executed before } a'\}$

Parameters

β_r^P	Capacity of resource $r \in \mathbf{R}^P$
$\beta_{ar}^{A,P}$	Requirement of resource $r \in \mathbf{R}^P$ required by activity $a \in \mathbf{A}$
τ_a^P	Duration of activity $a \in \mathbf{A}$

The first MIP model proposed to address the RCPSP was developed in the late 1960s[91]. However, restricted computational capabilities at the time put a halt on the use of mathematical programming to address the RCPSP and led to the development of other exact and approximate methods[81]. This model is now known as the aggregated discrete-time MIP formulation for the RCPSP.

Being a discrete time approach, a set of time points has to be defined. In general, the literature on RCPSP assumes that the activity durations are integer-valued so the standard step size used is one.

Let $\mathbf{T}^P = \{t \in \mathbb{Z}: 0 \leq t \leq \eta^P\}$ define the set of time points, where η^P is an estimate of the project duration.

Two dummy activities, $a = 0$ and $a = |\mathbf{A}| + 1$ are added to the set **A**. They respectively represent the formal start and end of the project. In order to minimize the duration of the project, the start time of activity $|\mathbf{A}| + 1$ minimized, i.e. a makespan minimization problem is solved.

The decision to assign an activity a to time interval t is made via a binary variable defined as $X_{at}^P \in \{0,1\}\ \forall a \in A \cup \{0, |\mathbf{A}| + 1\}, t \in \mathbf{T}^P$. Equations (3.15)-(3.18) define the aggregated RCPSP model. Equation (3.15) enforces precedence requirements by guaranteeing that the start time of a successor is greater than the finish time of a predecessor; equation (3.16) assigns the appropriate number of units of a resource to every activity, while satisfying resource capacity; equation (3.17) guarantees that every activity is executed once during the project; equation (3.18) defines the objective function.

$$\sum_{t \in \mathbf{T}^P} t X_{a't}^P \geq \sum_{t \in \mathbf{T}^P} t X_{at}^P + \tau_a^P \quad \forall (a, a') \in \mathbf{E} \tag{3.15}$$

$$\sum_{a \in \mathbf{A}} \sum_{t'=t-\tau_a^P+1}^{t} \beta_{ar}^{A,P} X_{at}^P \leq \beta_r^P \quad \forall t \in \mathbf{T}^P, r \in \mathbf{R}^P \tag{3.16}$$

$$\sum_{t \in \mathbf{T}^P} X_{at}^P = 1 \quad \forall a \in \mathbf{A} \cup \{0, |\mathbf{A}| + 1\} \tag{3.17}$$

$$\min \sum_{t \in \mathbf{T}^P} t X_{(|\mathbf{A}|+1)t}^P \tag{3.18}$$

This model is termed "aggregated" because the precedence constraint is only written for elements of the set **E** and the summations are made over time. By the end of the 1980s, a new model was proposed and compared to the original formulation[92]. The only difference is how the precedence is enforced. In this "disaggregated" model, precedence constraints are written for every pair of

activities in **E** and every time point. This modification results in a larger formulation, but it improves its relaxation and computational performance[92]. Equation (3.19) expresses this new constraint; it establishes that if $a, a' \in \mathbf{A}$ are project activities such that $(a, a') \in \mathbf{E}$ and activity a starts at time t then activity a' cannot start before time $t + \tau_a^P$ and vice versa.

$$\sum_{t'=t}^{\eta^P} X_{at'}^P + \sum_{t'=0}^{t+\tau_a^P-1} X_{a't'}^P \leq 1 \quad \forall (a, a') \in \mathbf{E}, t \in \mathbf{T}^P \tag{3.19}$$

The disaggregated RCPSP model is defined through equations (3.16)-(3.19). In section 3.3 we use the ideas introduced in equations (3.15) and (3.19) to derive constraints that enforce batch precedence in the MBPSP.

3.2.4. Solution methods for MIP models in network environments

Models SP&S and PRTN are rather simple formulations with only two or three basic equations needed to address the complex problem of scheduling in network environments. Despite their compact form, straightforward conceptual relation to actual restrictions of the production environment, and extendibility to most of the common scheduling features, their size (i.e. number of equations and variables) is considerably larger than their continuous-time counterparts[8, 9]. In principle, this might translate into significantly more computational effort if only commercially available solution schemes are applied. This is the main reason why researchers in the PSE area focused on the development of continuous-time models during the 1990s and 2000s and abandoned the discrete-time formulation because it was considered computationally intractable. However, most of the continuous-time models proposed do not achieve the level of generality models SP&S and PRTN provide.

During the past five years Maravelias and coworkers[51, 55, 59, 70] have presented several works that revisit discrete-time formulations and developed different solution methods that can be used to improve their computational performance. They have actually showed by means of extensive

computational studies that when models SP&S and PRTN are coupled with these solution strategies, significant improvement in computational tractability is achieved; in many instances, the results are orders of magnitude better than those obtained by continuous-time formulations. In addition, the methods they propose are shown to have better synergy with discrete-time models since the results with continuous-time counterparts are inferior, as discussed in Chapter 4. In this section we present an overview of these methods, whereas full details can be found in the references provided.

Reformulations

Velez and Maravelias[51] introduced a conceptually simple yet computationally powerful reformulation to model SP&S. It is based on introducing a new integer decision variable in order to force the branch-and-cut algorithm to generate a tree based on this new variable, thus reducing its size and the CPU time spent in proving optimality. This reformulation uses the number of batches of a given task $i \in \mathbf{I}^N$ in a particular unit $j \in \mathbf{J}_i^{STN}$, denoted as N_{ij}^{STN} as a variable, since it characterizes any given feasible schedule more effectively than the assignment variable X_{ijt}^{STN}. The fact is that multiple assignments can lead to the same number of batches processed in a unit. By branching on N_{ij}^{STN} at a given node, it is possible to prune multiple assignments at once.

Equation (3.20) defines and provides an upper bound for the new integer variable. This equation can be directly added to the SP&S formulation, and up to four orders of magnitude reduction in CPU time can be achieved as it has been observed in specific instances. Section 3.6 includes reformulation strategies applied to the MBPSP problem to complement the tightening methods we propose.

$$N_{ij}^{STN} = \sum_{t \in \mathbf{T}^N} X_{ijt}^{STN} \leq \left\lfloor \frac{\eta^N}{\tau_{ij}^{STN}} \right\rfloor \quad \forall i \in \mathbf{I}^N, j \in \mathbf{J}_i^{STN} \tag{3.20}$$

Tightening methods

Velez et al.[59] proposed a family of tightening constraints for minimization problems in network environments, based on the SP&S model. The basic premise on which these constraints are based is

the calculation of appropriate parameters that reflect external inputs which restrict the production of materials in the network. In general, the objective functions are formulated as linear combinations of the assignment variables X_{ijt}^{STN}. When minimization problems are considered, those combinations are "pushed" to be as small as possible. However, an external restriction based on customer demands implicitly determines the minimum number of variables that can vanish. The goal of these parameters is to explicitly quantify those restrictions by imposing lower bounds on the linear combinations and include them as constraints in the model. In particular, the authors defined two main parameters to quantify this effects: the minimum cumulative production of tasks and materials required to satisfy customer demand. The calculation of parameters for all tasks and materials is achieved through an algorithm that backward-propagates the known values for demand of final products.

Following these ideas, we proposed families of algorithms, parameters and constraints targeting maximization problems in network environments in Chapter 2. In this case, the external restrictions come from initial inventory and available time. Although the maximization tries to "push" linear combinations of X_{ijt}^{STN} to be as large as possible, it is possible to define upper bounds on these combinations. Analogous parameters are introduced to represent the maximum cumulative production of tasks and materials attainable with the available inventory and time. A forward-propagation algorithm propagates data on initial inventories for materials and processing times for tasks to calculate the parameter values for all tasks and materials.

There is a fundamental difference between the approaches just described for minimization and maximization problems in network environments. While the former only considers *amount-based* information, the latter also uses *time-based* data. This distinction is the key to define which methods can be extended to the sequential environments we consider in this chapter. In the MBPSP, decisions concerning amounts (i.e. the batching problem) are taken beforehand. Thus, the scheduling problem

does not require the amounts to be modeled and only time-based decisions are important. Therefore, only the methods developed for maximization problems in network environments are applicable to sequential environments. In Chapter 2 we consider the calculation of explicit time windows for each unit and task-unit combination to impose upper bounds on linear combinations of X_{ijt}^{STN}. Next, a summary of the parameters required to define these time windows is given using the notation of this chapter.

$\varepsilon_{ij}^{U,N}$ Earliest start time of task $i \in \mathbf{I}^N$ in unit $j \in \mathbf{J}_i^{STN}$

ε_j^N Earliest start time of unit $j \in \mathbf{J}^{STN}$, $\varepsilon_j^N = \min_{i \in \mathbf{I}_j^{STN}} \varepsilon_{ij}^{U,N}$

$\sigma_{ij}^{U,N}$ Shortest tail of task $i \in \mathbf{I}^N$ in unit $j \in \mathbf{J}_i^{STN}$, i.e. minimum time to the end of the horizon from the finishing of execution of i in j so that it leads to final products.

σ_j^N Shortest tail of unit $j \in \mathbf{J}^{STN}$, $\sigma_j^N = \min_{i \in \mathbf{I}_j^{STN}} \sigma_{ij}^{U,N}$

$\theta_{ij}^{U,N}$ Available time window for task $i \in \mathbf{I}^N$ in unit $j \in \mathbf{J}_i^{STN}$, $\theta_{ij}^{U,N} = \eta^N - \varepsilon_{ij}^{U,N} - \sigma_{ij}^{U,N}$

θ_j^N Available time window for unit $j \in \mathbf{J}^{STN}$, $\theta_j^N = \eta^N - \varepsilon_j^N - \sigma_j^N$

Equations (3.21) and (3.22) define the tightening constraints used by these methods. The calculation of earliest start times and shortest tails is based on processing times, as well as initial inventory and unit capacities. Full details are in Chapter 2. The corresponding formulations for sequential environments are further discussed in section 3.4.1.

$$\sum_{t \in \mathbf{T}^N} \tau_{ij}^{STN} X_{ijt}^{STN} \leq \theta_{ij}^{U,N} \quad \forall i \in \mathbf{I}^N, j \in \mathbf{J}_i^{STN} \tag{3.21}$$

$$\sum_{i \in \mathbf{I}_j^{STN}, t \in \mathbf{T}^N} \tau_{ij}^{STN} X_{ijt}^{STN} \leq \theta_j^N \quad \forall j \in \mathbf{J}^{STN} \tag{3.22}$$

Variable time windows

In Chapter 2 we also propose an extension that introduces additional decision variables instead of the fixed parameters defining earliest start time and shortest tail. Let $E_{ij}^{T,N}$, $S_{ij}^{T,N}$ respectively denote the actual start time and tail of task $i \in \mathbf{I}^N$ in unit $j \in \mathbf{J}_i^{STN}$. Equations (3.23) and (3.24) establish the relation between variables and parameters defining the time windows. Equation (3.25) generalizes equation (3.21) to variable start time and tail.

$$E_{ij}^{T,N} \geq \varepsilon_{ij}^{U,N} \quad \forall i \in \mathbf{I}^N, j \in \mathbf{J}_i^{STN} \tag{3.23}$$

$$S_{ij}^{T,N} \geq \sigma_{ij}^{U,N} \quad \forall i \in \mathbf{I}^N, j \in \mathbf{J}_i^{STN} \tag{3.24}$$

$$\sum_{t \in \mathbf{T}^N} \tau_{ij}^{STN} X_{ijt}^{STN} \leq \eta^N - E_{ij}^{T,N} - S_{ij}^{T,N} \quad \forall i \in \mathbf{I}^N, j \in \mathbf{J}_i^N \tag{3.25}$$

Additional constraints relating these new variables are defined in order to achieve the tightening effect. In section 3.4.2 we discuss how these constraints are extended to sequential environments.

3.3. Discrete-Time Mathematical Programming Models

In section 3.1 we introduced the problem under consideration and formally defined the major constraints a feasible schedule must satisfy, namely batch precedence, batch-stage assignment, and unit assignment. This section presents four different MIP models that enforce such constraints and the different objective functions we consider. These models implicitly enforce the condition that every batch has to be processed within its time window by defining appropriate time sets that restrict the number of decision variables for a given batch. Next, we define additional sets and parameters that are used across the four discrete-time models we present in this section.

Sets

$\mathbf{K}_{ik}^-$ Stages in which batch $i \in \mathbf{I}$ has to be processed before stage $k \in \mathbf{K}_i$, $\mathbf{K}_{ik}^- \equiv \{k' \in \mathbf{K}_i : k' < k\}$

$\mathbf{K}_{ik}^+$ Stages in which batch $i \in \mathbf{I}$ has to be processed after stage $k \in \mathbf{K}_i$, $\mathbf{K}_{ik}^+ \equiv \{k' \in \mathbf{K}_i : k' > k\}$

Parameters

δ	Step size for time discretization
ε_j	Earliest start time for unit $j \in \mathbf{J}$
$\varepsilon_{ij}^U / \varepsilon_{ik}^S$	Earliest start time for batch $i \in \mathbf{I}$ in unit $j \in \mathbf{J}_i$/on stage $k \in \mathbf{K}_i$
λ_j	Latest finish time for unit $j \in \mathbf{J}$
$\lambda_{ij}^U / \lambda_{ik}^S$	Latest finish time for batch $i \in \mathbf{I}$ in unit $j \in \mathbf{J}_i$/on stage $k \in \mathbf{K}_i$
μ_i	Release date for batch $i \in \mathbf{I}$ in terms of time grid, $\mu_i \equiv \lceil \bar{\mu}_i / \delta \rceil$
τ_{ij}	Processing time for batch $i \in \mathbf{I}$ in unit $j \in \mathbf{J}_i$ in terms of time grid, $\tau_{ij} \equiv \left\lceil \bar{\tau}_{ij} / \delta \right\rceil$
τ^0	Start of the scheduling horizon, $\tau^0 \equiv \min_{i \in \mathbf{I}} \mu_i$
τ^F	End of the scheduling horizon, $\tau^F \equiv \max_{i \in \mathbf{I}} \phi_i$
ϕ_i	Due date for batch $i \in \mathbf{I}$ in terms of time grid, $\phi_i \equiv \left\lfloor \bar{\phi}_i / \delta \right\rfloor$

Each of the four MIP models discussed in this section provide different mathematical formulations of these three constraints. A computational comparison between them is presented in section 3.6.

3.3.1. Modeling of time

We use the index/set notation $t \in \mathbf{T}$ to denote the position of a time point. The set of time points is defined as the horizon between the minimum release date and the maximum due date as expressed by equation (3.26). This horizon is divided into intervals of equal length δ, such that the position of time value $\bar{t}$ in the grid is given by $\bar{t} = \delta t$. We also use the elements of set $\mathbf{T} \backslash \{0\}$ to denote the position of time intervals, with interval t running between time points $t - 1$ and t. Figure 3.2 shows the time grid we use.

$$\mathbf{T} = \{t \in \mathbb{Z}_+ : \tau^0 \leq t \leq \tau^F\} \tag{3.26}$$

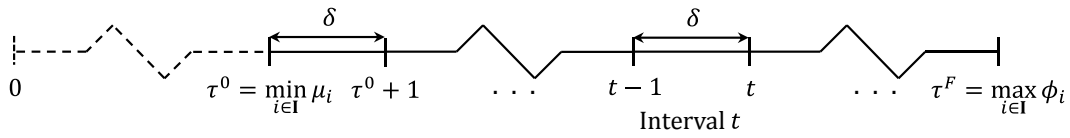


Figure 3.2. Global uniform grid used for discrete-time representation

Since every batch has its own time window which has to be implicitly enforced, we need to take into consideration the earliest start and latest finishing time of a batch in specific units/stages. Thus, we can define the set of time points on which the processing of a particular batch is feasible.

First, we start by defining the earliest start and latest finishing times of batch $i \in \mathbf{I}$ on every stage $k \in \mathbf{K}_i$ and denote them respectively with ε_{ik}^S and λ_{ik}^S. It is clear that the earliest batch i can start on its first stage κ_i^{FS} is given by its release date, $\varepsilon_{i\kappa_i^{FS}}^S = \mu_i \; \forall i$. Similarly, the latest batch i can finish on its last stage κ_i^{LS} is given by its due date, $\lambda_{i\kappa_i^{LS}}^S = \phi_i \; \forall i$. Equations (3.27) and (3.28) provide the definitions for all stages on which batch i is processed.

$$\varepsilon_{ik}^S = \mu_i + \sum_{k' \in \mathbf{K}_{ik}^-} \min_{j' \in \mathbf{J}_{ik'}} \tau_{ij'} \quad \forall i \in \mathbf{I}, k \in \mathbf{K}_i \tag{3.27}$$

$$\lambda_{ik}^S = \phi_i - \sum_{k' \in \mathbf{K}_{ik}^+} \min_{j' \in \mathbf{J}_{ik'}} \tau_{ij'} \quad \forall i \in \mathbf{I}, k \in \mathbf{K}_i \tag{3.28}$$

Second, we note that the earliest (latest) batch $i \in \mathbf{I}$ can start (finish) on a given unit $j \in \mathbf{J}_i$ only depends on the stage to which unit j belongs. Equation (3.29) provides the definition of ε_{ij}^U, the earliest start time for batch i in unit j, whereas equation (3.30) introduces λ_{ij}^U, the latest finish time for batch i in unit j.

$$\varepsilon_{ij}^{U} = \varepsilon_{ik}^{S} \quad \forall i \in \mathbf{I}, k \in \mathbf{K}_i, j \in \mathbf{J}_{ik} \tag{3.29}$$

$$\lambda_{ij}^{U} = \lambda_{ik}^{S} \quad \forall i \in \mathbf{I}, k \in \mathbf{K}_i, j \in \mathbf{J}_{ik} \tag{3.30}$$

Third, we can extend the concept of time window to specific units, which will be helpful in defining tightening constraints as explained in section 3.4.1. Equations (3.31) and (3.32) respectively define the earliest start time, ε_j, and latest finishing time, λ_j for unit $j \in \mathbf{J}$.

$$\varepsilon_j = \min_{i \in \mathbf{I}_j} \varepsilon_{ij}^{U} \quad \forall j \in \mathbf{J} \tag{3.31}$$

$$\lambda_j = \max_{i \in \mathbf{I}_j} \lambda_{ij}^{U} \quad \forall j \in \mathbf{J} \tag{3.32}$$

We can now formally define the set of feasible time points at which batch $i \in \mathbf{I}$ is allowed to start its execution in unit $j \in \mathbf{J}_i$, denoted by $\mathbf{T}_{ij}$, through equation (3.33)

$$\mathbf{T}_{ij} = \left\{t \in \mathbf{T}: \varepsilon_{ij} \leq t \leq \lambda_{ij} - \tau_{ij}\right\} \quad \forall i \in \mathbf{I}, j \in \mathbf{J}_i \tag{3.33}$$

Two related sets can be defined for batches and units[26]. Equation (3.34) defines $\mathbf{I}_{jt}$, the set of batches that can start in a given unit at a specific time point, whereas equation (3.35) defines $\mathbf{J}_{it}$, the set of units that can start processing a given batch at a specific time point.

$$\mathbf{I}_{jt} = \left\{i \in \mathbf{I}_j: \varepsilon_{ij} \leq t \leq \lambda_{ij} - \tau_{ij}\right\} \quad \forall j \in \mathbf{J}, t \in \left[\varepsilon_j, \lambda_j\right] \cap \mathbb{Z} \tag{3.34}$$

$$\mathbf{J}_{it} = \left\{j \in \mathbf{J}_i: \varepsilon_{ij} \leq t \leq \lambda_{ij} - \tau_{ij}\right\} \quad \forall i \in \mathbf{I}, t \in [\mu_i, \phi_i] \cap \mathbb{Z} \tag{3.35}$$

3.3.2. Batch-stage assignment

All four discrete-time models we present use a common assignment binary variable, X_{ijt}, which takes the value of one if batch $i \in \mathbf{I}$ starts being processed in unit $j \in \mathbf{J}_i$ at time point $t \in \mathbf{T}_{ij}$ and zero otherwise. Note that this decision variable is not indexed over the set of stages, since there is an implicit correspondence between units and stages as defined by $\mathbf{J}_k$.

In terms of constraints, the four models use a common expression to enforce the batch-stage assignment, as defined in equation (3.36).

$$\sum_{j\in\mathbf{J}_{ik},t\in\mathbf{T}_{ij}} X_{ijt} = 1 \quad \forall i \in \mathbf{I}, k \in \mathbf{K}_i \tag{3.36}$$

Next we present the four discrete-time models we study, which differ on the way they model the batch precedence and unit assignment.

3.3.3. STN-based model

Models based on the State-Task Network representation of a chemical facility rely on a clique constraint for the unit assignment and a material balance for the sequencing among different units. Although the clique constraint can readily be extended to sequential environments, the material balances cannot be expressed in terms of amounts, since they are not explicitly modeled in sequential processing. Instead, we can use the batches themselves as indicators of production of a particular batch after being processed on a given stage.

To provide a STN representation of a sequential facility, we define task $\mathbb{T}(i,k)$ as the processing of batch $i \in \mathbf{I}$ on any compatible unit of stage $k \in \mathbf{K}_i$ and state $\mathbb{S}(i,k)$ as the material obtained after batch i has been processed on stage k. Figure 3.3 depicts the STN representation of a multistage plant with tasks represented as rectangles and states as circles. Only a general batch $i \in \mathbf{I}$ is presented in Figure 3.3; it is understood that the complete facility is comprised of $|\mathbf{I}|$ parallel structures such the one depicted

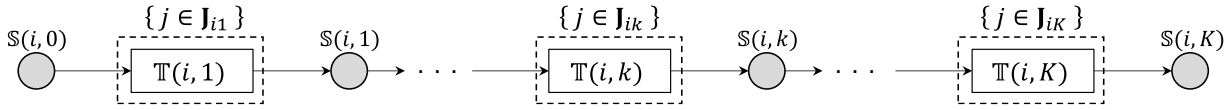


Figure 3.3. STN representation of a multistage facility.
Circles represent states, white boxes denote tasks, and dotted-line boxes define unit compatibility for a given task. Only a generic batch is depicted; the complete facility is represented by I parallel structures. $I \equiv |\mathbf{I}|$ denotes the total number of batches and $K \equiv |\mathbf{K}|$ defines the number of stages.

The unit assignment for the STN representation of the multistage batch process is expressed through equation (3.37), which is a clique constraint over time and is defined on the time window of unit $j \in \mathbf{J}$.

$$\sum_{i \in \mathbf{I}_j} \sum_{t'=t-\tau_{ij}+1}^{t} X_{ijt} \leq 1 \quad \forall j \in \mathbf{J}, t \in \left[\varepsilon_j, \lambda_j\right] \cap \mathbb{Z} \tag{3.37}$$

Batch precedence is enforced implicitly by requiring batch balance satisfaction at each time point. Let us define S_{ikt} as a binary variable to indicate whether or not material $\mathbb{S}(i,k)$ is present during interval $t \in \mathbf{T}\backslash\{\tau^0\}$. The batch balance expressed in equation (3.38) requires that a batch is present during interval $t+1$ only if it was present during interval t or if it is produced at time point t. It also requires that the batch is not present during interval $t+1$ if it is consumed at time point t. It is further assumed that material $\mathbb{S}(i,k)$ is not present at the beginning of the horizon.

$$S_{ik(t+1)} = S_{ikt}|_{t>\tau^0} + \sum_{j \in \mathbf{J}_{i(t-\tau_{ij})} \cap \mathbf{J}_k} X_{ij(t-\tau_{ij})} - \sum_{j \in \mathbf{J}_{it} \cap \mathbf{J}_{k+\nu_{ik}}} X_{ijt} \quad \forall i \in \mathbf{I}, k \in \mathbf{K}_i, t \in \mathbf{T}\backslash\{\tau^F\} \tag{3.38}$$

The STN-based model, hereinafter referred to as model M1, is given by equations (3.36), (3.37), and (3.38) plus the objective function.

3.3.4. RTN-based model

Castro and Grossmann[26, 93] extended the ideas about models based on the Resource-Task Network to sequential environments. Once again, the difference with respect to the STN-based model is that the units are explicitly considered as resources, instead of being implicitly mapped to tasks. The batch balance introduced in equation (3.38) remains valid to enforce batch precedence, but now a processing unit balance needs to be included to guarantee the unit assignment is satisfied.

The RTN representation of a sequential process requires the definition of tasks that explicitly include the unit in which they are being executed. Let $\mathbb{T}(i,j)$ be the task defined when batch $i \in \mathbf{I}$ is processed in unit $j \in \mathbf{J}_i$ that spans τ_{ij} time intervals. There are now two types of resources: states

$\mathbb{S}(i,k)$ defined the same way they were introduced for the STN representation, and units $\mathbb{U}_j: j \in \mathbf{J}$. Figure 3.4 shows the RTN representation of a multistage plant with tasks represented as rectangles and resources as circles. Similarly to Figure 3.3, only a general batch $i \in \mathbf{I}$ is presented in Figure 3.4; it is understood that the complete facility is comprised of $|\mathbf{I}|$ parallel structures such the one depicted.

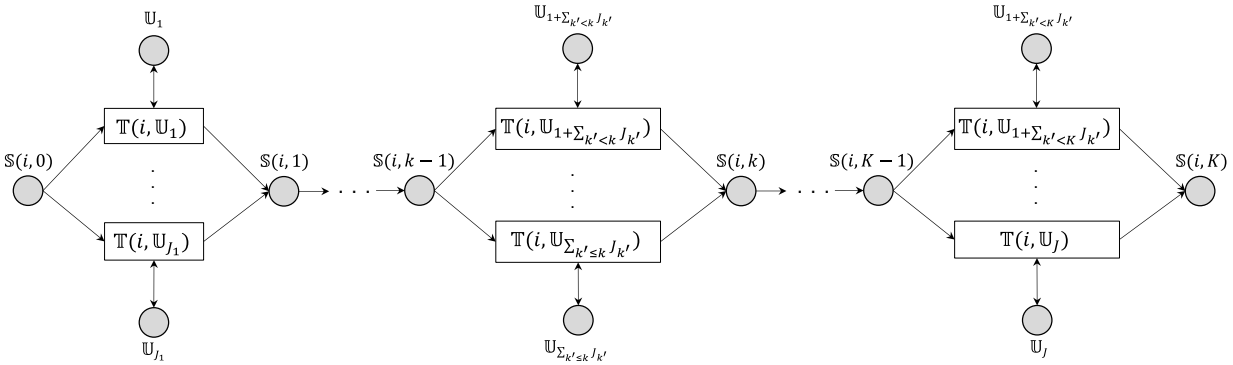


Figure 3.4. RTN representation of a multistage facility.
Circles represent resources (materials and units) and white boxes denote tasks. Only a generic batch is depicted; the complete facility is represented by I parallel structures. $I \equiv |\boldsymbol{I}|$ is the total number of batches, $J \equiv |\boldsymbol{J}|$ denotes the total number of units, $K \equiv |\boldsymbol{K}|$ defines the number of stages, and $J_k \equiv |\boldsymbol{J}_k|$ is the total number of units on stage k.

Unit assignment is explicitly enforced through a balance on the resource "processing units." Let R_{jt} be a binary variable to decide if unit $j \in \mathbf{J}$ is available during time interval $t \in \mathbf{T}\backslash\{\tau^0\}$. Equation (3.39) expresses that unit j is available during interval $t+1$ if it was available during interval t or if it is released by any task at time t. This balance also enforces that the unit becomes unavailable if any task starts being processed at time t. At the beginning of the horizon all the units are available.

$$R_{j(t+1)} = 1|_{t=\tau^0} + R_{jt}\big|_{t>\tau^0} + \sum_{i\in\mathbf{I}_{j(t-\tau_{ij})}} X_{ij(t-\tau_{ij})} - \sum_{i\in\mathbf{I}_{jt}} X_{ijt} \quad \forall j \in \mathbf{J}, t \in \mathbf{T}\backslash\{\tau^F\} \tag{3.39}$$

The RTN-based model defined by equations (3.36), (3.38), and (3.39) plus the objective function is referred to as model M2 in the following.

3.3.5. RCPSP-based aggregated model

Section 3.2.3 presents basic concepts and models to solve the Resource-Constrained Project Scheduling Problem. Although a different problem when compared to the MBPSP, the RCPSP

introduces an interesting way to enforce precedence constraints between a specific activity and its successors. The underlying idea is the direct calculation of times at which an activity ends and its successors start. We now extend these ideas to the MBPSP. We make the following observations to relate elements of the RCPSP and the MBPSP:

(i) There exists a unique correspondence between a project activity $a \in \mathbf{A}$ on the RCPSP and a task $\mathbb{T}(i,k)$ on the MBPSP.

(ii) The fact that a pair of activities a and a' belong to the set of precedence constraints $\mathbf{E}$, i.e. $(a,a') \in \mathbf{E}$, translates into the fact that a batch requires to be processed in consecutive stages, i.e. there is a precedence requirement for $\left(\mathbb{T}(i,k), \mathbb{T}(i,k+\nu_{ik})\right)$.

(iii) Resources $r \in \mathbf{R}^P$ on the RCPSP correspond to processing units $j \in \mathbf{J}$ on the MBPSP. Therefore the number of available units of resource r, β_r^P is always one for MBPSP. Similarly, the number of units of resource r required by activity $a \in \mathbf{A}$, $\beta_{ar}^{A,P}$, is also one for the MBPSP since task $\mathbb{T}(i,k)$ only requires one processing unit to be executed.

(iv) The main difference between the RCPSP and the MBPSP is that the latter requires an additional decision in order to assign a batch to a specific processing unit. This distinction introduces the need for appropriate unit aggregation in the constraints. Notice, for instance, the difference between equations (3.17) and (3.36). An extra summation is required in (3.36) to account for assignment to different units. Unit aggregation proves to be a crucial modeling element as we discuss in the next subsections.

Observation (iii) immediately leads to an RCPSP-based expression for the unit assignment of the MBPSP. If we substitute $\beta_{ar}^{A,P} = 1$ and $\beta_r^P = 1$ into equation (3.16), we obtain a clique over t, an expression completely analogous to equation (3.37) which enforces unit assignment for model M1. We conclude that the clique constraint in MBPSP is a simplification of the resource utilization constraint in RCPSP. Consequently we only need to focus on developing a constraint to enforce batch

precedence for the MBPSP. Observations (ii) and (iv) are the key to obtain such constraint; they are used next to derive RCPSP-based batch precedence constraints.

To develop an aggregated formulation, we want to derive a constraint which guarantees that the start time of a batch is larger than the finish time of its immediate predecessor. Equation (3.15) for the RCPSP can be transformed into a batch precedence constraint for the MBPSP, by taking into consideration observations (i), (ii), and (iv).

The binary variable X_{at}^{P} for activity $a \in \mathbf{A}$ in the RHS summation of equation (3.15) for the RCPSP corresponds to variable X_{ijt} for task $\mathbb{T}(i,k)$ in the MBPSP, i.e. the processing of batch $i \in \mathbf{I}$ in *any* unit $j \in \mathbf{J}_{ik}$. The fact that the unit is arbitrary means that a summation over $j \in \mathbf{J}_{ik}$ must be introduced to ensure that only one X_{ijt} is equal to one to calculate the finish time of $\mathbb{T}(i,k)$. Similarly, variable $X_{a't}^{P}$ for activity $a' \in \mathbf{A}$ in the LHS summation of equation (3.15) assimilates to variable X_{ijt} for task $\mathbb{T}(i,k+\nu_{ik})$, the execution of batch $i \in \mathbf{I}$ in any unit $j \in \mathbf{J}_{i(k+\nu_{ik})}$. Once again, only one X_{ijt} should be nonzero when calculating the start time of $\mathbb{T}(i,k+\nu_{ik})$ and thus aggregation over $j \in \mathbf{J}_{i(k+\nu_{ik})}$ is required. The final form of the MBPSP batch precedence constraint is given by equation (3.40); it expresses that the start time of task $\mathbb{T}(i,k+\nu_{ik})$ is greater than or equal to the finish time of its predecessor $\mathbb{T}(i,k)$. The aggregation over compatible units derived from observation (iv) guarantees that this batch precedence relation is independent of the actual unit assignment. Figure 3.5 shows the correspondence between elements of the RCPSP and MBPSP that leads to equation (3.40).

$$\sum_{j\in\mathbf{J}_{i(k+\nu_{ik})},t\in\mathbf{T}_{ij}} tX_{ijt} \geq \sum_{j\in\mathbf{J}_{ik},t\in\mathbf{T}_{ij}} \left(t+\tau_{ij}\right)X_{ijt} \quad \forall i \in \mathbf{I}, k \in \mathbf{K}_i \backslash \left\{\kappa_i^{LS}\right\} \tag{3.40}$$

The formulation consisting of equations (3.36), (3.37), and (3.40) plus the objective function is henceforth referred to as model M3.

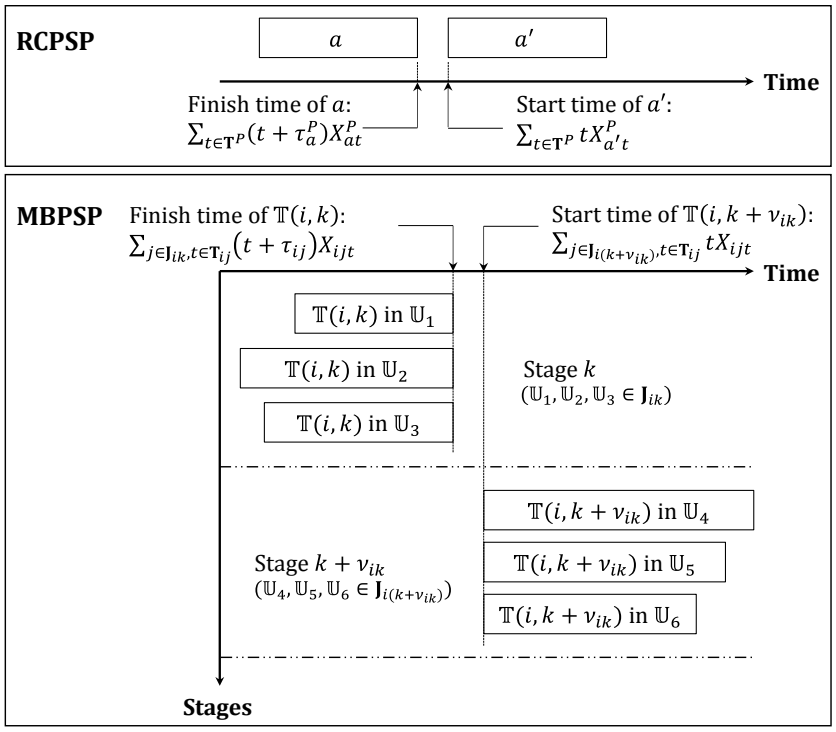


Figure 3.5. Time-aggregated RCPSP-based modeling of batch precedence constraints for MBPSP. Top and bottom boxes respectively represent precedence constraints for the RCPSP and the MBPSP. Note that the finish time of task $\mathbb{T}(i,k)$ and the start time of task $\mathbb{T}(i,k)$ are independent of the processing times of batch i on different units.

3.3.6. RCPSP-based disaggregated model

As discussed in section 3.2.3, the time-disaggregated version of the RCPSP model exhibits a tighter LP relaxation, although it has a larger size when compared to its aggregated counterpart. We now extend this idea to the MBPSP. The goal is to derive an expression analogous to equation (3.19), in order to establish a logical implication that relates the start times of a task and its successors.

We start by slightly modifying equation (3.19) in order to make it extendable to the MBPSP when processing units are introduced. The second term on the LHS of equation (3.19) only includes decision variables for the successor activity a'; however, the upper limit of this summation depends on the processing time for the predecessor activity a. This coupling prevents the unit aggregation required by observation (iv) above. Next, we redefine equation (3.19) to decouple the summations on the LHS.

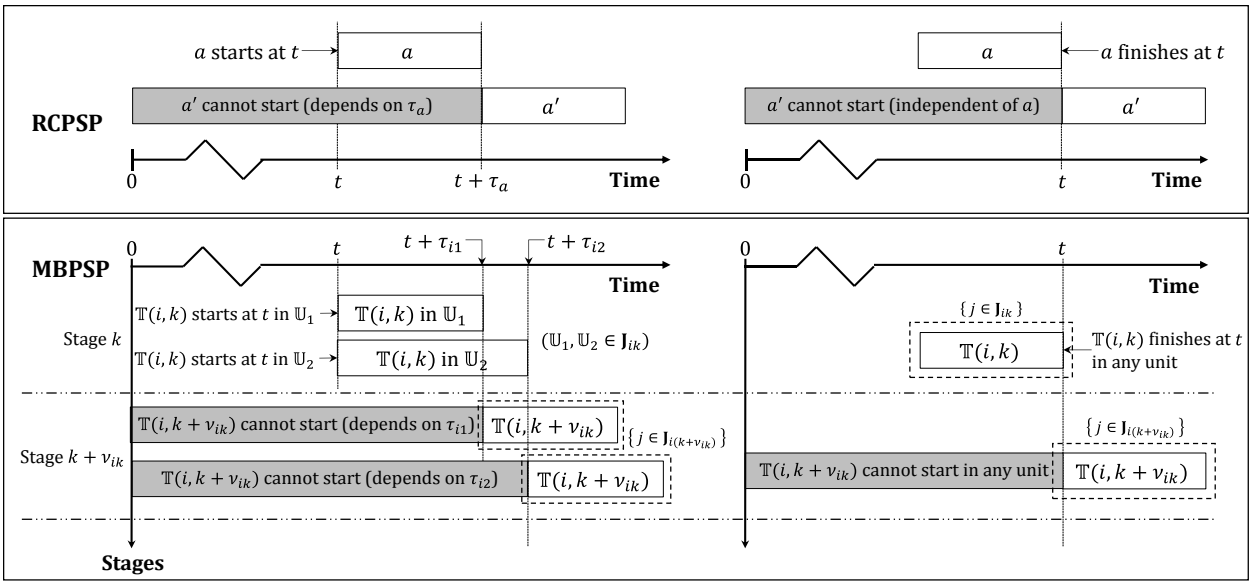
If $a, a' \in \mathbf{A}$ are project activities and $(a, a') \in \mathbf{E}$ then equation (3.41) establishes that if activity a finishes at time t then activity a' cannot start before time t and vice versa. In equation (3.41), each term on the LHS includes information for a single activity; therefore they are decoupled and unit aggregation is possible.

$$\sum_{t'=t}^{\eta^P} X^P_{a(t'-\tau_a)} + \sum_{t'=0}^{t-1} X^P_{a't'} \leq 1 \quad \forall (a, a') \in \mathbf{E}, t \in \mathbf{T}^P \tag{3.41}$$

Since activity a corresponds to task $\mathbb{T}(i,k)$, the binary variable $X^P_{a(t'-\tau_a)}$ in the first term of the LHS corresponds to variable $X_{ij(t'-\tau_{ij})}$ for the processing of batch $i \in \mathbf{I}$ in any unit $j \in \mathbf{J}_{ik}$. Aggregation over $j \in \mathbf{J}_{ik}$ is required to guarantee there is at most one nonzero assignment. Similarly, activity a' relates to task $\mathbb{T}(i, k+\nu_{ik})$; therefore variable $X^P_{a't'}$ in the second term of the LHS corresponds to variable $X_{ijt'}$ with $j \in \mathbf{J}_{i(k+\nu_{ik})}$ and aggregation over the processing units is required. Equation (3.42) is the precedence constraint for the disaggregated RCPSP-based model in MBPSP. It expresses that if task $\mathbb{T}(i,k)$ starts in any unit at time t, then its successor $\mathbb{T}(i, k+\nu_{ik})$ cannot start in any unit before t and vice versa. Figure 7 presents the conceptual extension that leads to equation (3.42), as well as an explanation of why decoupling is necessary.

$$\sum_{j \in \mathbf{J}_{ik}} \sum_{t'=t}^{\lambda_{ij}} X_{ij(t'-\tau_{ij})} + \sum_{j \in \mathbf{J}_{i(k+\nu_{ik})}} \sum_{t'=\varepsilon^S_{i(k+\nu_{ik})}}^{t-1} X_{ijt'} \leq 1 \tag{3.42}$$
$$\forall i \in \mathbf{I}, k \in K_i \backslash \{\kappa_i^{LS}\}, t \in [\varepsilon^S_{i(k+\nu_{ik})}, \lambda^S_{ik}] \cap \mathbb{Z}$$

The complete disaggregated model consists of equations (3.36), (3.37), and (3.42) plus the objective function. It is hereinafter referred to as model M4.



a) No precedence decoupling **b) Precedence decoupling**

Figure 3.6. Time-disaggregated RCPSP-based modeling of batch precedence constraints for MBPSP
Top and bottom boxes respectively represent precedence constraints for the RCPSP and the MBPSP. (a) Without precedence decoupling, (b) With precedence decoupling. If precedence decoupling is not used in the MBPSP (bottom left), then it is not possible to define a single window in which the successor task is not allowed to start. Therefore, aggregation on the units that process the predecessor task is not possible. This results in a larger and inefficient formulation for the batch precedence constraint.

3.3.7. Objective functions

Three different objective functions for minimization are considered in this chapter, namely cost, earliness, and makespan. Next, we provide the definitions of these objectives in terms of the assignment variable introduced for discrete-time models of the MBPSP.

Cost minimization is one of the main practical goals of any production facility. In this section we only consider processing costs, which are relevant for the basic MBPSP. Section 3.3.8 also considers utility costs. In addition, other types of costs such as inventory and switchovers could be included and directly related to decision variables. Let α_{ij} be the fixed cost of processing batch $i \in \mathbf{I}$ in unit $j \in \mathbf{J}_i$ which is added to the objective only of batch i is assigned to unit j at any time point $t \in \mathbf{T}_{ij}$. Equation (3.43) provides the definition of the cost minimization objective.

$$\min \sum_{i\in\mathbf{I}, j\in\mathbf{J}_i, t\in\mathbf{T}_{ij}} \alpha_{ij} X_{ijt} \tag{3.43}$$

Time-based objectives are very common in industry to ensure an adequate and efficient utilization of the installed equipment. They also are related to customer satisfaction and inventory management. An example of the latter is earliness minimization, which is sought with the aim of reducing storage costs. Earliness for a batch is defined as the positive difference between the due date and the actual finish date of that batch. Equation (3.44) gives one definition of the earliness minimization objective.

$$\min \sum_{i\in\mathbf{I}} \left(\bar{\phi}_i - \sum_{j\in\mathbf{J}_{i\kappa_i^{LS}}, t\in\mathbf{T}_{ij}} \left(\delta t + \bar{\tau}_{ij}\right) X_{ijt} \right) \tag{3.44}$$

The process makespan is defined as the total time it takes to complete the processing of all batches. We define it as a new variable $MS \in \mathbb{R}_+$, such that the makespan minimization objective is given by min MS. There are several ways to define MS, depending on which entity of the problem the definition is based. We present three variants: (i) based on batches, (ii) based on units with time disaggregation, and (iii) based on units with batch disaggregation.

Equation (3.45) provides the batch-based definition of makespan, guaranteeing that it is larger than the finish time of every batch.

$$MS \geq \sum_{j\in\mathbf{J}_{i\kappa_i^{LS}}, t\in\mathbf{T}_{ij}} \left(\delta t + \bar{\tau}_{ij}\right) X_{ijt} \quad \forall i \in \mathbf{I} \tag{3.45}$$

When a definition of makespan is to be based on processing units, it is necessary to disaggregate the constraints to avoid additional variables. Equations (3.46) and (3.47) ensure that the makespan is larger than the finish time of every unit. They provide unit-based definitions of makespan with time and batch disaggregation respectively.

$$MS \geq \sum_{i \in \mathbf{I}_{jt}} (\delta t + \bar{\tau}_{ij}) X_{ijt} \quad \forall j \in \mathbf{J}, t \in [\varepsilon_j, \lambda_j] \cap \mathbb{Z} \tag{3.46}$$

$$MS \geq \sum_{t \in \mathbf{T}_{ij}} (\delta t + \bar{\tau}_{ij}) X_{ijt} \quad \forall j \in \mathbf{J}, i \in \mathbf{I}_j \tag{3.47}$$

Note that any combination of equations (3.45), (3.46), and (3.47) is redundant, but there might be computational benefits if more than one equation is included. Section 3.6 presents computational results and an extended discussion on the performance of the different variants.

3.3.8. Limited resources

The fact models M1-M4 use a common time-grid allows for the seamless inclusion of limited resources such as utilities and manpower. In this work we focus on the additional constraints imposed by the presence of limited utilities that might be simultaneously required by multiple batches. Since these four models adopt a discrete representation of time, it is possible to monitor variations in the utility levels and model utility costs while preserving the linearity of the mathematical formulation. Maravelias and coworkers[43, 94] proposed formulations to account for utility constraints suitable for the discrete-time representation, which we incorporate into our models.

Let $u \in \mathbf{U}$ be the set of limited utilities (e.g. high-pressure steam, cooling water, electricity, etc.) and let ξ_{iju} and ψ_{ut} respectively denote the fixed requirement of utility u for batch $i \in \mathbf{I}$ in unit $j \in \mathbf{J}_i$ and the maximum availability of utility u during interval $t \in \mathbf{T}\backslash\{0\}$. Furthermore, let us define decision variable $W_{ut} \in \mathbb{R}_+$ to account for the amount of utility u consumed during time interval t. Then equation (3.48) provides the definition of W_{ut} through a balance in terms of the total consumption of u in the previous interval, the consumption of u by batches starting at t and the release of u by batches finishing at t.

$$W_{ut} = W_{u(t-1)} + \sum_{i\in\mathbf{I}, j\in\mathbf{J}_{it}} \xi_{iju} X_{ijt} - \sum_{i\in\mathbf{I}, j\in\mathbf{J}_{i(t-\tau_{ij})}} \xi_{iju} X_{ij(t-\tau_{ij})} \leq \psi_{ut} \quad \forall u \in \mathbf{U}, t \in \mathbf{T}\backslash\{\tau^F\} \tag{3.48}$$

When utilities are considered, the cost minimization objective can be modified as shown in equation (3.49) to include α_{ut}^U, the cost of utility u during interval t.

$$\min \sum_{i\in\mathbf{I}, j\in\mathbf{J}_i, t\in\mathbf{T}_{ij}} \alpha_{ij} X_{ijt} + \sum_{u\in\mathbf{U}, t\in\mathbf{T}} \alpha_{ut}^U W_{ut} \tag{3.49}$$

Section 3.5.1 presents various small-scale illustrations of the coupling of the discrete-time models we are studying with utility constraints.

3.3.9. Product diversification in multistage facilities

Hitherto we have considered that every batch processed in a sequential multistage facility defines different intermediate materials after it is processed on each stage. However, chemical facilities are usually characterized by the transformation of a few raw materials into several different products, that is, there is a degree of *diversification* as material flows along processing units. For instance, a set of batches processed in a three-stage facility might share a single raw material and single intermediates after the first and second stages; only after the third stage the products are really different. A common example of this situation occurs when the last stage consists of packing operations, where products are differentiated according to packing type or size.

Traditional modeling approaches, including the discrete-time formulations just defined, rely on the fact that batches are different from the beginning of the process. This means that we have to define as many intermediate materials after each stage as batches are processed. Diversification is not taken into account, which might lead to inefficiency in the solution process. The goal of this section is to show that a representation based on the STN is a natural framework to handle diversification in sequential environments.

Let us consider a facility in which all batches share a common raw material and intermediates up to stage $|\mathbf{K}| - 1$ and only the products of stage $|\mathbf{K}|$ are different. The STN representation shown in Figure 3.3 can be modified to account for diversification as depicted in Figure 3.7. In this representation, the first $|\mathbf{K}| - 1$ tasks and $|\mathbf{K}|$ states are independent of the batch and they are only indexed by k. Note that Figure 3.3 is the representation of only one generic batch; the representation of the complete facility requires $|\mathbf{I}|$ parallel structures, one for every batch. In contrast, Figure 3.7 represents the complete facility. When this representation change is translated into the mathematical model, a considerable reduction in number of variables and constraints can be achieved.

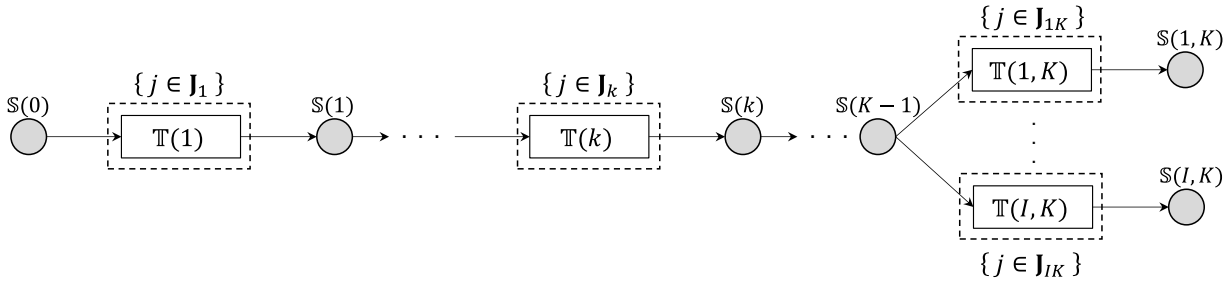


Figure 3.7. STN representation of a multistage facility with diversification.
The complete facility is represented. $I \equiv |\boldsymbol{I}|$ denotes the total number of batches and $K \equiv |\boldsymbol{K}|$ defines the number of stages.

The key element to determine if a task is batch-independent is the analysis of the materials it consumes and produces. In Figure 3.7, we see that material $\mathbb{S}(0)$ is common to all batches, which can be formally stated as $\mathbb{S}(i,0) \equiv \mathbb{S}(i',0) \equiv \mathbb{S}(0)\ \forall (i,i') \in \mathbf{I}^2 : i \neq i'$. Similarly, $\mathbb{S}(1)$ is also common for all batches. Therefore, $\mathbb{T}(i,1) \equiv \mathbb{T}(i',1) \equiv \mathbb{T}(1)\ \forall (i,i') \in \mathbf{I}^2 : i \neq i'$. We can informally say that all the tasks $\mathbb{T}(i,1)\ \forall i \in \mathbf{I}$ collapse into a single task $\mathbb{T}(1)$ since they all consume/produce the same materials.

We can generalize the concept of diversification if we consider that subgroups of tasks $\mathbb{T}(i,k)$ can be collapsed into a single task for specific subsets of batches. Let $l \in \mathbf{L}$ be an index/set to denote subgroups of batches that share common intermediates at a given stage. We can define $\mathbf{I}_{lk}$ as the lth set of batches on stage k for which tasks $\mathbb{T}(i,k)$ consume/produce the same set of materials.

Informally we say that all the tasks $\mathbb{T}(i,k)\ \forall i \in \mathbf{I}_{lk}$ collapse into a single task $\mathbb{T}(l,k)$. Unit compatibility is given by $\mathbf{J}_{lk} = \bigcap_{i\in\mathbf{I}_{lk}} \mathbf{J}_{ik}$. Figure 3.8 shows an example of this generalized sequential environment with diversification.

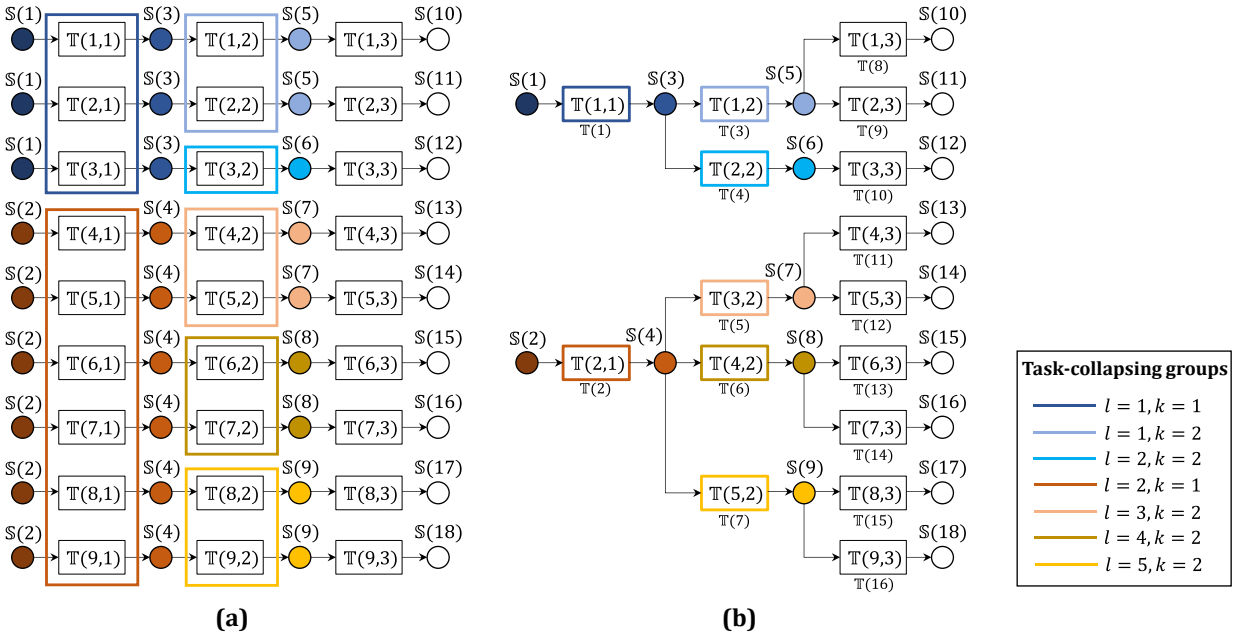


Figure 3.8. STN representation of a multistage facility with subgroup diversification.
Three-stage facility processing 9 batches with common intermediates. (a) Full STN representation. (b) Simplified STN representation using groups defined by diversification. In (b), the caption below every box is the re-enumeration of tasks. The color key indicates groups of tasks on (a) that collapse to a single task on (b).

Application of model SP&S

We now propose a modified STN-based model to take diversification into account. Figure 3.8b suggests that when diversification is present, the concepts of batch and stage might be dropped in favor of introducing the concept of task into the model. In fact, it is possible to re-enumerate tasks/states and use model SP&S defined for network environment.

After re-enumeration, task $\mathbb{T}(i)$ replaces $\mathbb{T}(l,k)$ and material $\mathbb{S}(r)$ takes the place of $\mathbb{S}(l,k)$. In this context, we use index $i \in \mathbf{I}^N$ to represent *tasks* (not the batches $i \in \mathbf{I}$, which we just dropped from the problem). We assume that inventory of raw materials –$\mathbb{S}(l,0)$– is always available and therefore

those states are not explicitly modeled. Importantly, due dates are now enforced for a set of final products, $\mathbf{R}^{MF}$, instead of the original set of orders. We denote these due dates as $\bar{\phi}_r^N \ \forall r \in \mathbf{R}^{MF}$.

Referring to model SP&S we observe that (i) equation (3.10) can be directly used to enforce unit assignment for the MBPSP with diversification, (ii) equation (3.11) does not apply since amounts are not modeled in the MBPSP, and (iii) equation (3.12) guarantees the batch precedence requirement for the MBPSP but it must be modified to account for appearance/disappearance of a material instead of its amount. Let $S_{mt}^{B,N} \in \{0,1\} \ \forall m \in \mathbf{M}^{STN} \backslash \mathbf{M}^{R,N}, t \in \mathbf{T}^N \backslash \{0\}$ define if material m is present during interval t, where $\mathbf{M}^{R,N}$ is the set of raw materials.. Equation (3.50) provides the final form of the batch precedence equation.

$$S_{m(t+1)}^{B,N} = S_{mt}^{B,N} + \sum_{i \in \mathbf{I}_m^+, j \in \mathbf{J}_i^{STN}} X_{ij\left(t-\tau_{ij}^N\right)}^{STN} - \sum_{i \in \mathbf{I}_m^-, j \in \mathbf{J}_i^{STN}} X_{ijt}^{STN} \quad \forall m \in \mathbf{M}^{STN} \backslash \mathbf{M}^{R,N}, t \in \mathbf{T}^N \backslash \{\eta^N\} \qquad (3.50)$$

The sets $\mathbf{I}_m^+/\mathbf{I}_m^-$ can readily be defined according to Figure 3.8, so that task $\mathbb{T}(i) \equiv \mathbb{T}(l,k)$ consumes material $\mathbb{S}(m_1) \equiv \mathbb{S}(l,k-1)$ and produces $\mathbb{S}(m_2) \equiv \mathbb{S}(l,k)$, for values of i, m_1, m_2 given by the re-enumeration of tasks and states.

There are two important facts that are worth to mention. First, it is not necessary to include an explicit equation to enforce the batch-stage assignment constraint of the MBPSP. In fact, when we defined the mapping between $\mathbb{T}(i)$ and $\mathbb{T}(l,k)$, the batch-stage assignment is automatically satisfied. Second, equation (3.50) guarantees that the no mixing/splitting conditions are satisfied because $S_{mt}^{B,N}$ are binary variables and cannot be aggregated or divided. Therefore, there is no need to write the additional equations required in network environments to enforce batch integrity[60, 94].

Equations (3.51) and (3.52) are respectively defined to enforce due dates and guarantee that each product is delivered.

$$\sum_{i\in\mathbf{I}_m^+, j\in\mathbf{J}_i^{STN}, t\in\mathbf{T}^N} \left(t+\tau_{ij}^{STN}\right) X_{ijt}^{STN} \le \phi_m^N \quad \forall m \in \mathbf{M}^{F,N} \tag{3.51}$$

$$S_{m\eta^N}^{B,N} = 1 \quad \forall m \in \mathbf{M}^{N,F} \tag{3.52}$$

Where $\eta^N = \max_{m\in\mathbf{M}^{N,F}} \phi_m^N$ and $\phi_m^N = \left\lfloor \bar{\phi}_m^N / \delta^N \right\rfloor$. The tree objective functions we considered can be defined through equations (3.53)-(3.55) for cost, earliness and makespan minimization respectively.

$$\min \sum_{i\in\mathbf{I}^N, j\in\mathbf{J}_i^{STN}, t\in\mathbf{T}} \alpha_{ij}^N X_{ijt}^{STN} \tag{3.53}$$

$$\min \sum_{m\in\mathbf{M}^{F,N}} \bar{\phi}_m^N - \sum_{m\in\mathbf{M}^{F,N}, i\in\mathbf{I}_m^+, j\in\mathbf{J}_i^{STN}, t\in\mathbf{T}^N} \left(\delta^N t + \bar{\tau}_{ij}^{STN}\right) X_{ijt}^{STN} \tag{3.54}$$

$$\min MS: MS \ge \sum_{i\in\mathbf{I}_j^{STN}} \left(\delta^N t + \bar{\tau}_{ij}^{STN}\right) X_{ijt}^{STN} \quad \forall j \in \mathbf{J}^N, t \in \mathbf{T}^N \tag{3.55}$$

Note that equation (3.55) uses a time-disaggregated unit-based definition for makespan, analogous to equation (3.46). We could also use a product-based definition for MS, similar to equations (3.45), but we omit it here for brevity. Note that the batch-disaggregated version of MS, equation (3.47), cannot be extended, since tasks are used instead of batches. The final model for MBPSP with diversification, hereinafter denoted SP&Sdiv, is comprised of equations (3.10), (3.50)-(3.52) and the appropriate objective function (3.53)-(3.55). In section 3.5.2, we present an illustration for the MBPSP with diversification.

Multiple orders

Model SP&Sdiv relies on the fact that each product is produced and delivered once. In fact, equations (3.51), (3.52), and (3.54) were derived based on this assumption. However, a realistic scheduling scenario might include multiple *orders* of the same product, so it has to be produced

several times, as well as all the intermediates that lead to it. The general model presented by Sundaramoorthy and Maravelias[94] suggests that in these cases we can introduce the notion of pseudo-customers and treat them as additional units of the network, so that multiple batches of the same product are delivered to different units.

Let $\mathbf{J}^{C,N}$ denote the set of customers and let $\bar{\phi}_{mj}^{C,N}$ be the due date of material $m \in \mathbf{M}^{F,N}$ for customer $j \in \mathbf{J}^{C,N}$. It is assumed that a single customer cannot order the same product twice; if that occurs, then an additional pseudo-customer is defined. The number of customers that are required is then equal to the maximum numbers of orders for a single product.

In order to model the product-customer assignment, a new decision variable is required. Let Z_{mjt}^{STN} be a binary variable that is equal to one if material $m \in \mathbf{M}^{F,N}$ is delivered to customer $j \in \mathbf{J}^{C,N}$ at time $t \in \mathbf{T}^{N}$. Then equations (3.56) and (3.57) respectively ensure due date satisfaction and single delivery to a customer. They replace equations (3.51) and (3.52).

$$Z_{mjt}^{STN} = 0 \quad \forall m \in \mathbf{M}^{F,N}, j \in \mathbf{J}^{C,N}, t > \phi_{mj}^{C,N} \tag{3.56}$$

$$\sum_{t \le \phi_{mj}^{C,N}} Z_{mjt}^{STN} = 1 \quad \forall m \in \mathbf{M}^{F,N}, j \in \mathbf{J}^{C,N} \tag{3.57}$$

Where $\phi_{mj}^{C,N} = \left\lfloor \bar{\phi}_{mj}^{C,N} / \delta^{N} \right\rfloor$. The material balance in equation (3.50) is modified to include the deliveries to customer during the scheduling horizon according to equation (3.58).

$$S_{m(t+1)}^{B,N} = S_{mt}^{B,N} + \sum_{i \in \mathbf{I}_m^+, j \in \mathbf{J}_i^{STN}} X_{ij\left(t-\tau_{ij}^{N}\right)}^{STN} - \sum_{i \in \mathbf{I}_m^-, j \in \mathbf{J}_i^{STN}} X_{ijt}^{STN} - \sum_{j \in \mathbf{J}^{C,N}} Z_{mjt}^{STN} \quad \forall m \in \mathbf{M}^{N} \backslash \mathbf{M}^{R,N}, t \in \mathbf{T}^{N} \backslash \{\eta^{N}\} \tag{3.58}$$

Equations (3.53) and (3.55) are still valid for cost and makespan minimization. Nevertheless, the definition of the earliness minimization objective has to be modified, as shown in equation (3.59).

$$\min \sum_{m\in\mathbf{M}^{F,N},j\in\mathbf{J}^{C,N}} \bar{\phi}_{mj}^{C,N} - \sum_{m\in\mathbf{M}^{F,N},j\in\mathbf{J}^{C,N}} \sum_{t\le\phi_{mj}^{C,N}} \delta t Z_{mjt}^{N} \tag{3.59}$$

The generalized model for MBPSP with diversification and multiple orders, SP&SdivM, is defined by equations (3.10), (3.56)-(3.58) and the appropriate objective function from (3.53), (3.55), or (3.59). Section 3.5.2 also presents an illustrative example of this case.

3.4. Solution Methods

3.4.1. Tightening based on fixed time windows

In section 3.3.1 we discussed how the earliest start and latest finish times of a batch are used to define a set of feasible time points to enforce the requirement that a batch has to be processed between its release and due dates. In this section we extend these concepts to define a tightening constraint for the MBPSP.

When we consider cost and earliness minimization objectives, the time window for a unit $j \in \mathbf{J}$ is fixed and defined as the difference between its latest finish and earliest start times, i.e. $\lambda_j - \varepsilon_j$. However, when minimizing makespan, we do not know *a priori* whether $\lambda_j > MS$ or $\lambda_j \le MS$. Thus we need to introduce a new parameter to characterize the time window of a unit, regardless of the objective considered. Let T^F represent the end of the horizon; it can be fixed or variable as shown in equation (3.60).

$$T^F = \begin{cases} \tau^F & \text{if end time is fixed (min Cost, Earliness)} \\ MS & \text{if end time is variable (min Makespan)} \end{cases} \tag{3.60}$$

We define the *shortest tail* of unit j, denoted σ_j, as the minimum elapsed time between the finishing of the last batch processed in unit j and T^F. This value only depends on the processing times in stages that come after unit j. Equation (3.61) gives the definition of shortest tail.

$$\sigma_j = \min_{i \in \mathbf{I}_j} \sum_{k \in \mathbf{K}^+_{i\kappa_j}} \min_{j' \in \mathbf{J}_{ik}} \tau_{ij'} \tag{3.61}$$

Where κ_j is the stage to which unit j belongs. The time window for unit j can then be defined as $T^F - \sigma_j - \varepsilon_j$. Note that for fixed end time $\sigma_j = T^F - \lambda_j$, whereas for variable end time there is no general relation between σ_j and λ_j. Equation (3.62) defines a tightening constraint to enforce that the actual processing time of a unit does not exceed its time window.

$$\sum_{i \in \mathbf{I}_j, t \in \mathbf{T}_{ij}} \tau_{ij} X_{ijt} \leq T^F - \sigma_j - \varepsilon_j \quad \forall j \in \mathbf{J} \tag{3.62}$$

We test the effect of adding this constraint to the four discrete-time models defined before and discuss the results in sections 3.5 and 3.6.

3.4.2. Tightening based on variable time windows

In Chapter 2 we developed tightening methods based on variable time windows for network environments. Earliest start time and shortest tail parameters were extended to define new decision variables which can be used in constraints that are tighter than their fixed counterparts. In this section, we extend those concepts to sequential environments.

Start time and tail of a batch

Section 3.4.1 describes a tightening constraint for *units* based on fixed parameters for the earliest start time and shortest tail. We now consider the case when these quantities are not fixed, so new decision variables are introduced in the model. However, our discussion is based on start times and tails for *batches* rather than units in order to model these values for the general task $\mathbb{T}(i,k)$. Let E^T_{ik} represent the actual start time of batch i on stage k and let S^T_{ik} denote the tail of batch i on stage k. Equations (3.63) and (3.64) give the definition of these new variables.

$$E_{ik}^{T} = \sum_{j \in \mathbf{J}_{ik}, t \in \mathbf{T}_{ij}} t X_{ijt} \quad \forall i \in \mathbf{I}, k \in \mathbf{K}_i \tag{3.63}$$

$$S_{ik}^{T} = T^{F} - \sum_{j \in \mathbf{J}_{ik}, t \in \mathbf{T}_{ij}} (t + \tau_{ij}) X_{ijt} \quad \forall i \in \mathbf{I}, k \in \mathbf{K}_i \tag{3.64}$$

Where T^F is given by equation (3.60). A lower bound for the start time can be defined using parameter ε_{ik}^{S} introduced in section 5.1, as shown in equation (3.65). Similarly, the actual tail can be lower-bounded with σ_{ik}^{S}, the shortest tail of batch i on stage k, according to equation (3.66). Equation (3.67) provides the definition of σ_{ik}^{S}. Figure 3.9 shows a small illustration of the relation between fixed and variable start times and tails.

$$E_{ik}^{T} \geq \varepsilon_{ik}^{S} \quad \forall i \in \mathbf{I}, k \in \mathbf{K}_i \tag{3.65}$$

$$S_{ik}^{T} \geq \sigma_{ik}^{S} \quad \forall i \in \mathbf{I}, k \in \mathbf{K}_i \tag{3.66}$$

$$\sigma_{ik}^{S} = \sum_{k' \in \mathbf{K}_{ik}^{+}} \min_{j' \in \mathbf{J}_{ik'}} \tau_{ij'} \quad \forall i \in \mathbf{I}, k \in \mathbf{K}_i \tag{3.67}$$

	τ_{ij}	
i	$\mathbb{U}_1/\mathbb{U}_2$	$\mathbb{U}_3/\mathbb{U}_4$
1	2	3
2	3	2
3	2	3
4	3	2

$\mu_i = 2, \phi_i = 10 \; \forall i$

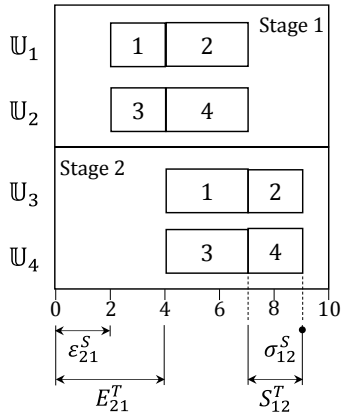


Figure 3.9. Start times and tails
Two-stage facility with two units per stage in which four batches are to be processed. All the batches have a common earliest start date of $\varepsilon_{i1}^{S} = 2$ for stage 1 and a shortest tail of $\sigma_{i2}^{S} = 0$ on stage 2. The values of E_{ik}^{T} and S_{ik}^{T} are lower-bounded by ε_{ik}^{S} and σ_{ik}^{S} respectively, but their actual values are larger. For instance, batch 2 has an actual start date of $E_{21}^{T} = 4$ on the first stage, but it could start as early as $\varepsilon_{21}^{S} = 2$. Similarly, batch 1 has an actual tail of $S_{12}^{T} = 2$ on the second stage but its shortest tail is $\sigma_{12}^{S} = 0$.

Precedence relations through start times and tails

The fact that we use variables based on tasks allows us to explicitly enforce the unique sequence in which all the batches follow the stages. Equation (3.68) ensures that the start time of a batch on any stage exceeds the finish time of the batch on the immediately previous stage. Similarly, equation (3.69) enforces that the tail of a batch on any stage is larger than the sum of the tail and processing times of the batch on the next stages.

$$E^T_{i(k+\nu_{ik})} \geq E^T_{ik} + \sum_{j \in \mathbf{J}_{ik}, t \in \mathbf{T}_{ij}} \tau_{ij} X_{ijt} \quad \forall i \in \mathbf{I}, k \in \mathbf{K}_i \tag{3.68}$$

$$S^T_{ik} \geq S^T_{i(k+\nu_{ik})} + \sum_{j \in \mathbf{J}_{i(k+\nu_{ik})}, t \in \mathbf{T}_{ij}} \tau_{ij} X_{ijt} \quad \forall i \in \mathbf{I}, k \in \mathbf{K}_i \tag{3.69}$$

It is important to note that equations (3.68) and (3.69) relate actual time values for start and finishing of a batch in consecutive stages. Consequently, these equations can be used directly to enforce the batch precedence constraint of the MBPSP. We could write new discrete-time models by combining (i) equation (3.68) –or equation (3.69)– and the definition of E^T_{ik} –or S^T_{ik}– , (ii) the batch-stage assignment constraint given through equation (3.36), and (iii) the unit assignment constraint defined through either equation (3.37) or (3.39). In particular, equations (3.63) and (3.68) are equivalent to equation (3.40), the batch precedence constraint for model M3. In this work we study the four models described before, but our formulations are flexible enough to allow the definition of up to ten different models, whose performances could be compared.

Linear combinations of start times and tails

Simply defining, lower-bounding and enforcing precedence for E^T_{ik} and S^T_{ik} is not enough to achieve a tightening effect on the original discrete-time model. It is necessary to develop constraints that describe the dependency between start times –and between tails– of different tasks. In Chapter 2 we describe three cases in which constraints relating time-based variables can be enforced: (i)

subsequent tasks, (ii) tasks that consume a common material, and (iii) tasks that share a common unit.

We now analyze what these cases represent for the MBPSP. Case (i) is already covered through equations (3.68) and (3.69), since in sequential environments task $\mathbb{T}(i, k + \nu_{ik})$ is the only one that can consume the product of $\mathbb{T}(i, k)$. Case (ii) cannot be used in sequential environments, because it involves batch splitting, which is not allowed in the MBPSP. Finally, case (iii) could be directly extended to sequential environments by using the RTN-based tasks $\mathbb{T}(i, j)$. However, stages can be viewed as supersets of units, and then it is more general to use STN-based tasks $\mathbb{T}(i, k)$, which in addition are directly related to the variables E_{ik}^{T} and S_{ik}^{T} already defined. Hence, for sequential environments we study the dependence of variable times for batches that share the same stage.

The method described in section 2.3.3 is based on explicitly studying all the possible sequences among tasks assigned to the same unit which is factorial in nature. The result is a linear combination of these variables that can be directly added to the model formulation. In general, units in network environments are specialized and the number of tasks assigned to a single unit remains low; thus, the exhaustive enumeration of sequences remains computationally tractable. However, in sequential environments the number of batches that have to be processed and can be assigned to a single stage is significantly higher (i.e. several dozens of orders can be obtained after solving the batching problem), causing a combinatorial explosion if explicit sequences were to be considered. Therefore, we adopt a different strategy in order to obtain the final constraints. Instead of aiming for general linear combinations for start times and tails, we focus on lower-bounding their summations. Equations (3.70) and (3.71) express these constraints.

$$\sum_{i \in \mathbf{I}_k} E_{ik}^T \geq \gamma_k \quad \forall k \in \mathbf{K} \tag{3.70}$$

$$\sum_{i \in \mathbf{I}_k} S_{ik}^T \geq \zeta_k \quad \forall k \in \mathbf{K} \tag{3.71}$$

Lower bounds γ_k and ζ_k are calculated by solving minimization problems subject to the original MBPSP constraints. These minimization problems are conceptually defined through equations (OP1) and (OP2). If we refer to Figure 3.9, the values of these parameters can be directed calculated as $\gamma_1 = 12(= 2 + 4 + 2 + 4), \gamma_2 = 22(= 4 + 7 + 4 + 7), \zeta_1 = 14(= 5 + 2 + 5 + 2), \zeta_2 = 4(= 2 + 0 + 2 + 0)$. For more complex facilities, these values have to be calculated through (OP1) and (OP2).

$$\gamma_k = \min \sum_{i \in \mathbf{I}_k} E_{ik}^T \qquad \forall k \in \mathbf{K}$$
$$s.t. \quad \text{Batch precedence} - \text{eqn (3.38) or (3.40) or (3.42)}$$
$$\text{Batch-stage assignment} - \text{eqn (3.36)}$$
$$\text{Unit assignment} - \text{eqn (3.37) or (3.39)}$$
$$\text{Start time definition} - \text{eqn (3.63)}$$
$$\text{Start time bound} - \text{eqn (3.65)}$$
$$\text{Start time sequence} - \text{eqn (3.68)} \tag{OP1}$$

$$\zeta_k = \min \sum_{i \in \mathbf{I}_k} S_{ik}^T \qquad \forall k \in \mathbf{K}$$
$$s.t. \quad \text{Batch precedence} - \text{eqn (3.38) or (3.40) or (3.42)}$$
$$\text{Batch-stage assignment} - \text{eqn (3.36)}$$
$$\text{Unit assignment} - \text{eqn (3.37) or (3.39)}$$
$$\text{Tail definition} - \text{eqn (3.64)}$$
$$\text{Tail bound} - \text{eqn (3.66)}$$
$$\text{Tail sequence} - \text{eqn (3.69)} \tag{OP2}$$

Problem (OP2) includes equation (3.64) as one of its constraints. This equation includes T^F and thus depends on the objective considered. However, when solving for makespan minimization the value of MS is not known, so we cannot set $T^F = MS$. To overcome this problem, we use a "large enough" value to estimate MS, and relax the due dates for all the batches, so that no artificial extra time is added to the summation of tails due to the presence of early due dates. Thus, we set $T^F = \tau^F$

and $\phi_i = \tau^F \ \forall i \in \mathbf{I}$ when solving (OP2). Figure 3.10 presents a simple instance to explain why the due date relaxation is necessary.

		τ_{ij}	
i	ϕ_i	$\mathbb{U}_7$	$\mathbb{U}_8$
1	46	2	-
2	50	2	-
3	50	3	-
4	50	-	2
5	50	-	2
6	50	-	2

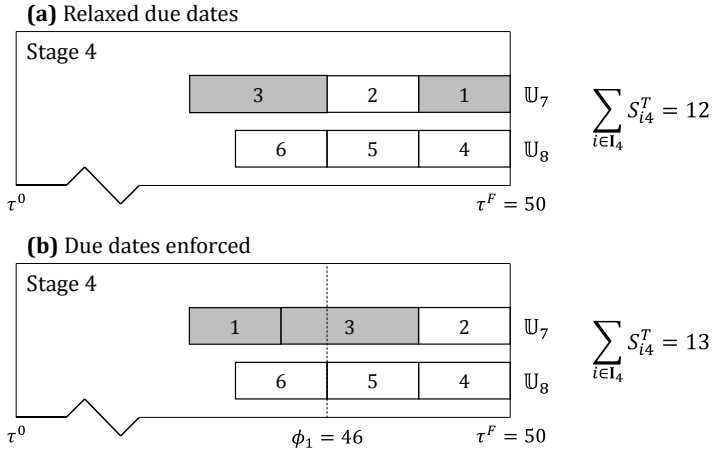


Figure 3.10. Due date relaxation to estimate ζ_k.
The Gantt chart for the last stage of a four-stage facility with two units per stage is shown. Due dates, unit compatibility and processing times for six batches that need to be processed are given. (a) For relaxed due dates, batches with the smaller processing times are closer to τ^F; therefore the summation of tails has the smallest possible value of 12. (b) For enforced due dates, batch 1 is pushed far from τ^F, and the summation of tails is increased to 13. Setting $\zeta_4 = 13$ incorrectly cuts off the feasible solution in (a) for $MS \leq 46$. The correct value is $\zeta_4 = 12$.

Tightening constraint

We can now define a tightening constraint that can be added to the original formulations of the four discrete-time models we are studying. Equation (3.72) expresses that the total time batch $i \in \mathbf{I}$ spends being processed in all stages cannot exceed its available time window. Section 7 shows computational results that include this and other constraints defined in this section.

$$\sum_{k\in\mathbf{K}_i, j\in\mathbf{J}_{ik}, t\in\mathbf{T}_{ij}} \tau_{ij} X_{ijt} \leq T^F - E^T_{i\kappa_i^{FS}} - S^T_{i\kappa_i^{LS}} \quad \forall i \in \mathbf{I} \tag{3.72}$$

3.4.3. Improving the accuracy of discrete-time scheduling solutions

In this section, we propose a novel method to improve the solution quality of the results that are obtained from discrete-time models with a large step size δ. One of the main disadvantages of discrete-time models is the approximation it introduces on time-dependent parameters, which is reflected in the accuracy of the solution. The method we propose permits to refine solutions obtained

with a discrete-time formulation in order to obtain realistic timing of events. Difficult problems can be solved with a relatively large value of δ without loss in the solution quality. We apply this method to makespan minimization problems since it is the hardest objective studied in this chapter.

The key idea behind is to solve the problem in two steps: (i) obtain assignment and sequencing solutions using a discrete-time model, (ii) refine the solution with a continuous-time model. The discrete-time solutions are converted into decision variables of the continuous-time model MH&C introduced in section 3.2.1, i.e. assignment (X_{ij}^{C}), timing (C_{ik}), and sequencing ($Y_{ii'k}$), using equations (3.73)-(3.75).

$$X_{ij}^{C} = \sum_{t \in \mathbf{T}_{ij}} X_{ijt} \quad \forall i, j \in \mathbf{J}_i \tag{3.73}$$

$$C_{ik} = \sum_{j \in \mathbf{J}_{ik}, t \in \mathbf{T}_{ij}} (\delta t + \bar{\tau}_{ij}) X_{ijt} \quad \forall i, k \in \mathbf{K}_i \tag{3.74}$$

$$Y_{ii'k} = 1 \quad \forall (i,i') \in \mathbf{I}^2 : i < i', k \in \mathbf{K}_i \cap \mathbf{K}_{i'} : (\exists j \in \mathbf{J}_{ik} \cap \mathbf{J}_{i'k} : X_{ij}^{C} + X_{i'j}^{C} = 2), \left(C_{i'k} \geq C_{ik} + \sum_{j \in \mathbf{J}_{i'k}} \tau_{i'j} X_{i'j}^{C} \right) \tag{3.75}$$

Next, the assignment and sequencing variables, X_{ij}^{C} and $Y_{ii'k}$ are fixed in equations (3.1), (3.3)-(3.5) and the problem is solved to obtain the timing variables, C_{ik}. Table 1 presents an algorithm that allows to eliminate redundant constraints from the model MH&C; some of the equations (3) and (4) are excluded. We introduce parameters $\omega_{ii'jk}^{1}$, $\omega_{ii'jk}^{2}$, and $\omega_{ii'jk}^{3}$ as indicators to identify different cases. $\omega_{ii'jk}^{1} = 1$ means that both batches i and i' are assigned to a common unit $j \in \mathbf{J}_{ik}$. $\omega_{ii'jk}^{2} = 1$ ensures that batch i is processed before i', whereas $\omega_{ii'jk}^{3} = 1$ indicates that batch i is processed after i'.

The modified MH&C model, MH&C_LP, consists of equations (3.6), (3.9) and (3.76)-(3.79). Since all the binary variables are fixed, the resulting problem is an LP that can be solved with little

computational effort. As a result of this process, we obtain a schedule with a continuous timing of events and an improved objective, i.e. the refined solution. Although we note that this method does not guarantee that the optimal solution can be found, it turns out to be a powerful tool in solving difficult problems as shown later in section 3.6.

$$C_{i(k+\nu_{ik})} \geq C_{ik} + \bar{\tau}_{ij} \quad \forall i \in \mathbf{I}, k \in \mathbf{K}_i \backslash \{\kappa_i^{LS}\}, j \in \mathbf{J}_{i(k+\nu_{ik})}: X_{ij}^C = 1 \tag{3.76}$$

$$C_{i'k} - \bar{\tau}_{i'j} \geq C_{ik} \quad \forall (i,i') \in \mathbf{I}^2: i' > i, k \in \mathbf{K}_i \cap \mathbf{K}_{i'}, j \in \mathbf{J}_{ik} \cap \mathbf{J}_{i'k}: \omega_{ii'jk}^1 = \omega_{ii'jk}^2 = 1 \tag{3.77}$$

$$C_{ik} - \bar{\tau}_{ij} \geq C_{i'k} \quad \forall (i,i') \in \mathbf{I}^2: i' > i, k \in \mathbf{K}_i \cap \mathbf{K}_{i'}, j \in \mathbf{J}_{ik} \cap \mathbf{J}_{i'k}: \omega_{ii'jk}^1 = \omega_{ii'jk}^3 = 1 \tag{3.78}$$

$$C_{i\kappa_i^{FS}} \geq \bar{\rho}_i + \bar{\tau}_{ij} \quad \forall i \in \mathbf{I}, j \in \mathbf{J}_{i\kappa_i^{FS}}: X_{ij}^C = 1 \tag{3.79}$$

Table 3.1. Algorithm for elimination of redundant constraints in model MH&C.

Set $\omega_{ii'jk}^1 = \omega_{ii'jk}^2 = \omega_{ii'jk}^3 = 1 \quad \forall (i,i') \in \mathbf{I}^2: i' > i, k \in \mathbf{K}_i \cap \mathbf{K}_{i'}, j \in \mathbf{J}_{ik} \cap \mathbf{J}_{i'k}$
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 $X_{ij}^C + X_{i'j}^C < 2$ then $\omega_{ii'jk}^1 = 0$
If $\omega_{ii'jk}^1 = 1$ and $Y_{ii'k} = 0$ then $\omega_{ii'jk}^2 = 0$
If $\omega_{ii'jk}^1 = 1$ and $Y_{ii'k} = 1$ then $\omega_{ii'jk}^3 = 0$
End loop (× 4)

3.5. Illustrative examples

3.5.1. Utility constraints

This section considers two different examples to illustrate the modeling flexibility and computational capabilities of the proposed discrete-time models. Example 1 is a modification of a small-scale instance presented by Sundaramoorthy et al.[43], for which we consider that batching and storage decisions are addressed separately. Example 2 is a medium-scale problem we propose to test the performance of our formulations as the model size increases. Complete data for these examples can be found in the Supporting Information.

We use model M4 as an illustration to solve the examples in this section. The performance of M1-M3 is similar in these small and medium-scale instances. For comparison between different models, please refer to section 3.6 which presents a comprehensive computational study including larger instances. The instances in this section were solved using GAMS 24.4.6/CPLEX 12.6.2 on a computer with 8 GB of RAM and a dual Intel Core i7-930 processor at 2.8 GHz running on Windows 7.

Example 1: We consider a facility with two stages and two units per stage. Twelve batches have to be processed with the objective of minimizing cost. Three different instances of this facility are considered. In the first instance, cooling water (CW) is required in both stages with a cost of \$4/(ton/h) and a maximum availability of 7 ton/h. The second instance introduces time variability on the maximum available amount of CW; initially the availability is 5 ton/h but after 20 hours it increases to 7 ton/h. The third instance adds periodic variability to the CW price with two levels at \$5/(ton/h) and \$4/(ton/h) that are interchanged every 5 hours.

All three instances are modeled with 425 constraints, 37 continuous variables, and 816 binary variables. Computational results are summarized in Table 3.2 with the corresponding Gantt charts and utility profiles shown in Figure 3.11. All instances are solved relatively fast (<30 sec) using model M4.

We observe that when utility availability is increased after 20 h (Figure 3.11b), the schedule is shifted to the right so batches can be processed as late as possible, while keeping the total cost low (<1.5% increase in cost with respect to fixed availability). When the CW price increases (Figure 3.11c), some batches are shifted back to the left so they are processed early when the utility cost is low. However, the fact that CW is more scarce at the beginning of the horizon forces many batches to be processed late and the total cost is significantly impacted (>8% increase in cost with respect to fixed cost and availability).

Table 3.2. Computational results for example 1, instances 1-3

	Instance 1	Instance 2	Instance 3
CW Cost	Constant	Constant	Time-varying
CW Availability	Constant	Time-varying	Time-varying
Objective value	1,631.00	1,650.00	1,768.00
MIP time (s)	<1	24.8	15.6

Example 2: Let us consider a sequential facility with four stages in which 15 orders are to be processed. The first and fourth stages have three processing units each, while the second and third stages have two units. High-pressure steam (HPS) is required for stages 1 and 3; and refrigerated water (RW) is used in stages 2 and 4. The objective is to minimize makespan. We consider two instances of this facility. The first instance assumes that there is unlimited availability for both HPS and RW. The second instance limits the amount of HPS to 10 ton/h and RW to 15 ton/h.

The unlimited availability assumption of instance 1 allows to decouple the scheduling and utility assignment problems, so they can be solved in series. We use this fact to compare the performance of our models/methods with the continuous-time model MH&C introduced in section 3.2.1 when solving the scheduling problem only. Table 3.3 presents the results for model M4 with and without the tightening constraint from fixed time windows, equation (3.62), as well as for model MH&C. We obtained a difference in computational time of at least four orders of magnitude between models MH&C and the tightened version of M4. Even without tightening, at least a two order-of-magnitude decrease in CPU time is observed with model M4, although it has five times more binary variables than MH&C.

Table 3.3. Computational results and model statistics for example 2 with unlimited resources

	M4	M4+eq.(3.62)	MH&C
Constraints	1,415	1,425	1,927
Continuous variables	1	1	56
Binary variables	2,372	2,372	480
Objective value (h)	21.00	21.00	22.00
LP relaxation (h)	2.95	20.00	13.00
MIP time (s)	134.2	7.0	10,800
Nodes	1,010	206	4,931,001
Gap (%)	---	---	12.5

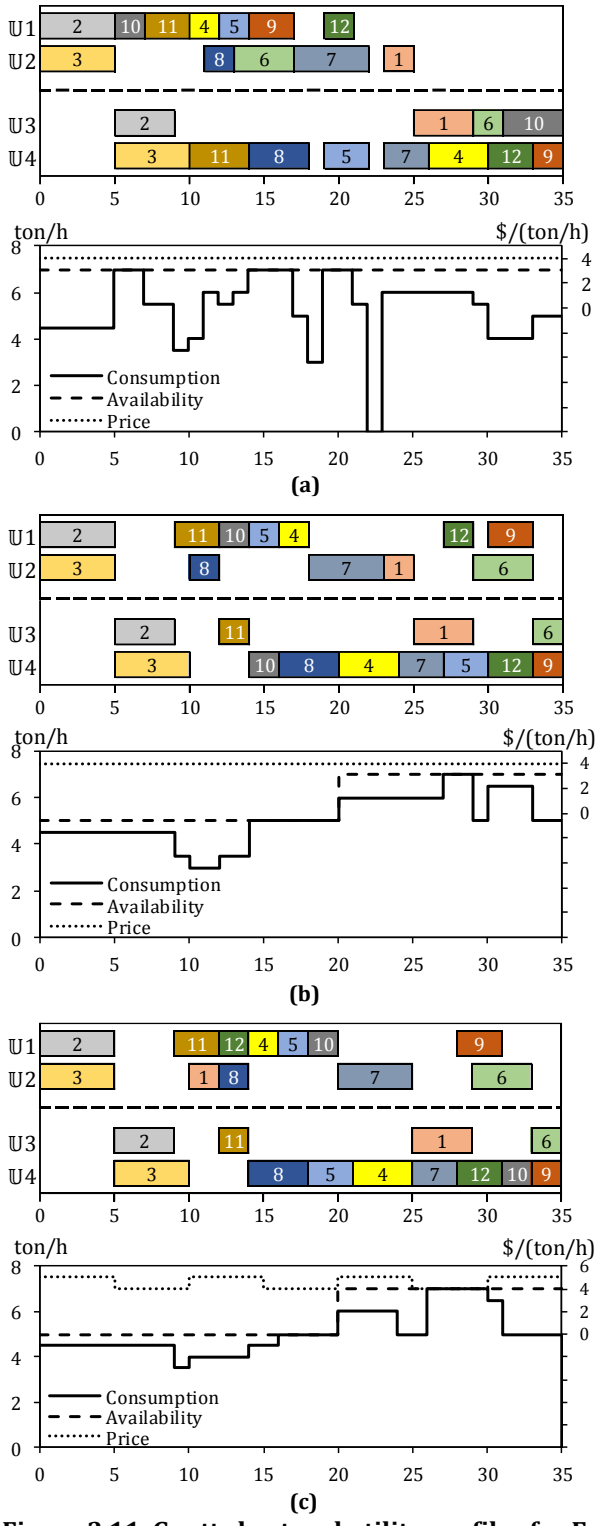


Figure 3.11. Gantt chart and utility profiles for Example 1
(a) Constant CW cost and availability, (b) constant CW cost and time-varying availability, (c) time-varying CW cost and availability

To solve the coupled scheduling/utility problem that results when limited resources are enforced, we used model M4 coupled with equation (3.62) as before. Table 3.4 provides model statistics and computational results for this instance. Figure 13 shows the Gantt chart and utility profiles for the optimal solution. An increase of 7 hours in makespan is necessary to accommodate the restrictions of limited HPS and RW.

Table 3.4. Computational results and model statistics for example 2 with limited HPS and RW

	M4+eq.(3.62)
Constraints	1,589
Continuous variables	84
Binary variables	2,372
Objective value (h)	28.00
LP relaxation (h)	20.00
MIP time (s)	28.7
Nodes	1,025

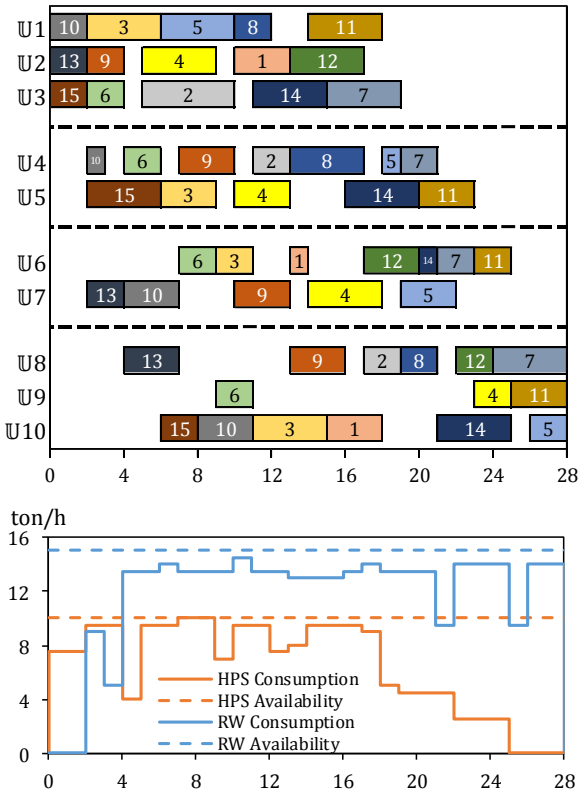


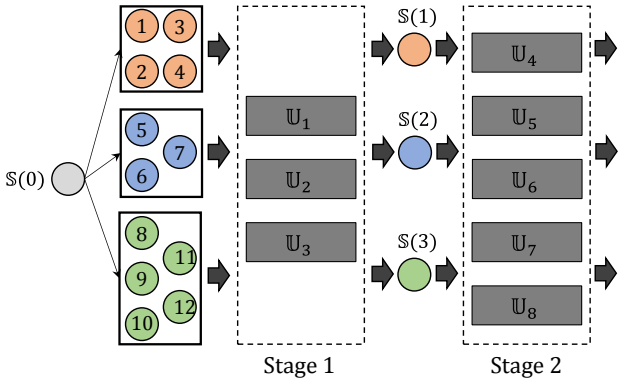
Figure 3.12. Gantt chart and utility profiles for Example 2 with limited resources
Fifteen batches are processed in a 4-stage facility, which uses HPS in stages 1 and 3, and RW in stages 2 and 4. The objective is to minimize makespan with fixed utility availability.

3.5.2. Diversification in sequential environments

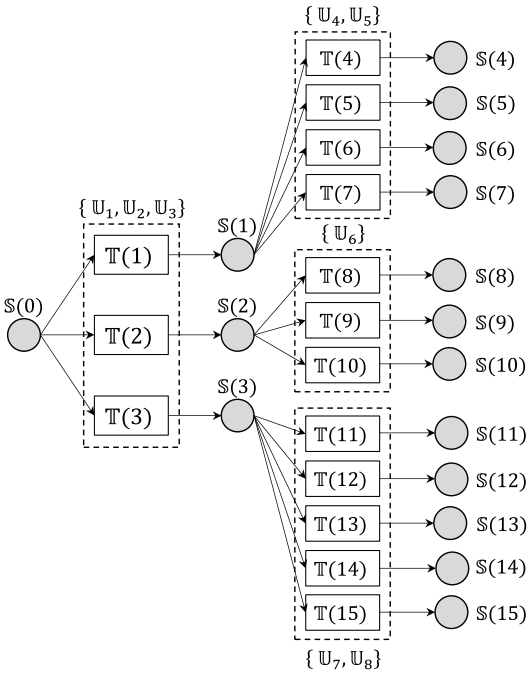
Single order

In this section we consider the sequential process with diversification first addressed by Sundaramoorthy and Maravelias[94] using a general model for hybrid environments. Inventory storage and batching decisions are not addressed, since they are not included in the MBPSP definition.

Let us consider a facility with two stages; the first stage has three units and the second has five units. Twelve batches are to be processed starting from a common raw material $\mathbb{S}(0)$. The first four batches have a common intermediate, $\mathbb{S}(1)$ after the first stage and use the first two units of the second stage. The next three batches share $\mathbb{S}(2)$ and are compatible with the third unit of stage 2. Finally, the last five batches use $\mathbb{S}(3)$ and can be processed in the las two units of the second stage. Detailed data for this facility are provided in the Supporting Information. Figure 3.13(a) provides a representation of this process with common materials depicted.



(a)



(b)

Figure 3.13. Sequential process with diversification.
(a) Standard representation of a two-stage facility where twelve batches are processed. All the batches share $\mathbb{S}(0)$ as raw material. Intermediates $\mathbb{S}(1)$, $\mathbb{S}(2)$, and $\mathbb{S}(3)$ are common for three different subgroups. (b) STN representation. Stages and batches are not explicitly shown. The original batches are represented by final products $\mathbb{S}(4)$- $\mathbb{S}(15)$. Tasks and materials are re-enumerated

We can use the STN-based representation and model introduced in section 3.3.9 to address this problem. In this case we have $|\mathbf{L}| = 3$ subsets of batches with $\mathbf{I}_1 = \{1,2,3,4\}, \mathbf{I}_2 = \{5,6,7\}, \mathbf{I}_3 = \{8,9,10,11,12\}$. The STN representation consists of 15 tasks; the first three to transform raw material $\mathbb{S}(0)$ into common intermediates $\mathbb{S}(1)$-$\mathbb{S}(3)$ and the last 12 to produce individual batches from the intermediates. Figure 3.13 (b) shows the STN representation of the process under study.

Note that tasks and states have been re-enumerated, but they can be traced back to original tasks $\mathbb{T}(l,k)$ and states $\mathbb{S}(l,k)$, so the traceability of the batches is retained. For instance, $\mathbb{T}(i=3)$ corresponds to $\mathbb{T}(l=3,k=1)$ and state $\mathbb{S}(m=2)$ represents $\mathbb{S}(l=2,k=1)$.

We consider three different instances of this process, one for each objective we are studying. All three instances were solved in less than 1 CPU second. Table 5 summarizes results and model statistics for the three objectives considered.

Table 3.5. Computational results and statistics for sequential process with diversification

Model	SP&Sdiv		
Problem	Cost	Earliness	Makespan
Constraints	741	741	753
Continuous variables	466	466	467
Binary variables	930	930	930
Objective value	955	0	14

Multiple orders

We now consider the same facility depicted in Figure 3.13 when multiple orders are placed for the same product. In particular, we consider three orders of products $\mathbb{S}(4)$- $\mathbb{S}(15)$, for a total of 36 batches. Pertinent data can be found in the Supporting Information.

If we were to use a batch-based model such the ones described in section 3.2.1 and 3.3, we would have to make three copies of the original batches in order to consider 36 independent orders, each

of them with given processing data and due dates. In contrast, using model SP&SdivM just requires adjusting the set $\mathbf{J}^{C,N}$ and a single parameter $\phi_{mj}^{C,N}$ to reflect multiplicity of orders.

We solve the makespan minimization problem using model SP&SdivM. Moreover, we solve the same problem using STN-based model M1 with and without tightening to compare. Table 3.6 presents the results and model statistics for the runs with instances of the MBPSP with diversification and multiple orders. It is clear from Table 3.6 the significant advantage of model SP&SdivM in terms of computational performance. First, there is a considerable reduction in model size; model SP&SdivM requires 50% less constraints and 68% less binary variables (46% less total variables). Second, there is an important reduction in computational time: one order of magnitude when compared with the untightened order-based model and half the time after tightening. Note that the tightened version of STN is slower than SP&SdivM even though its relaxation is very close to the actual solution. Figure 3.14 presents the Gantt chart obtained for the minimum makespan problem.

Table 3.6. Computational results and statistics for sequential process with diversification and multiple orders

	SP&SdivM	M1	
Tightening	---	No	Equation (62)
Constraints	2,248	4,483	4,491
Continuous variables	3,062	3,674	3,674
Binary variables	1,530	4,731	4,731
Objective value (h)	37.00	37.00	37.00
LP relaxation (h)	6.41	6.41	36.00
MIP time (s)	4.26	30.03	8.36
Total elapsed time (s)	1.56	36.39	14.63
Nodes	4	79	38

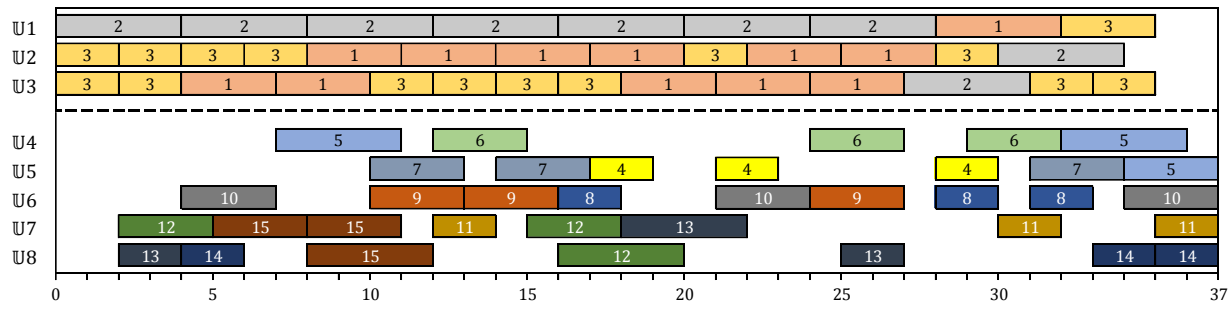


Figure 3.14. Gantt chart for MBPSP with makespan minimization with diversification/multiple orders.

3.6. Computational Study

In this section we test the performance of the four discrete-time models M1-M4 introduced in section 3.3, as well as the effectiveness of the solution methods proposed in section 3.4 by means of a computational study with several instances of six multistage facilities. For comparison purposes, we use model MH&C, referred to as M5 in this section, which has proven to be one of the best continuous-time models for the MBPSP and has been used as a benchmark formulation in different studies[22, 26, 61]. Next, we define the notation used throughout the study.

(i) Multistage facilities are identified with [P#], where # ∈ {1, … ,6}. P1 is a modified version of a facility used by Harjunkoski and Grossmann[22]; P2-P3 were studied by Castro and Grossmann[26]; P4 appears in Liu and Karimi[25]; and P5-P6 are taken from Castro et al.[61].

(ii) Objective functions are represented by [@@], where @@ ∈ {CT, EA, MS#} for cost, earliness and makespan minimization problems respectively. For makespan minimization we include the additional field # ∈ {1,2,3} to respectively indicate whether equation (3.45), (3.46), or (3.47) is used to define MS.

(iii) Models are denoted by [M#], with # ∈ {1, … ,5} as previously defined.

(iv) Inclusion of tightening constraints is identified with the pair [T##], where # ∈ {0,1}. The first position corresponds to the fixed-window constraint –equation (3.62)–, and the second position represents the variable-window constraint –equation (3.72)–.

(v) The use of integer reformulation, as presented in Merchan et al. is identified with [R#], where # ∈ {0,1}. Priority utilization is indicated by the triplet [P##@] with # ∈ {0,1} and @ ∈ {N, X}. The first field indicates if priorities are active. The second and third field are not used if priorities are not defined. The second field indicates if busyness-type hierarchy is used. The third field indicates if higher priorities are assigned to either the reformulation variable or the assignment variable X_{ijt}.

The complete notation for a single run is given by [P#]. [@@].[M#]. [T##].[R#].[P##@]. For example, a run labeled "P4.CT.M4.T10.R1.P100.Z0" indicates that a cost minimization problem for facility P4 is solved, using the disaggregated RCPSCP-based model with fixed-window tightening and using reformulation with regular priorities on the assignment variable.

A total of 1,808 runs are included in this study. All the runs were solved using GAMS 24.4.3/CPLEX 12.6.1 on a cluster of 21 identical computers. Each computer has 16 GB of RAM and an Intel Xeon (E5520) processor at 2.27GHz running on CentOS Linux 7. Default CPLEX options are used unless specified otherwise. A time limit of one hour was enforced for all the runs. Model and solution statistics for all runs are given in the Supporting Information.

We study a total of 16 formulations for each of the discrete-time models M1-M4 defined in section 3.3. Formulations are defined through different combinations of solution methods, i.e. different assignments of the notation fields [T##].[R#].[P##@]. Table 3.7 shows the definition of the six basic formulations we consider. Note that priorities are only considered when the reformulation is used and that higher priorities are assigned to the reformulation variable. The cases when priorities are used without reformulation or higher priorities are assigned to assignment variables proved not to have significant impact on the models so they are not included in this study.

Table 3.7. Basic formulations F0-F5

Formulation	T##		R#	P##@			Equations
F0 (Original)	0	0	0	0	---	---	M1: (3.36), (3.37), (3.38) M2: (3.36), (3.38), (3.39) M3: (3.36), (3.37), (3.40) M4: (3.36), (3.37), (3.42)
F1	1	0	0	0	---	---	F0+(3.62)
F2	0	1	0	0	---	---	F0+(3.72) Preprocessing: Best bound at node 0 of (OP1) and (OP2)[a]
F3	0	0	1	0	---	---	F0+Reformulation
F4	0	0	1	1	0	N	F0+Priorities Preprocessing: LP relaxation of F0
F5	0	0	1	1	1	N	F0+Priorities Preprocessing: LP relaxation of F0

[a] It was found through computational experimentation that the optimal MIP solution for (OP1) and (OP2) is not required; a lower bound at node zero is enough.

The remaining ten formulations are combinations of two or three methods from F1-F5. They are denoted by [F#.#.#], explicitly indicating which formulations are combined.

3.6.1. Performance Analysis

To better summarize the vast amount of results and identify trends that can be used as guidelines for our analysis, we first present performance charts for different objectives and models. In general, a performance chart shows how efficient a formulation is in terms of both percentage of instances solved to optimality and relative computational time compared to the fastest formulation. The abscissa corresponds to the ratio between solution times of a particular instance and the fastest instance for a given formulation; the ordinate is the fraction of instances solved to optimality within a given relative time. A general discussion on this type of charts is given in Appendix B

Figure 3.15 shows the best four formulations for each model-objective [M#].[@@] combination, as well as the original formulation F0. Several trends can be observed in the subplots of Figure 3.15, as discussed next.

For earliness minimization, the original formulation F0 performs as good as the best formulations that include our methods. This means that the original discrete-time models are good enough to minimize earliness effectively. A possible explanation for this fact lies in the relatively short CPU times required to solve this problem to optimality. Adding the constant-window tightening constraint (F1) and using reformulation (F3) appear to be the best options, but it does not outperform F0. For formulations F2 and F4, the improvement achieved with our methods is balanced with the extra time required for preprocessing. Earliness minimization proved to be the easiest objective we consider; all the instances were solved to optimality within the allocated time limit.

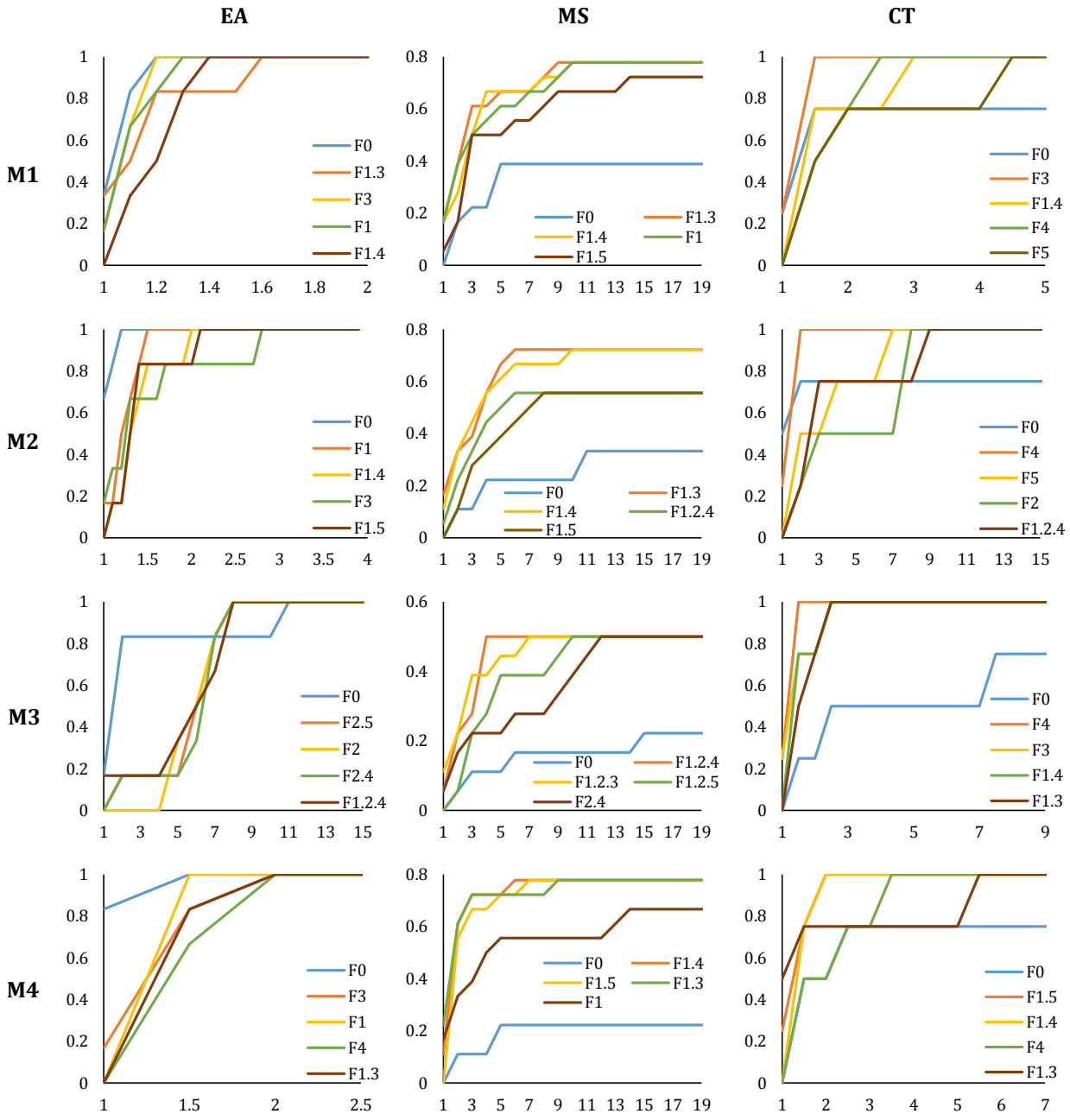


Figure 3.15. Best formulations for different model-objective combinations.
Combinations of [M#].[@@] are presented with # ∈ {1, … ,4} and @@ ∈ {CT, EA, MS}. The abscissa is the ratio R = (CPU time)/(CPU time for fastest formulation) . The ordinate is the fraction of instances solved to optimality in less than R times the fastest instance.

A very different behavior is observed for makespan minimization, which exhibits significant improvements in performance when our methods are used. An average 42% of additional instances are solved with the best formulations for each model in comparison to the original formulation. It can be observed that formulations that combine tightening constraints and reformulations appear

consistently as the best formulations to solve the makespan minimization problem. In particular, a formulation that combines F1 (constant time window tightening) and F3 (reformulation without priorities) appears in the performance chart for every model. Makespan minimization is the hardest problem we consider with only 78% of instances solved to optimality with the best formulations within the time limit.

Cost minimization is the intermediate objective in terms of difficulty and performance improvement when our methods are used. Although 100% of the instances were solved to optimality within an hour using our methods, the average times are higher than those for earliness minimization. In terms of performance, we observe that formulation F0 only solves 75% of the instances within 15 times the fastest formulation; this represents a 25% improvement in performance when the best of our methods is combined with each model. Although a general trend in terms of models is not as clear as it is for the other two objectives, it can be observed that the reformulations (F3, F4 and combinations) consistently rank among the best performers.

We now use performance charts to compare among different discrete-time models. We also include the continuous-time model M5 in order to show the computational capabilities of formulations currently available in the PSE literature. Figure 3.16 presents performance curves for each objective considered, when the best formulation for each model is used. From Figure 3.16 we observe that the best model for both earliness and makespan minimization is the STN-based formulation M1. For cost minimization, however, the continuous-time formulation, model M5, produces the best results, closely followed by the aggregated RCPSP-inspired model M3. Nonetheless, we observe for the objectives where discrete-time models are superior, the performance is much better. The percentage of additional instances solved to optimality with the discrete-time models is significant: 67% for earliness and 33% for makespan. Even if our solution methods are applied to model M5 for earliness and makespan minimization, its performance does not improve (results not shown).

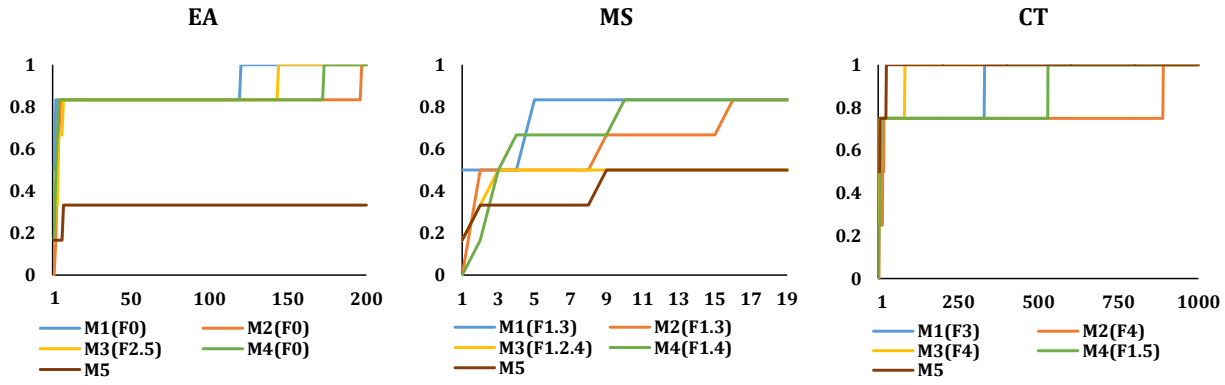


Figure 3.16. Best models for different objectives
Performance curves of models M1-M5 for each objective, earliness (EA), makespan (MS), and cost (CT). Models M1-M4 use the best formulation for a particular objective. The abscissa is the ratio $R = (\text{CPU time})/(\text{CPU time for fastest model})$. The ordinate is the fraction of instances solved to optimality in less than R times the fastest instance.

3.6.2. Makespan minimization in large instances

In this section we present a more detailed analysis for the solution of the makespan minimization problem for the largest instances in the references we used for the computational study. As mentioned before, makespan minimization is by far the hardest objective we consider and therefore additional solution techniques can be addressed in order to reduce the computational burden. We first revisit formulation F2 based on variable time windows and then present results of the solution refinement algorithm discussed in section 3.4.3.

Variable time window

Based on the performance analysis of the previous section, formulation F2 based on variable time windows does not seem to be competitive when compared against formulations F1 (constant time window) or F3 (reformulation). However, this apparent low performance is the result of averaging over both small- and large-scale instances. The preprocessing time required by F2 is comparable to the solution times of the former; however, this preprocessing is much shorter than the solution times of the latter. We now focus on facilities P2, P3, P4 and P6 which are the largest instances presented in their respective references[25, 26, 61] to show that significant improvements in computational time and optimality gap are achieved by using F2.

A total of 48 instances including these facilities are solved using different combinations of the four discrete-time models and three definitions of the makespan minimization objective. Table 3.8 summarizes aggregated computational results over the 48 instances. It can be observed that formulation F2 outperforms F0 in all the categories included in Table 3.8. By including tightening equation (3.72) based on variable time windows, we achieved significant reduction in total computational time (includes preprocessing for F2), optimality gap and number of instances for which a feasible solution was not returned within the time limit. The improvement in terms of optimality gap derived from formulation F2 is especially significant and is greater than that of any other formulation. This improvement is not reflected on the performance analysis of the previous section, which is based on fraction of instances solved to optimality and relative solution times.

Table 3.8. Aggregated computational results for formulations F0 and F2 for large-scale instances.

	F0	F2
Average CPU time (s)	3,171.6	2,470.4
Average gap (%)	29.4	6.2
No feasible solution returned (%)	25	6.25

For further illustration, Table 3.9 presents results for four particular instances, one for each facility considered. It is evident the benefits of using F2 for these large instances.

Table 3.9. Computational results for one instance of each large-scale facility

Instance	F0		F2			
	CPU time (s)	Gap (%)	Preproc. (s)	MIP time (s)	CPU time (s)	Gap (%)
P2.MS1.M2	3,600.1	99.0	90.3	3,600.1	3,690.4	7.25
P3.MS2.M1	3,600.0	4.76	17.2	495.5	512.7	---
P4.MS3.M3	3,600.1	14.2	8.5	555.7	564.2	---
P6.MS3.M4	3,600.0	NA[a]	45.4	3,600.3	3,645.7	3.3

[a] No feasible solution returned

Solution refinement

We now apply the method described in section 3.4.3 to a particular large-scale instance to test its effectivity. Let us revisit facility P6, denoted P13 in the original reference[61], which is adapted from a real industrial case study. Hitherto, a value of $\delta = 1$ has been used to solve approximations of this instance, but if exact solutions were to be found, the step size would need to be $\delta = 1 \times 10^{-3}$, given

the values of processing times (complete data can be found in the Supporting Information). Such small value for the discretization parameter renders the discrete-time intractable; even with the value of $\delta = 1$ the problem is hard to solve as discussed in the previous subsection.

The two-step refinement method is applied to P6: (i) an optimal solution with $\delta = 0.5$ is found, and (ii) model MH&C_LP is used to refine the timing decision variables using the assignment and sequencing obtained with the discrete-time model. For the first step, we note that optimality cannot be proved within the original time limit if equations (3.45)-(3.47) are used to explicitly define the variable MS, even for small estimates of the makespan value. Instead, the iterative approach used by Castro and Grossmann[26] is adapted to models M1-M4. An initial estimate for the makespan value is calculated based on minimum completion dates for batches and stages. Then the feasibility problem is iteratively solved with any objective function; each time a value of δ is added to the initial estimate. The makespan is determined when the problem first becomes feasible.

Table 3.10 presents the results of this procedure and compares them with the best estimate reported for this facility. We observe that a close value to the best reported solution has been found in a relatively short time considering the size of the problem. Both solution with $\delta \in \{0.5,1\}$ are better than the value reported for the full-space model.

Table 3.10. Comparison of makespan values for P6 with continuous-time solution and discrete-time refinement.

	This work		Previous work[53]	
Model	STN, F1.4, $\delta = 1$	STN, F1.4, $\delta = 0.5$	Full-space model	Decomposition alg
Makespan (h)	33.615	30.376	36.136	30.053
CPU time (s)	96.7	1,761.6	60,000[a,b]	5,833[a]

[a] Solved in a computer with 4 GB of RAM and an Intel Core2 Duo (T9300) processor at 2.5GHz running on Windows Vista
[b] Time limit reached

3.7. Conclusions

This chapter presents a new approach to modeling and solving scheduling problems in multistage batch facilities using discrete-time MIP models. Four different models are studied; two are network-based and two are inspired on the formulations for the RCPSP problem. The major advantages of

using this type of models as opposed to continuous-time counterparts are the simplicity to model different extensions and features common to scheduling problems in chemical facilities, as well as considerably better LP relaxations. In particular, we show how modeling utility requirements and product diversification can be achieved with small changes in the proposed formulations.

In terms of solution strategies, we explore two different ideas. On one hand we formulate tightening constraints based on time windows available for both tasks and units. A constant-window-based constraint for processing units uses the parameters for earliest start time and shortest tail that have been used before exclusively to limit the number of binary variables. Moreover, an extension to variable start times and tails is proposed in order to further tighten the formulations. On the other hand we use a reformulation in terms of number of batches processed by each unit and take advantage of the prioritization capabilities of CPLEX to improve the solution process and achieve lower computational time and optimality gap.

An extensive computational study revealed that the original discrete-time models are effective in solving the easiest of objectives discussed in this work, namely earliness minimization. However, a combination of methods is required to effectively reduce the solution times of both makespan and cost minimization. For the makespan objective we found that the constant-window constraint combined with the reformulation has the best performance in terms of computational time and percentage of instances solved to optimality. For the cost minimization problem, we could observe that reformulation methods were more effective than the tightening constraints.

In terms of model comparison, we found that all the proposed discrete-time models significantly outperform one of the best continuous-time models in the literature. In some cases, up to three order of magnitude reduction in computational time is achieved. Each objective has a discrete-time model for which the best performances are obtained. Both earliness and makespan minimization achieves the best results with the disaggregated RCPSP-inspired model M4. Cost minimization has its best

performance when solved using the aggregated RCPSP-inspired formulation M3. However, both network models M1 and M2 have comparable performances in term of number of instances solved to optimality and relative solution times.

We finally observe that for the hardest problem we consider, makespan minimization, additional techniques are required in order to obtain solution in reasonable time. In particular, we show how the optimality gap is dramatically improved using the variable-window formulation F2. Moreover, a refinement procedure is explained and tested to obtain competitive discrete-time solutions when the discretization introduces an approximation to the parameters and results.

We expect our methods to be readily applicable to a vast range of industrial and academic problems given the compact model formulations and straightforward implementation of the solutions strategies we propose. This work settles the basis for future research that might include extensions to other common features such as sequence-dependent changeovers, limited/shared storage and simultaneous batching and scheduling.

3.8. Notation

Indices

$a \in \mathbf{A}$	Activities (RCPSP)
$i \in \mathbf{I}$	Batches (MBPSP)
$i \in \mathbf{I}^N$	Tasks (Network)
$j \in \mathbf{J}/\mathbf{J}^{STN}$	Processing units (MBPSP/Network)
$k \in \mathbf{K}$	Stages (MBPSP)
$l \in \mathbf{L}$	Subsets of batches that share common raw materials and intermediates (MBPSP)
$m \in \mathbf{M}^{STN}$	Materials (Network)
$r \in \mathbf{R}^P/\mathbf{R}^{RTN}$	Resources (RCPSP/ Network)
$t \in \mathbf{T}/\mathbf{T}^N/\mathbf{T}^P$	Time points (MBPSP/Network/RCPSP)

$u \in \mathbf{U}$	Utilities (MBPSP)

Sets

$\mathbf{E}$	Precedence relations (RCPSP)
$\mathbf{I}_j$	Batches that can be processed in unit $j \in \mathbf{J}$ (MBPSP)
$\mathbf{I}_j^{STN}$	Tasks that can be processed in unit $j \in \mathbf{J}^{STN}$ (Network)
$\mathbf{I}_{jt}$	Batches that can start in unit $j \in \mathbf{J}$ at time point $t \in \mathbf{T}$ (MBPSP)
$\mathbf{I}_m^+/\mathbf{I}_m^-$	Tasks that produce/consume material $m \in \mathbf{M}^N$ (Network)
$\mathbf{I}_k$	Batches that are processed on stage $k \in \mathbf{K}$ (MBPSP)
$\mathbf{I}_l$	Batches that belong to subset $l \in \mathbf{L}$ (MBPSP)
$\mathbf{I}_r^{RTN}$	Tasks that interact with resource $r \in \mathbf{R}^{RTN}$
$\mathbf{J}^{C,N}$	Customers (Network)
$\mathbf{J}_i$	Units that can process batch $i \in \mathbf{I}$ (MBPSP)
$\mathbf{J}_i^{STN}$	Units that can process task $i \in \mathbf{I}^N$ (Network)
$\mathbf{J}_{it}$	Units that can start processing of batch $i \in \mathbf{I}$ at time point $t \in \mathbf{T}$ (MBPSP)
$\mathbf{J}_k$	Units that belong to stage $k \in \mathbf{K}$ (MBPSP)
$\mathbf{J}_{ik}$	Units that can process batch $i \in \mathbf{I}$ on stage $k \in \mathbf{K}$, i.e. $\mathbf{J}_{ik} \equiv \mathbf{J}_i \cap \mathbf{J}_k$ (MBPSP)
$\mathbf{K}_i$	Stages on which batch $i \in \mathbf{I}$ is processed (MBPSP)
$\mathbf{K}_{ik}^-$	Stages in which batch $i \in \mathbf{I}$ has to be processed before stage $k \in \mathbf{K}_i$ (MBPSP)
$\mathbf{K}_{ik}^+$	Stages in which batch $i \in \mathbf{I}$ has to be processed after stage $k \in \mathbf{K}_i$ (MBPSP)
$\mathbf{M}^{F,N}$	Final products (Network)
$\mathbf{M}^{R,N}$	Raw materials (Network)
$\mathbf{T}_{ij}$	Time points at which batch $i \in \mathbf{I}$ is allowed to start its execution in unit $j \in \mathbf{J}_i$ (MBPSP)

Parameters

α_{ij} Fixed cost of processing batch $i \in \mathbf{I}$ in $j \in \mathbf{J}_i$ (MBPSP)

α_{ij}^{N} Fixed cost of processing task $i \in \mathbf{I}^N$ in $j \in \mathbf{J}_i^N$ (Network)

α_{ut}^{U} Cost of utility $u \in \mathbf{U}$ during interval $t \in \mathbf{T}\backslash\{\tau^F\}$ (MBPSP)

$\beta_{ir}^{m,RTN}/\beta_{ir}^{M,RTN}$ Minimum/maximum capacity utilization of resource $r \in \mathbf{R}^{RTN}$ by task $i \in \mathbf{I}_r^{RTN}$ (Network)

$\beta_{ij}^{m,STN}/\beta_{ij}^{M,STN}$ Minimum/maximum batch size of task $i \in \mathbf{I}^N$ in unit $j \in \mathbf{J}_i^{STN}$ (Network)

β_r^P Number of available units of resource $r \in \mathbf{R}$ (RCPSP)

$\beta_{ar}^{A,P}$ Number of units of resource $r \in \mathbf{R}$ required by activity $a \in \mathbf{A}$ (RCPSP)

γ_k Lower bound on the sum of actual start times on stage $k \in \mathbf{K}$ (RCPSP)

γ_{rt}^{RTN} Maximum storage allowed for resource $r \in \mathbf{R}^{RTN}$ during interval $t \in \mathbf{T}^N\backslash\{0\}$ (Network)

γ_m^{STN} Storage capacity for material $m \in \mathbf{M}^{STN}$ (Network)

δ Step size for time discretization (MBPSP)

δ^N Step size for time discretization (Network)

$\varepsilon_{ij}^{U}/\varepsilon_{ik}^{S}$ Earliest start time for batch $i \in \mathbf{I}$ in unit $j \in \mathbf{J}_i$/on stage $k \in \mathbf{K}_i$ (MBPSP)

ε_j Earliest start time for unit $j \in \mathbf{J}$ (MBPSP)

$\varepsilon_{ij}^{U,N}$ Earliest start time of task $i \in \mathbf{I}^N$ in unit $j \in \mathbf{J}_i^N$ (Network)

ε_j^N Earliest start time of unit $j \in \mathbf{J}^N$ (Network)

ζ_k Lower bound on the sum of actual tails on stage $k \in \mathbf{K}$ (RCPSP)

η^N Scheduling horizon (Network)

η^P Project duration (RCPSP)

θ_j Time window for unit $j \in \mathbf{J}$ (MBPSP)

$\theta_{ij}^{U,N}$ Available time window for task $i \in \mathbf{I}^N$ in unit $j \in \mathbf{J}_i^N$ (Network)

θ_j^N Available time window for unit $j \in \mathbf{J}^N$ (Network)

$\kappa_i^{FS}/\kappa_i^{LS}$	First/last stage on which batch $i \in \mathbf{I}$ can be processed (MBPSP)
κ_j	Stage to which unit $j \in \mathbf{J}$ belongs (MBPSP)
λ_j	Latest finish time for unit $j \in \mathbf{J}$ (MBPSP)
$\lambda_{ij}^U/\lambda_{ik}^S$	Latest finish time for batch $i \in \mathbf{I}$ in unit $j \in \mathbf{J}_i$/on stage $k \in \mathbf{K}_i$ (MBPSP)
$\bar{\mu}_i/\mu_i$	Release time for batch $i \in \mathbf{I}$ in real time/in terms of time grid (MBPSP)
ν_{ik}	Defines the next stage on which batch $i \in \mathbf{I}$ can be processed after stage $k \in \mathbf{K}_i$, i.e. batch i is processed on stage $k + \nu_{ik}$ immediately after stage k (MBPSP)
ξ_{iju}	Fixed requirement of utility $u \in \mathbf{U}$ for batch $i \in \mathbf{I}$ in unit $j \in \mathbf{J}_i$ (MBPSP)
ξ_{rt}^{RTN}	Amount of resource $r \in \mathbf{R}^{RTN}$ made available or unavailable at time $t \in \mathbf{T}^N$ (Network)
ξ_{mt}^{STN}	Delivery or demand for material $m \in \mathbf{M}^{STN}$ (Network)
ρ_{irt}^{RTN}	Number of normalized continuous units of resource $r \in \mathbf{R}^{RTN}$ that interact with task $i \in \mathbf{I}_r^{RTN}$ during interval $t \in \mathbf{T}^N \backslash \{0\}$ (Network)
ρ_{im}^{STN}	Conversion coefficient for material $m \in \mathbf{M}^{STN}$ produced or consumed by task $i \in \mathbf{I}^N$ (Network)
σ_j	Shortest tail for unit $j \in \mathbf{J}$ (MBPSP)
$\sigma_{ij}^{U,N}$	Shortest tail of task $i \in \mathbf{I}^N$ in unit $j \in \mathbf{J}_i^N$ (Network)
σ_j^N	Shortest tail of unit $j \in \mathbf{J}^N$ (Network)
τ^0	Start of the scheduling horizon (MBPSP)
τ^F	End of the scheduling horizon (MBPSP)
τ_a^P	Duration of activity $a \in \mathbf{A}$ (RCPSP)
$\bar{\tau}_{ij}/\tau_{ij}$	Processing time of batch $i \in \mathbf{I}$ in unit $j \in \mathbf{J}$ in real time/in terms of time grid (MBPSP)
$\bar{\tau}_i^{RTN}/\tau_i^{RTN}$	Duration of task $i \in \mathbf{I}^N$ in real time units/in terms of time points (Network)

$\bar{\tau}_{ij}^{STN}/\tau_{ij}^{STN}$ Processing time of task $i \in \mathbf{I}^N$ in unit $j \in \mathbf{J}_i^{STN}$ in real time/in terms of time grid (Network)

$\bar{\phi}_i/\phi_i$ Due date for batch $i \in \mathbf{I}$ in real time/in terms of time grid (MBPSP)

$\bar{\phi}_{mj}^{C,N}/\phi_{jm}^{C,N}$ Due date of product $m \in \mathbf{M}^{F,N}$ for customer $j \in \mathbf{J}^{C,N}$ in real time/in terms of time grid (Network)

$\bar{\phi}_m^N/\phi_m^N$ Due date for final product $m \in \mathbf{M}^{F,N}$ in real time/in terms of time grid (Network)

χ_{irt}^{RTN} Number of discrete units of resource $r \in \mathbf{R}^{RTN}$ that interact with task $i \in \mathbf{I}_r^{RTN}$ during interval $t \in \mathbf{T}^N \backslash \{0\}$ (Network)

ψ_{ut} Maximum availability of utility $u \in \mathbf{U}$ during interval $t \in \mathbf{T} \backslash \{\tau^F\}$ (MBPSP)

$\omega_{ii'jk}^{\#}$ Indicator parameters to identify redundant constraints when defining model MH&G_LP used to refine discrete-time schedules, $\# \in \{1,2,3\}$ (MBPSP)

M Large number for continuous-time big-M formulations (RCPSP)

Binary/integer variables

N_{ij}^{STN} Number of batches of task $i \in \mathbf{I}^N$ in unit $j \in \mathbf{J}_i^{STN}$ (Network)

N_j Number of batches assigned to unit $j \in \mathbf{J}$ (MBPSP)

R_{jt} One if unit $j \in \mathbf{J}$ is available during time interval $t \in \mathbf{T} \backslash \{\tau^F\}$ (MBPSP)

S_{ikt} One if the material produced by batch $i \in \mathbf{I}$ after stage $k \in \mathbf{K}_i$ is available during interval $t \in \mathbf{T} \backslash \{\tau^F\}$ (MBPSP)

$S_{mt}^{B,N}$ One if material $m \in \mathbf{M}^N$ is available at time $t \in \mathbf{T}^N$ (Network)

X_{ij}^C One if batch $i \in \mathbf{I}$ is assigned to unit $j \in \mathbf{J}_i$ (MBPSP, continuous-time)

X_{ijt} One if batch $i \in \mathbf{I}$ starts being processed in unit $j \in \mathbf{J}_i$ at time $t \in \mathbf{T}_{ij}$ (MBPSP)

X_{it}^{RTN} Discrete extent. It is equal to one if task $i \in \mathbf{I}^N$ starts at time $t \in \mathbf{T}^N$ (Network)

X_{ijt}^{STN} One if task $i \in \mathbf{I}^N$ starts in unit $j \in \mathbf{J}_i^{STN}$ at time $t \in \mathbf{T}^N$ (Network)

$Y_{ii'k}$ One if batch $i \in \mathbf{I}$ is processed before batch $i' \in \mathbf{I}$ on stage $k \in \mathbf{K}_i \cap \mathbf{K}_{i'}$ (MBPSP, continuous-time)

Z_{mjt}^{N} One if product $m \in \mathbf{M}^{F,N}$ is delivered to customer $j \in \mathbf{J}^{C,N}$ at time $t \in \mathbf{T}^{N}$ (Network)

Continuous variables

B_{it}^{RTN} Continuous extent. $\rho_{irt}^{RTN} B_{it}^{RTN}$ is the total number of continuous units of resource $r \in \mathbf{R}^{RTN}$ that interact with task $i \in \mathbf{I}_r^{RTN}$ during interval $t \in \mathbf{T}^{N} \backslash \{0\}$ (Network)

B_{ijt}^{STN} Batch size of task $i \in \mathbf{I}^{N}$ that starts in unit $j \in \mathbf{J}_i^{STN}$ at time $t \in \mathbf{T}^{N}$ (Network)

C_{ik} Completion time for batch $i \in \mathbf{I}$ on stage $k \in \mathbf{K}_i$ (MBPSP, continuous-time)

E_{ik}^{T} Actual start time of batch $i \in \mathbf{I}$ on stage $k \in \mathbf{K}_i$ (MBPSP)

$E_{ij}^{N,T}$ Actual start time of task $i \in \mathbf{I}^{N}$ in unit $j \in \mathbf{J}_i^{N}$ (Network)

MS Makespan (MBPSP, Network)

R_{rt}^{RTN} Excess resource $r \in \mathbf{R}^{RTN}$ during interval $t \in \mathbf{T}^{N} \backslash \{0\}$ (Network)

S_{ik}^{T} Actual tail of batch $i \in \mathbf{I}$ on stage $k \in \mathbf{K}_i$ (MBPSP)

$S_{ij}^{N,T}$ Actual tail of task $i \in \mathbf{I}^{N}$ in unit $j \in \mathbf{J}_i^{N}$ (Network)

S_{mt}^{STN} Inventory level of material $m \in \mathbf{M}^{STN}$ during interval $t \in \mathbf{T}^{N} \backslash \{0\}$ (Network)

T^{F} Variable end of the scheduling horizon (MBPSP)

W_{ut} Amount of utility $u \in \mathbf{U}$ consumed during time interval $t \in \mathbf{T} \backslash \{\tau^{F}\}$ (MBPSP)

Other symbols

$\mathbb{S}(\cdot)$ Materials (states) in the STN representation

$\mathbb{T}(\cdot)$ Tasks in the STN representation

$\mathbb{U}.$ Processing units in the RTN representation and the MPSPS description

Chapter 4

Solution methods for continuous-time models in network environments[3]

The main idea of this chapter is extend the reformulations[51] and tightening methods[59] developed for material-based discrete-time models to continuous-time models. At the same time, we explore different options that are particular to the nature of continuous-time models and evaluate their performance in terms of computational time and optimality gap improvement.

This chapter is structured as follows. In section 4.1 we present background material, describing the general problem, the continuous-time models to which the methods are applied, and the objective functions we consider. In section 4.2, we discuss the application of reformulations to continuous-time models and section 4.3 addresses tightening methods for these models. Section 4.4 presents specific computational results. Finally, in section4.5 we draw general conclusions based on the insights we are able to develop.

4.1. Background

4.1.1. Problem statement

We consider network facilities that do not include material-handling restrictions (i.e. batch splitting/mixing is allowed), for which the scheduling problem determines the number, size, assignment, sequencing, and timing of a set of bathes consuming/producing a set of materials and being processed on a given set of units. Its input includes data for equipment availability, unit capacity and connectivity, unit-task compatibility, as well as production recipes, targets, and costs. We use the STN representation[15], which employs the following sets:

$\mathbf{I} = \{i: i \text{ is a task}\}$

[3] This chapter is modified from Merchan et al.[95] and Merchan and Maravelias[96]

$\mathbf{J} = \{j: j \text{ is a unit}\}$

$\mathbf{J}_i = \{j \in \mathbf{J}: \text{unit } j \text{ can process task } i\}$

$\mathbf{K} = \{k: k \text{ is a material}\}$

$\mathbf{I}_k^+ = \{i \in \mathbf{I}: \text{task } i \text{ produces material } k\}$

$\mathbf{I}_k^- = \{i \in \mathbf{I}: \text{task } i \text{ consumes material } k\}$

The main assumptions for the STN-based representation we use in this work are: (a) a task cannot be interrupted once it has started (no preemption); (b) every task ends within the scheduling horizon; (c) task processing times are constant; (d) utilities and other shared resources are not included; (e) each material has a dedicated storage vessel; (f) connections among processing units and between processing units and storage vessels are available as required by the production recipe; (g) material transfer between units is instantaneous; (h) changeover/setup times and costs are not included; and (i) problem data are deterministic and fixed over the scheduling horizon.

Assumption (a) is necessary in chemical production scheduling. Assumptions (b) – (d) can be trivially relaxed, while assumptions (e) and (f) can be relaxed with minor model modifications. Relaxing assumptions (g) and (h) has been done using STN- and RTN-based models but requires extensive changes. Finally, assumption (i) leads to substantially harder problems.

It is important to note that only assumptions (a) and (b) are strictly required for the reformulations we will present. In other words, our reformulations can be applied to models that account for all remaining problem features and constraints (changeovers, shared storage, intermediate demands, non-instantaneous transfers, etc.).

Under assumptions (a) – (i), each set is associated with specific parameters that define a particular instance of the problem,

$\tau_{ij}^{F}/\tau_{ij}^{V}$ Fixed/variable processing time for task $i \in \mathbf{I}$ in unit $j \in \mathbf{J}_i$

$\beta_j^{max}/\beta_j^{min}$ Maximum/minimum batch size for unit $j \in \mathbf{J}$

γ_k Storage capacity for material $k \in \mathbf{K}$

ρ_{ik} Fraction of material $k \in \mathbf{K}$ produced ($\rho_{ik} > 0 \ \forall k \in \mathbf{K}, \forall i \in \mathbf{I}_k^+$) or consumed ($\rho_{ik} < 0 \ \forall k \in \mathbf{K}, \forall i \in \mathbf{I}_k^-$) by task $i \in \mathbf{I}$

4.1.2. Mathematical models

Based on the classification recently introduced by Maravelias[10], the models we consider in this work use a material-based approach and a global time-grid-based framework with continuous time representation to handle the sequencing and timing of tasks among units. We chose global rather than unit-specific time-grid models, so that we can (1) readily model consumption/production of materials by different tasks carried out on different units and (2) accommodate shared resources (e.g. utilities) into our reformulations without added complexity. In general, a global continuous-time formulation divides the production horizon η into $(N-1)$ *time periods* of unknown length that are common across all units. A new set of *time points* $\mathbf{N} = \{n \in \mathbb{Z}: 1 \leq n \leq N\}$ is defined, so that the task-unit assignment is mapped onto this time grid. An interval n runs between time points n and $(n+1)$. One important advantage of continuous-time models is that they can seamlessly account for variable processing times, which is harder to achieve in discrete-time formulations[57].

For the most part, global continuous-time models employ at least the following decision variables:

T_n Continuous. Position of time point $n \in \mathbf{N}$

X_{ijn}/Y_{ijn} Binary. It is equal to one if task $i \in \mathbf{I}$ starts/finishes in unit $j \in \mathbf{J}_i$ at time $n \in \mathbf{N}$

BS_{ijn}/BF_{ijn} Continuous. Batch size of task $i \in \mathbf{I}$ that starts/finishes in unit $j \in \mathbf{J}_i$ at time $n \in \mathbf{N}$

BP_{ijn} Continuous. Batch size of task $i \in \mathbf{I}$ that continues to be processed in unit $j \in \mathbf{J}_i$ at time $n \in \mathbf{N}$

S_{mn} Continuous. Inventory level of material $m \in \mathbf{M}$ at point time $n \in \mathbf{N} \cup \{0\}$; $S_{m0} = \gamma_m^0$ is a known parameter that corresponds to the initial inventory of material $m \in \mathbf{M}$

Material-based approaches explicitly optimize the number and size of batches and guarantee the sequencing through material balances. As a result, most global continuous-time models share similar ideas for the modeling of those two components of the scheduling problem. It is the modeling of the assignment and timing that introduces the major differences between them. In order to assess the effectiveness of our methods when applied to different models, three representative global continuous-time models are used as the basis for the present work. We expect the results we obtain and the conclusions we draw in this work to be similar for other continuous-time models. Next, a description of each model is provided. Equations and additional variables required for each model can be found in Appendix A.

Model M&G

The first model we consider was developed by Maravelias and Grossmann[30] using Boolean logic to enforce the task-unit assignment followed by a combined convex hull/big-M reformulation of the resulting hybrid generalized disjunctive programming (GDP). The time modeling guarantees a task must start at a time point, although it can finish anytime during a subsequent interval. We will refer to this model as M&G.

Model S&K

Sundaramoorthy and Karimi[31] used the concept of *recipe diagram* of a batch production facility to develop a formulation for the scheduling problem that is based on four fundamental balances,

[95]namely, resources (process units), processing times, material residing in process units, and material inventory in storage vessels. We will refer to this formulation as S&K. It is important to note that model S&K introduces a slight variation in the way the set of tasks is defined. An *idle task*, $i = 0$, is defined so that it occupies a time slot when no task is assigned to a given unit, but no batch sizing variables are assigned to it. As a result we redefine $\mathbf{I} = \{i: i \text{ is a task}\} \cup \{0\}$.

Model GH&M

In 2009, Giménez et al.[32, 95] published two papers to present models that are able to handle many features that typically appear in scheduling problems, such as temporal material storage in units, material transfers, shared resources and additional operational tasks (e.g. maintenance, cleaning). This formulation is based on two fundamental concepts: process unit states (i.e. execution, storage or idle) and time balances. A task formally starts (ends) at a time point, but its actual starting (finishing) time is not required to coincide with a specific time point, which implies that materials could be stored temporarily in the unit before (after) the actual task starts (ends). We will refer to this model as GH&M and focus on the features that are also handled by the models M&G and S&K.

4.1.3. Objective functions

There are two main categories for objective functions in scheduling problems: economic functions and time-based functions. The former include sales, cost and profit, whereas the latter include makespan, tardiness/lateness and earliness. In the present study we consider sales, cost and makespan. The following additional parameters are required for the evaluation of the objective functions.

π_m	Per-unit price of material $m \in \mathbf{M}$
$\alpha_{ij}^F / \alpha_{ij}^V$	Fixed/variable processing cost for task $i \in \mathbf{I}$ in unit $j \in \mathbf{J}_i$
ξ_m	Demand of material $m \in \mathbf{M}$ at the end of the horizon

The sales objective function can be expressed by equation (4.1). This is only used for the reformulation methods. Tightening methods involving maximization problems are described in Chapter 2 of this thesis.

$$\max \sum_{m \in \mathbf{M}} \pi_m S_{mN} \tag{4.1}$$

In order to define objective functions involving cost and makespan, constraint (4.2) needs to be enforced to meet the demand requirements at the end of the horizon. The production cost and makespan objectives are defined through (4.3) and (4.4) respectively.

$$S_{mN} \geq \xi_m \qquad \forall m \in \mathbf{M} \tag{4.2}$$

$$\min \sum_{i \in \mathbf{I}} \sum_{j \in \mathbf{J}_i} \sum_{n \in \mathbf{N}} \left(\alpha_{ij}^F X_{ijn} + \alpha_{ij}^V BS_{ijn} \right) \tag{4.3}$$

$$\min MS \tag{4.4}$$

Where $MS \in \mathbb{R}_+$ represents the makespan. Additional constraints and/or model modifications are required for makespan minimization as explained in Appendix A.

4.2. Reformulations in continuous-time models.

In order to extend the reformulation introduced by Velez and Maravelias[51] to the three continuous-time models described in section 4.1.2, we study methods 1 and 2 in their work, which define, respectively, an integer variable and a set of binary variables to quantify the number of batches of a particular task. We also introduce two additional methods as alternatives to method 2, with the aim of exploring formulations with a number of binary variables and constraints that is logarithmic, rather than linear, in the number of batches. Moreover, since most continuous-time models employ two distinct binary variables to account for the start and the end of a task, we present three variants for each method. A particular variant depends on whether the number of batches is

defined through the starting or finishing binary variables or both. A detailed explanation of each method is presented below, with the equations written in terms of a general binary variable U_{ijn}.

Depending on what binary variable is used in place of U_{ijn}, the variants are defined as follows:

(i) Variant A: $U_{ijn} = X_{ijn}$

(ii) Variant B: $U_{ijn} = Y_{ijn}$

(iii) Variant C: The equations are written twice, for $U_{ijn} = X_{ijn}$ and for $U_{ijn} = Y_{ijn}$, based on the assumption that if a task starts at some time point, it must finish within the given horizon.

4.2.1. Method 1

Velez and Maravelias[51] declared an explicit integer variable N_{ij} to be the total number of times task $i \in \mathbf{I}$ runs in unit $j \in \mathbf{J}_i$, as stated in equation (4.5).

$$N_{ij} = \sum_{n \in \mathbf{N}} U_{ijn} \qquad \forall i \in \mathbf{I}, \forall\, j \in \mathbf{J}_i \tag{4.5}$$

It is possible to bound N_{ij} using the maximum number of times task $i \in \mathbf{I}$ can run in unit $j \in \mathbf{J}_i$ before the end of the time horizon,

$$0 \leq N_{ij} \leq \left\lfloor \frac{\eta}{\tau_{ij}^F + \tau_{ij}^V \beta_j^{min}} \right\rfloor \qquad \forall i \in \mathbf{I}, \forall j \in \mathbf{J}_i \tag{4.6}$$

$$N_{ij} \in \mathbb{Z} \qquad \forall i \in \mathbf{I}, \forall j \in \mathbf{J}_i \tag{4.7}$$

Where τ_{ij}^F (τ_{ij}^V) is the fixed (variable) processing time for task $i \in \mathbf{I}$ in unit $j \in \mathbf{J}_i$.

4.2.2. Method 2

Velez and Maravelias[51] also used the equivalence between general-integer and binary-integer programming, to implicitly define and bound the total number of times task $i \in \mathbf{I}$ runs in unit $j \in \mathbf{J}_i$. Let us define the set $\mathbf{K}_{ij} = \left\{k \in \mathbb{Z}: 0 \leq k \leq \left\lfloor \eta / \left(\tau_{ij}^F + \tau_{ij}^V \beta_{ij}^{min}\right) \right\rfloor\right\}$ $\forall i \in \mathbf{I}, \forall\, j \in \mathbf{J}_i$, and let W_{ijk} be a

binary variable that is one if there are $k \in \mathbf{K}_{ij}$ batches of task $i \in \mathbf{I}$ running in unit $j \in \mathbf{J}_i$. Then we can write,

$$\sum_{k \in \mathbf{K}_{ij}} kW_{ijk} = \sum_{n \in \mathbf{N}} U_{ijn} \qquad \forall i \in \mathbf{I}, \forall\, j \in \mathbf{J}_i \tag{4.8}$$

Furthermore, only one binary variable can be nonzero for each task-unit pair,

$$\sum_{k \in \mathbf{K}_{ij}} W_{ijk} = 1 \qquad \forall i \in \mathbf{I}, \forall j \in \mathbf{J}_i \tag{4.9}$$

$$W_{ijk} \in \{0,1\} \qquad \forall i \in \mathbf{I}, \forall j \in \mathbf{J}_i, \forall k \in \mathbf{K}_{ij} \tag{4.10}$$

4.2.3. Method 3

Equation (4.8) introduces a number of binary variables that is linear in $|\mathbf{K}_{ij}|$, i.e., it requires as many binary variables as elements in the set $\mathbf{K}_{ij}$. There exists an alternative formulation for the equivalence between general-integer and binary-integer programming due to Watters[96], that is logarithmic in $\left|\mathbf{K}_{ij}\right|$, i.e. only $\left\lceil \log_2 |\mathbf{K}_{ij}| \right\rceil$ binary variables are required.

Let us define the set $\mathbf{LK}_{ij} = \left\{ l \in \mathbb{Z}\colon 0 \leq l \leq \left\lfloor \log_2 \left\lfloor \eta / \left(\tau_{ij}^F + \tau_{ij}^V \beta_{ij}^{min} \right) \right\rfloor \right\rfloor \right\} \forall i \in \mathbf{I}, \forall j \in \mathbf{J}_i$, and let W_{ijl}^* be a binary variable. Equations (4.11) and (4.12) implicitly define the total number of times task $i \in \mathbf{I}$ runs in unit $j \in \mathbf{J}_i$.

$$\sum_{l \in \mathbf{LK}_{ij}} 2^k W_{ijl}^* = \sum_{n \in \mathbf{N}} U_{ijn} \qquad \forall i \in \mathbf{I}, \forall\, j \in \mathbf{J}_i \tag{4.11}$$

$$\sum_{n \in \mathbf{N}} U_{ijn} \leq \left\lfloor \frac{\eta}{\tau_{ij}^F + \tau_{ij}^V \beta_j^{min}} \right\rfloor \qquad \forall i \in \mathbf{I}, \forall j \in \mathbf{J}_i \tag{4.12}$$

$$W_{ijl}^* \in \{0,1\} \qquad \forall i \in \mathbf{I}, \forall\, j \in \mathbf{J}_i, \forall l \in \mathbf{LK}_{ij} \tag{4.13}$$

4.2.4. Method 4

From equation (4.9) it is clear that the set $\{W_{ijk}\}_{k\in\mathbf{K}_{ij}}$ is a specially ordered set of type 1 (SOS1) However, since W_{ijk} can only take binary values, there is no practical difference in declaring them as binary or SOS1 variables. In a recent work, Vielma and Nemhauser[97] proposed a new type of formulation for specially ordered sets, in which only a logarithmic number of variables and additional constraints is required. This formulation proved to be superior to both the traditional SOS formulation with a linear number of binary variables and constraints and the specialized branching schemes embedded in commercial solvers.

In addition to $\mathbf{K}_{ij}$ and $\mathbf{LK}_{ij}$ defined in methods 2 and 3 respectively, Vielma and Nemhauser[95] introduced the sets $\mathbf{K}_{ij}^{+}(l,B)=\{k\in\mathbf{K}_{ij}:(B(k))_l=1\}$ and $\mathbf{K}_{ij}^{0}(l,B)=\{k\in\mathbf{K}_{ij}:(B(k))_l=0\}$, where $B:\mathbf{K}_{ij}\rightarrow\{0,1\}^{\lceil\log_2|\mathbf{K}_{ij}|\rceil}$ is any injective function (i.e. every element in the range of B is the image of at most one element in its domain). Equations (4.9) and (4.10) in method 2 are then replaced by equations (4.14) through (4.18).

$$\sum_{k\in\mathbf{K}_{ij}} W_{ijk}\leq 1 \qquad \forall i\in\mathbf{I},\forall j\in\mathbf{J}_i \tag{4.14}$$

$$\sum_{k\in\mathbf{K}_{ij}^{+}(l,B)} W_{ijk}\leq W_{ijl}^{*} \qquad \forall i\in\mathbf{I},\forall\, j\in\mathbf{J}_i \tag{4.15}$$

$$\sum_{k\in\mathbf{K}_{ij}^{0}(l,B)} W_{ijk}\leq 1-W_{ijl}^{*} \qquad \forall i\in\mathbf{I},\forall j\in\mathbf{J}_i \tag{4.16}$$

$$W_{ijk}\geq 0 \qquad \forall i\in\mathbf{I},\forall\, j\in\mathbf{J}_i,\forall k\in\mathbf{K}_{ij} \tag{4.17}$$

$$W_{ijl}^{*}\in\{0,1\} \qquad \forall i\in\mathbf{I},\forall j\in\mathbf{J}_i,\forall l\in\mathbf{LK}_{ij} \tag{4.18}$$

4.3. Tightening methods in continuous-time models

The proposed methodology consists of (1) using the network structure, unit capacities, recipes and demand information to calculate four parameters, and (2) defining valid inequalities that use

these parameters to tighten the original formulations[59]. The first parameter, ω_k, defines the minimum amount of material k that is required to meet the given demand. The second parameter, μ_i, gives the minimum production that task i needs to yield in order to satisfy customer demand. A third parameter, κ_k, is introduced to represent the minimum number of batches required to produce material k. Finally, a fourth parameter, λ_i, defines the minimum number of batches of task i. Figure 4.1 illustrates how the available information is used to calculate the parameters involved in the tightening constraints. Figure 4.2 shows how the demand information is propagated backwards to calculate the important parameters for a small illustrative network.

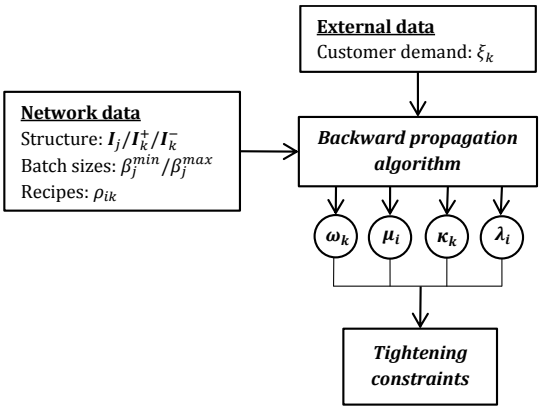
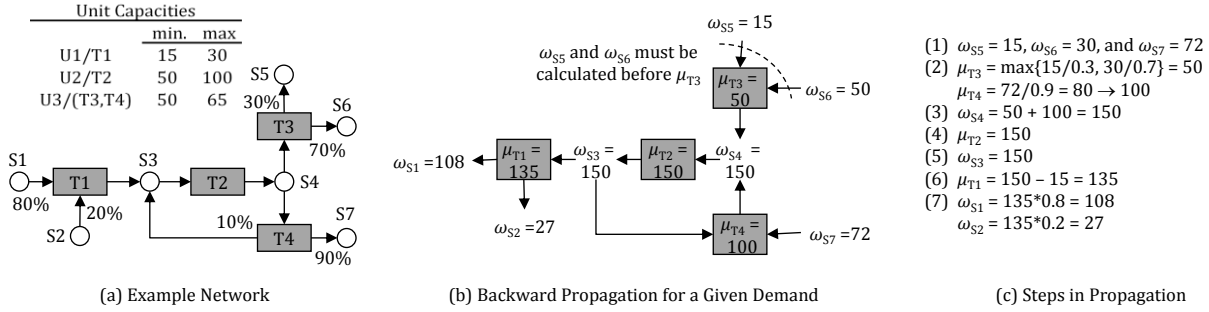


Figure 4.1. Simplified flowchart for tightening methods of Velez et al.[59]
Different calculated parameters interactions to define valid inequalities are shown



(a) Example Network (b) Backward Propagation for a Given Demand (c) Steps in Propagation

Figure 4.2. Backward propagation algorithm applied to a small network with recycle of material.
The steps in propagation are obtained as follows. (1) Given customer demand. (2) T3 processes enough for S5 and S6. Initially neglect S3, U3 can produce 50-65, 100-130, and so on (based on capacity), therefore μ_T4 is increased from 80 to 100. (3) Requirement for T3 and T4. (4) and (5) are straightforward. (6) T4 can supply up to 15(=150*0.1) of S3, so the remaining comes from T1. (7) From recipe.

Two different types of tightening constraints can be defined. The minimum number of batches processed by a given task provides a lower bound for the sum of assignment variables as defined by equation (4.19). When a task is carried out in units with different capacities, then equation (4.20), based on the minimum production requirement for a given task, leads to tighter formulations.

$$\sum_{j\in\mathbf{J}_i}\sum_{n\in\mathbf{N}} X_{ijn} \geq \lambda_i \qquad \forall i \in \mathbf{I} \tag{4.19}$$

$$\sum_{j\in\mathbf{J}_i}\sum_{n\in\mathbf{N}} \beta_j^{max} X_{ijn} \geq \mu_i \qquad \forall i \in \mathbf{I} \tag{4.20}$$

When a material is produced by multiple tasks, equation (4.21) lower-bounds the number of all batches producing material k, and if these tasks are carried out in units with different capacities, equation (4.22) provides additional tightening.

$$\sum_{i\in\mathbf{I}_k^+}\sum_{j\in\mathbf{J}_i}\sum_{n\in\mathbf{N}} X_{ijn} \geq \kappa_k \qquad \forall k \in \mathbf{K}^{MT} \tag{4.21}$$

$$\sum_{i\in\mathbf{I}_k^+}\sum_{j\in\mathbf{J}_i}\sum_{n\in\mathbf{N}} \rho_{ik}\beta_j^{max} X_{ijn} \geq \omega_k \qquad \forall k \in \mathbf{K}^{MT} \tag{4.22}$$

Where $\mathbf{K}^{MT} = \{k \in \mathbf{K}: k \text{ is produced by multiple tasks}\}$.

The tightening constraints defined above can be used separately or combined. In the present work, we define three distinct formulations. Formulation 1 only uses equations (4.19) and (4.21) based on the minimum number of batches; formulation 2 only uses equations (4.20) and (4.22) based on minimum production requirements; and formulation 3 uses all four equations. Other combinations of equations proved to be ineffective to tighten the original formulation. In addition, equations (4.19)-(4.22) are written for the starting binary variable, X_{ijn}, but it is also possible to define the tightening constraints by replacing X_{ijn} with the finishing binary variable, Y_{ijn}. Moreover, if we assume that every task that starts must finish during the scheduling horizon, the same

constraint can be written twice, once for X_{ijn} and once for Y_{ijn}. In this work we distinguish these possibilities as *variants* of a given formulation. Variant A corresponds to using X_{ijn}, variant B uses Y_{ijn}, and variant C uses both.

4.4. Computational studies

4.4.1. Problems, formulations, instances and runs

We follow the general ideas presented in Sundaramoorthy and Maravelias[68] to define, classify and identify every computational run in the present study.

A *problem* type is defined as a combination of objective function OBJ and specific feature FEAT and is represented by the pair [OBJ].[FEAT]. We consider sales (SLS) maximization, as well as makespan (MS) and cost (CT) minimization. The two features in which we are interested are constant and variable processing times (CPT, VPT).

A *problem formulation* is a unique MIP representation of a given problem, once the model (MOD) and solution method have been selected. For each of the three continuous-time models under study (M&G, S&K, GH&M), we consider several problem formulations, which include the original, identified as F0, and the combinations of methods and variants described above, represented by F#X. For reformulations # corresponds to the method (1-4), whereas for tightening constraints it denotes a specific formulation (1-3). In both cases X defines the variant (A-C). The representation for a problem formulation is therefore [MOD].[F#X].[OBJ].[FEAT].

An *instance* of a formulation is obtained by fixing the process network (PN#) and the horizon (H#). In order to assess the effectiveness of the proposed methods, five different networks found in literature are considered. PN1 is a modified version of the network presented in the seminal work of Kondili et al.[15], PN2 and PN4 are taken from Papageorgiou and Pantelides[67], PN3 is from Sundaramoorthy and Maravelias[92] and PN5 can be found in Maravelias and Papalamprou[46] . Figures and data for individual networks can be found in the supporting information. The time horizon values

we considered are relatively small in order to obtain optimal solutions for the majority of the instances. In particular, we use horizons of 12, 15, 18, 24 and 36 hours. Instances are labeled [MOD].[F#X].[OBJ].[FEAT].[PN#].[H#].

A *run* of a particular instance is an attempt to solve it, by iteratively increasing the number of time points N until its optimal value, N^*, has been determined, as it is customary in the literature on continuous-time models. We say that the optimal number of time points has been attained when the optimal value of the objective function does not change during three consecutive iterations and N^* is defined as the first of those three iterations. We keep track of the values at $(N^* + 1)$ and $(N^* + 2)$ to draw conclusions on the behavior of a given formulation. A run is labeled as [MOD].[F#X].[OBJ].[FEAT].[PN#].[H#].[N#]. A total of 1,170 runs are included for reformulations and 2,880 for tightening methods.

All the runs were solved using CPLEX 12.5 on GAMS 24.0 on a computer with a 2.8 GHz Intel Core i7-930 processor and 8 GB of RAM, running a 64-bit Windows 7 operating system. Default CPLEX settings were used unless specified otherwise. Time limits of 30 and 60 minutes were enforced for the tightening and reformulation runs respectively.

4.4.2. Results for reformulations in continuous-time models

Aggregate performance results

First we use performance charts in order to effectively summarize the vast amount of computational results we collected and draw general conclusions about the improvement introduced by the proposed reformulations. A general discussion on this type of charts is given in Appendix B. Since the main goal of the present work is to discuss how different objective functions benefit from particular reformulations, this section presents an analysis based on *problems*, as defined in the previous section. Figures 4.3-4.5 summarize the results for each of the three problems we study.

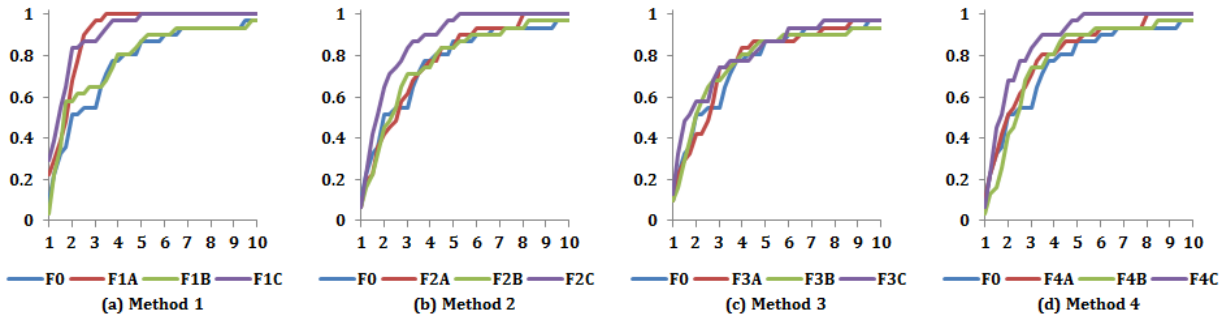


Figure 4.3. Aggregate performance charts for sales maximization including original formulation (F0) and proposed reformulations (F#X) for (a) Method 1, (b) Method 2, (c) Method 3, and (d) Method 4. [The vertical axis is the fraction of instances solved in less than the time for the fastest instance multiplied by a given value on the horizontal axis]

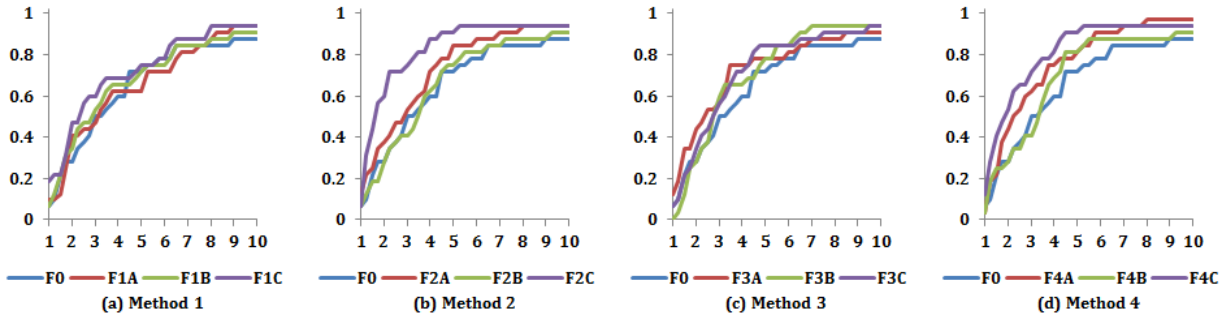


Figure 4.4. Aggregate performance charts for makespan minimization including original formulation (F0) and proposed reformulations (F#X) for (a) Method 1, (b) Method 2, (c) Method 3, and (d) Method 4. [The vertical axis is the fraction of instances solved in less than the time for the fastest instance multiplied by a given value on the horizontal axis]

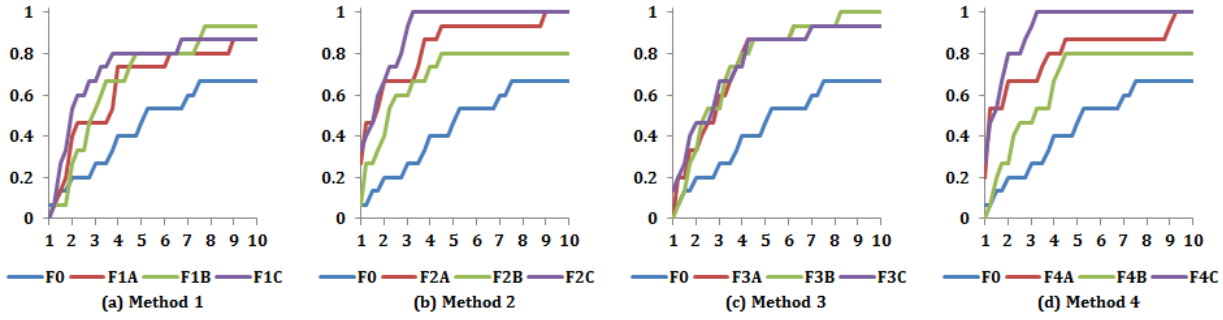


Figure 4.5. Aggregate performance charts for problem cost minimization including original formulation (F0) and proposed reformulations (F#X) for (a) Method 1, (b) Method 2, (c) Method 3, and (d) Method 4. [The vertical axis is the fraction of instances solved in less than the time for the fastest instance multiplied by a given value on the horizontal axis]

Figure 4.3 shows that method 1 outperforms the other three methods for sales maximization. In fact, 100% of the instances are solved to optimality in less than four times the fastest instance when formulation F1A is used. In contrast, the original formulation F0 only solves 74% of the instances in

the same time. In terms of variants, we observe that variants A and C perform similarly and much better than variant B, which is closer in performance to F0.

Figure 4.4 shows that for makespan minimization, methods 2 and 4 exhibit the best performance, solving more than 95% if the instances within five times the solution time for the fastest instance. Formulation F0 only solves less than 75% of the instances within the same time window. In terms of variants, it is clear that variant C exhibits the best performance.

Finally, Figure 4.5 clearly demonstrates that the improvement obtained for cost minimization is more dramatic, with methods 2 and 4, and variant C being superior in terms of number of instances solved and relative solution time. We observe that 100% of the instances are solved within three times the fastest instance when the reformulation is used, whereas only 27% are solved by F0 in the same time. It is interesting to note that F0 is able to solve only 67% of the instances within one order of magnitude of the lowest solution time.

Based on this basic analysis, we choose the following reformulations for further analysis in the following sections: F1C for sales maximization (SLS), F2C for cost minimization (CT), and F4C for makespan minimization (MS). Note that we use the same reformulation for all models although larger enhancements can be obtained if a reformulation is chosen for a combination of model and objective function. Also, in practice, users concerned with the scheduling of a given facility can perform extensive testing so they choose the best reformulation for a given model, objective function, and specific process network. This testing has to be performed only once.

Selected reformulations: solution time and optimality gap

Absolute computational times and optimality gaps are arguably the most commonly used factors in the comparative analysis of different reformulations. Accordingly, this section is devoted to present these results, using the general conclusions drawn from the previous section as guidelines to compare the original formulation against the best reformulation for each objective function.

We first focus on the improvements obtained in terms of computational time when the proposed reformulations are used. Figure 4.6(a) clearly reveals a significant reduction in average solution time for each of the objective functions when we use the best reformulation obtained from the previous section. It is important to note that these averages are calculated using only those instances solved to optimality within the assigned time limit, in order to avoid the artificial bias that would be introduced by adding the time limit value multiple times. A total of 92% of the 1,170 runs were solved to optimality. For those instances not solved to optimality, a separate analysis based on optimality gap is given later. Additional data on how many instances are solved to optimality for particular reformulations are included in the Supporting Information.

Next we analyze the impact of the reformulations on specific models. Figure 4.6(b) presents these results and once more we observe that there is a significant reduction in computational time for each of the models when we compare the original formulation and the best reformulation.

Finally, we discuss how our methods reduce the computational effort needed to identify the minimum number of points, N^*, necessary to represent the optimal solution; a task usually achieved by solving models with increasing number of periods until the optimal objective function value does not change in consecutive iterations. Therefore, we include in this analysis not only the runs with the optimal number of time points, but also runs with $(N^* + 1)$ and $(N^* + 2)$ points, which are required to empirically "confirm" that the necessary number of points has been reached. Figure 4.7 summarizes these findings based on objective function. We observe a consistent behavior across objective functions. For N^* points the reformulation is comparable to the original formulation in terms of computational time but the optimality gap was decreased to zero. Similarly, for $(N^* + 1)$ points all the runs were solved to optimality, but now we notice that the average computational time is reduced. Finally, for $(N^* + 2)$ points both the computational time and the optimality gap are significantly decreased. In particular, we observe that for sales maximization the average gap is cut in half, whereas for both makespan and cost minimization it is reduced by a factor of around 5.

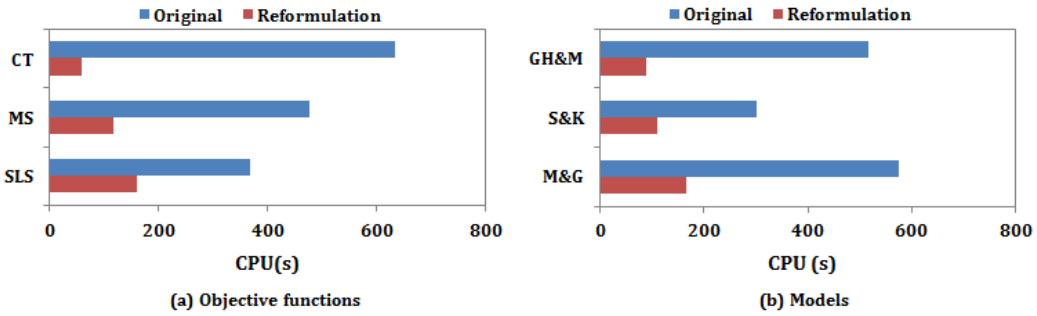


Figure 4.6. Average computational time for original formulation and best reformulation for (a) different objective functions, and (b) different models.

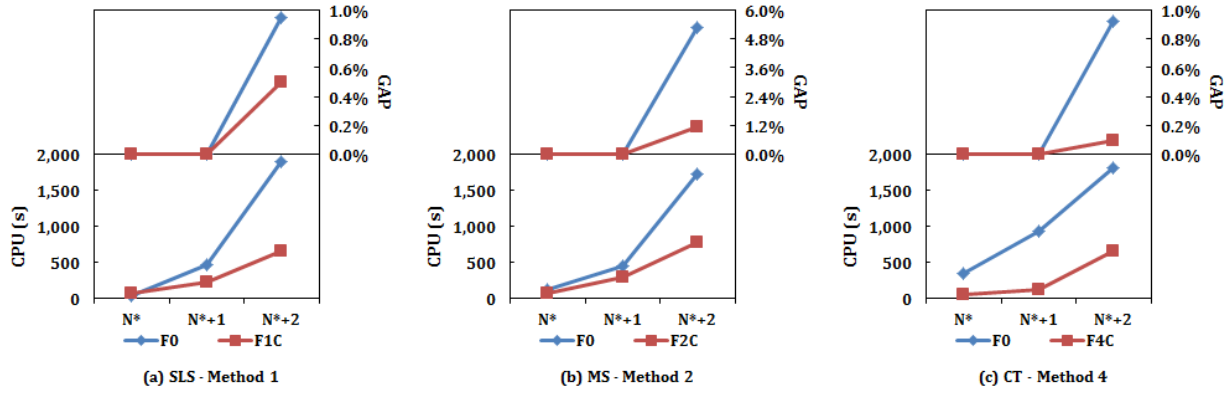


Figure 4.7. Average computational time and optimality gap for problems (a) Sales (SLS), method 1, (b) Makespan (MS), method 2, and (c) Cost (CT), method 4, including original formulation (F0) and proposed reformulations (FXC).

Improvement factors

An additional measure we can use to quantify the increase in computational efficiency derived from the introduction of our reformulations is the *improvement factor*, defined as the ratio between the computational times of the original formulation and the best reformulation. Once again, in order to avoid introducing an artificial bias derived from the time limit, we only consider those instances that were solved to optimality by both formulations in less than 3,600 CPU seconds. Moreover, we calculate this value for each instance and then take the average over all possible instances. Figure 4.8 presents values for this improvement factor (a) aggregated by objective function, (b) aggregated by model, and (c) for different combinations of objective function and model. We observe from figure 4.8(c) that depending on the combination it is possible to obtain improvement factors up to almost

20 (e.g. makespan minimization with model GH&M), which means that, on average, the proposed reformulation solves 20 times faster than the original formulation. We also notice that every objective and every model benefit from our reformulations.

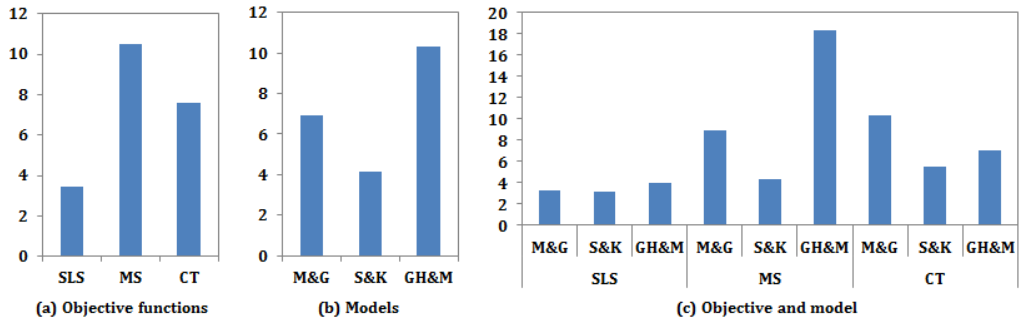


Figure 4.8. Computational improvement factors based on (a) objective function, (b) model, and (c) combinations of objective and model.

Additional examples

This section presents detailed results for particular instances of each problem and model we consider, in order to further illustrate the advantages of the proposed reformulations and gain additional insight into their behavior. Using the identification we introduced in section 4.4.1, we chose nine instances, one for each combination of objective function and model. The selection of process network and horizon value is random. For each instance we present a comparison of the computational time between the original formulation and the best reformulation as previously discussed. In the cases where either or both of them were not able to solve the instance to optimality, the optimality gap is used for the analysis. Table 4.1 presents the solution statistics for each instance with the comparison made for runs with $N^* + 2$ points, where N^* is the optimal number of time points. Additional statistics are included in the Supporting Information.

From table 4.1 we observe that only four of the nine instances were solved to optimality within the allotted time limit of one hour with the original formulation. In contrast, when the reformulation was introduced, only two instances were not solved within the time limit. In fact, the three additional instances that are solved only when the reformulation is used (1, 2, and 8), exhibit reduction in

computational time of one or two orders of magnitude. The remaining two instances that were not solved to optimality in one hour when the reformulation was introduced exhibit a significant decrease in the optimality gap. Moreover, when the time limit was increased to 2 hours, both instances were solved to optimality, while the original formulation was still unsolved (statistics not shown).

Table 4.1. Solution statistics for nine instances comparing original formulation (F0) and proposed reformulation (F#X) for $N^* + 2$ points.

		F0		F#X	
Instance	N^*	CPU (s)	Gap (%)	CPU (s)	Gap (%)
(1) M&G.F1C.SLS.PN2.H18	12	3,600.0	3.01	79.1	0
(2) S&K.F1C.SLS.PN1.H18	12	3,600.0	0.83	128.2	0
(3) GH&M.F1C.SLS.PN5.H24	9	1,155.3	0	242.8	0
(4) M&G.F4C.MS.PN1.H24	10	3,276.9	0	83.9	0
(5) S&K.F4C.MS.PN4.H24	10	2,450.4	0	1,913.0	0
(6) GH&M.F4C.MS.PN4.H24	10	3,600.0	21.15	3,600.0	13.33
(7) M&G.F2C.CT.PN3.H15	6	3,600.0	3.75	3,600.0	0.59
(8) S&K.F2C.CT.PN3.H15	6	3,600.0	0.59	30.84	0
(9) GH&M.F2C.CT.PN5.H24	10	29.4	0	1.79	0

We now analyze the behavior of the solution time for the selected instances as the number of time points increases. Figure 4.9 presents a comparison between the original formulation, F0, and the best reformulation for the particular instance, F#X, for N^*, $N^* + 1$, and $N^* + 2$ time points. As discussed before, the time required to solve the models with additional periods is substantial. We observe that the selected reformulation consistently outperforms the original formulation in the three cases. Using the best reformulation, only instances 6 and 7 reach the time limit with $N^* + 2$ time points, but the optimality gap is decreased as discussed before. Those instances are particularly hard to solve because of their sizes; instance 6 consists of a network with 19 tasks, 8 units and 27 materials, whereas instance 7 has 15 tasks, 8 units and 16 materials.

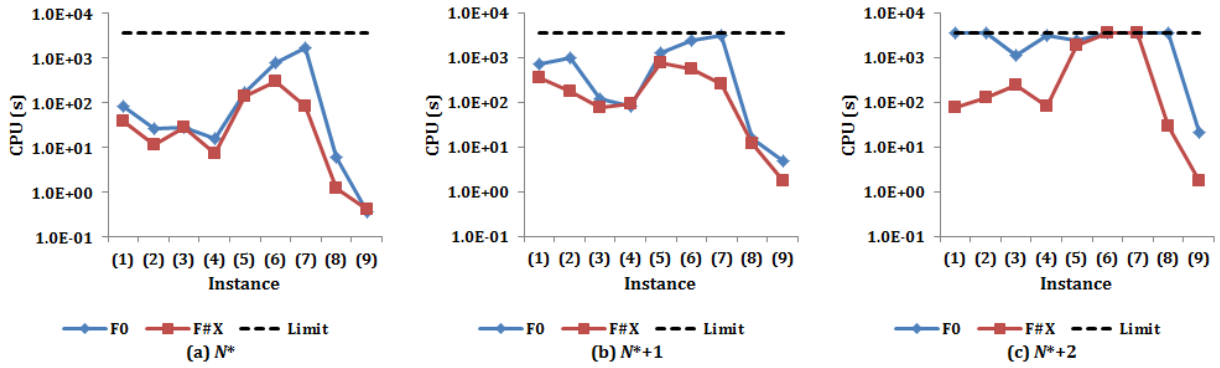


Figure 4.9. Computational time for nine instances and runs for (a) N^*, (b) $N^* + 1$, and (c) $N^* + 2$ time points, across different objective functions and models.

Finally we note that the size of the formulations is not greatly affected by the introduction of the proposed reformulations. For the nine instances presented in table 4.1, the average increase in the total number of variables is 9%, whereas the number of equations increases by only 4%. Individual results for each instance and additional statistics are available in the Supporting Information

4.4.3. Results for tightening constraints in continuous-time models

Cumulative results

In the interest of providing a general conclusion on the effectiveness of the proposed tightening methods, a performance assessment is made first by comparing the tightened formulations to the original formulations across all problems, models and instances. Figure 4.10 shows performance charts for each formulation, including the three variants and the original formulation as a reference. It is clear that the tightened formulations are much better than the original, which is only able to solve around 55% of the instances within one order of magnitude of the solution time for the fastest instance. Tightened formulation 1, which includes equations (4.19) and (4.21) is able to move this percentage up to 90%, whereas formulation 2, introducing equations (4.20) and (4.22), and formulation 3, including all equations (4.19)-(4.22), are able to increase it to 95%. This suggests that including constraints based on the minimum total amount a task is required to process and the total amount of material that needs to be produced is more effective than using constraints obtained from

simply bounding the number of batches for a single task and material. Additional testing (statistics not shown) showed that equation (4.20) is the most effective. In terms of variants, we notice that variant B, based on the finishing binary variable is less effective than the other two.

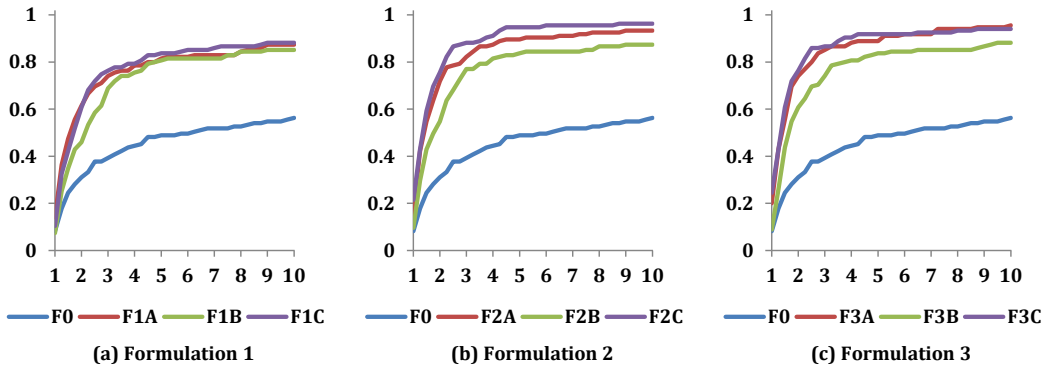


Figure 4.10. Performance charts for all four problems {MS.CPT, MS.VPT, CT.CPT, CT.VPT}.
Includes original formulation (F0) and proposed tightened formulations (F#X) based on (a) Minimum number of batches, (b) Minimum production requirements, (c) Combined minimum requirements. The vertical axis is the fraction of instances solved in less than the time for the fastest instance multiplied by a given value on the horizontal axis.

In order to discern the impact of the proposed tightening methods on each objective function, we separate the results based on problems. Figure 4.11 presents the performance results for cost minimization, including both features, constant and variable processing time. Tightened formulations solve an additional 70% of the instances compared to the original formulation, which only solves around 25% of the instances within one order of magnitude of the solution time for the fastest instance. The results for makespan (not shown) are completely different, showing less than 10% improvement in the number of instances solved. Figure 4 also confirms that the tightened formulations that include the minimum production requirements, equations (4.20) and (4.22), are more effective. In addition, no significant difference between formulations 2 and 3 is evident after this comparison.

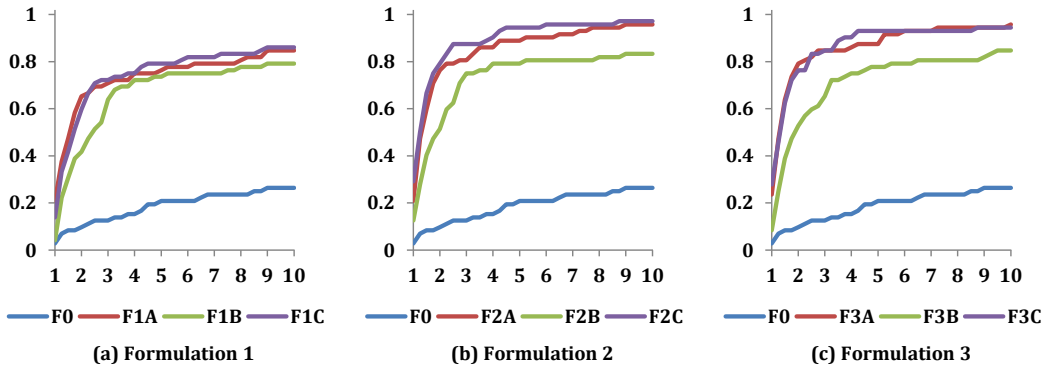


Figure 4.11. Performance charts for problems {CT.CPT, CT.VPT}.
Includes original formulation (F0) and proposed reformulations (F#X) tightened formulations (F#X) based on (a) Minimum number of batches, (b) Minimum production requirements, (c) Combined minimum requirements. The vertical axis is the fraction of instances solved in less than the time for the fastest instance multiplied by a given value on the horizontal axis.

Although not the primary goal of this work, we now compare the behavior of the models under study when the tightening methods are used. Figure 4.12 shows the performance profiles for the three models, before and after the valid inequalities with formulation 3 and variant A are added. All models benefit from the tightening constraints, but to different extents. Model GH&M exhibits the largest improvement with an additional 50% of the instances solved to optimality within one order of magnitude of the fastest instance, followed by model S&K with 40% and M&G with less than 20%. However, model S&K is the only one that solves all the instances in less than 4 times the solution time of the fastest instance. Interestingly model M&G is the best model for the original formulation but the worst after the tightening constraints are included

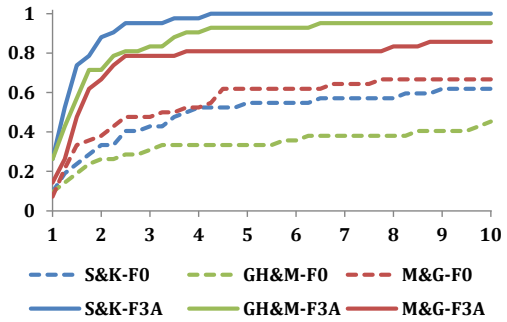


Figure 4.12. Performance charts for all problems with different models with formulations F0 and F3A.

The vertical axis is the fraction of instances solved in less than the time for the fastest instance multiplied by a given value on the horizontal axis.

Selected instances

To further illustrate the computational advantages of the proposed methods, we present detailed results for particular instances. Following the analysis presented in the previous subsection, we use formulation 3, variant A (F3A) for all the selected instances because it led to the best average results. We chose nine instances, which cover all models, networks, and horizon values. Using the identification introduced above, Table 4.2 summarizes the instances studied and their optimal number of points. Complete statistics for these instances are included in the Supporting Information. To better assess the computational enhancement, we do not use a resource limit of 1,800 CPU seconds.

Table 4.2. Instances selected for solution time analysis.

Number	Instance	N^*
(1)	GH&M.F3A.MS.VPT.PN2.H24	13
(2)	S&K.F3A.CT.CPT.PN1.H36	21
(3)	S&K.F3A.CT.VPT.PN1.H24	11
(4)	M&G.F3A.CT.CPT.PN4.H36	11
(5)	M&G.F3A.MS.CPT.PN4.H24	10
(6)	GH&M.F3A.MS.CPT.PN3.H24	7
(7)	M&G.F3A.CT.CPT.PN1.H24	15
(8)	GH&M.F3A.CT.VPT.PN2.H36	17
(9)	S&K.F3A.CT.VPT.PN3.H24	7

Figure 4.13 presents the results in terms of computational time for each instance for both the original and the tightened formulation. It confirms the general behavior derived from the performance charts, in which a clear improvement was introduced by using the valid inequalities derived from the proposed algorithm. We observe that the instances in which makespan is the objective function do not improve as much as the ones in which cost minimization is the objective. On the other hand, for instances where cost is minimized we observe significant improvements

ranging from 1 to 4 orders of magnitude. The behavior exhibited by Figure 4.13 is representative of the enhancements we observed in other instances.

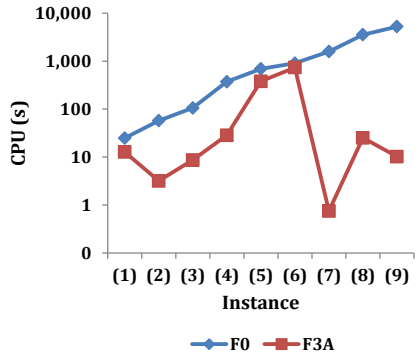


Figure 4.13. Computational time for nine instances run with optimal number of points.
Instances ranked in increasing order of difficulty for formulation F0.

4.5. Conclusions

4.5.1. Reformulation methods

We have presented reformulation strategies for material-based continuous-time models for the solution of chemical production scheduling problems, based on defining and bounding the total number of batches of a given task in a particular unit, as recently suggested by Velez and Maravelias[51] for discrete-time formulations. In particular, we presented four methods and three variants for each method. The introduction of these reformulations enhances the solution of the problem in terms of both computational time and optimality gap reduction for three objective functions commonly used: cost and makespan minimization and sales maximization.

We found that our reformulations are effective regardless of the objective function or model. Average improvement factors ranging from 3 to 20 were obtained for different combinations of objectives and models. In particular, the aggregate improvement factors obtained for each objective function can be used to rank them in ascending order of impact as follows (improvement factors are

given in parenthesis): (1) sales maximization (3.5), (2) cost minimization (7.6), and (3) makespan minimization (10.5).

Regarding the different models, we found that all of them benefit from our reformulations. With respect to the original formulation, model M&G requires the highest average computational times, followed by model GH&M and then model S&K. When the reformulations are introduced, model GH&M benefits the most with an aggregate improvement factor of 10.3, followed by model M&G (factor of 7.0) and model S&K (factor 4.1). Model S&K exhibited the most consistent behavior across all problems when combined with the appropriate method and either variant A or C. Model GH&M is significantly improved when combined with variant C. This fact can be explained if we consider that variant C enforces the equality of the sum of starting and finishing variables, a constraint which is not explicitly enforced in model GH&M unlike the other two models.

In terms of objective functions, we showed that sales maximization benefits the most when an integer variable is explicitly defined (method 1), whereas for cost and makespan minimization the disaggregation of the integer variable into binary variables (methods 2 and 4) is more effective. It is interesting to note that out of the three methods that introduce integer disaggregation, method 3 did not lead to any significant advantage, despite the fact that it uses fewer binary variables. A potential explanation is that the binary variables defined by method 2 do not infer any particular structure; they do not have a clear correspondence to elements of the problem. On the other hand, the binary variables used by method 2 define a special-ordered set of type 1 (SOS1) and those used by method 4 induce an independent branching scheme; the existence of these structured relationships between variables greatly benefits the solution algorithm[95]. In terms of variants, we showed that defining the integer variable for the number of batches of a given task using both the starting and the finishing binary variables (variant C) produces the best results.

Velez and Maravelias[51] showed that the reformulations of discrete-time models lead to dramatic computational improvements of up to three orders of magnitude and they are consistently effective across objective functions. Although our results are not as dramatic, we found that the introduction of the proposed reformulations enhances the performance of continuous-time models, if a careful selection of the reformulation (model/method/variant) is made. Average improvement factors of up to an order of magnitude are possible, with specific instances exhibiting even greater enhancements. Nevertheless, our reformulations inherit the limitations of continuous-time models, which lead to looser relaxations and require an additional initialization step to determine the optimal number of time points. This latter step can be time-consuming for medium- to large-scale instances.

Importantly, since practically all time-grid-based continuous-time models in the literature employ the two types of binary variables we considered in this study, our methods are applicable to all continuous-time models. Similarly, our methods can be applied to models that account for a wide range of additional characteristics, including variable processing times, utility and other shared resource constraints, different inventory policies and shared storage vessels, and changeover costs and times. Furthermore, similar if not greater enhancements are expected for any formulation employed routinely for the scheduling of a given facility. This is because the user can perform an extensive analysis to identify the best reformulation for the specific model, objective function, and facility. Finally, note that our reformulations are trivially applicable to models based on the Resource-Task Network (RTN) representation, as has been illustrated by Velez and Maravelias[51].

4.5.2. Tightening methods

We presented tightening methods for general material-based continuous-time models for the solution of chemical production scheduling problems. In particular, we extended the ideas presented by Velez et al.[59] for discrete-time formulations to take into account the demand information that is available as an input for the solution of the MIP model. Using the algorithm and valid inequalities

discussed in their work, we were able to achieve significant improvement for three representative continuous-time models found in the literature.

Specifically, we were able to show that cost minimization consistently benefits from the introduction of the tightening methods, in contrast to makespan minimization, for which only a few instances exhibited any improvement. Moreover, we also showed that constraints based on the minimum production amounts required by a particular task are more effective than the ones based on minimum number of batches. In addition, we determined that the variants do not introduce a significant difference, although it seems to be better to use those that include the starting binary variable.

The results obtained for continuous-time models are comparable to those reported by Velez et al.[59] for discrete-time formulations. The introduction of the proposed tightening methods led to order-of-magnitude improvements in computational time; in some instances up to four orders of magnitude were achieved. In both time frameworks, we found that cost minimization, which is probably the most widely used objective in practice for short-term scheduling, benefits the most from our methodologies. An important difference, however, is that makespan minimization was found to be easier than cost minimization in discrete-time models, so the method were most effective for the *harder* problem, whereas for continuous-time formulations cost minimization is easier than makespan minimization.

Since the calculation of the parameters included in the algorithm only requires information on network topology and customer demand and this information is available *a priori*, it is possible to extend our methodologies to include several common features in scheduling problems, such as utilities and shared resources, changeover information, shared storage and special policies, and non-instantaneous material transfers. Moreover, given that the formulation of the valid inequalities

themselves is based on a starting binary variable which is present in every time-grid-based model, we expect our methods to work with all continuous-time models found in literature.

It is important to note that the calculation of the parameters required to formulate the tightening constraints is done in very short times (less than 10 seconds) compared to the computational times for solving the MIP itself. Thus, it is guaranteed that the running time for the algorithm is negligible in the overall solution process. In addition, the number of additional constraints introduced by the algorithm is also very small compared with the number of equations defining the model. Therefore we expect our methods will almost always lead to net improvements in the solution process.

In summary, we showed that by using the demand propagation algorithm and defining appropriate valid inequalities, a great enhancement in the solution of chemical production scheduling problems is achieved. Our methods are applicable to all continuous-time models and can be readily used in problems with a broad range of processing characteristics and constraints.

4.6. Notation

Indices and sets

$i \in \mathbf{I}$	Tasks
$j \in \mathbf{J}$	Units
$k \in \mathbf{K}_{ij}$	Number of batches for task i in unit j
$l \in \mathbf{LK}_{ij}$	Logarithmic scale for the number of batches for task i in unit j
$m \in \mathbf{M}$	Materials
$n \in \mathbf{N}$	Time points
$\mathbf{I}_m^+/\mathbf{I}_m^-$	Tasks that produce/consume material m
$\mathbf{J}_i$	Units that can perform task i

$\mathbf{K}_{ij}^{+}(l,B)/\mathbf{K}_{ij}^{-}(l,B)$	Batches required for left/right branching in logarithmic approach
$\mathbf{K}^{MT}$	Materials that are produced by multiple tasks

Parameters

$\alpha_{ij}^{F}/\alpha_{ij}^{V}$	Fixed/variable processing cost for running task i in unit j
$\beta_j^{max}/\beta_j^{min}$	Maximum/minimum batch size for unit j
γ_m	Storage capacity for material m
γ_m^0	Initial inventory of material m ($\gamma_m^0 = S_{m0}$)
η	Horizon
κ_k	Lower bound on the number of batches of tasks producing material k
λ_i	Lower bound on the number of batches of task i
μ_i	Lower bound on the production of task i required to meet final demand
ξ_m	Demand of material m at the end of the horizon
π_m	Per-unit price for material m
ρ_{im}	Conversion coefficient of material m produced/consumed by task i
$\tau_{ij}^{F}/\tau_{ij}^{V}$	Fixed/variable processing time for task i in unit j
ω_m	Lower bound on the amount of material m required to meet final demand

Binary/integer variables

N_{ij}	Number of times task i runs in unit j
U_{ijn}	General binary variable to represent either X_{ijn} or Y_{ijn}

W_{ijk}	One if there are k batches of task i running in unit j
W^*_{ijl}	One if there are $\lfloor 2^l \rfloor$ batches of task i running in unit j
X_{ijn}/Y_{ijn}	One if task i starts/finishes in unit j at time n

Continuous variables

BS_{ijn}/BF_{ijn}	Batch size of task i that starts/finishes in unit j at time n
BP_{ijn}	Batch size of task i that continues to be processed in unit j at time n
S_{mn}	Inventory level of material m at point time n
T_n	Position of time point n

Chapter 5

Applications to large-scale instances

In this chapter we present results of the application of the methods we have derived to problems that have been regarded as of special interest due to either their use for benchmarking or their relevance in industrial applications. The goal of this chapter is to show that the methods we propose have a significant impact in reducing the computational burden of modeling and solving real-life problems with mathematical programming techniques. We also extend some of the concepts presented in previous chapter to account for additional features that were not taken into account in the original derivation of the solution methods we propose.

5.1. The Dow Problem

In this section we present the application of the methods proposed in Chapter 2 to an industrial-scale instance. We use a simplified version of the problem introduced by Nie et al.[98] and shown in Figure 5.1. The process consists of five main product lines, A-E, that go through six common steps. Step 1 is a batch process for which two identical units are available (UB1, UB2). Steps 2, 3, and 5 are buffer tanks (UT1, UT2, UT3) and steps 4 and 6 are continuous processes (UC1, UC2). An intermediate F is required for lines A-C and is produced by a single execution of the batch operation. In addition, the capacity of the second continuous operation is reduced with each execution; therefore a replenishing task must be run when the capacity falls below 25% of the maximum. Figure 5.2 is the STN representation of this process.

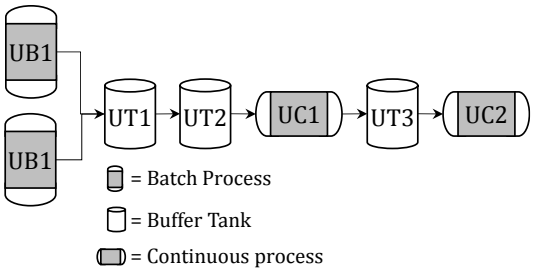


Figure 5.1. The Dow Process used as industrial-scale instance.

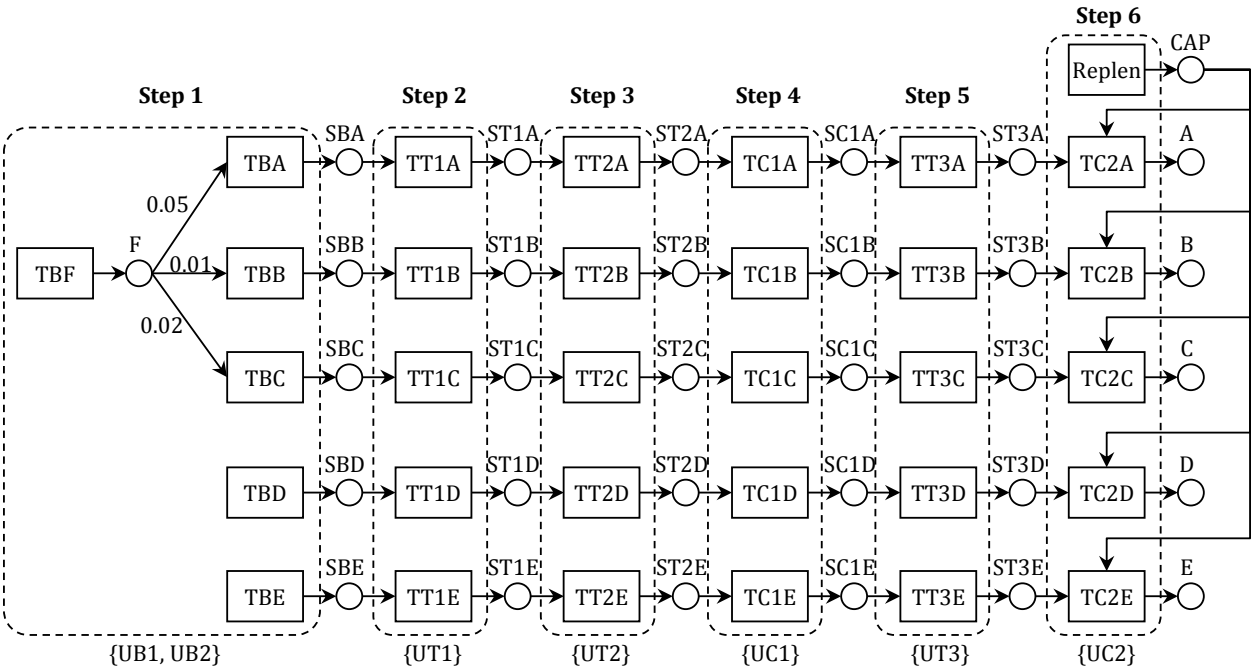


Figure 5.2. STN representation of the Dow Process.

We do not consider the changeovers included in the original reference. A scheduling horizon of five days (120 hours) is used with a scheduling period of 1 hour. We use the discrete-time model SP&S and formulation F1 to solve this problem. Table 5.1 provides a summary of the results and Figure 5.3 contains the Gantt chart of the optimal schedule. Note that, although the preprocessing time is significant, the effect on the MIP solution is to reduce the computational time by at least two orders of magnitude.

Table 5.1. Comparison of results of original formulation and best proposed formulation on the Dow Example.

	F0	**F1**
Constraints	17,828	17,968

Discrete variables	4,598	4,682
Total variables	13,109	13,193
Integer solution (*USD*)	15,339.79	15,339.79
LP relaxation solution (*USD*)	15,406.46	15,406.46
Preprocessing time (*s*)	-	678.91
MIP solution time (*s*)	3,600	31.24
Total CPU time (*s*)	3,600	710.15
Gap (%)	0.43	0.00

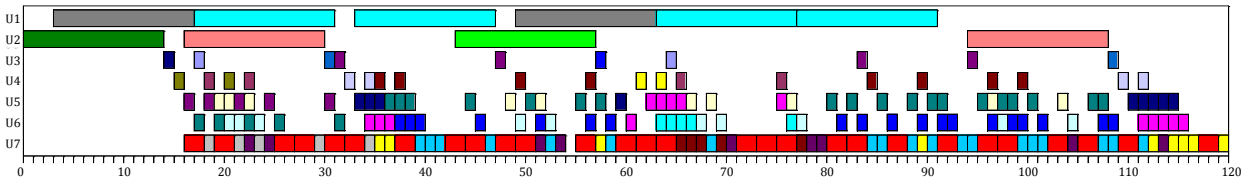


Figure 5.3. Gantt chart of optimal solution to the Dow example.

5.2. The Kallrath Example

The Kallrath network[7] (Figure 5.4) is a benchmark problem that has been used in the literature as a challenge problem to test the effectiveness of different formulations and solution methods. It consists of batch tasks and does not have storage in units, buffer tanks, or changeovers. A particular characteristic of this network is that one of its tasks (T2) has a variable conversion coefficient, where 20-70% of a batch is converted to S21 and the remainder is converted to S22. We consider profit maximization using the methods derived in Chapter 2.

In order to be able to solve this problem we first need to introduce some changes in the equations that define the original model SP&S, as well as the constraints derived from the application of the four algorithms described in section 2.2.

First, the unit capacity (2.2) and material balance (2.3) constraints are replaced by equations 5.1 and 5.2 respectively.

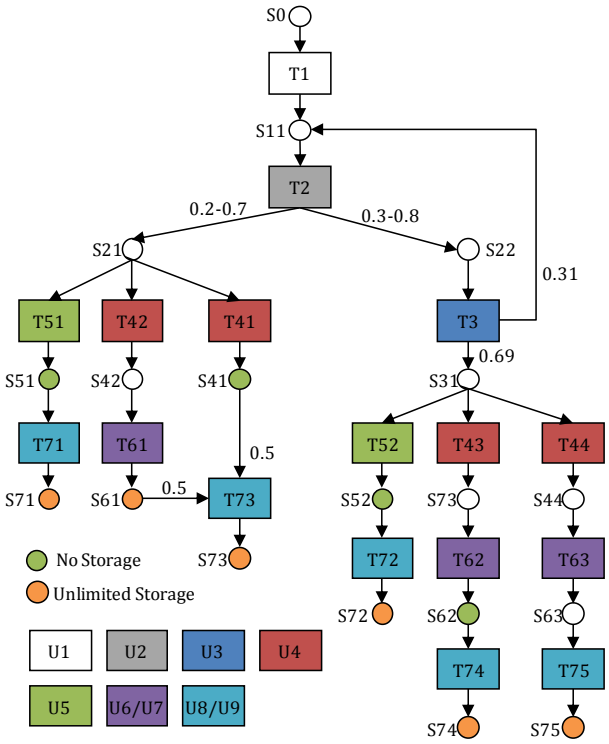


Figure 5.4. STN representation of the Kallrath example.

$$\rho_{ik}^{min} B_{ijt} \leq B_{ijkt}^{\mathrm{V}} \leq \rho_{ik}^{max} B_{ijt} \quad \forall i \in \mathbf{I}^{V}, k \in \mathbf{K}_i^{+} \tag{5.1}$$

$$S_{kt} = S_{k(t-1)} + \sum_{i \in \mathbf{I}_k^{+} \setminus \mathbf{I}^{V}} \rho_{ik} \sum_{j \in \mathbf{J}_i} B_{ij(t-\tau_{ij})} + \sum_{i \in \mathbf{I}_k^{+} \cap \mathbf{I}^{V}, j \in \mathbf{J}_i} B_{ijk(t-\tau_{ij})}^{\mathrm{V}} + \sum_{i \in \mathbf{I}_k^{-}} \rho_{ik} \sum_{j \in \mathbf{J}_i} B_{ijt} + \xi_{kt} \quad \forall k \in \mathbf{K}, t \in \mathbf{T} \tag{5.2}$$

A new set, $\mathbf{I}^{V}$, is introduced in equation (5.1) to denote the tasks that can produce materials with variable conversion coefficients. Moreover, bounds on the conversion coefficients are introduced with ρ_{ik}^{min} and ρ_{ik}^{max}. The new variable B_{ijkt}^{V} is the amount of material k that is produced by task i in unit j starting at time t and is only defined for $i \in \mathbf{I}^{V}$. Equation (5.3) provides the relation of this variable with the original batch size variable.

$$\sum_{k\in\mathbf{K}_i^+} B^{\mathrm{V}}_{ijkt} = B_{ijt} \quad \forall i \in \mathbf{I}^V \tag{5.3}$$

Next, a modification in the values of the tightening parameters is required to take into account the variability in the conversion coefficients. Equations (2.16) and (2.18) are respectively replaced with (5.4) and (5.5). Variable Q^V_{ik} represents a material-based disaggregation of the original variables Q_{ik} and is analogous in use to B^{V}_{ijkt}.

$$\mu_i^{LP} = \max \left\{ Q_i: \begin{array}{l} \xi_{k0} + \sum_{i\prime\in\mathbf{I}_k^+\setminus\mathbf{I}^V} \rho_{i\prime k} Q_{i\prime} + \sum_{i\prime\in\mathbf{I}_k^+\cap\mathbf{I}^V} Q^V_{i\prime k} + \sum_{i\prime\in\mathbf{I}_k^-} \rho_{i\prime k} Q_{i\prime} \geq 0 \ \ \forall k \\ \sum_{i\prime\in\mathbf{I}_k^+\setminus\mathbf{I}^V} \rho_{i\prime k} Q_{i\prime} + \sum_{i\prime\in\mathbf{I}_k^+\cap\mathbf{I}^V} Q^V_{i\prime k} \leq \omega_k \ \ \forall k \\ \sum_{k\in\mathbf{K}_{i\prime}^+} Q^V_{i\prime k} = Q_{i\prime} \ \ \forall i' \in \mathbf{I}^V \\ \rho^{min}_{i\prime k} Q_{i\prime} \leq Q^V_{i'k} \leq \rho^{max}_{i'k} Q_{i'} \ \ \forall i' \in \mathbf{I}^V, k \in \mathbf{K}_{i\prime}^+ \end{array} \right\} \tag{5.4}$$

$$\forall i \in \mathbf{I}: \omega_k \ is \ known \ \forall k \in \mathbf{K}_i^-$$

$$\omega_k = \xi_{k0} + \sum_{i\in\mathbf{I}_k^+} \rho^{max}_{ik} \mu_i^1 \quad \forall k \in \mathbf{K}: \mu_i \ is \ known \ \forall i \in \mathbf{I}_k^+ \tag{5.5}$$

The maximum number of batches for a given dependence is now calculated with equation (5.6) instead of (2.19).

$$\zeta^{max}_{id} = \min \left\{ \sum_{j\in\mathbf{J}_i} \left\lfloor \frac{\theta_{ij}}{\tau_{ij}} \right\rfloor, \max_{j\in\mathbf{J}_i} \left(\min_{k\in\mathbf{K}_i^-} \left\lfloor \frac{\omega_k}{\rho^{min}_{ik} \beta^{min}_j} \right\rfloor \right) \right\} \quad \forall d \in \mathbf{D}, i \in \mathbf{I}_d \tag{5.6}$$

Using these modifications we can now solve the Kallrath problem. Table 5.2 summarizes the results for the original Kallrath instance, denoted P0, and a horizon of 60 hours. The original formulation is able to find the optimal solution, but it fails to prove optimality after one hour. When the tightening and reformulations methods are added, optimality is proven in about 12 min. There are no available data in the literature to compare the results for profit maximization, since the

solution usually focuses on makespan minimization. Figure 5.5 shows the resulting Gantt chart. Additional data and model statistics can be found in the Supporting Information.

Table 5.2. Computational results for Kallrath example.

	Objective	CPU time (s)			Gap (%)
		Preproc.	MIP	Total	
Original	4599.9	0.00	3600	3600	<0.01
Tightened	4599.9	121.73	565.11	686.84	0.00

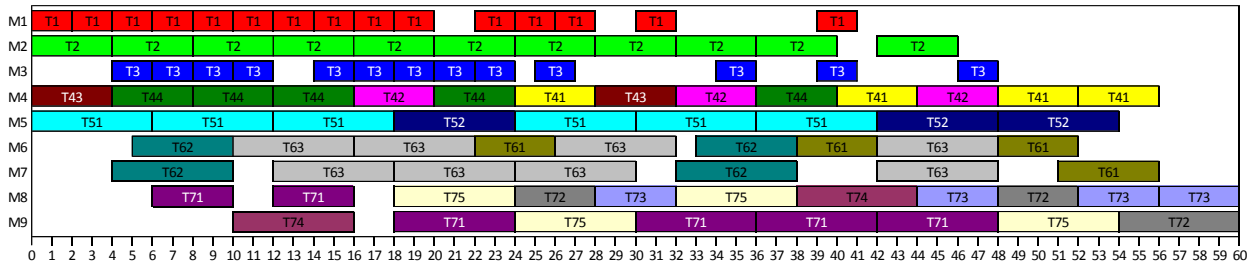


Figure 5.5. Gantt chart for the optimal solution of the Kallrath example with profit maximization.

5.3. Notation

Only new sets, parameters and variables are listed here. For a comprehensive list of symbols, please refer to 2.7.

Sets

$\mathbf{I}^V$ Tasks that produce materials with variable conversion coefficients

Parameters

$\rho_{ik}^{min}/\rho_{ik}^{max}$ Minimum/maximum conversion coefficients of task $i \in \mathbf{I}$ to material $k \in \mathbf{K}_i^+$

Variables

B_{ijkt}^{V} Amount of material $k \in \mathbf{K}_i^+$ produced by task $i \in \mathbf{I}^V$ in unit $j \in \mathbf{J}_i$ starting at time t

Q_{ik}^{V} Amount of material $k \in \mathbf{K}_i^+$ produced by task $i \in \mathbf{I}^V$

Chapter 6

Conclusions and Recommendations

6.1. Concluding Remarks

In this thesis we focused on developing mathematical models and methods that improve the solution process of scheduling problems in chemical production. Methods for Mixed-Integer Programming models for different types of environments (sequential and network) and time representation (discrete and continuous) were developed. With this thesis, the spectrum of scheduling problems that can be represented and solved using mathematical programming concepts is greatly expanded.

First, we developed new methods for the enhancement of the solution of MIP models for scheduling in network environments. These methods consist of four preprocessing algorithms based on instance-specific information (network structure, recipe, processing times, initial inventory, and unit capacities) that allow calculating parameters that are used to generate tightening constraints. Moreover, we presented a new family of constraints based on constraint propagation. The proposed methods are applicable to all material-based, time-indexed, MIP scheduling models, discrete- and continuous-time. We showed that the constraints derived from the sequence of preprocessing algorithms produce the best results for both discrete- and continuous-time models. In many cases, up to two orders of magnitude decrease in computational time was achieved when these constraints were applied to the discrete-time model.

Second, we presented a new approach to modeling and solving scheduling problems in sequential environments using discrete-time MIP models. It is important to note that existing methods rely almost exclusively on continuous-time models. We developed three new discrete-time models that were compared to the only existing discrete-time formulation for this problem, as well as the best

continuous-time model available. The first model is based on the STN representation of network facilities, whereas the other two are inspired on the formulations for the RCPSP problem. The major advantages of using this type of models as opposed to continuous-time counterparts are the simplicity to model processing features common to chemical facilities, as well as considerably better LP relaxations. In particular, we showed how modeling utility requirements and product diversification can be achieved with small changes in the proposed formulations. In terms of solution strategies, we introduced tightening constraints based on time windows available for both tasks and units. These windows can be fixed or variable. We showed that for fixed windows, only additional parameters are required. We also discussed how variable windows require a few new variables and constraints that do not have a significant impact in the model size. In both cases, we derived procedures to properly define and characterize the related parameters and variables.

Interestingly, the solution methods we developed for both network and sequential environments are largely based on the concepts of earliest start time and shortest tail. Although these quantities have been used in the scheduling literature in the past, they had never been studied as both fixed and variables quantities that can be used to define different types of tightening constraints. The novelty of our approaches is twofold: on one hand, we directly use the parameter version to limit different linear combinations of decision variables. On the other hand, we use concepts from constraint propagation and deeper insights about the problem nature to introduce a variable version of these quantities and further tighten the formulations.

Third, we developed specific reformulations and tightening constraints for material-based continuous-time models. Special modeling characteristics required in continuous-time formulations, such as time balancing and matching to grid points were taken into account. Moreover, recent developments in the area of specially ordered sets were brought into the scheduling problem. We observed that, although these methods are effective in improving the solution process for continuous-time models, their impact is less significant than their discrete-time counterparts. Our

methods inherit the original limitations of continuous-time models, which lead to looser relaxations and require an additional iterative step to determine the optimal number of time points. In terms of reformulations, average improvement factors of up to an order of magnitude are possible, with specific instances exhibiting even greater enhancements. Regarding demand-based tightening constraints for minimization problems, we observed that the introduction of these methods also led to order-of-magnitude improvements in computational time; in some instances up to four orders of magnitude were achieved. Given that both the reformulation and tightening constraints are based on linear combinations of an assignment binary variable which is present in every time-grid-based model, we expect our methods to work with all continuous-time models found in literature.

Finally, we were able to apply our methods to solve large-scale instances that are closer to industrial applications. Results are promising in terms of both computational time reduction and improvement of the solution quality. We expect that similar enhancements can be achieved in similar instances with industrial relevance.

6.2. Future Research Directions

Research and its quest for the expansion of knowledge is a never-ending activity that continuously reshapes and invigorates itself. With this thesis we were able to answer many questions and complete an important part of the spectrum of models and solution methods required to solve general problems in production scheduling. Throughout this process, interesting new questions and problems were discovered; some of them remain unanswered and constitute the basis for future studies.

For network environments we developed algorithms and valid inequalities that employed time and inventory restrictions, but did not consider cleaning and transition operations. A natural research direction would be to generalize the algorithms and constraints we developed to include information about these operations either for sequence-independent setups or sequence-dependent

changeovers. Another possible research direction would be exploring the tradeoff between computational performance and model tightness if some of the parameters in the algorithms are calculated using reduced versions of the original MIP problem. For instance, parameters that appear in constraints for variable time windows could be obtained this way and compared to the combinatorial procedures we proposed.

Based on the assumptions we made for derivation of models and methods in sequential environments, it is clear that adding common features to the problem can lead to new research projects. Three possible directions would be (1) considering simultaneous batching and scheduling, (2) studying restrictions on intermediate storage, and (3) including changeovers. A more fundamental generalization that should be explored is concerned with multipurpose batch plants, in which specific products have different routes, although each of them can be thought of as following a multistage path.

Appendices

A. Material-based continuous-time models

The equations that define each of the three continuous-time models for network environments considered in chapters 2 and 4 of this thesis are summarized next. The common variables are defined next.

T_n: Continuous nonnegative. Actual value of time point $n \in \mathbf{N}$

Y_{ijn}: Binary. It is equal to one if task $i \in \mathbf{I}$ finishes in unit $j \in \mathbf{J}_i$ at time $n \in \mathbf{N}$

BF_{ijn}/BP_{ijn}: Continuous nonnegative. Batch size of task $i \in \mathbf{I}$ that starts/continues to be processed in unit $j \in \mathbf{J}_i$ at time $n \in \mathbf{N}$

The following additional sets also appear in the formulations.

$\mathbf{I}_j = \{i \in \mathbf{I}$: task i can be processed in unit $j\}$

$\mathbf{I}^{ZW+} = \{i \in \mathbf{I}$: task i produces a material with zero–wait storage policy$\}$

$\mathbf{I}^{ZW-} = \{i \in \mathbf{I}$: task i consumes a material with zero–wait storage policy$\}$

A.1. Model M&G

Additional variables

TS_{ijn}/TF_{ijn} Continuous. Starting/finishing time of task $i \in \mathbf{I}$ in unit $j \in \mathbf{J}_i$ during interval $n \in \mathbf{N}\backslash\{N\}$

D_{ijn} Continuous. Duration of processing of task $i \in \mathbf{I}$ that starts in unit $j \in \mathbf{J}_i$ during interval $n \in \mathbf{N}\backslash\{N\}$

Assignment constraints

$$\sum_{i \in \mathbf{I}_j} X_{ijn} \leq 1 \qquad \forall j \in \mathbf{J}, \forall n \in \mathbf{N} \tag{A1}$$

$$\sum_{i \in \mathbf{I}_j} Y_{ijn} \leq 1 \qquad \forall j \in \mathbf{J}, \forall n \in \mathbf{N} \tag{A2}$$

$$\sum_{n \in \mathbf{N}} X_{ijn} = \sum_{n \in \mathbf{N}} Y_{ijn} \qquad \forall i \in \mathbf{I}, \forall j \in \mathbf{J}_i \tag{A3}$$

$$\sum_{i \in \mathbf{I}_j} \sum_{n' \leq n} (X_{ijn'} - Y_{ijn'}) \leq 1 \qquad \forall j \in \mathbf{J}, \forall n \in \mathbf{N} \tag{A4}$$

$$Y_{ij1} = 0 \qquad \forall i \in \mathbf{I}, \forall j \in \mathbf{J}_i \tag{A5}$$

$$X_{ijN} = 0 \qquad \forall i \in \mathbf{I}, \forall j \in \mathbf{J}_i \tag{A6}$$

Timing constraints

$$T_1 = 0 \tag{A7}$$

$$T_N = \eta \tag{A8}$$

$$T_{n+1} \geq T_n \qquad \forall n \in \mathbf{N} \backslash \{N\} \tag{A9}$$

$$D_{ijn} = \tau_{ij}^F X_{ijn} + \tau_{ij}^V BS_{ijn} \qquad \forall i \in \mathbf{I}, \forall j \in \mathbf{J}_i, \forall n \in \mathbf{N} \backslash \{N\} \tag{A10}$$

$$TF_{ijn} \leq TS_{ijn} + D_{ijn} + \eta\,(1 - X_{ijn}) \qquad \forall i \in \mathbf{I}, \forall j \in \mathbf{J}_i, \forall n \in \mathbf{N} \backslash \{N\} \tag{A11}$$

$$TF_{ijn} \geq TS_{ijn} + D_{ijn} - \eta\,(1 - X_{ijn}) \qquad \forall i \in \mathbf{I}, \forall j \in \mathbf{J}_i, \forall n \in \mathbf{N} \backslash \{N\} \tag{A12}$$

$$TF_{ijn} - TF_{ij(n-1)} \leq \eta\, X_{ijn} \qquad \forall i \in \mathbf{I}, \forall j \in \mathbf{J}_i, \forall n \in \mathbf{N} \backslash \{1, N\} \tag{A13}$$

$$TF_{ijn} - TF_{ij(n-1)} \geq D_{ijn} \qquad \forall i \in \mathbf{I}, \forall j \in \mathbf{J}_i, \forall n \in \mathbf{N}\backslash\{1, N\} \tag{A14}$$

$$TS_{ijn} = T_n \qquad \forall i \in \mathbf{I}, \forall j \in \mathbf{J}_i, \forall n \in \mathbf{N}\backslash\{N\} \tag{A15}$$

$$TF_{ij(n-1)} \leq T_n + \eta\,(1 - Y_{ijn}) \qquad \forall i \in \mathbf{I}, \forall j \in \mathbf{J}_i, \forall n \in \mathbf{N}\backslash\{1\} \tag{A16}$$

$$TF_{ij(n-1)} \geq T_n - \eta\,(1 - Y_{ijn}) \qquad \forall i \in \mathbf{I}^{ZW+}, \forall j \in \mathbf{J}_i, \forall n \in \mathbf{N}\backslash\{1\} \tag{A17}$$

Batch sizing constraints

$$\beta_j^{min} X_{ijn} \leq BS_{ijn} \leq \beta_j^{max} X_{ijn} \qquad \forall i \in \mathbf{I}, \forall j \in \mathbf{J}_i, \forall n \in \mathbf{N} \tag{A18}$$

$$\beta_j^{min} Y_{ijn} \leq BF_{ijn} \leq \beta_j^{max} Y_{ijn} \qquad \forall i \in \mathbf{I}, \forall j \in \mathbf{J}_i, \forall n \in \mathbf{N} \tag{A19}$$

$$\begin{aligned} \beta_j^{min}\left(\sum_{n'<n} X_{ijn'} - \sum_{n'\leq n} Y_{ijn'}\right) &\leq BP_{ijn} \\ &\leq \beta_j^{max}\left(\sum_{n'<n} X_{ijn'} - \sum_{n'\leq n} Y_{ijn'}\right) \qquad \forall i \in \mathbf{I}, \forall j \in \mathbf{J}_i, \forall n \in \mathbf{N}\backslash\{1, N\} \end{aligned} \tag{A20}$$

$$BS_{i(n-1)} + BP_{i(n-1)} = BP_{in} + BF_{in} \qquad \forall i \in \mathbf{I}, \forall j \in \mathbf{J}_i, \forall n \in \mathbf{N}\backslash\{1\} \tag{A21}$$

Material balance constraints

$$S_{mn} = S_{m(n-1)} + \sum_{i \in \mathbf{I}_m^+} \sum_{j \in \mathbf{J}_i} \rho_{im} BF_{ijn} + \sum_{i \in \mathbf{I}_m^-} \sum_{j \in \mathbf{J}_i} \rho_{im} BS_{ijn} \leq \gamma_m \qquad \forall m \in \mathbf{M}, \forall n \in \mathbf{N} \tag{A22}$$

Bounding constraints

$$X_{ijn}, Y_{ijn} \in \{0,1\} \qquad \forall i \in \mathbf{I}, \forall j \in \mathbf{J}_i, \forall n \in \mathbf{N} \tag{A23}$$

$$BS_{ijn}, BP_{ijn}, BF_{ijn} \in [0, +\infty) \qquad \forall i \in \mathbf{I}, \forall j \in \mathbf{J}_i, \forall n \in \mathbf{N} \tag{A24}$$

$$TS_{ijn}, TF_{ijn}, D_{ijn} \in [0, +\infty) \qquad \forall i \in \mathbf{I}, \forall j \in \mathbf{J}_i, \forall n \in \mathbf{N}\backslash\{N\} \tag{A25}$$

$$S_{mn} \in [0, +\infty) \qquad \forall m \in \mathbf{M}, \forall n \in \mathbf{N} \tag{A26}$$

$$T_n \in [0, +\infty) \qquad \forall n \in \mathbf{N} \tag{A27}$$

Tightening constraints

$$\sum_{i \in \mathbf{I}_j} \sum_{n<N} D_{ijn} \leq \eta \qquad \forall j \in \mathbf{J} \tag{A28}$$

$$\sum_{i \in \mathbf{I}_j} \sum_{n \leq n' < N} D_{ijn'} \leq \eta - T_n \qquad \forall j \in \mathbf{J}, \forall n \in \mathbf{N} \tag{A29}$$

$$\sum_{i \in \mathbf{I}_j} \sum_{n' \leq n} \left(\tau_{ij}^F Y_{ijn'} + \tau_{ij}^V B F_{ijn'} \right) \leq T_n \qquad \forall j \in \mathbf{J}, \forall n \in \mathbf{N} \tag{A30}$$

Makespan minimization

The horizon parameter η is substituted by the variable MS in equations (A8), (A28) and (A29), and it is kept as an upper bound for MS in equations (A11)-(A13), (A16) and (A17).

A.2. Model S&K

Additional variables

Z_{jn} Binary. It equals one if any task begins in unit $j \in \mathbf{J}$ at time $n \in \mathbf{N}$

YP_{ijn} Binary. It equals one if task $i \in \mathbf{I}$ continues to be processed in unit $j \in \mathbf{J}_i$ at time $n \in \mathbf{N}$

SL_n Duration of interval $n \in \mathbf{N}\backslash\{N\}$

TR_{jn} Continuous. Time remaining in unit $j \in \mathbf{J}$ to complete an ongoing task at time $n \in \mathbf{N}$

Time balances

$$SL_n = T_{n+1} - T_n \qquad \forall n \in \mathbf{N}\backslash\{N\} \tag{A31}$$

$$\sum_{n\in\mathbf{N}\backslash\{N\}} SL_n \leq \eta \tag{A32}$$

$$Z_{jn} = \sum_{i\in\mathbf{I}_j} X_{ijn} \qquad \forall j \in \mathbf{J}, \forall n \in \mathbf{N}\backslash\{N\} \tag{A33}$$

$$Z_{jn} = \sum_{i\in\mathbf{I}_j} Y_{ijn} \qquad \forall j \in \mathbf{J}, \forall n \in \mathbf{N}\backslash\{1, N\} \tag{A34}$$

$$YP_{ijn} = YP_{ij(n-1)} + X_{ij(n-1)} - Y_{ijn} \qquad \forall i \in \mathbf{I}, \forall j \in \mathbf{J}_i, \forall n \in \mathbf{N}\backslash\{1, N\} \tag{A35}$$

Processing time balances

$$TR_{j(n+1)} \geq TR_{jn} + \sum_{i\in\mathbf{I}_j} \left(\tau_{ij}^F X_{ijn} + \tau_{ij}^V BS_{ijn}\right) - SL_n \qquad \forall j \in \mathbf{J}, \forall n \in \mathbf{N}\backslash\{N\} \tag{A36}$$

Material balances in process units

$$BP_{ijn} = BP_{ij(n-1)} + BS_{ij(n-1)} - BF_{ijn} \qquad \forall i \in \mathbf{I}\backslash\{0\}, \forall j \in \mathbf{J}_i, \forall n \in \mathbf{N}\backslash\{1\} \tag{A37}$$

$$\beta_j^{min} X_{ijn} \leq BS_{ijn} \leq \beta_j^{max} X_{ijn} \qquad \forall i \in \mathbf{I}\backslash\{0\}, \forall j \in \mathbf{J}_i, \forall n \in \mathbf{N} \tag{A38}$$

$$\beta_j^{min} YP_{ijn} \leq BP_{ijn} \leq \beta_j^{max} YP_{ijn} \qquad \forall i \in \mathbf{I}\backslash\{0\}, \forall j \in \mathbf{J}_i, \forall n \in \mathbf{N} \tag{A39}$$

$$\beta_j^{min} Y_{ijn} \leq BF_{ijn} \leq \beta_j^{max} Y_{ijn} \qquad \forall i \in \mathbf{I}\backslash\{0\}, \forall j \in \mathbf{J}_i, \forall n \in \mathbf{N} \tag{A40}$$

$$TR_{jn} \leq \sum_{i\in\mathbf{I}_j} \left(\tau_{ij}^F YP_{ijn} + \tau_{ij}^V BP_{ijn}\right) \qquad \forall j \in \mathbf{J}, \forall n \in \mathbf{N} \tag{A41}$$

Material balances in storage vessels

$$S_{mn} = S_{m(n-1)} + \sum_{i \in \mathbf{I}_m^+} \sum_{j \in \mathbf{J}_i} \rho_{im} BF_{ijn} + \sum_{i \in \mathbf{I}_m^-} \sum_{j \in \mathbf{J}_i} \rho_{im} BS_{ijn} \leq \gamma_m \qquad \forall m \in \mathbf{M}, \forall n \in \mathbf{N} \tag{A42}$$

Bounding/fixing constraints

$$SL_n \leq \max_{j \in \mathbf{J}} \left[\max_{i \in \mathbf{I}_j} \left(\tau_{ij}^F + \tau_{ij}^V \beta_j^{max} \right) \right] \qquad \forall n \in \mathbf{N} \backslash \{N\} \tag{A43}$$

$$TR_{jn} \leq \max_{i \in \mathbf{I}_j} \left(\tau_{ij}^F + \tau_{ij}^V \beta_j^{max} \right) \qquad \forall j \in \mathbf{J}, \forall n \in \mathbf{N} \tag{A44}$$

$$X_{ijN} = 0 \qquad \forall i \in \mathbf{I}, \forall j \in \mathbf{J}_i \tag{A45}$$

$$Y_{ij1} = 0 \qquad \forall i \in \mathbf{I}, \forall j \in \mathbf{J}_i \tag{A46}$$

$$YP_{ijn} = 0 \qquad \forall i \in \mathbf{I}, \forall j \in \mathbf{J}_i, n \in \{1, N\} \tag{A47}$$

$$Z_{jN} = 1 \qquad \forall j \in \mathbf{J} \tag{A48}$$

$$X_{ijn}, Y_{ijn}, YP_{ijn} \in \{0,1\} \qquad \forall i \in \mathbf{I}, \forall j \in \mathbf{J}_i, \forall n \in \mathbf{N} \tag{A49}$$

$$Z_{jn} \in \{0,1\} \qquad \forall j \in \mathbf{J}, \forall n \in \mathbf{N} \tag{A50}$$

$$BS_{ijn}, BP_{ijn}, BF_{ijn} \in [0, +\infty) \qquad \forall i \in \mathbf{I} \backslash \{0\}, \forall j \in \mathbf{J}_i, \forall n \in \mathbf{N} \tag{A51}$$

$$S_{mn} \in [0, +\infty) \qquad \forall m \in \mathbf{M}, \forall n \in \mathbf{N} \tag{A52}$$

$$TR_{jn} \in [0, +\infty) \qquad \forall j \in \mathbf{J}, \forall n \in \mathbf{N} \tag{A53}$$

$$T_n, SL_n \in [0, +\infty) \qquad \forall n \in \mathbf{N} \tag{A54}$$

Makespan minimization

Equation (A55) is enforced in order to define the makespan and replaces equation (A32)

$$MS = \sum_{n<N} SL_n \tag{A55}$$

A.3. Model GH&M

Additional variables

E_{jn}/V_{jn}	Binary. It equals one if unit $j \in \mathbf{J}$ is at execution/idle state during interval $n \in \mathbf{N}\backslash\{N\}$
Z_{jn}	Binary. It equals one if at time $n \in \mathbf{N}$, unit $j \in \mathbf{J}$ continues executing a task that started at a previous time point
T_{jn}^{LB}/T_{jn}^{EE}	Continuous. Amount of time the beginning/end of a task is delayed/anticipated in unit $j \in \mathbf{J}$ with respect to its formal beginning/end at time $n \in \mathbf{N}$
T_{jn}^{V}	Continuous. Time spent by unit $j \in \mathbf{J}$ in the idle state during interval $n \in \mathbf{N}\backslash\{N\}$

Unit state constraints

$$E_{jn} + V_{jn} = 1 \qquad \forall j \in \mathbf{J}, \forall n \in \mathbf{N}\backslash\{N\} \tag{A56}$$

$$E_{jn} = Z_{jn} + \sum_{i \in \mathbf{I}_j} X_{ijn} \qquad \forall j \in \mathbf{J}, \forall n \in \mathbf{N}\backslash\{N\} \tag{A57}$$

$$Z_{jn} = Z_{j(n-1)} + \sum_{i \in \mathbf{I}_j} X_{ij(n-1)} \qquad \forall j \in \mathbf{J}, \forall n \in \mathbf{N}\backslash\{1\} \tag{A58}$$

Slack time and idle state constraints

$$T_{jn}^{LB} \leq \eta \sum_{i \in \mathbf{I}_j \backslash \mathbf{I}^{ZW-}} X_{ijn} \qquad \forall j \in \mathbf{J}, \forall n \in \mathbf{N}\backslash\{N\} \tag{A59}$$

$$T_{jn}^{EE} \leq \eta \sum_{i \in \mathbf{I}_j \setminus \mathbf{I}^{ZW+}} Y_{ijn} \qquad \forall j \in \mathbf{J}, \forall n \in \mathbf{N} \setminus \{1\} \tag{A60}$$

$$T_{jn}^{V} \leq \eta V_{jn} \qquad \forall j \in \mathbf{J}, \forall n \in \mathbf{N} \setminus \{N\} \tag{A61}$$

$$T_{n+1} - T_n - \eta (1 - V_{jn}) \leq T_{jn}^{V} \leq T_{n+1} - T_n \qquad \forall j \in \mathbf{J}, \forall n \in \mathbf{N} \setminus \{N\} \tag{A62}$$

Time balance constraints

$$T_n \geq \sum_{1 < n' \leq n} \left[T_{jn'}^{EE} + \sum_{i \in \mathbf{I}_j} \left(\tau_{ij}^{F} Y_{ijn'} + \tau_{ij}^{V} BF_{ijn'} \right) \right] + \sum_{n' < n} \left(T_{jn'}^{LB} + T_{jn'}^{V} \right) \quad \forall j \in \mathbf{J}, \forall n \in \mathbf{N} \setminus \{N\} \tag{A63}$$

$$\eta - T_n \geq \sum_{n \leq n' < N} \left[T_{jn'}^{LB} + T_{jn'}^{V} + \sum_{i \in \mathbf{I}_j} \left(\tau_{ij}^{F} X_{ijn'} + \tau_{ij}^{V} BS_{ijn'} \right) \right] + \sum_{n' > n} T_{jn'}^{EE} \quad \forall j \in \mathbf{J}, \forall n \in \mathbf{N} \setminus \{N\} \tag{A64}$$

$$\eta = \sum_{n > 1} \left[T_{jn}^{EE} + \sum_{i \in \mathbf{I}_j} \left(\tau_{ij}^{F} Y_{ijn} + \tau_{ij}^{V} BF_{ijn} \right) \right] + \sum_{n < N} \left(T_{jn}^{LB} + T_{jn}^{V} \right) \qquad \forall j \in \mathbf{J} \tag{A65}$$

Batch sizing constraints

$$\beta_j^{min} X_{ijn} \leq BS_{ijn} \leq \beta_j^{max} X_{ijn} \qquad \forall i \in \mathbf{I}, \forall j \in \mathbf{J}_i, \forall n \in \mathbf{N} \tag{A66}$$

$$\beta_j^{min} Y_{ijn} \leq BF_{ijn} \leq \beta_j^{max} Y_{ijn} \qquad \forall i \in \mathbf{I}, \forall j \in \mathbf{J}_i, \forall n \in \mathbf{N} \tag{A67}$$

$$\beta_j^{min} Z_{jn} \leq \sum_{i \in \mathbf{I}_j} BP_{ijn} \leq \beta_j^{max} Z_{jn} \qquad \forall j \in \mathbf{J}, \forall n \in \mathbf{N} \tag{A68}$$

$$BS_{ijn} + BP_{ijn} = BP_{ij(n+1)} + BF_{ij(n+1)} \qquad \forall i \in \mathbf{I}, \forall j \in \mathbf{J}_i, \forall n \in \mathbf{N} \setminus \{N\} \tag{A69}$$

Material balance constraints

$$S_{mn} = S_{m(n-1)} + \sum_{i \in \mathbf{I}_m^+} \sum_{j \in \mathbf{J}_i} \rho_{im} BF_{ijn} + \sum_{i \in \mathbf{I}_m^-} \sum_{j \in \mathbf{J}_i} \rho_{im} BS_{ijn} \leq \gamma_m \qquad \forall m \in \mathbf{M}, \forall n \in \mathbf{N} \tag{A70}$$

Bounding/fixing constraints

$$X_{ijN} = 0 \qquad \forall i \in \mathbf{I}, \forall j \in \mathbf{J}_i \tag{A71}$$

$$Y_{ij1} = 0 \qquad \forall i \in \mathbf{I}, \forall j \in \mathbf{J}_i \tag{A72}$$

$$Z_{jn} = 0 \qquad \forall j \in \mathbf{J}, n \in \{1, N\} \tag{A73}$$

$$X_{ijn}, Y_{ijn} \in \{0,1\} \qquad \forall i \in \mathbf{I}, \forall j \in \mathbf{J}_i, \forall n \in \mathbf{N} \tag{A74}$$

$$Z_{jn} \in \{0,1\} \qquad \forall j \in \mathbf{J}, \forall n \in \mathbf{N} \tag{A75}$$

$$E_{jn}, V_{jn} \in \{0,1\} \qquad \forall j \in \mathbf{J}, \forall n \in \mathbf{N} \backslash \{N\} \tag{A76}$$

$$BS_{ijn}, BP_{ijn}, BF_{ijn} \in [0, +\infty) \qquad \forall i \in \mathbf{I}, \forall j \in \mathbf{J}_i, \forall n \in \mathbf{N} \tag{A77}$$

$$S_{mn} \in [0, +\infty) \qquad \forall m \in \mathbf{M}, \forall n \in \mathbf{N} \tag{A78}$$

$$T_{jn}^{LB}, T_{jn}^{EE} \in [0, +\infty) \qquad \forall j \in \mathbf{J}, \forall n \in \mathbf{N} \tag{A79}$$

$$T_{jn}^{V} \in [0, +\infty) \qquad \forall j \in \mathbf{J}, \forall n \in \mathbf{N} \backslash \{N\} \tag{A80}$$

$$T_n \in [0, +\infty) \qquad \forall n \in \mathbf{N} \tag{A81}$$

Makespan minimization

The horizon η is replaced by MS in equations (A64) and (A65) but is still used as an upper bound in equations (A59)-(A62).

B. Performance charts

A performance chart is a type of Cartesian plot in which the abscissa corresponds to the ratio between the solution times of a particular instance of a formulation/model and the fastest formulation/model, whereas the ordinate defines the fraction of instances that were solved within a given ratio of solution times. This type of plot was originally introduced to compare different solvers in computational studies[99]. Figure B1 shows a general performance chart comparing two hypothetical models.

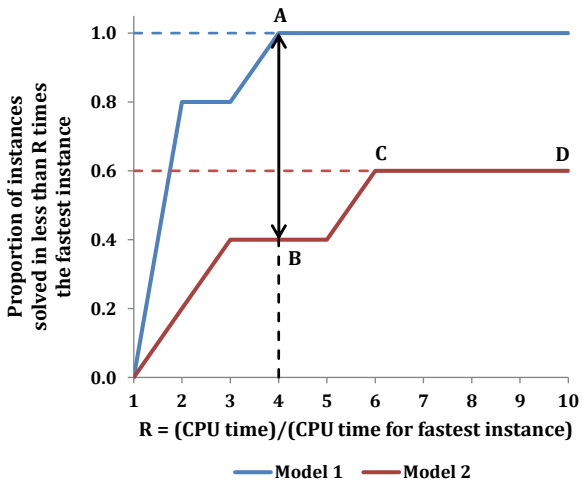


Figure B1. General performance chart.

Figure B1 reveals many interesting facts regarding the comparative behavior of models 1 and 2. Point A (with coordinates (4, 1.0)) shows that for model 1, 100% of the instances were solved within four times the CPU time of the fastest instance. Similarly, point C (6, 0.6) shows that 60% of the instances solved with model 2 required less than six times the CPU time for the fastest instance. Moreover, point D (10, 0.6) reveals that for model 2, only 60% of the instances were solved within one order of magnitude of the fastest solution time. In addition, since at point A, model 1 has solved all the instances, the arrow AB illustrates the difference between models 1 and 2 in terms of performance: 60% more instances are effectively solved using model 1 within four times the fastest solution time. Figure A1 shows that model 1 is better than model 2 in terms of performance. In

general, we expect a model whose performance curve is close to one in the ordinate and shifted towards the left top corner to be better than a model whose performance curve lies *underneath* it.

Bibliography

1. Pinedo, M. L., *Scheduling: theory, algorithms, and systems*. Springer Science & Business Media: 2012.
2. Salveson, M. E., On a Quantitative Method in Production Planning and Scheduling. *Econometrica* **1952,** 20, (4), 554-590.
3. Kelley, J. E.; Walker, M. R., Critical-path planning and scheduling. In *Proceedings of the Eastern Joint Computer Conference*, ACM: Boston, Massachusetts, 1959; pp 160-173.
4. Malcolm, D. G.; Roseboom, J. H.; Clark, C. E.; Fazar, W., Application of a Technique for Research and Development Program-Evaluation. *Operations Research* **1959,** 7, (5), 646-669.
5. Reklaitis, G. V., Overview of scheduling and planning of batch process operations. In *Batch processing systems engineering*, Springer: 1996; pp 660-705.
6. Pinto, J. M.; Grossmann, I. E., Assignment and sequencing models for the scheduling of process systems. *Annals of Operations Research* **1998,** 81, 433-466.
7. Kallrath, J., Planning and scheduling in the process industry. *Or Spectrum* **2002,** 24, (3), 219-250.
8. Floudas, C. A.; Lin, X. X., Continuous-time versus discrete-time approaches for scheduling of chemical processes: a review. *Computers & Chemical Engineering* **2004,** 28, (11), 2109-2129.
9. Mendez, C. A.; Cerda, J.; Grossmann, I. E.; Harjunkoski, I.; Fahl, M., State-of-the-art review of optimization methods for short-term scheduling of batch processes. *Computers & Chemical Engineering* **2006,** 30, (6-7), 913-946.
10. Maravelias, C. T., General framework and modeling approach classification for chemical production scheduling. *Aiche Journal* **2012,** 58, (6), 1812-1828.
11. Harjunkoski, I.; Maravelias, C. T.; Bongers, P.; Castro, P. M.; Engell, S.; Grossmann, I. E.; Hooker, J.; Mendez, C.; Sand, G.; Wassick, J., Scope for industrial applications of production scheduling models and solution methods. *Computers & Chemical Engineering* **2014,** 62, 161-193.
12. Mauderli, A.; Rippin, D. W. T., Production Planning and Scheduling for Multipurpose Batch Chemical Plants. *Computers & Chemical Engineering* **1979,** 3, (1-4), 199-206.
13. Reklaitis, G. In *Review of scheduling of process operations*, AIChE Symposium Series, 1982; 1982; pp 119-133.
14. Egli, U. M.; Rippin, D. W. T., Short-Term Scheduling for Multiproduct Batch Chemical-Plants. *Computers & Chemical Engineering* **1986,** 10, (4), 303-325.
15. Kondili, E.; Pantelides, C. C.; Sargent, R. W. H., A General Algorithm for Short-Term Scheduling of Batch-Operations – I. MILP Formulation. *Computers & Chemical Engineering* **1993,** 17, (2), 211-227.
16. Shah, N.; Pantelides, C. C.; Sargent, R. W. H., A General Algorithm for Short-Term Scheduling of Batch-Operations – II. Computational Issues. *Computers & Chemical Engineering* **1993,** 17, (2), 229-244.
17. Pantelides, C. C. In *Unified frameworks for optimal process planning and scheduling*, Proceedings on the second conference on foundations of computer aided operations, 1994; Cache Publications New York: 1994; pp 253-274.

18. Schilling, G.; Pantelides, C. C., A simple continuous-time process scheduling formulation and a novel solution algorithm. *Computers & Chemical Engineering* **1996,** 20, S1221-S1226.

19. Shah, N. In *Single and Multisite Planning and Scheduling: Current Status and Future Challenges*, AIChE Symposium Series, 1998; New York, NY: American Institute of Chemical Engineers, 1971-c2002.: 1998; pp 75-90.

20. Mendez, C. A.; Henning, G. P.; Cerda, J., Optimal scheduling of batch plants satisfying multiple product orders with different due-dates. *Computers & Chemical Engineering* **2000,** 24, (9-10), 2223-2245.

21. Mendez, C. A.; Henning, G. P.; Cerda, J., An MILP continuous-time approach to short-term scheduling of resource-constrained multistage flowshop batch facilities. *Computers & Chemical Engineering* **2001,** 25, (4-6), 701-711.

22. Harjunkoski, I.; Grossmann, I. E., Decomposition techniques for multistage scheduling problems using mixed-integer and constraint programming methods. *Computers & Chemical Engineering* **2002,** 26, (11), 1533-1552.

23. Gupta, S.; Karimi, I. A., An improved MILP formulation for scheduling multiproduct, multistage batch plants. *Industrial & Engineering Chemistry Research* **2003,** 42, (11), 2365-2380.

24. Pinto, J. M.; Grossmann, I. E., A Continuous-Time Mixed-Integer Linear-Programming Model for Short-Term Scheduling ff Multistage Batch Plants. *Industrial & Engineering Chemistry Research* **1995,** 34, (9), 3037-3051.

25. Liu, Y.; Karimi, I. A., Scheduling multistage, multiproduct batch plants with nonidentical parallel units and unlimited intermediate storage. *Chemical Engineering Science* **2007,** 62, (6), 1549-1566.

26. Castro, P. M.; Grossmann, I. E., New continuous-time MILP model for the short-term scheduling of multistage batch plants. *Industrial & Engineering Chemistry Research* **2005,** 44, (24), 9175-9190.

27. Zhang, X.; Sargent, R. W. H., The optimal operation of mixed production facilities - A general formulation and some approaches for the solution. *Computers & Chemical Engineering* **1996,** 20, (6-7), 897-904.

28. Giannelos, N. F.; Georgiadis, M. C., A novel event-driven formulation for short-term scheduling of multipurpose continuous processes. *Industrial & Engineering Chemistry Research* **2002,** 41, (10), 2431-2439.

29. Mockus, L.; Reklaitis, G. V., Continuous time representation approach to batch and continuous process scheduling. 1. MINLP formulation. *Industrial & Engineering Chemistry Research* **1999,** 38, (1), 197-203.

30. Maravelias, C. T.; Grossmann, I. E., New general continuous-time state-task network formulation for short-term scheduling of multipurpose batch plants. *Industrial & Engineering Chemistry Research* **2003,** 42, (13), 3056-3074.

31. Sundaramoorthy, A.; Karimi, I. A., A simpler better slot-based continuous-time formulation for short-term scheduling in multipurpose batch plants. *Chemical Engineering Science* **2005,** 60, (10), 2679-2702.

32. Gimenez, D. M.; Henning, G. P.; Maravelias, C. T., A novel network-based continuous-time representation for process scheduling: Part I. Main concepts and mathematical formulation. *Computers & Chemical Engineering* **2009,** 33, (9), 1511-1528.

33. Ierapetritou, M. G.; Floudas, C. A., Effective continuous-time formulation for short-term scheduling. 1. Multipurpose batch processes. *Industrial & Engineering Chemistry Research* **1998,** 37, (11), 4341-4359.

34. Janak, S. L.; Lin, X. X.; Floudas, C. A., Enhanced continuous-time unit-specific event-based formulation for short-term scheduling of multipurpose batch processes: Resource constraints and mixed storage policies. *Industrial & Engineering Chemistry Research* **2004,** 43, (10), 2516-2533.

35. Janak, S. L.; Floudas, C. A., Improving unit-specific event based continuous-time approaches for batch processes: Integrality gap and task splitting. *Computers & Chemical Engineering* **2008,** 32, (4-5), 913-955.

36. Susarla, N.; Li, J.; Karimi, I. A., A Novel Approach to Scheduling Multipurpose Batch Plants Using Unit-Slots. *Aiche Journal* **2010,** 56, (7), 1859-1879.

37. Ku, H. M.; Karimi, I. A., Scheduling in Serial Multiproduct Batch Processes with Finite Interstage Storage - A Mixed Integer Linear Program Formulation. *Industrial & Engineering Chemistry Research* **1988,** 27, (10), 1840-1848.

38. Mendez, C. A.; Cerda, J., An MILP Continuous-Time Framework for Short-Term Scheduling of Multipurpose Batch Processes Under Different Operation Strategies. *Optimization and Engineering* **2003,** 4, (1-2), 7-22.

39. Prasad, P.; Maravelias, C. T., Batch selection, assignment and sequencing in multi-stage multi-product processes. *Computers & Chemical Engineering* **2008,** 32, (6), 1106-1119.

40. Sundaramoorthy, A.; Maravelias, C. T., Simultaneous batching and scheduling in multistage multiproduct processes. *Industrial & Engineering Chemistry Research* **2008,** 47, (5), 1546-1555.

41. Castro, P. M.; Erdirik-Dogan, M.; Grossmann, I. E., Simultaneous batching and scheduling of single stage batch plants with parallel units. *Aiche Journal* **2008,** 54, (1), 183-193.

42. Sundaramoorthy, A.; Maravelias, C. T., Modeling of Storage in Batching and Scheduling of Multistage Processes. *Industrial & Engineering Chemistry Research* **2008,** 47, (17), 6648-6660.

43. Sundaramoorthy, A.; Maravelias, C. T.; Prasad, P., Scheduling of Multistage Batch Processes under Utility Constraints. *Industrial & Engineering Chemistry Research* **2009,** 48, (13), 6050-6058.

44. Baumann, P.; Trautmann, N., A continuous-time MILP model for short-term scheduling of make-and-pack production processes. *International Journal of Production Research* **2013,** 51, (6), 1707-1727.

45. Pochet, Y.; Warichet, F., A tighter continuous time formulation for the cyclic scheduling of a mixed plant. *Computers & Chemical Engineering* **2008,** 32, (11), 2723-2744.

46. Maravelias, C. T.; Papalamprou, K., Polyhedral results for discrete-time production planning MIP formulations for continuous processes. *Computers & Chemical Engineering* **2009,** 33, (11), 1890-1904.

47. Maravelias, C. T., On the combinatorial structure of discrete-time MIP formulations for chemical production scheduling. *Computers & Chemical Engineering* **2012,** 38, 204-212.

48. Bassett, M. H.; Pekny, J. F.; Reklaitis, G. V., Decomposition techniques for the solution of large-scale scheduling problems. *Aiche Journal* **1996,** 42, (12), 3373-3387.

49. Kelly, J. D.; Zyngier, D., Hierarchical decomposition heuristic for scheduling: Coordinated reasoning for decentralized and distributed decision-making problems. *Computers & Chemical Engineering* **2008,** 32, (11), 2684-2705.

50. Burkard, R. E.; Hatzl, J., Review, extensions and computational comparison of MILP formulations for scheduling of batch processes. *Computers & Chemical Engineering* **2005,** 29, (8), 1752-1769.

51. Velez, S.; Maravelias, C. T., Reformulations and Branching Methods for Mixed-Integer Programming Chemical Production Scheduling Models. *Industrial & Engineering Chemistry Research* **2013,** 52, (10), 3832-3841.

52. Velez, S.; Merchan, A. F.; Maravelias, C. T., On the solution of large-scale mixed integer programming scheduling models. *Chemical Engineering Science* **2015,** 136, 139-157.

53. Subrahmanyam, S.; Kudva, G. K.; Bassett, H. H.; Pekny, J. F., Application of distributed computing to batch plant design and scheduling. *Aiche Journal* **1996,** 42, (6), 1648-1661.

54. Ferris, M. C.; Maravelias, C. T.; Sundaramoorthy, A., Simultaneous Batching and Scheduling Using Dynamic Decomposition on a Grid. *Informs Journal on Computing* **2009,** 21, (3), 398-410.

55. Velez, S.; Maravelias, C. T., A branch-and-bound algorithm for the solution of chemical production scheduling MIP models using parallel computing. *Computers & Chemical Engineering* **2013,** 55, 28-39.

56. Jain, V.; Grossmann, I. E., Algorithms for hybrid MILP/CP models for a class of optimization problems. *Informs Journal on Computing* **2001,** 13, (4), 258-276.

57. Roe, B.; Papageorgiou, L. G.; Shah, N., A hybrid MILP/CLP algorithm for multipurpose batch process scheduling. *Computers & Chemical Engineering* **2005,** 29, (6), 1277-1291.

58. Maravelias, C. T., A decomposition framework for the scheduling of single- and multi-stage processes. *Computers & Chemical Engineering* **2006,** 30, (3), 407-420.

59. Velez, S.; Sundaramoorthy, A.; Maravelias, C. T., Valid Inequalities Based on Demand Propagation for Chemical Production Scheduling MIP Models. *Aiche Journal* **2013,** 59, (3), 872-887.

60. Velez, S.; Maravelias, C. T., Mixed-Integer Programming Model and Tightening Methods for Scheduling in General Chemical Production Environments. *Industrial & Engineering Chemistry Research* **2013,** 52, (9), 3407-3423.

61. Castro, P. M.; Hariunkoski, I.; Grossmann, I. E., Optimal Short-Term Scheduling of Large-Scale Multistage Batch Plants. *Industrial & Engineering Chemistry Research* **2009,** 48, (24), 11002-11016.

62. Maravelias, C. T.; Grossmann, I. E., Minimization of the makespan with a discrete-time state-task network formulation. *Industrial & Engineering Chemistry Research* **2003,** 42, (24), 6252-6257.

63. Merchan, A. F.; Maravelias, C. T., Preprocessing and Tightening Methods for Time-Indexed MIP Chemical Production Scheduling Models. *Computers & Chemical Engineering* **2015,** doi: 10.1016/jcompchemeng.2015.10.003.

64. Nemhauser, G. L.; Wolsey, L. A., Integer and Combinatorial Optimization. Interscience Series in Discrete Mathematics and Optimization. *ed: John Wiley & Sons* **1988**.

65. He, Y. H.; Hui, C. W., Rule-evolutionary approach for single-stage multiproduct scheduling with parallel units. *Industrial & Engineering Chemistry Research* **2006,** 45, (13), 4679-4692.

66. Dijkstra, E. W., A note on two problems in connexion with graphs. *Numerische mathematik* **1959,** 1, (1), 269-271.

67. Barber, C. B.; Dobkin, D. P.; Huhdanpaa, H., The Quickhull algorithm for convex hulls. *Acm Transactions on Mathematical Software* **1996,** 22, (4), 469-483.

68. Papageorgiou, L. G.; Pantelides, C. C., Optimal campaign planning scheduling of multipurpose batch semicontinuous plants .2. A mathematical decomposition approach. *Industrial & Engineering Chemistry Research* **1996,** 35, (2), 510-529.

69. Merchan, A. F.; Lee, H.; Maravelias, C. T., Discrete-Time Mixed-Integer Programming Models for Production Scheduling in Multistage Facilities. *Computers and Chemical Engineering* **2015,** Submitted.

70. Sundaramoorthy, A.; Maravelias, C. T., Computational Study of Network-Based Mixed-Integer Programming Approaches for Chemical Production Scheduling. *Industrial & Engineering Chemistry Research* **2011,** 50, (9), 5023-5040.

71. Graves, S. C., A Review of Production Scheduling. *Operations Research* **1981,** 29, (4), 646-675.

72. Maccarthy, B. L.; Liu, J. Y., Addressing the Gap in Scheduling Research - A Review of Optimization and Heuristic Methods in Production Scheduling. *International Journal of Production Research* **1993,** 31, (1), 59-79.

73. Wiers, V. C. S., A review of the applicability of OR and AI scheduling techniques in practice. *Omega-International Journal of Management Science* **1997,** 25, (2), 145-153.

74. Pinedo, M., *Planning and scheduling in manufacturing and services*. Springer: 2005; Vol. 24.

75. Linn, R.; Zhang, W., Hybrid flow shop scheduling: A survey. *Computers & Industrial Engineering* **1999,** 37, (1-2), 57-61.

76. Wang, H., Flexible flow shop scheduling: optimum, heuristics and artificial intelligence solutions. *Expert Systems* **2005,** 22, (2), 78-85.

77. Ruiz, R.; Vazquez-Rodriguez, J. A., The hybrid flow shop scheduling problem. *European Journal of Operational Research* **2010,** 205, (1), 1-18.

78. Brah, S. A. Scheduling in a Flow Shop with Multiple Processors. PhD thesis, University of Houston, 1988.

79. Brucker, P.; Drexl, A.; Mohring, R.; Neumann, K.; Pesch, E., Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research* **1999,** 112, (1), 3-41.

80. Demeulemeester, E. L.; Herroelen, W., *Project scheduling: a research handbook*. Springer Science & Business Media: 2002; Vol. 102.

81. Hartmann, S.; Briskorn, D., A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research* **2010,** 207, (1), 1-14.

82. Schmidt, C. W.; Grossmann, I. E., Optimization models for the scheduling of testing tasks in new product development. *Industrial & Engineering Chemistry Research* **1996,** 35, (10), 3498-3510.

83. Jain, V.; Grossmann, I. E., Resource-constrained scheduling of tests in new product development. *Industrial & Engineering Chemistry Research* **1999,** 38, (8), 3013-3026.

84. Papageorgiou, L. G.; Rotstein, G. E.; Shah, N., Strategic supply chain optimization for the pharmaceutical industries. *Industrial & Engineering Chemistry Research* **2001,** 40, (1), 275-286.

85. Colvin, M.; Maravelias, C. T., R&D pipeline management: Task interdependencies and risk management. *European Journal of Operational Research* **2011,** 215, (3), 616-628.

86. Kyriakidis, T. S.; Kopanos, G. M.; Georgiadis, M. C., MILP formulations for single- and multi-mode resource-constrained project scheduling problems. *Computers & Chemical Engineering* **2012,** 36, 369-385.

87. Kopanos, G. M.; Kyriakidis, T. S.; Georgiadis, M. C., New continuous-time and discrete-time mathematical formulations for resource-constrained project scheduling problems. *Computers & Chemical Engineering* **2014,** 68, 96-106.

88. Talbot, F. B., Resource-Constrained Project Scheduling with Time-Resource Tradeoffs – The Non-Preemptive Case. *Management Science* **1982,** 28, (10), 1197-1210.

89. Weglarz, J.; Jozefowska, J.; Mika, M.; Waligora, G., Project scheduling with finite or infinite number of activity processing modes - A survey. *European Journal of Operational Research* **2011,** 208, (3), 177-205.

90. Zapata, J. C.; Hodge, B. M.; Reklaitis, G. V., The multimode resource constrained multiproject scheduling problem: Alternative formulations. *Aiche Journal* **2008,** 54, (8), 2101-2119.

91. Pritsker, A. A. B.; Waiters, L. J.; Wolfe, P. M., Multiproject Scheduling with Limited Resources: A Zero-One Programming Approach. *Management Science* **1969,** 16, (1), 93-108.

92. Christofides, N.; Alvarez-Valdes, R.; Tamarit, J. M., Project Scheduling with Resource Constraints – A Branch and Bound Approach. *European Journal of Operational Research* **1987,** 29, (3), 262-273.

93. Castro, P. M.; Grossmann, I. E., An efficient MILP model for the short-term scheduling of single stage batch plants. *Computers & Chemical Engineering* **2006,** 30, (6-7), 1003-1018.

94. Sundaramoorthy, A.; Maravelias, C. T., A General Framework for Process Scheduling. *Aiche Journal* **2011,** 57, (3), 695-710.

95. Gimenez, D. M.; Henning, G. P.; Maravelias, C. T., A novel network-based continuous-time representation for process scheduling: Part II. General framework. *Computers & Chemical Engineering* **2009,** 33, (10), 1644-1660.

96. Watters, L. J., Reduction of Integer Polynomial Programming Problems to Zero-One Linear Programming. *Operations Research* **1967,** 15, (6), 1171-&.

97. Vielma, J. P.; Nemhauser, G. L., Modeling disjunctive constraints with a logarithmic number of binary variables and constraints. *Mathematical Programming* **2011,** 128, (1-2), 49-72.

98. Nie, Y. S.; Biegler, L. T.; Villa, C. M.; Wassick, J. M., Reactor Modeling and Recipe Optimization of Ring-Opening Polymerization: Block Copolymers. *Industrial & Engineering Chemistry Research* **2014,** 53, (18), 7434-7446.

99. Dolan, E. D.; More, J. J., Benchmarking optimization software with performance profiles. *Mathematical Programming* **2002,** 91, (2), 201-213.