Advances on mathematical modelling and optimization framework for process scheduling

A thesis submitted to The University of Manchester for the degree

of Doctor of Philosophy

in the Faculty of Science and Engineering

2020

Nikolaos Rakovitis

School of Engineering

Department of Chemical Engineering and Analytical Science

Blank page

Contents

Word count: 83690

Blank page

Abstract

Chemical production scheduling is responsible for providing the allocation, sequencing and timings of operations into units to produce several valuable products. As a result, optimal scheduling is crucial for the vitality and prosperity of the chemical industry as it directly affects its productivity and its operational costs. Although many mathematical models have been developed in the past three decades, most models either lead to large model sizes and intractable computational time or generate suboptimal solutions in some cases. Additionally, mathematical models for scheduling of multipurpose batch plants do not allow related production and consumption tasks in different units to start or/and end at the same time points, which is different from the models for scheduling of semi-continuous/continuous and multistage multiproduct batch plants. Therefore, there is no generic and efficient framework for chemical production scheduling problems.

In this Thesis, a generic and efficient modelling framework is proposed using the unit-specific event-based time representation. The main features of this framework include (a) defining all timing variables based on units instead of tasks, (b) allowing related non-recycling production and consumption tasks to take place at the same event-point where a new definition for recycling tasks is presented, (c) sequencing different units processing related production and consumption tasks only if there is an indirect material transfer (i.e. there are not enough materials in the storage for consuming tasks), (d) aligning different units processing related tasks only if there is a direct material transfer (i.e. there is not enough storage for producing materials), (e) allowing processing units to hold materials for multiple event points. It is demonstrated that the proposed framework outperforms existing approaches in both solution quality and computational expenses. For large-scale problems, which require significantly high computational time, an enhanced rolling-horizon decomposition approach is developed in which a grouping strategy using the mixed-integer programming is proposed to divide the entire problem into subproblems. It is shown that the enhanced decomposition approach can generate optimal or near-optimal solutions in significantly less computational time. Finally, a hybrid solution approach through a combination of gene expression programming with the mathematical programming approach is explored to solve large-scale energy-efficient flexible job-shop scheduling problems. The results demonstrate that the hybrid approach can significantly improve the solution quality.

Blank page

Declaration

No portion of the work referred to in the thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Nikolaos Rakovitis

Blank page

Copyright statement

Blank page

Acknowledgements

First of all, I would like to thank my supervisor Dr Jie Li for his valuable guidance as well as his important and on-to-the-point suggestions regarding my work during the whole period of my PhD studies. Thank you for the limitless hours that you spend with me, despite your limited time, to teach me the concept of scheduling, to advise me of how to conduct and improve this work and to provide me with a necessary boost in difficult times. I truly believe that I would not be able to develop such work, without your guidance. I would also like to thank my co-supervisor, Dr Nan Zhang for the useful meetings I had with him, especially during the beginning of my PhD studies that helped me proceed with my studies. Finally, I would like to thank Prof Liping Zhang from Wuhan University, with whom I collaborated during her academic visit to The University of Manchester in 2019. I would like to thank you for advising me and for providing me with the necessary data to successfully conduct part of this work.

My studies in Manchester wouldn't be so easy without the help I had from my old friends Panagiotis Petsagkourakis and Anna Maria Tsakiroglou, which helped me a lot to have a smooth settlement in Manchester. Thank you for helping me feel much more relaxed, especially after a hard-working week. I would also like to thank all my colleagues from the Centre for Process Integration. During my studies in Manchester, I made a lot of new friends from and outside CPI, which I hope to see again any time soon. A big thanks to the "late lunch" crew, Fernando Hector Almeida Trasviña, Gonzalo Mauricio Figueroa and Jose Loyola Fuentes and of course my "office neighbour" Julia Jimenez Romero, with whom we always had so nice conversations during our lunch break. A special thanks to Yinjie Ma, Dan Li, Xi Cheng, Francesca Wang, Mohamed Al Jamri, Isabel Pazmino Mayorga, Zekun Yang, Merve Ceylan, Allesandro Usai, Rahma Mutia, Kexin Xu, Minerva Martinez Ledesma, Giannis Zacharopoulos and Rahul Kadam.

I would also like to thank my parents Panagiotis and Afroditi and my brother Dimitris for their unlimited support and love they offered me during my studies even though they were so far away.

Finally, I would like to thank my partner Danai who was always next to me even in all good and bad times. There are not enough words to express my gratitude and love to you. Without you, nothing would be the same.

Blank page

Chapter 1: Introduction

It is undeniable that the industrial revolution has drastically affected the everyday life of most people around the globe during the past two centuries. Processing facilities construct most of the tools and products that an average household uses every day. Furthermore, the process industry has significantly advanced the health industry, transportation and communications, which further improved the everyday life of numerous people. Only the chemical industry is responsible for producing more than 70,000 different products in the USA (SelectUSA, 2020). As a result, the process industry significantly affects the global economy. More specifically, only in the UK, the process industry has contributed £200 billion in 2016, which is approximately 15% of the total UK economic output (office for national statistics, 2016) However, the highly competitive market makes challenging for an individual process facility to withstand. Therefore, a facility needs to produce one or more valuable products at the minimum possible cost. Apart from surviving to such a competitive market, there is also one more factor that leads facilities in the same direction; their environmental footprint. By optimizing their processes, facilities can minimize the use of raw material resources as well as to reduce their energy needs which leads to less fuel consumption and as a result fewer gas emission without affecting their productivity. Furthermore, facilities receive several orders every day and therefore, they require the proper managerial tools that not only help them to optimize their process but also to provide a quick solution to manage to meet their due dates.

1.1 Classification of process industry

In general, there are different types of processes that a facility can process to produce a product. Therefore, the process industry can be classified into two main categories; batch and continuous process industry. A facility can perform batch processes, continuous processes or both, based on the type and the quality of the product that the facility produces.

1.1.1 Batch process industry

Facilities that perform batch processes produce one or more products by processing several raw materials. More specifically, the raw materials enter a batch vessel at the beginning of the process, which converts them into final products after a specified time. At the end of the processing time, the final products exit the batch vessel. It is suitable for processing facilities to use batch processes if they require to produce multiple products with different specifications in small quantities. Furthermore, it is preferable to use a batch process in cases where there is a high risk of contamination. Batch process industry can be further classified into the multipurpose, multistage and multitasking batch process industry.

1.1.1.1 Multipurpose batch process industry

In the multipurpose batch process industry, a batch can split into one or more parts. In this case, two different processing units can process each fraction. Additionally, mixing two or more streams is also possible in such a facility. Therefore, a multipurpose batch process facility can produce a final product by mixing several raw materials. Such facilities also use several recycling streams mainly to increase product yields. Commonly a multipurpose batch process facility produces more than one final products and each product follows a different processing path. In other words, the number and the type of processes that each raw material follows to produce a final product may differ. Finally, a processing unit can process multiple tasks from different stages. For instance, a reactor may be suitable for two reactions within the same facility.

Figure 1 depicts all different features of a multipurpose batch process facility. For instance, raw materials 1 and 2 can mix into the same batch B1. Furthermore, the intermediate product of batch B2 (material 4) can split into two different parts (batches B3 and B4). Figure 1 also depicts a case of a recycling stream. In this case, batch B3 produces intermediate product 3 together with final product 5, which is separated and mixed with the rest intermediate product 3 (Batch B2).

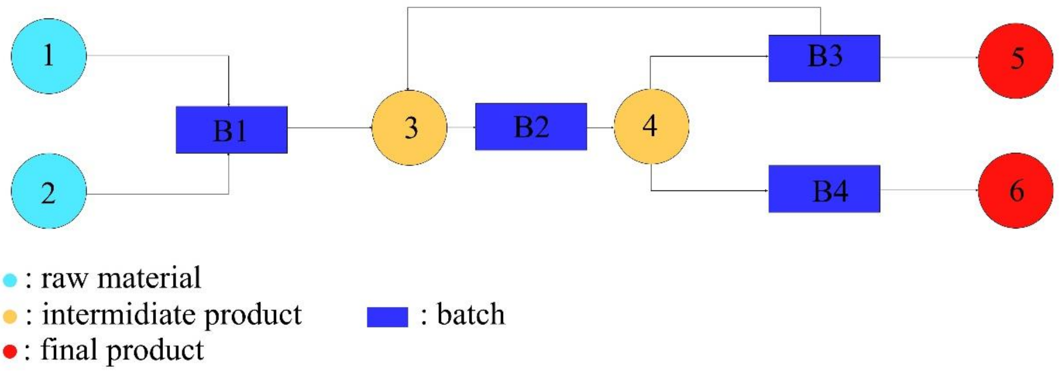

Figure 1 A multipurpose batch process facility

A specific case of a multipurpose batch process industry is the job-shop industry. In the job-shop facility, only one out of the several available processing units (machines) can process a specific operation in a job-shop facility. There are also cases, though where two or more processing units that are suitable for the same operation/task within a facility. Such facilities are commonly known as flexible job-shop facilities. Both job-shop and flexible job-shop facilities can process multiple jobs, which consist of one or more operations. The main difference between common multipurpose and job-shop/flexible job-shop facilities is that the batch size is not involved in this problem. More specifically, in such facilities, a processing unit receives one or more parts of an object/tool and performs several modifications to an object or assembly the different parts.

1.1.1.2 Multistage batch process industry

In multi-stage batch processes industry, processing units process several batches, which consist of one or more raw materials, mixed before the start of the whole process, in several predefined stages. The processing stages (processing path) are the same for all batches, while no splitting or mixing is allowed during the whole process. Furthermore, a multi-stage batch process facility does not contain any recycling stream. In each stage, one or more processing units are available to process each batch. However, each processing unit can only process operations of this stage, while it can only process one operation at a time, similar to multipurpose batch processes. A specific case of multi-stage batch processes is the single-stage batch process, where there is only one stage in the processing path. Figure 2 depicts a general single and multi-stage batch process facility.

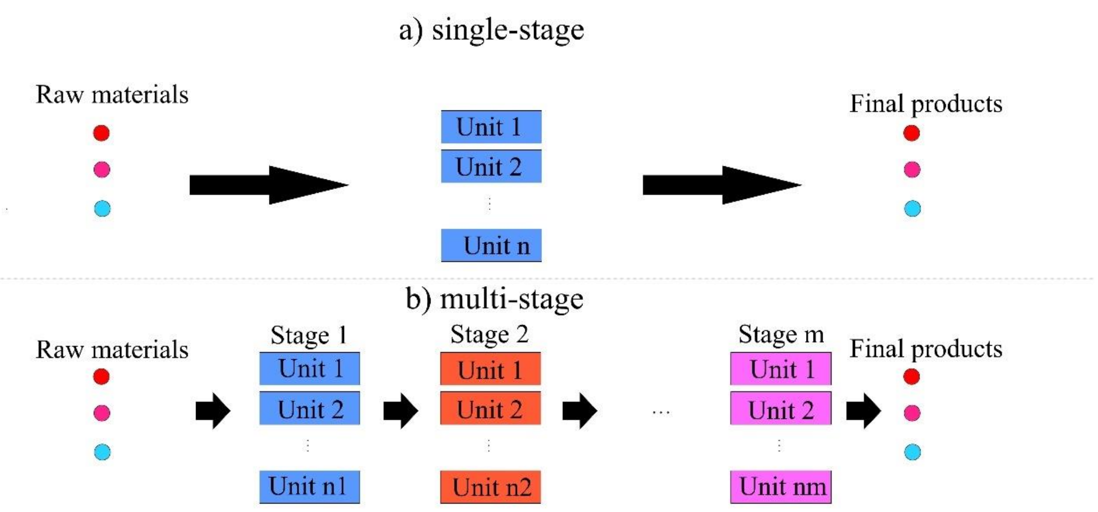

Figure 2 A general single-stage and multi-stage representation

The food industry is an example of a multi-stage batch process industry. In the food industry, it is crucial to avoid contamination between batches. Therefore, there is no split or mix between food product batches. In this case, if there is any contamination, then it only affects a small number of products (i.e. food processed in the same processing units right before or after the contaminated product). Furthermore, in the case of a faulty product, the facility can identify all affected products and remove them from the market, before consumed by the final customer.

1.1.1.3 Multitasking batch process industry

As already discussed, in both multipurpose and multi-stage batch processes, a processing unit can only process one task at a time. For instance, a reactor can only process one reaction, while a separator can only perform one separation at a time. However, there are several cases where a processing unit contains multiple departments or slots, and it can process more than two tasks simultaneously. These are commonly known as multitasking batch processes. A multitasking batch process facility can also perform both multi-stage and multipurpose batch processes.

The scientific service industry is an example of a multitasking batch process industry. Such facilities examine several samples from different customers for physical and chemical properties, using multiple processing units. Each unit can process a large number of those samples in their departments/slots. Furthermore, each department can examine a property independently. As a result, a processing unit can process multiple tasks simultaneously.

1.1.2 Continuous process industry

Continuous process industry processes one or more materials without interruption. In this case, raw materials continuously enter a processing unit, while final products exit uninterrupted. The use of continuous processes is desirable when the facility requires to produce a large quantity of a specific product with the same specification. Using a batch process is not suitable for generating such large amounts of a product, since the quality may slightly differ from batch to batch. Additionally, a significantly large number of batch vessels, which lead to high capital cost are required to fulfil the market demands. Continuous processes can avoid those issues. However, their main drawback is that unscheduled interruption of a processing unit can significantly affect the performance of the facility. Therefore, it is more crucial to perform proper maintenance occasionally to avoid such an unscheduled interruption than facilities that perform batch processes.

The refinery is an example of a continuous process facility. A refinery facility processes large quantities of raw materials (crude oil) to produce enough products (fuel) to fulfil the market demands. Additionally, there are strict specifications in the quality of products, since it is essential to ensure optimal performance by simultaneously fulfilling the environmental specifications. Therefore, only continuous processes are suitable for a refinery.

1.2 Supply chain management

The number and the type of processes between facilities may significantly differ, even if they produce similar final products. However, despite such differences, there are three managerial levels at which every process industry needs to manage its operations. In the first level, a processing facility should decide the number of different products is going to produce. Multiple factors affect such a decision, including the predicted market demands and the available technology for producing such products. Furthermore, the facility must ensure that it follows the health and safety procedures to process the final products. In this level, it also examines the effect of possible extensions or upgrades in the profitability. For instance, if the market demand for a given product is high, then the use of additional processing units will lead to larger productivity and profitability for the facility. Likewise, a facility requires new types of processing units, if it decides to produce new products. Such a facility should also examine if it contains a suitable amount of storage tanks to store all products, which should operate within the safety limits. During a process, a facility should also stock several intermediate products except for raw materials and final products. Therefore, it should examine if the available storage tanks can temporarily store such intermediate products. Finally, the existing or potential new equipment should operate in the conditions required to produce intermediate or final products. Such first-level operational decisions are commonly known as long-term planning or strategic planning. Usually, strategic planning decisions consider periods which can vary for one to multiple years, since changes the strategic planning require high investments.

In the next step, a processing facility needs to determine which products and in which quantity is going to produce for a shorter period than the strategic planning period.

Different factors can affect these decisions, including the availability and the price of raw materials, as well as the actual market needs. The available raw materials and final products (produced in a previous period) can also affect such decisions. Those decisions usually change more frequently than strategic planning decisions since most of these factors fluctuate significantly during short periods. Such planning decisions are commonly known as medium-term planning or tactical planning, the period of which usually varies from multiple weeks to several months. In tactical planning, the processing facility only decides for the type and amount of products among the products specified in the strategic planning that will produce in this period. The structure remains the same, even if a structural change can potentially lead to increased profitability. Similarly, the processing facility can only produce a specific amount of products based on its storage availability, even in cases of high demand for a product. In such a case, the strategic planning of the next period should consider the extension of the facility with additional processing units or storage tanks.

1.2.1 Scheduling decisions in the process industry

The third and last managerial level is commonly known as short-term planning or scheduling. Facilities usually take scheduling decisions for shorter periods, which can vary from multiple hours to one week. Despite the short period of the scheduling decisions, this level is crucial in supply chain management. In this stage, the processing facility decides the processes that are going to take place within the scheduling horizon, which significantly affects the efficiency of the facility. Scheduling decisions includes the allocation, the sequence and the timing of operation into units. More specifically, scheduling decides the processing unit, as well as the start and end processing times for all operations. Such a decision can significantly vary based on the objective of the facility; maximizing the productivity, minimizing the makespan, or minimizing the cost are some of the goals of a typical facility. Additionally, the facility needs to decide if maintenance should take place in one or more processing units. In such a case, it should examine what is the best schedule, that both fulfils the objectives of the facility and successfully perform the maintenance.

It is crucial for the smooth operation of each facility, to have the best schedule, that not only fulfils the customer demands but also ensures that it also produces all products in the minimum possible cost. The importance of such managerial level can exceed the ones of the strategic and tactical planning since the scheduling decisions are responsible for putting the plan into action. Taking bad scheduling decisions can significantly affect the facility's vitality by leading to increased costs or losing customer trust. Additionally, a facility needs to be able to generate a schedule in a short time. As already discussed, a facility should take a scheduling decision for the next few hours. Therefore, facilities should use a method that provides a schedule within a few seconds to up to several minutes. In a case of an unplanned change (i.e. breakdown of a processing unit, increase in product demand, additional customers and orders) the facility needs to develop a new schedule that affects the functionality of the process as less as possible. Even though several approaches can generate the best solution on a given problem, they usually require high computational even for small-scale examples. Many other methods can provide a solution to the scheduling problem in significantly less time. However, they generate far for optimum solutions. It is, therefore, a challenging task up to this day for facilities to find a methodology that can provide optimum or near-optimum schedules in a reasonable time.

In Figure 3, a general diagram of planning decisions that a processing facility should take is presented.

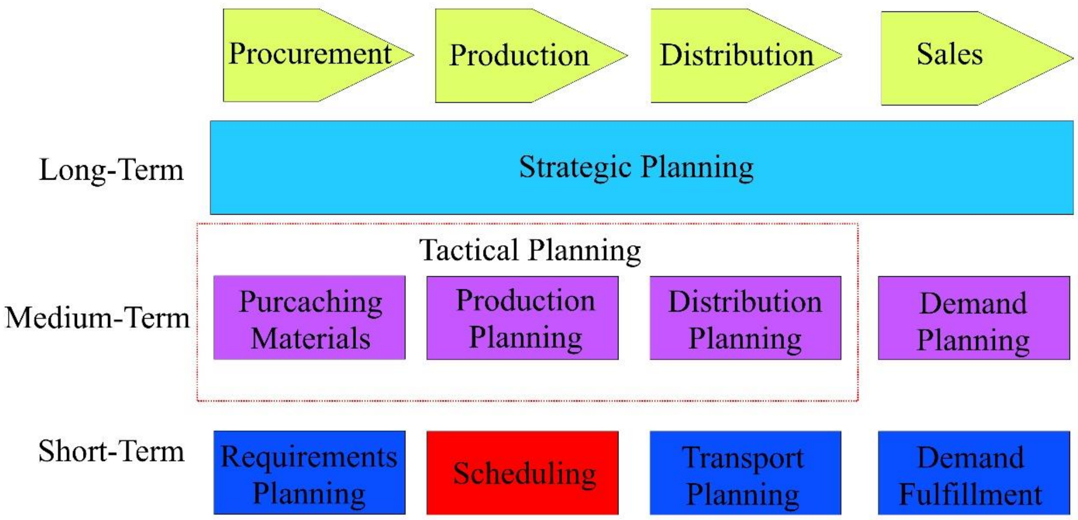

Figure 3 Supply chain planning matrix (Sung and Maravelias 2007; Stadler *et al.* 2015)

1.3 Motivation and Objectives

Even though the optimization of strategic and tactical planning has been well established since the 1950s, scheduling decisions have not gathered enough attention. The main burden of developing an efficient methodology for optimal scheduling was the significantly smaller computational power during the past few decades as well as the fact that the most scheduling problems are NP-hard problems (Garey et al. 1976; Bruker et al. 1998). As a result, the first approaches use heuristic or spreadsheet approaches for scheduling decisions. In most facilities in process industries, such methods are used even up to this day. Since heuristic or spreadsheet approaches are only limited to generate a feasible solution, which in some cases is far from optimum, there is still much room from improvement.

Development of mathematical models for scheduling of process scheduling has gathered some attention during the past three decades. In most cases, those models can solve the scheduling problem of a specific facility (i.e. refineries, steelmaking) or a type of process (i.e. batch process or semi-continuous/continuous process). However, each facility has notable differences and, as a result, it is not possible to use an approach dedicated to a specific type of industry to solve a different scheduling problem. More importantly, there is not a general mathematical modelling framework which can develop the optimal scheduling decisions for all different types of processes. Such limitation is crucial for the existing approaches since a facility may include more than one type of operations.

Another limitation of most existing mathematical models for developing scheduling decisions is the lack of efficiency and robustness. Even though there are mathematical models for most types of industries, they usually require high computational time to generate optimal solutions. Additionally, such approaches fail to incorporate all facility's features in some cases, which limits the capability of those approaches. Such limitations are the main reason that industries prefer using heuristic or spreadsheet approaches over mathematical modelling approaches for their scheduling decisions. In this case, even though they are only able to generate a feasible solution, they can provide such a solution in significantly less computational time. It is, therefore, essential to develop efficient and robust mathematical models to tackle such high computational burdens and inefficiency. Such mathematical model should include all features of the process industry in the mathematical framework and incorporate strategies that significantly reduce the computational time, by reducing the resulting model size and tightening the MILP relaxation of the problem.

Although existing mathematical approaches usually require high computational time to generate the optimal solution for small-scale examples, there are also large-scale scheduling problems, where it is impossible, even for a very efficient mathematical model, to generate the optimal solution in acceptable computational time. For such hard-to-solve scheduling problems, different decomposition approaches have been developed. Most of such algorithms use the strategy of dividing the scheduling horizon into smaller sub-horizons, solving one subproblem at a time and fixing the resulted scheduling decisions before solving the next subproblem. Such approaches are commonly known as rolling horizon decomposition approaches. The division into smaller subproblems is usually performed based on the due dates of each product. More specifically, orders with earlier due dates are assigned to before those orders with later due dates. However, the rolling horizon decomposition approach may fail to successfully divide the scheduling problem if there are many orders with the same due date. Currently, there is not an efficient rolling horizon decomposition approach to handle such cases.

With significant advances in machine learning evolutionary techniques presented, the use of such techniques combined with mathematical modelling can improve the efficiency of developing scheduling decisions. For instance, an evolutionary approach (i.e. gene expression programming) can generate effective rules based on existing information on scheduling decisions for previous scheduling horizons and their effect in the final solution. These rules can efficiently decide the allocation and sequencing of tasks into units. Then, a mathematical model can generate the best timing and batching of each process for the given task allocation and sequencing. This linear programming (LP) problem require acceptable computational time. As a result, such an efficient hybrid algorithm can significantly decrease the computational time. Despite the potential of such hybrid algorithms, it still seems that there is not such an approach developed for process scheduling.

Based on those limitations of the existing mathematical models, the objective of this thesis is:

1) To develop a new generic, robust and efficient framework for the process industry, which lead to smaller model sizes and require less computational time to generate the optimal solution, by introducing several features in the proposed framework, including;

a) Allowing related production and consumption tasks to take place at the same event point;

b) Sequence processing units that process production and consumption tasks related to the same state only if there is an indirect material transfer between those units. Such indirect material transfer should take place if a consumption task consumes more materials than the materials stored in the storage tanks;

c) Align processing units that process production and consumption tasks related to the same state only if there is a direct material transfer between those units. Such direct material transfer should take place if storage capacity is not enough to store all the producing materials. Therefore, they should immediately transfer to another unit;

d) Allow processing units to store the materials that they produced for multiple event points;

2) To enhance the rolling-horizon decompositions algorithms, by using mixed-integer programming to group different products/orders that have the same due dates;

3) To explore a combination of gene expression programming and the mathematical programming approach for energy-efficient scheduling of flexible job-shop scheduling problems.

1.4 Research Contributions and Thesis Structure

The thesis format is "journal format" containing several published or submitted academic papers in peer-reviewed scientific journals. Chapter 1 presents a brief introduction of the thesis, while Chapter 2 presents a detailed literature review for existing approaches for scheduling of process industry. The rest of the chapters contains the research contributions as follows.

1.4.1 Chapter 3 - Research contribution 1

Chapter 3 contains a new approach for scheduling of multipurpose batch processes is presented. In this approach, a new, slightly different definition of recycling and non-recycling tasks, is proposed. Additionally, non-recycling production and consumption tasks can take place at the same event point. Two mathematical models are developed, based on unit-specific event-based time representation. While the first model uses task-based timing variables, the second model uses unit-based timing variables. By solving several well-established examples, it seems that the proposed approach can reduce the number of event points, and it leads to smaller model sizes which improve the efficiency of the model.

This research contribution is published in Frontiers of Chemical Science and Engineering.

Rakovitis, N., Zhang, N., Li, J. Zhang, L. A new approach for scheduling of multipurpose batch processes with unlimited intermediate storage policy. *Front. Chem. Sci. Eng.* **13,** 784–802 (2019) doi: doi.org/10.1007/s11705-019-1858-4

Author's contribution

Nikolaos Rakovitis developed the two mathematical models, examined the models by conducting all the computational studied and wrote the presented manuscript.

Nan Zhang reviewed and edited the manuscript.

Jie Li contemplated and supervised the work, reviewed and edited the manuscript.

Liping Zhang reviewed and edited the manuscript.

1.4.2 Chapter 4 - Research contribution 2

In Chapter 4, a generic framework is developed and implemented in the scheduling problem of multipurpose batch processes. The features of the approach presented in research contribution 1, was included together with the new features of indirect and direct material transfer. Additionally, in the proposed formulation, processing units can store materials for multiple event points. The solutions generated by solving several benchmark examples demonstrate that the proposed model can generate the optimal solution in all cases and it leads to the smallest model sizes and, as a result, it is more efficient.

This research contribution is submitted for publication to AIChE journal.

Rakovitis, N., Pan Y, Zhang, N., Li, J. Kopanos, G. Generic mathematical formulations for scheduling of multipurpose batch plants, AIChE journal, submitted

Author's contribution

Nikolaos Rakovitis developed the generic mathematical model, performed the computational studies and wrote the presented manuscript.

Yueting Pan prepared several GAMS codes for the examples solved

Nan Zhang reviewed and edited the manuscript.

Jie Li contemplated and supervised the work, reviewed and edited the manuscript.

Giorgos Kopanos reviewed and edited the manuscript.

1.4.3 Chapter 5 - Research contribution 3

In Chapter 5, the proposed framework is implemented in the continuous process industry. The results demonstrate that the proposed formulation requires significantly less computational time than a recent existing formulation for scheduling of continuous processes.

Rakovitis, N., Hasnuddin, W. M. A. W., Zhang, N., Li, J. A Generic Approach for Scheduling of Semi-continuous and Continuous Processes, to be submitted to Chemical Engineering Science

Nikolaos Rakovitis developed the generic mathematical model, performed the computational studies and wrote the presented manuscript.

Wan Mohd Azril bin Wan Hasnuddin used the developed mathematical models to solve a number of benchmark examples.

Nan Zhang reviewed and edited the manuscript.

Jie Li contemplated and supervised the work, reviewed and edited the manuscript.

1.4.4 Chapter 6 - Research contribution 4

In Chapter 6, the proposed framework is used for scheduling of multitasking batch processes. Except from the proposed framework, another mathematical model is developed, which is based on unit-specific event-based time representation with task-based timing variables. The proposed framework leads to significantly less computational time than all mathematical models and it can generate significantly better solutions than the non-uniform discrete-time model of Lagzi *et al.* 2017b.

This research contribution is published in Computers and Chemical Engineering

Rakovitis, N., Zhang, N., Li, J. A novel unit-specific event-based formulation for short-term scheduling of multitasking processes in scientific service facilities, Computers and Chemical Engineering, 133(2), (2020) doi: doi.org/10.1016/j.compchemeng.2019.106626

Author's contribution

Nikolaos Rakovitis developed the two mathematical models, conducted all the computational studied and wrote the presented manuscript.

Nan Zhang reviewed and edited the manuscript.

Jie Li contemplated and supervised the work, reviewed and edited the manuscript.

1.4.5 Chapter 7 - Research contribution 5

Chapter 7 presents three novel mathematical models for scheduling of energy-efficient flexible job shops. The first model implements the framework developed in the previous chapters to solve this model, while the second and third model uses the local sequence-based representation. An enhanced rolling horizon decomposition approach is also presented, where a grouping strategy using the mixed-integer programming divides the entire problem into different subproblems. Such decomposition approach can successfully decompose a large-scale problem, where all products/orders have the same due date. Furthermore, the combination of those mathematical models with existing evolutionary approaches has been examined. The results demonstrate that the models are more efficient and robust than all existing mathematical models. Furthermore, combining the models with the proposed rolling horizon decomposition approach leads to significantly better solutions and less computational time. Several comparative studies have shown that the proposed algorithm can generate better solutions than the best-reported approach for this scheduling problem.

Rakovitis, N., Zhang, N., Li, J. Zhang, L. Novel Approaches for Energy-Efficient Scheduling of Flexible Job-Shop Problems, to be submitted to European Journal of Operational Research

Author's contribution

Nikolaos Rakovitis developed the mathematical models, the enhanced rolling horizon decomposition approach and the hybrid mathematical programming and evolutionary approach algorithm, conducted the computational studies for the mentioned approaches and wrote the presented manuscript.

Nan Zhang reviewed and edited the manuscript.

Jie Li contemplated and supervised the work, reviewed and edited the manuscript.

Liping Zhang provided the computational results of GEP approach, reviewed and edited the manuscript.

Chapter 2: Background & Literature review

Scheduling problems in the process industry has gathered significant attention during the past three decades. Multiple research groups proposed different approaches, especially mathematical models, to generate optimal or near-optimal schedules for both batch and continuous processes. In this chapter, a brief background on different programming optimization approaches, as well as the process and time representations, will be presented. Additionally, the formulations proposed for scheduling of single-stage, multi-stage, multipurpose, flexible job-shop and multi-tasking batch processes, as well as continuous processes, will be presented and discussed.

2.1 Introduction in optimization

In a processing facility, several actions, as well as physical and chemical phenomena, take place. For instance, raw materials occasionally enter and exit a processing unit during the scheduling horizon. Additionally, the facility should distribute the final products in the market. Furthermore, multiple processes such as reactions, heating of materials and separations take place to produce such products. Mathematical relations such as equalities, inequalities and logical conditions can describe such activities and phenomena. The combination of all these relations creates a mathematical model (Floudas 1995) which describes the processing facility.

Several factors can affect the performance of the facility. Each facility aims to choose those factors that lead to the best performance. An objective function mathematically describes the performance of the facility. The objective usually differs in each facility, since what is the best performance is subjective. For instance, it can be desirable to either minimize the likelihood of undesirable events such as breakdowns or to maximize the productivity of the facility. Even though both cases aim to maximize the profitability of the facility, they may lead to different solutions and different process performance. The objective function together with the mathematical model consisting of all constraints is an optimization problem (Edgar and Himmelblau, 1989). A general optimization problem can have the following structure.

$$
\begin{aligned}
\max_{x} \quad & f(x) \\
s.t. \quad & g_i(x) = 0, \quad i = 1,2,\dots,n \\
& h_j(x) \leq 0, \quad j = 1,2,\dots,m \\
& \qquad\qquad x \in X \subseteq R^n
\end{aligned}
$$

Where x is the vector of continuous variables, $g_i(x)$ is a set of equality constraints, $h_j(x)$ is a set of inequality constraints and $f(x)$ is the objective function.

Optimization problems are classified based on the type of variables and constraints that the mathematical model contains (Figure 4). In the simplest case, the mathematical model includes linear constraints and continuous variables. This optimization problem is a linear programming (LP) problem. If the model also contains integer variables, then the mathematical model is a mixed-integer linear programming (MILP) problem. Such variables are necessary if the scheduling problem considers logical conditions. For instance, assigning the process of a task to a processing unit requires several binary decision variables. Examining if a processing unit is active during a specific time is also imposed by using several binary variables. It is also possible to deal with restricted cases, where, it is not possible to have a decimal number (i.e. the number of samples processed in a processing unit). These types of variables should only take integer variables. If all the variables are integer though, the optimization problem is named integer programming (IP) problem. Furthermore, if the optimization problem contains non-linear terms (e.g. to explain complicated phenomena), it is called non-linear programming (NLP) problem if there are only continuous variables. Finally, if there are both continuous and discrete variables, it is called mix-integer non-linear programming (MINLP) problem (Edgar and Himmelblau, 1989; Floudas, 1995).

A process industry needs to make multiple decisions in all planning periods, such as choosing the producing products, the processing units, as well as the detailed sequencing and allocation of tasks into units. Such decisions can only be described in a mathematical model by introducing several binary variables. As a result, most of the planning and scheduling problems are MILP problems. Next, more details for the MILP problems will be presented. Additionally, the branch and bound method will be presented, which is the most common method to find the optimal solution of a MILP problem.

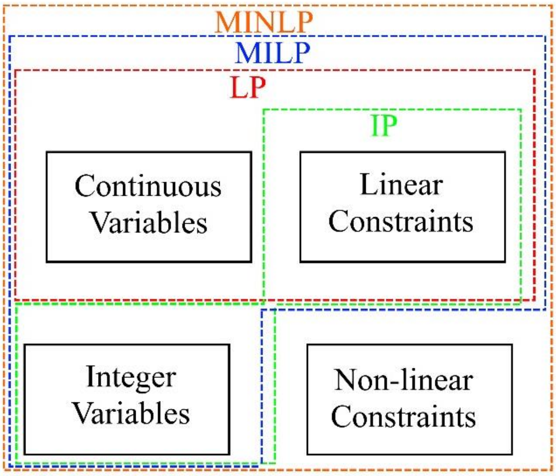

Figure 4 Classification of mathematical models

2.1.1 MILP & relaxed MILP optimization problem

As discussed, a MILP optimization problem contains multiple linear constraints and several continuous and binary/integer variables. A general MILP problem has the following structure.

$$
\begin{aligned}
\max_{x,y} \quad & f(x,y) \\
s.t. \quad & g_i(x,y) = 0, \quad i = 1,2,\dots,n \\
& h_j(x,y) \le 0, \quad j = 1,2,\dots,m \\
& \qquad\qquad x \in X \subseteq R^n \\
& \qquad\qquad y \in Y \ integer
\end{aligned}
$$

Where x is the vector of continuous variables, y is the vector of the integer variables $g_i(x,y)$ is a set of equality constraints, $h_j(x,y)$ is a set of inequality constraints and $f(x,y)$ is the objective function.

Another optimization problem that is also solved is the relaxed mixed-integer linear programming (rMILP) problem. The rMILP problem contains the same objective function and constraints with the MILP problem. The main difference with the MILP problem is that the integer variables are denoted as continuous variables instead. In other words, all binary variables of the MILP problem can take values within the interval [0,1] in the rMILP problem. Therefore, the rMILP problem is an LP problem which is usually easy to solve. The solution to this problem provides an upper/lower bound to the MILP problem. More specifically, in maximization problems, the solution of the rMILP problem is the upper bound for the MILP problem.

An important factor that affects the efficiency of a mathematical model is the difference between the solution of the MILP and the rMILP problem. A mathematical model is tight if there is a small gap between those solutions. In this case, the rMILP solution usually contains multiple relaxed variables with an integer solution, even though those variables are continuous. It is desirable that the rMILP solution only contains relaxed variables with integer solution. In such a case, the rMILP solution is also a solution to the MILP problem. If the gap between the MILP and the rMILP solution is large, then the problem may require excessive computational time even to generate a feasible solution. A set of tightening constraints can tight the relaxation of the problem. Those constraints, which may or may not have a physical meaning, they force relaxed variables to have integer or close to integer values in the rMILP solution.

2.1.2 Branch and bound algorithm

Usually, MILP problems are hard to solve. One method to find the optimal solution in small examples is to investigate the best solution by examining all possible permutations of the integer variables. For instance, in a batch scheduling problem with two batch processes and two processing units, there are four possible permutations in total. In such an example, it is easy to examine all four permutations to find the best solution. However, the possible permutations exponentially increase as the example size increases. As a result, it is computationally expensive to examine all of them in a common problem. Branch and bound is an algorithm that can solve MILP problems, mainly because it can prove that a solution is the best solution without examining all permutations.

The branch and bound algorithm finds the best solution to a MILP problem as follows. In the first step, the algorithm relaxes all integer variables. Therefore, the resulting rMILP problem is solved first. The solution of the rMILP problem is the upper bound (or lower bound in minimization problems) of the MILP problem. In other words, it is not possible to find an integer solution with a better objective value. The rMILP result is the root node of the branch and bound algorithm. In the root node, there are three different cases. In the first case, the root node is infeasible, and as a result, it is not necessary to examine any permutation since the MILP problem will also be infeasible. In the second case, all relaxed variables have integer values. In that case, it is also unnecessary to examine any permutation since the rMILP solution is also a solution of the MILP problem. Since rMILP provides the upper bound of the problem, there is not an integer solution with a better objective. Finally, in the last case, some or all relaxed variables have non-integer values. In such a case, several permutations, to find the best solution, should be examined.

The branch and bound algorithm does not examine the permutations randomly. Instead, it introduces additional constraints in the relaxed model, and it evaluates the solution before proceeding to the examination of a branch. Similar to the root node, the algorithm does not examine any additional permutations of the branch if the node is infeasible or if all relaxed values have integer values. However, there is one more case that the algorithm stops examining a branch; if the solution of a node has an objective value less than the best integer solution found so far. In this case, there is not a better solution in this branch. The whole procedure continues until all the branches are examined or pruned, or until the difference between the best integer solution and the upper bound is less than the specified accuracy.

After the branch and bound algorithm examines a node, the algorithm continues as follows. Let's assume that the solution of a node contains relaxed variables with decimal values. In this case, the algorithm chooses one of those variables, usually the one that is further from the closest integer. For instance, if two relaxed variables have the values x1 = 2.3 and x2 = 4.4 respectively, then the algorithm chooses variable x2. In the next step, the algorithm generates two new nodes by adding a constraint. In the first node, the chosen variable can take values less or equal to the greatest integer value that is smaller than the value of the variable, while in the second node, it takes values greater or equal to the smallest integer values that are greater than the value of the variable. For instance, if the same example is considered, where x2 is chosen, the algorithm adds the constraint $x2 \leq 4$ to create the first node and the constraint $x2 \geq 5$, to create the second node. For both nodes, optimization takes place to find the best solution by using the new set of constraints.

In Figure 5, a simple tree, where all the possible cases are depicted. In each node, the objective value is depicted. For instance, the root node has an objective value of 15, while the best integer solution has a value of 10. Additionally, in each arrow, the constraint included is depicted.

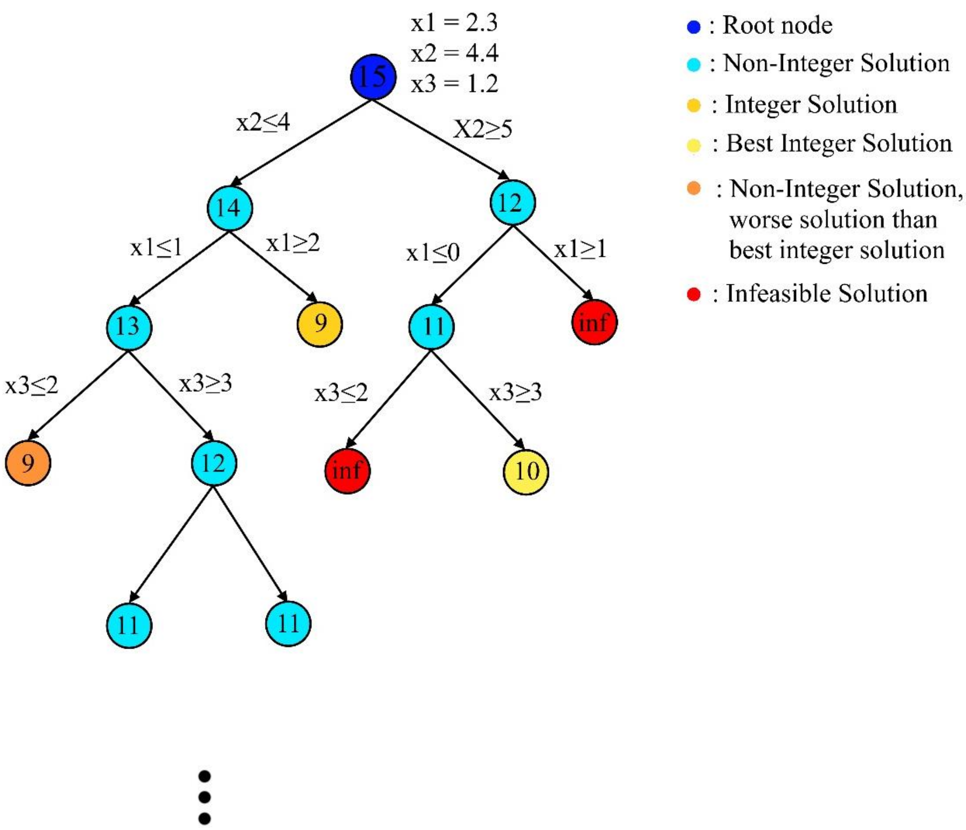

Figure 5 Branch and bound tree

2.2 Process representations

Process representation is essential to develop efficient mathematical models for the process industry. Such representation should contain information for the available units, the tasks that they are processing as well as the materials produced and consumed in each process. Additional details for connection between processes, processing paths, conversion rates and resources are also necessary. The most common-used representations developed are the State Task Network (STN), the Resource Task Network (RTN), the State Sequence Network (SSN) and the Disjunctive graphs. This chapter presents a brief introduction to those process representations.

2.2.1 State Task Network representation

Kondili *et al.* (1993) was the first to introduce the state task network (STN) representation. In STN representation, tasks denote all processes/operations in a facility while states denote the consuming/producing materials. Simple shapes such as rectangles and circles represent tasks and states of a facility, respectively Simple shapes such as rectangles and circles represent tasks and states of a facility, respectively. For instance, if the arrow points out the task, then it consumes the related state. Similarly, if the arrow points out the state, then the related task produces this state.

Figure 6 depicts the STN representation of an illustrative example. In this example, three states represent one raw material, one intermediate product and one final product. Additionally, two tasks represent the two processes the processing facility uses to produce the final product.

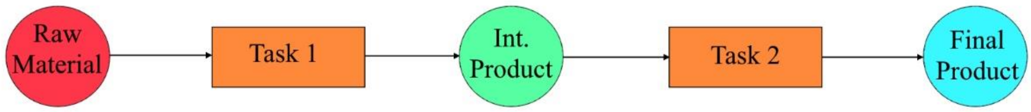

Figure 6 STN representation

2.2.2 Resource Task Network representation

The STN representation even though it is useful to represent a facility, it does not contain any resources. As a result, the STN representation cannot provide all the necessary information in examples with resource constraints. Pantelides (1994) tackled this issue, with the Resource Task Network (RTN) representation. In RTN representation, circles except for states, they also represent resources. If a task requires one of those resources, then a dotted arrow depicts this relation.

Figure 7 presents an example of an RTN representation. In this example, there are two resources available. The first task requires both resources, while the second task only requires the second resource.

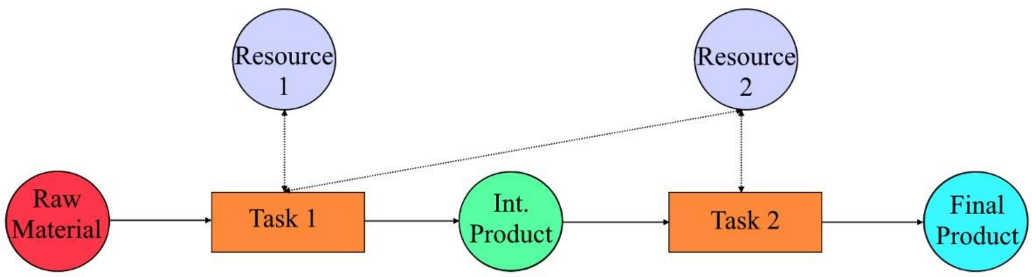

Figure 7 RTN representation

2.2.3 State Sequence Network representation

Majozi and Zhu (2001) proposed the State Sequence Network (SSN) representation. This representation only represents the states by using rectangles. In contrast to STN and RTN representation, an arrow represents the relation between two states. The direction of the arrow denotes the processing path. For instance, if the arrow points out a state, then a process produces this state by consuming the related state. In this way, the SSN representation does not immediately represent the processes. Instead, it assumes that a processing unit performs the conversion of one state to another one. Finally, a node denotes the mixing of two states or the splitting of a batch.

Figures 8 and 9 present the SSN representation of two examples. The first example in Figure 8 is the same as the example presented in Figures 6 and 7. Figure 9 depicts the second example with five states in total; one raw material, two intermediate products and two final products. In node 1, the raw material splits since two processing units consume the same state. Each processing unit produces a different intermediate state. In node 2, intermediate product 1 splits into two parts. A process consumes the first part to produce final product 2. Finally, in node 3, another unit consumes a mixture of intermediate products 1 and 2 to produce final product 2.

Figure 8 SSN representation a

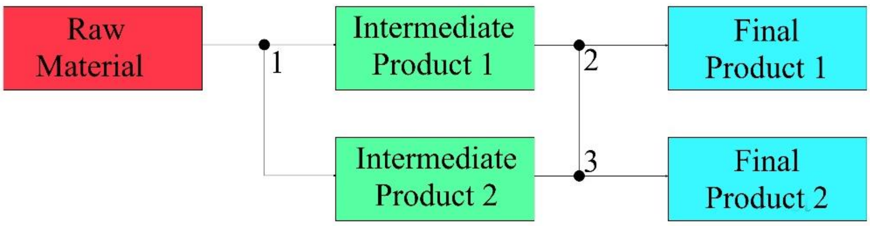

Figure 9 SSN representation b

2.2.4 Disjunctive graphs

Roy and Sussmann (1964) developed disjunctive graphs to represent a job-shop facility. In a disjunctive graph, a node represents an operation/task. Each node usually contains two numbers to denote the job and the operation. Furthermore, two dummy nodes represent the start and the end of all jobs. All operations can only start after the dummy start node and before the dummy end node. Solid arks depict the relation between two consecutive operations in a job, the direction of which denotes the sequence. In a job-shop facility, each processing unit can process different operations/tasks. If a processing unit can process two or more operations/tasks, then disjunctive arks connect all these operations. Each type of ark (dotted, dashed) or colour denotes a different processing unit. Finally, a disjunctive graph may also depict the earlier start time that an operation can start based on the processing time.

The disjunctive graphs can successfully represent both classical and flexible job-shop scheduling problems. Figure 10 depicts an example of a disjunctive graph with three jobs and three operations/tasks in each job. Nodes 0 and 1 are the dummy start and the dummy end node, respectively. This figure also depicts the relation between processing units and operations. For instance, unit 1 can process operations (1,1), (2,1) and (2,2) while unit 2 can process operations (1,2), (3,1) and (3,3). Finally, unit 3 processes operations (1,3), (3,2) and (2,3).

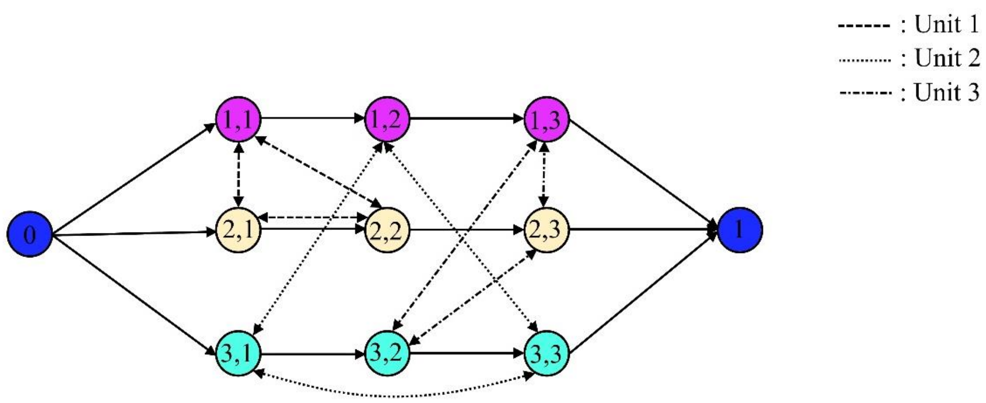

Figure 10 disjunctive graph

2.3 Time representations

Before developing a mathematical model, it is necessary to decide the time representation of the developed formulation. Different time representations lead to different model sizes and relaxations, and as a result, they significantly affect the efficiency of the model. There are two different types of time representations; discrete-time and continuous-time.

2.3.1 Discrete time representation

One of the first attempts to develop mathematical models for scheduling of process facilities were based on the discrete-time representation (Bowman 1959). Such approach divides the scheduling horizon into several time intervals. Each time interval has a fixed and known length before the optimization problem, while the start and the end of a process or activity can only take place at the bounds of a time interval. Mathematical models based on discrete-time representation can either be uniform or non-uniform. In the former models, the time intervals for all processing units have the same length during the whole scheduling horizon. On the other hand, in non-uniform discrete-time models, the duration of the time intervals can differ from unit to unit. Furthermore, for a given processing unit, the length of the time intervals can also be different during the scheduling horizon. However, in both models, the duration of each time interval cannot change during optimization.

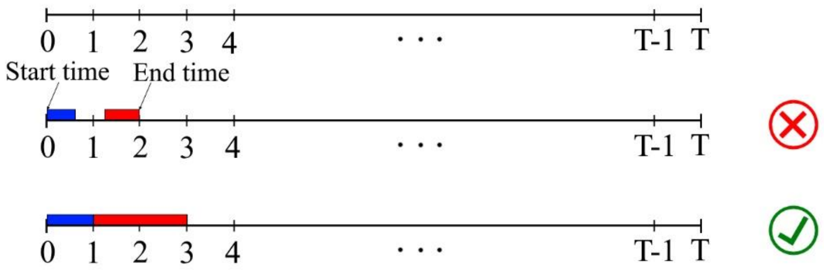

Figure 11 Division of scheduling horizon in discrete time formulations. The start and end time of a task/operation/process must be exactly at the time interval

Discrete-time representation models are easy to implement, and they lead to simple formulations with tight relaxations, especially if the objective is the maximization of productivity. However, since the length of each time interval remains fixed, significantly many time intervals are required in most cases. More specifically, their duration should be equal to the greatest common factor of the processing time of all tasks in all available units in uniform discrete-time models. Since the processing time can significantly differ, usually a large number of time intervals are required to generate the optimal solution. As a result, discrete-time models lead to large model sizes, even for small examples. The use of time intervals with different length can significantly reduce the model size of the problem. However, since a task can only start or finish at the bounds time intervals, rounding the processing time to the closest multiple of the length is required. In this case, it is possible to overestimate or underestimate the productivity of a given processing unit.

Non-uniform discrete-time models can also lead to smaller model sizes. However, since the length of time intervals between processing units can differ, the time intervals between two units processing related tasks may not match. In this case, the consumption task starts in the next available time interval. As a result, using such models may lead to suboptimum solutions, where one or more units remain idle for specific periods.

2.3.2 Continuous-time representations

In contrast to discrete-time representations, in continuous time representations, the scheduling horizon is not divided into time intervals of equal length. Instead, the division of the scheduling horizon takes place during the optimization. Based on how the division takes place, continuous-time representations are classified into slot-based, global event-based and unit-specific event-based representations.

2.3.2.1 Global event-based representation

The global event-based representation uses multiple event points to divide the scheduling horizon. In contrast to the discrete-time formulations, the position of each event point is unknown. The optimization problem determines the location of each event point. As a result, the length between consecutive event points can differ. Global event-based representation requires fewer event points than discrete-time, which leads to significantly smaller model sizes. However, models based on this representation usually have worse relaxations since they contain several constraints with big-M terms. Another disadvantage is that the start time for all processing units is the same for a given event point. In other words, global event-based representation divides the scheduling horizon uniformly for all event points (Reklaitis and Mockus 1995).

The optimal number of event points are unknown in advance. Instead, an iterative procedure determines the best number of event points. More specifically, the model first uses the minimum number of event points to solve the problem. In the next step, it uses an additional event point to solve the same problem. If there is an improvement in the solution, then the problem is further solved with more number of event points. The procedure continues until there is no improvement in the solution.

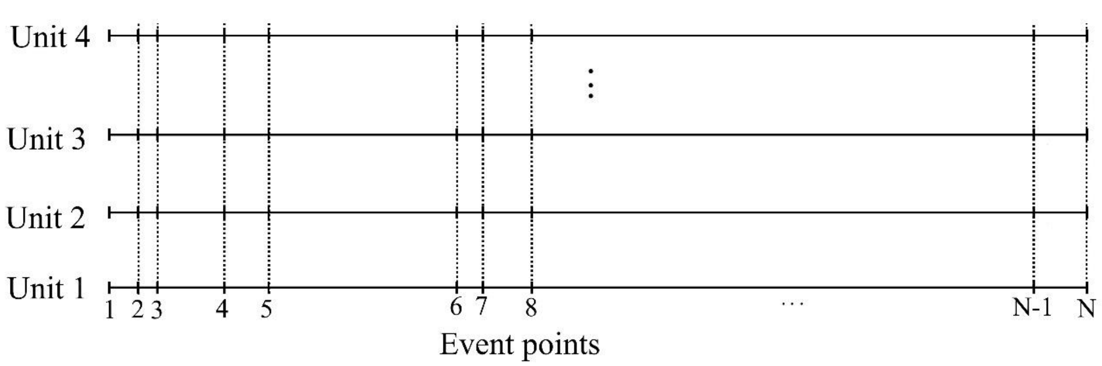

Figure 11 Global event-based representation

2.3.2.2 Unit-specific event-based representation

Similar to the global event-based, the unit-specific event-based representation divides the scheduling horizon using several event points (Ierapetritou and Floudas 1998a). However, in the unit-specific event-based approach, the event points split the scheduling horizon differently for each processing unit. As a result, the start time during a specific event point can differ between two units. This representation leads to less number of event points and as a result, to smaller model sizes than global event-based representation. However, it also leads to worse relaxations in some cases since they introduce constraints with big-M terms. Additionally, the iteration procedure is also required to generate the optimal number of event points in unit-specific event-based representation.

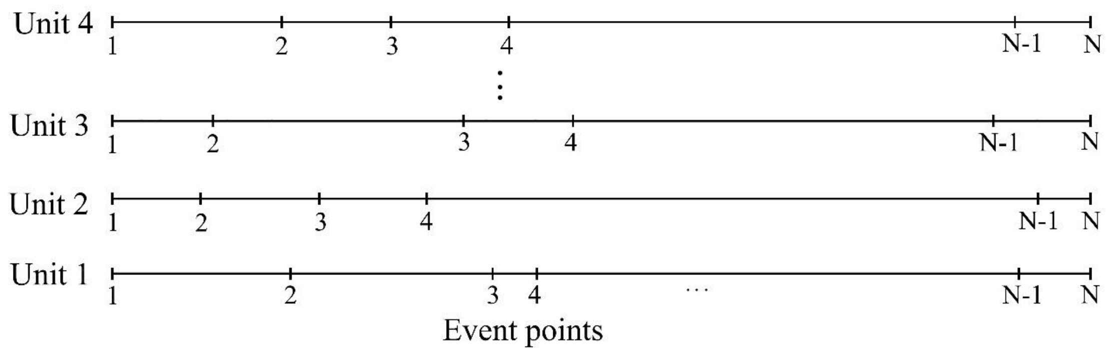

Figure 12 Unit-specific event-based representation

In the literature, there are two different types of unit-specific event-based mathematical models. The main difference in these models lays in the modelled timing variables. The first type of models, which is the most common ones, use timing variables based on tasks. More specifically, the event points divide the scheduling horizon differently for each task. In a mathematical model based on a unit-specific event-based representation using task-based timing variables, the start or/and the finish time of a task during an event point is defined as a variable. On the other hand, in the rest of unit-specific event-based mathematical models, the start or/and the finish time of a processing unit during an event point is defined as a variable. In other words, the event points divide the scheduling horizon based on units. Using task-based timing variables usually leads to worse relaxation and as a result, such models require tightening constraints to achieve the same rMILP solution with models using unit-based models.

2.3.2.3 Slot-based representation

Slot-based representations divide the scheduling horizon using several time slots (Pinto and Grossmann, 1994). Similar to other continuous-time representations, the optimization problem specifies the length of each time slot. In contrast to global and unit-specific event-based representation, the end time of a time slot should coincide with the start time of the next time slot. There are two different types of models using slot-based representation; process slot-based and unit-slot models. In process slot-based models, the time slots are common to all processing units, similar to the global event-based formulations. In unit-slot models, the time and length of time slots, as well as the start and end times, can differ in each processing unit, which is the same as unit-specific event-based representation. The iteration procedure determines the optimal number of time slots for both types of models. The main difference is that slot-based representation models introduce the duration of each slot as continuous variables. On the contrary, in global event-based and unit-specific event-based, the position of each event point is defined instead.

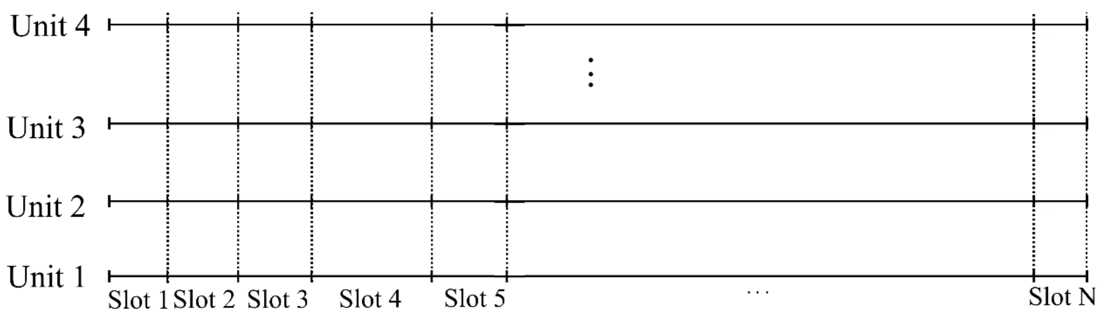

Figure 13 Process slot-based representation

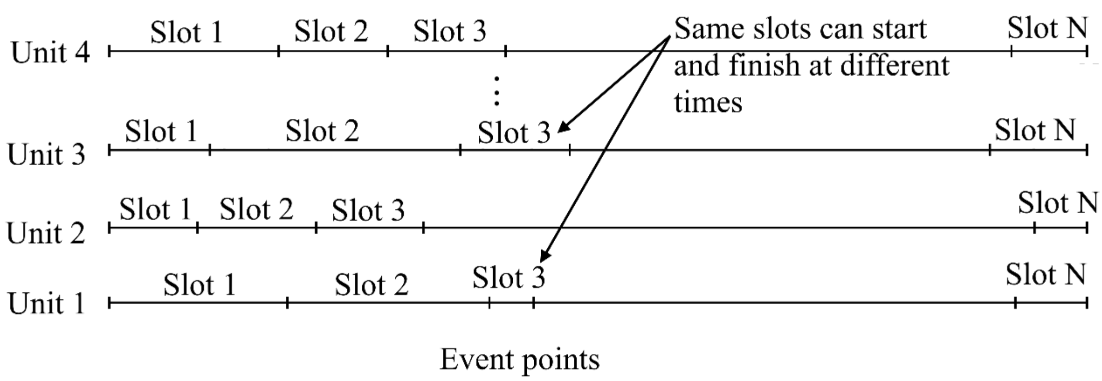

Figure 14 Unit slot-based representation

2.3.3 Sequence-based representation

Sequence-based representation, do not divide the scheduling horizon into time intervals/time slots/event points. Instead, they define the sequencing of operations into units (Ku and Karimi, 1988). There are two different types of sequence-based models; local sequence-based and global sequence-based representations. The former models define the sequencing between two successive operations, while the later models only examine whether an operation precedes another operation in a processing unit. Since such formulations determine the sequence of two tasks, time is not explicitly modelled. As a result, there are no event points or time slots that need to define a priori with an iteration procedure. One of their disadvantages, However, the number of batches have to be determined a priori. Additionally, they do suffer from the difficulty in monitoring resource levels.

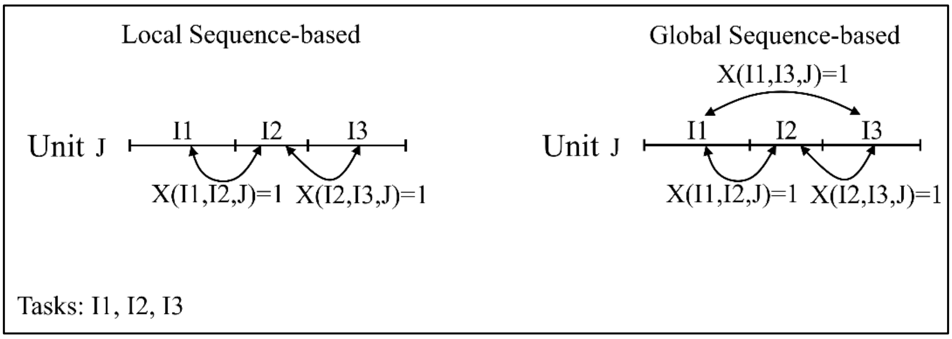

Figure 15 Local Sequence-based and Global-Sequence-based representations

2.4 Scheduling of multipurpose batch processes

Developing methodologies for scheduling of multipurpose batch processes is not a recent trend. Instead, it has gathered much attention since the 1990s. Kondili *et al.* (1993) were one of the first attempts to tackle this problem. To generate the schedule of a general multipurpose batch process, they proposed the STN representation based on which they developed a simple discrete-time formulation. The pioneering work of Kondili *et al.* (1993) inspired multiple researchers to formulate efficient mathematical models and the examples solved are used to examine and compare new mathematical models in the past three decades.

Despite the novelty of the Kondili’s *et al.* (1993) work, it seems that the proposed model leads to significantly large model sizes which affect the performance of the model, even after reformulating some of the constraints to improve the relaxation of the problem

(Shah *et al.* 1993). Such inefficiency is due to the many time intervals required even for the small examples. This issue motivated the research community to develop mathematical models, using continuous-time representations to reduce the number of time intervals required and as a result, reduce the model size the improve the efficiency. Mockus and Reklaitis presented the first global event-based mathematical model (Reklaitis and Mockus 1995; Mockus and Reklaitis 1997). They simplified their MINLP model by using exact linearization. Ierapetritou and Floudas (1998a, b) introduced the unit-specific event-based time representation for the same problem. Both models based on continuous time-representations require significantly fewer slots/event points, and as a result, they lead to smaller model sizes. Between those models, the model of Ierpetritou and Floudas (1998a) is the most efficient since it requires the least number of event points.

The models followed the one of Kondili *et al.* (1993), even though they managed to reduce the model size, they still require excessive computational time to generate the optimal solution. One of the reasons is that those models are MINLP models, which need linearisation and as a result, it increases the complexity of the model (Reklaitis and Mockus 1995; Mockus and Reklaitis 1997). Furthermore, the model of Ierapetritou (1998a, b) generates schedules with the scheduling horizon violation (Castro *et al.* 2001). As a result, in later attempts, a new type of binary and continuous variables and constraints were examined to reduce the model size, as well as different time representations. Additional features were also added in many formulations to create a more general formulation. Zhang and Sargent (1996) developed a new global-event based mathematical model. Even though the constraints used contain non-linear terms which lead to an MINLP model linearisation can convert it into an MILP problem. Schilling and Pantelides (1996) presented an MINLP model for scheduling of multipurpose batch processes, which they converted to MILP by using Glover's transformation (Glover 1975). They also used global event-based time representation. Both models still require intractable time, even with the developed simplifications. Castro *et al.* (2001) managed to reduce the number of event points in global event-based representations, by allowing the length between two event points to be larger than the processing time of a task processed in the first event point. As a result, their model is significantly more efficient than the mathematical model of Schiling and Pantelides (1996), and it can generate a schedule with no scheduling horizon violation by using the same number of event points. Their model does not contain any non-linear term, and as a result, it leads to a MILP problem. Castro *et al.* (2004) later improved the model of Castro *et al.* (2001) by introducing a different set of constraints that lead to tighter relaxation. Majozi and Zhu (2001) proposed the SSN representation. Based on the SSN representation, they presented two mathematical models, where they use different timing variables for each state. These models do not introduce binary variables for tasks, since in SSN only introduce states. However, it seems that the proposed model requires significantly larger model sizes with more continuous variables and constraints. Lee *et al.* (2001) used three sets of binary variables to denote whether a unit starts, continues or ends processing a task during an event point, which can reduce the number of binary variables. They implemented their approach in a unit-specific event-based model. Gianelos and Georgiadis (2002) reduced the number of event points required in unit-specific event-based formulations by introducing a different set of sequencing constraints than Ierapetritou and Floudas (1998a). Their proposed model does not lead to scheduling horizon violation, in contrast to the model of Ierapetritou and Floudas (1998a). Maravelias and Grossmann (2003) used the global event-based and the unit-specific event-based representation for tasks that produce or do not produce a state with zero-wait policy, respectively. Janak *et al.* (2004) modified and extended the model of Ierapetritou and Floudas (1998a) to include different storage policies. Sundaramoorthy and Karimi (2005) developed a slot-based mathematical model for scheduling of multipurpose batch processes. Shaik and Floudas (2008) implemented the RTN process representation in a unit-specific event-based mathematical model for the first time. Shaik and Floudas (2009) introduced a parameter to control the number of event points that a task can span, which leads to smaller model sizes than the model of Janak *et al.* (2004). They also extended this model to solve problems with limited resources. Vooradi and Shaik (2012) improved the mathematical model of Shaik and Floudas (2009), by introducing a single set of allocation constraints and removing the big-M terms from duration and different task in different unit sequencing constraints. Even though the improved model leads to smaller model sizes and tighter relaxation, it seems that in some cases, the model leads to more number of event points, which leads to significant increases in computational expenses. Lee and Maravelias (2017) attempted to improve the efficiency of discrete-based models by presenting two models using the STN and RTN representation, respectively. Finally, Lee and Maravelias (2018) developed a solution approach by combining discrete and continuous-time formulations to reduce the computational time required. Even though they managed to reduce the computational time, their model can lead to a suboptimum solution, especially if the necessary parameters are not correctly tuned.

Despite the multiple proposed models, extensions and improvements presented in the literature, it is still computationally expensive to solve the multipurpose batch process scheduling problem. One of the reasons that those mathematical models lead to high computational time is the unnecessary sequence and alignment of related production and consumption tasks, which was leading to large model sizes. Seid and Majozi (2012) managed to reduce the model size by conditionally sequence all related production and consumption tasks, based on the availability of the consuming state. They also aligned all those production and consumption tasks, based on the availability of storage. However, their model leads to schedules with real-time storage violations. Vooradi and Shaik (2013) also conditionally sequence and align related production tasks, based on whether the consumption task consumes materials from the production task, or whether the materials produced by the production task can be stored, respectively. Even though they avoided to generate schedules with a real-time violation, their formulation introduced a significantly large number of binary variables, which deteriorate the performance of the model in some cases.

As discussed before, mathematical models based on continuous-time representations require an unknown number of time slots/event points. In this case, the iterative procedure finds the optimum amount of time slots or event points. This procedure first solves the problem using the minimum number of event points. The number of time slots/event points are increased by one until there is no improvement in the solution using two consecutive event points. However, using such a procedure to find the optimal solution may lead to intractable computational time. Li and Floudas (2010) developed a framework for optimal event point determination in unit-specific event-based mathematical models to decrease the time required to find the number of event points to generate the optimal solution.

Recently, the use of metaheuristics to solve the multipurpose batch process problems gained attention. Research groups using metaheuristics aim to develop near-optimum solutions in significantly less computation time than the existing mathematical models. He and Hui (2010) analyzed an example from Kondili *et al.* (1993). For this example, they defined the crucial factors that significantly affect the makespan, such as units, tasks and products. Based on this analysis, they developed a genetic algorithm to assign the key-tasks in the key-units. By using the classical approaches of selection, mutation and crossover, they were able to generate a good schedule in small computational time. Woolway and Majozi (2018) developed a general framework for scheduling of multipurpose batch processes. Similar to He and Hui (2010), they proposed a genetic algorithm which uses the techniques for selection crossover and mutation. They also used a chromosome with two distinct parts, which determine the assignment of a unit to an event point and the length between two event points, respectively. Finally, Woolway and Majozi (2019) modified the approach of Woolway and Majozi (2018) to consider a discrete-time framework. They also tested the simulated annealing (SA) algorithm and the migrating bird optimization (MBO) algorithm. Even, though such approaches can significantly reduce the computational time, it still seems that they are unable to prove the optimality of the solution and it is still possible to generate a far from the optimum solution.

2.5 Scheduling of single and multi-stage batch processes

Scheduling of single- and multi-stage batch processes has also gathered considerable attention in the past three decades. In some of the early attempts, researchers developed models for the single-stage batch process with multiple parallel processing units, due to its simplicity. Cerdá *et al.* (1997) were the first to propose a mathematical model to solve the single-stage batch process problem. Their model uses immediate sequence-based time representation. Méndez and Cerdá (2000) included the problem of limited storage by introducing a separate stage. They also used the same time representation as with Cerdá *et al.* (1997). Both models of Cerdá *et al.* (1997) and Méndez and Cerdá (2000) predefine the number and the size of batches that are going to be processed. Méndez *et al.* (2000) dropped this assumption by considering the batching problem. More specifically, they propose a two-step approach, where the first step determines the optimum number and size of batches are determined. In the second step, a direct sequence-based model solves the scheduling problem, based on the first step. Lamba and Karimi (2002) solved the single-stage problem with limited resources. Finally, Castro and Grossmann (2006) examined the performance of different time representations, including discrete and continuous-time representations (global- and unit-specific event-based representations) in the single-stage problem. The authors concluded that the unit-specific event-based time representation leads to significantly smaller model sizes than the rest of the time representations.

Despite the progress in developing schedules for single-stage batch process examples, such approaches cannot directly solve process industry problems. Processing of a batch in more than one stages is familiar in the process industry. In this case, an efficient mathematical model should not only determine the best assignment and timing of tasks into products but also ensure that a process of a batch starts before the finish time of all other processes in all previous stages. Such case led researchers to extend their models to consider multi-stage models or to develop new ones. Pinto and Grossmann (1995) developed a mathematical model for scheduling of multi-stage batch processes, even before the first models for single-stage problems. The authors used unit-based and task-based timing variables, which they connect by using a set of time matching constraints. However, since the model requires significant computational time, they examined the preordering of the sequencing of tasks. Pinto and Grossmann (1996) proposed an improved model of Pinto and Grossmann (1995), which requires less continuous variables and constraints to generate the optimal solution. Even though they improved the efficiency of their model, they still lead to significant computational expenses. Hui and Gupta (2000) and Hui *et al.* (2000) used a different time representation. More specifically, they developed a direct sequence-based model for scheduling of multi-stage batch processes. To improve the efficiency of their approach, they also proposed a preordering heuristic, which assigns the sequencing of tasks based on their due dates. Méndez *et al.* (2001) also solved the problem of scheduling of multi-stage batch processes with limited resources by developing an indirect sequence-based model. Harjunkoski and Grossmann (2002) developed two additional efficient decomposition strategies for multi-stage scheduling problems to improve computational efficiency. Méndez and Cerdá (2003) considered more than one clusters producing the same resources as well as unit-dependent resources. Gupta and Karimi (2003a) developed an improved direct sequence-based mathematical model for scheduling of multi-stage batch processes, which outperforms the models of Pinto and Grossmann (1995), Hui and Gupta (2000) and Hui *et al.* (2000), by examining multiple different unit assignment constraints. Gupta and Karimi (2003b) developed a two-step method for solving the batching and the scheduling problem, where the first step, determines the optimal number and size of batches, while the second stage sequences the operations into units. Castro and Grossmann (2005) extended the studies presented in Castro and Grossmann (2006) for the multi-stage problem, while Liu and Karimi (2007) examined multiple variations of models based on slot-based representation for the multi-stage problem. Sundaramoorthy and Maravelias (2008) solved the batching and scheduling problem simultaneously by developing an indirect sequence-based mathematical model. Sundaramoorthy *et al.* (2009) considered the simultaneous batching and scheduling of multi-stage batch processes with limited resources by proposing a uniform discrete-time representation. Fumero *et al.* (2012) developed a slot-based mathematical model for multi-stage batch plants operating in campaign mode. They also presented a simplified model where they define the maximum number of slots postulated in each unit and used several preordering constraints. The same research group also proposed an optimization framework for multiple (multisite) multi-stage batch process facilities (Ackermann *et al.* 2018) Finally, Novara *et al.* (2016), developed an efficient constraint programming model for multi-stage batch plants, by considering limited resources and campaign mode operation.

2.6 Scheduling of multitasking batch processes

In contrast to the scheduling of multipurpose and multi-stage batch processes, scheduling of multitasking batch process has not gathered adequate attention. Only a few models, developed in the past five years, consider this problem. Patil *et al.* (2015) were the first to propose a mathematical model for scheduling of multitasking batch processes. They developed a model based on uniform discrete-time representation, and they solved several examples from scientific service facilities. Lagzi *et al.* (2017a), solved the same problem by using a process-slot formulation to generate the optimal solution. Lagzi *et al.* (2017b) developed a non-uniform discrete-time model. They also performed comparative studies, where they concluded that the non-uniform discrete-time model is the most efficient. Santos *et al.* (2018) extended the non-uniform discrete-time model of Lagzi *et al.* (2017b) to consider personnel allocation for multitasking environments. Finally, Lee *et al.* (2019) investigated the multitasking problem of conflicting objectives for the same problem. Despite those attempts, it still seems that excessive computational time is required to generate a schedule for multitasking batch processes, which makes it intractable to use such models

2.7 Scheduling of job-shops

The job-shop scheduling problem has gathered significant attention since the late 50s. Even though the first attempts to solve this problem used mathematical modelling programming approaches (Bowman 1959; Manne 1960; Greenberg 1966), the complexity of the problem and the small computational power of computing machines during this period made it impossible to solve this problem using such models. Instead, to solve this NP-hard problem, researchers developed various dispatching rules. During the next two decades, a great variety of such dispatching rules proposed and examined in existing job-shop scheduling problems. Panwalkar and Iskander (1977) reviewed and analyzed all these dispatching rules.

Dispatching rules, even though they can generate a sequence of jobs into units fast, they can only generate a feasible schedule. As a result, later attempts focused on using enumeration procedures as well as branch and bound methods. Balas (1969) developed an implicit enumeration technique for the first time. In this work, he randomly generated an initial solution (root node), while for the next solution, he examined whether reversing an arc can lead to a better solution. They were multiple works that followed the work of Balas (1969) to develop more efficient enumeration techniques. Schrage (1970) presented five different cases where he proved that the schedule does not improve, even if there is any change in the sequence. As a result, Schrage showed there is no need to examine all cases to prove optimality. Florian *et al.* (1971), used the disjunctive graph to generate the nodes of each branch of their approach. More specifically, the root node contains all the disjunctive arcs, while for the next nodes, they removed those arcs one by one until the schedule is feasible. Ashour and Hiremath (1973) also propose a branch and bound method. This method assigns all operations of a job to the available units in the root node, without considering any other jobs. In the next nodes, this approach refines the schedule, which violates the allocation constraints by modifying the timing and sequence of the conflicting operations that lead to the best solution. Fisher *et al.* (1983) presented two mathematical models with surrogate duality relaxation in the capacity and precedence constraints. Barker and McMahon (1985) developed a similar methodology with Balas (1969). The authors used different rearrangement techniques to improve the efficiency of their approach. Adams *et al.* (1988) used an approximation method to solve the job-shop scheduling problem.

Heuristic and metaheuristic methods were also applied to the job-shop scheduling problem, the use of which led to more efficient approaches to solve this problem. Applegate and Cook (1991) combined the branch and bound algorithm with a new heuristic method. They also introduced several branch cuts to obtain better bounds. Falkenauer and Bouffouix (1991) proposed a genetic algorithm to generate feasible solutions for the job-shop scheduling problem, while Laarhoven *et al.* (1992) solved multiple small examples by using a simulating annealing algorithm. For the same problem, Dell'Amico and Trubian (1993) presented a tabu search algorithm, while Colorni *et al.* (1994) developed an ant colony approach. Park *et al.* (2003) developed a hybrid genetic algorithm. Watanabe *et al.* (2005) also developed a genetic algorithm, which they combined with an approach that can find an area with a high probability of containing higher quality solutions. Finally, Sha and Hsu (2006) examined the particle swarm optimization algorithm for developing schedules for job-shop problems.

The flexible job-shop scheduling problem has also gathered significant attention. Wagner (1959) proposed a mixed-integer mathematical model for this problem. However, similar to the job-shop scheduling problem, it was impossible to solve such mathematical models. Additionally, in the flexible job-shop scheduling problem except for the sequence, the assignment of tasks to units should also be determined, which increases the difficulty. As a result, the first approaches attempted to solve the assignment and sequencing problem independently. Brandimarte (1993) developed a two-level tabu search algorithm, where the first level determines the assignment of tasks into units. Based on this assignment, the second level examines the best sequencing of operations in all machines. Paulli (1995) also used a hierarchical algorithm to solve this problem. More specifically, the algorithm first develops a feasible schedule by using several dispatching rules, while in the next step, the approach reassigns the operations into units. Hussain and Joshi (1998) developed a genetic algorithm to define the assignment problem and an NLP mathematical model to find the best sequence.

Later attempts focused on solving the assignment and sequencing problem simultaneously. Using genetic algorithms to determine both cases in the flexible job-shop scheduling problem gained significant attention during the past decades. To develop an efficient genetic algorithm approach, the research community examined multiple different aspects, including encoding, initialization of the population, decoding, selection, crossover and mutation. Mesghouni *et al.* (1997) was the first work to propose the parallel job representation for the chromosome. They presented a simple genetic algorithm, where it randomly initializes the population. According to the selection methodology used, the fittest individuals have a polynomial increase from generation to generation. Lee *et al.* (1998) presented a similar genetic algorithm for solving the same problem. Chen *et al.* (1999) used two chromosomes to represent the assignment and the sequence of operations into machines, respectively. This algorithm also examines if the given chromosomes can generate a feasible solution, while an order-preserving crossover also ensures that the new chromosomes will not lead to an infeasible schedule. Kacem *et al.* (2002) used the parallel job representation for the chromosomes, similar to Mesghouni *et al.* (1997). The authors also proposed an assignment algorithm that investigates the assignments that will fail to generate the optimal solution (forbidden assignments) and the ones that will lead to the optimal solution (obligatory assignments). Jia *et al.* (2003) developed a genetic algorithm for scheduling of the distributed flexible job-shop problem. They proposed an appropriate encoding, which includes both the information for jobs and facilities. Ho and Tay (2004) used composite dispatching rules to initialize the population of their genetic algorithm approach. They also used a new chromosome representation which consists of two parts; the operation order part and the selection machine part. Tay and Wibowo (2004) combined the approaches of Chen *et al.* (1999) and Ho and Tay (2004). They also used two parts to denote the assignment and sequence of operations to machines. Chan *et al.* (2006) developed an improved genetic algorithm for the distributed flexible job-shop problem. Pezzella *et al.* (2008) examined three new chromosome selection methods in the genetic algorithm of Kacem *et al.* (2002). Gao *et al.* (2008) combined the genetic algorithm with a variable neighbourhood search algorithm to reduce the generations required for the genetic algorithm to terminate. Giovani and Pezella (2010) also developed an improved genetic algorithm for the distributed and flexible job-shop scheduling problem. Zhang *et al.* (2011) used a similar chromosome representation with Chen *et al.* (1999). They also developed two algorithms to generate the initial population. They also examined different existing crossover and mutation methods to create new chromosomes (Watanabe *et al.* 2005; Gao *et al.* 2008; Lee *et al.* 1998). Al-Hinai and ElMekkawy (2011) used an operation-based representation for the chromosomes. To improve the efficiency of their approach, they combined the genetic algorithm with a local search algorithm. According to their method, local search slightly modifies the solution generated by a chromosome to examine if the new assignment leads to a better solution. The local search algorithm only analyzes chromosomes after several generations.

Using different metaheuristics for effectively solving the job-shop scheduling problem was also examined in the past three decades. For instance, Hurink *et al.* (1994), Mastrolilli *et al.* (2000), Saidi-Mehrabad and Fattahi (2007), Fattahi *et al.* (2007) and Liouane *et al.* (2007) developed tabu search algorithms to solve this problem. Bagheri *et al.* (2010) and Roshainaei *et al.* (2013) used artificial immune algorithm and hybrid artificial immune algorithm and simulated annealing respectively to solve this scheduling problem. Apart from the tabu search algorithm, Liouane *et al.* (2007) also developed an ant colony algorithm, while Fattahi *et al.* (2007) proposed a simulating annealing approach. Gao *et al.* (2006) and Gao *et al.* (2008) combined a genetic algorithm with a local search methodology to improve the efficiency of the developed approach. Zhang *et al.* (2009) developed a hybrid tabu search and particle swarm optimization. Finally, Yazdani *et al.* (2010) proposed a variable neighbourhood search algorithm.

Mathematical modelling also gained attention to solve the flexible job-shop problem in the past two decades. Choi and Choi (2002) developed a direct sequence-based mathematical model for flexible job-shop scheduling problem. Fattahi *et al.* (2007) used both unit-based and task-based timing variables to model the problem, while they matched those variables for operation processed in the same processing unit. Özgüven *et al.* (2010) also developed a mathematical for the same problem. Roshainaei *et al.* (2013) presented an improved mathematical model also based on direct sequence-based time approach Finally, Karimi *et al.* (2017) developed a sequence-based mathematical model to solve this problem.

2.7.1 Scheduling of energy-efficient job-shop and flexible job-shops

Despite the interest into the job-shop and the flexible job-shop scheduling problem, it seems that most of these cases consider minimization of makespan, tardiness or cost as objective. On the contrary, limited approaches consider the examination of minimizing energy consumption. May *et al.* (2015) developed a genetic algorithm to solve the multi-objective problem of both minimizing makespan and total energy consumption in a job-shop. Dai *et al.* (2013) developed a hybrid genetic algorithm and simulating annealing approach to solve the same multi-objective problem for the flexible job-shop problem. Zhang *et al.* (2017) proposed two different methods to solve the flexible job-shop scheduling model with minimization of energy consumption as an objective. In the first method, they developed an MINLP mathematical model which they later linearize to a MILP model. In the second method, they generated efficient dispatching rules by using genetic evolutionary programming approach. Wu and Sun (2018) solved the multi-objective problem of minimizing total energy consumption and makespan in a flexible job-shop environment by using a non-dominated sorting genetic algorithm. In their work, they assumed that each processing unit process the available operations in multiple processing times, while the processing units must remain idle for a specified time after before they can switch off. Wang *et al.* (2018) separately solved the assignment and sequencing in a two-stage optimization method. A genetic algorithm performs the sequencing of operations into units, while a hybrid genetic and particle swarm optimization approach is responsible for sequencing operations. In this work, they assumed that a processing unit remains idle if it does not process any operations. Zhang *et al.* (2019) used the non-dominated sorting genetic algorithm to solve the simultaneous assignment and sequencing problem in the flexible job-shop scheduling problem with the multi-objective of minimization of makespan and energy consumption. Finally, Meng *et al.* (2019) developed six mathematical models based on Wanger's modelling approach (Wanger 1959) for scheduling of flexible job-shop with minimization of makespan as objective. A comparative study between those models and the proposed models showed that the model of Zhang *et al.* (2017) is less efficient than the proposed models.

2.8 Scheduling of continuous processes

Developing methods for scheduling of continuous processes have also been considered. While in some cases, researchers developed mathematical models only for this problem, others incorporated both batch and continuous processes in their models. For instance, Schiling and Pantelides (1996) developed their slot-based model to handle both batch and continuous processes. Similarly, Lee *et al.* (2001) also considered continuous processes in their model. On the other hand, Karimi and McDonald (1997) developed a slot-based mathematical model solely for semi-continuous processes. The models proposed also use different time representations, similar to the models for batch process problems. In several cases, the researchers improved their existing mathematical models for scheduling of multipurpose batch processes. For instance, Zhang and Sargent (1998) extended their global event-based model of Zhang and Sargent (1996) to consider continuous processes. Ierapetritou and Floudas (1998b) used their unit-specific event-based formulation to consider continuous and semi-continuous processes. Mockus and Reklaitis (1999 a, b) examined the same problem in their extended global event-based mathematical model (Mockus and Reklaitis 1997), which does not use direct linearization, and as a result, it avoids solving a series of large-scale MILPs. In their model, they also considered the case that the customer demands are not strictly satisfied in the given time, but at a later time (soft due date constraints). Méndez and Cerdá (2002) developed a sequence-based mathematical model for multipurpose facilities with continuous processes. Catro *et al.* (2004) improved the model Castro *et al.* (2001) by using a new set of timing constraints and considering zero-wait policies. In their updated model, they also included continuous processes. Shaik and Floudas (2007) developed an improved mathematical model of Ierapetritou and Floudas (1998b). This model, except for continuous processes, it also considers different storage requirements such as flexible, finite, unlimited and no-intermediate storage policies. Li *et al.* (2010) developed a unit-specific event-based mathematical model for continuous gasoline blending operations. Finally, Li *et al.* (2012) developed a unit-specific event-based model for continuous steel casting.

2.9 Rolling horizon decomposition

Even though the process industry takes scheduling decisions for short periods (one day), in some cases, it also needs to develop a detailed schedule for several weeks. Furthermore, a processing facility may process a significantly high amount of products during the scheduling period. Such large-scale examples are hard to solve, and as a result, it is common to decompose them into smaller subproblems. Decomposing a problem can significantly affect the solution since a unit can only produce a product within a part of the whole scheduling period or to all processing units.

Rolling horizon is a commonly used approach to decompose large-scale problems. This approach divides the scheduling horizon into smaller sub-horizons and determines the materials included in each sub-horizon. The availability of materials and units as well as the due dates significantly affects those decisions. After the rolling horizon successfully divides the problem into smaller sub-problems, a mathematical model determines the best schedule for each sub-problem. More specifically, the model determines and fixes the assignment, allocation and timings of tasks into units at the first sub-problem before solving the next sub-problem. The procedure continues for all sub-problems.

By using the rolling horizon decomposition approach, it is possible to generate near-optimum solutions in significantly less computational time than directly solving the large-scale problem. Several research groups successfully implemented the rolling horizon decomposition in the scheduling of batch and continuous processes. Singer (2001) developed a rolling horizon decomposition approach for scheduling of job-shops. In their model, they divided the scheduling horizon in time windows, and they included each job in a time window using three heuristic rules, based on the total workload, the variation of processing times included in each time window and the overlapping factor. Lin *et al.* (2002) developed a rolling horizon decomposition method for scheduling of multipurpose batch processes. They divided the scheduling horizon into days and the mathematical model proposed defines how many days each subproblem includes. They also extended their algorithm to take into consideration tradeoffs between demand satisfaction, unit utilization and model complexity. Shaik *et al.* (2009) and Li *et al.* (2012) implemented Lin's algorithm in industrial cases for continuous processes with small improvements and modifications. Yan *et al.* (2013) also divided the scheduling horizon in time windows similar to Singer (2001) for the job-shop scheduling problem. An optimization model determines the operations included in each time window in their approach. Finally, Mohammadi and Poursabzi (2014) developed two heuristics for decomposing the job-shop scheduling problem. In their formulation, they divide the scheduling horizon into three parts, where they fix all binary variables in the first part, while they relax them in the third part.

2.10 Summary

As presented in this chapter, there are several process representations to efficient represent different types of processes and scheduling problems (i.e. problems with or without resource constraints). Additionally, the research community proposed multiple timing representations to improve the efficiency of the mathematical modelling approach. Even though numerous mathematical models have been presented in the literature during the past three decades, using such process and time representations, it still seems that the majority of those models fail to generate a feasible solution in reasonable computational time. Furthermore, there is not a general efficient framework that can directly solve all various types of process industry.

For large-scale or computationally expensive problems, there is a common approach to decompose them by implementing a rolling horizon decomposition algorithm. Various rolling horizon decomposition algorithms have been presented in the literature to effectively divide the problem into smaller subproblems, based on the due dates or on fixed time windows. However, it seems that there is not an efficient rolling horizon decomposition approach which can decompose problems that contain orders/products with the same due date for all of them. Additionally, up to this date, it seems that there is any hybrid gene-expression programming and mathematical programming approach to solve large-scale process scheduling problems. Combining those programming approaches could potentially lead to a significant reduction in the computational time required. Such improvement is possible by using effective dispatching rules, generated by gene-expression programming, to define the assignment and sequencing of operation into units and mathematical programming to determine the optimal batching and timing of operations for the given allocation and sequencing.

Chapter 3: A new approach for Scheduling of multipurpose batch processes

3.1 Introduction

Process industry commonly uses batch processes, especially from facilities that produce multiple high-value products. A processing facility prefers batch processes instead of continuous processes when its long-term planning is to develop several high-value materials with distinct differences in properties in small quantities. In most of these cases, a product requires a combination of different materials, and each product may follow a different processing path. Furthermore, facilities should increase their conversion rate of raw materials, and therefore they recycle the unreacted materials back to the upstream process. As already discussed, such facilities are known as multipurpose batch process facilities, the optimal scheduling of which is crucial to ensure the prosperity of the processing facility.

During the past three decades, multiple mathematical models for scheduling of multipurpose batch processes have been presented based on discrete and continuous-time representations including slot-based, global event-based, unit-specific event-based and sequence-based time representations. More and more improved mathematical models attempt to develop an efficient approach to generate optimal solutions. Despite such attention on developing mathematical models for this type of problem, it still seems that the proposed models lead to large model sizes that significantly affect the efficiency of those models. As a result, process industry usually refrains from using such mathematical models. The main reason lays into the fact that all proposed models require an unnecessarily large number of time intervals/time slots/event points to generate the optimal solution, which affects the model size. Such increase in the model size can significantly affect the efficiency of the model even for small examples.

Among existing formulations, models based on unit-specific event-based time representation require the least number of event points than the time intervals/time slots/event points of different time representations to generate the optimal solution. As discussed before, unit-specific event-based models divide the scheduling horizon independently for each unit and, as a result, it is possible for the start time of two processing units during the same event point to differ. Therefore, models based on unit-specific event-based time representations lead to smaller model sizes in most cases. However, similar to models based on different time representations, they do not allow related production and consumption tasks to take place at the same event point. More specifically, a task that consumes a state during event point n can only start after all related production tasks finish at event point (n - 1). In this case, the model requires two event points to generate the optimal solution, while both processing units only process one task. Allowing related production and consumption tasks to take place at the same event point can eliminate the excess event points required.

In this chapter, the effect of allowing related production and consumption tasks to take place at the same event point is examined. First, it is investigated whether all those production and consumption tasks are allowed to take place at the same event point. Based on this analysis, a new definition of recycling tasks is developed. Two unit-specific event-based mathematical models for scheduling of multipurpose batch processes are also developed, where related non-recycling production and consumption tasks are allowed to take place at the same event point. While in the first model uses timing variables based on tasks, similar to the most common unit-specific event-based models in the literature, the second model uses unit-based timing variables. The proposed model can reduce the model sizes and computational time. Therefore, using such an approach in a generic framework for process scheduling (Chapter 4) can be beneficial.

3.2 Research contribution 1

Rakovitis, N., Zhang, N., Li, J. Zhang, L. A new approach for scheduling of multipurpose batch processes with unlimited intermediate storage policy. *Front. Chem. Sci. Eng.* **13,** 784–802 (2019) doi: doi.org/10.1007/s11705-019-1858-4

Blank page

A new approach for scheduling of multipurpose batch processes with unlimited intermediate storage policy

Nikolaos RAKOVITIS,[1] Nan ZHANG,[1] Jie LI(✉),[1] Liping ZHANG[2]

[1]Centre for Process Integration, School of Chemical Engineering and Analytical Science, The University of Manchester, Manchester, M13 9PL, United Kingdom

[2]Department of Industrial Engineering, School of Machinery and Automation, Wuhan University of Science and Technology, Wuhan, Hubei, 430081 P. R China

Received MM DD, 2018; accepted MM DD, 2018

E-mail: jie.li-2@manchester.ac.uk

Abstract

The increasing demand of goods, the high competitiveness in the global marketplace as well as the need to minimize the ecological footprint lead multipurpose batch process industries to seek ways to maximize their productivity with a simultaneous reduction of raw materials and utility consumption and efficient use of processing units. Optimal scheduling of their processes can lead facilities towards this direction. Although a great number of mathematical models have been developed for such scheduling, they may still lead to large model sizes and computational time. In this work, we develop two novel mathematical models using the unit-specific event-based modelling approach in which consumption and production tasks related to the same states are allowed to take place at the same event points. The computational results demonstrate that both proposed mathematical models reduce the number of event points required. The proposed unit-specific event-based model is the most efficient since it both requires a smaller number of event points and significantly less computational time in most cases, especially for those examples which are computationally expensive from existing models.

Keywords Scheduling, multipurpose batch processes, simultaneous transfer, mixed-integer linear programming

1 Introduction

Nowadays, it is more important than ever for the multipurpose batch process industry to maximize their productivity by simultaneously minimize their costs, fuel and raw material consumption and ecological footprint to be able to survive in a highly competitive market. Developing optimal schedules is one of the main tools that multipurpose batch process industry can utilize to optimize their processes. Although heuristics-based and spreadsheet-based methods are often used to generate schedules, they are restricted to simple batch processes and often produce suboptimal schedules. Mathematical programming especially mixed-integer programming approaches have been received much attention in the past three decades because they can be used for more complicated batch processes and often provide optimal schedules. Before developing mathematical models, it is crucial to well represent the multipurpose batch process. Two representations have been proposed including state-task network and resource-task network representations. The state-task network representation (STN) is proposed by Kondili et al. [1], in which all materials in the process are represented by states and processing operations in units are treated as tasks. While states are represented with circles (state nodes), tasks are depicted with rectangles (task nodes). The connections between states and task nodes are depicted with arrows. No resources such as processing units, storage tanks, utilities and manpower are demonstrated in the STN representation. Therefore, the resource-task network representation (RTN) is proposed by Pantelides [2], in which resources used by tasks are explicitly included.

Based on the STN and RTN representations, several modelling approaches have been proposed for optimal scheduling of multipurpose batch processes resulting in a great number of mathematical models in the last three decades [3-7]. These modelling approaches include discrete-time [1, 8, 9], and continuous-time modelling approaches. The continuous-time modelling approaches include slot-based [10]-[12], global event-based [13-15], unit-specific event-based [16-19] and sequence-based modelling approaches [20-22]. The slot-based modelling approaches can be further classified into process-slot [11, 12] and unit-slot [12, 23] modelling approaches.

In the discrete-time modelling approach, the scheduling horizon is divided into time intervals of uniform or non-uniform lengths, where the start and end times of each interval are known, and batches, tasks, or activities are assigned to intervals. Mathematical models developed using this modelling approach are often simple, and they usually lead to tight mixed-integer linear programming (MILP) relaxation. A batch, task or activity should start or end exactly at the time interval points. The model sizes largely depend on the number of time intervals required. A great number of time intervals are often required to generate exact solutions, leading to computationally intractable model sizes even for small-scale problems since the length of each time interval is equal to the greatest common factor of the processing times of all units. To avoid an intractable number of time intervals required, continuous-time modelling approaches have been proposed in which the scheduling horizon is divided into ordered slots or event points with non-uniform unknown lengths. Batches, tasks, or activities are assigned to slots or event points. A batch, task or activity should start or end exactly at the slot points or event points. The model sizes also largely depend on the number of slots or event points required. The continuous-time modelling approaches require a significantly smaller number of time slots or event points. However, they often lead to worse MILP relaxation than the discrete-time modelling approach mainly since they have to introduce several big-M terms in sequencing constraints. In the process slot-based and global event-based continuous-time modelling approaches, time slots or event points are common or shared for all processing units in the process. In other words, batches, tasks or activities in all processing units must start or end at the same slots or event points. In the unit-specific event-based and unit-slot modelling approaches, each unit has independent or separate time slots or event points. The same time slots or event points for different units can start or end at different times. Therefore, the unit-specific event-based or unit-slot modelling approaches often require a smaller number of slots or event points compared to the process slot-based and global event-based modelling approaches, leading to smaller model size and less computational time in general. While the unit-specific event-based modelling approach divides the scheduling horizon using event points where the next event point is not necessarily immediately start after its previous event point end, the unit-slot modelling approach divides the scheduling horizon based on slots where the next slot must immediately start after the end of its previous slot. In general, the unit-slot modelling approach is very similar to the unit-specific event-based modelling approach. Finally, the sequence-based modelling approach employs direct (immediate) or indirect (general) sequencing (precedence) of task-pairs on units to define a schedule. Time is not explicitly modelled in terms of slots or event points. Although it is not necessary for the sequence-based modelling approach to postulate the numbers of slots or event points a priori, they must postulate the number of batches, tasks or activities a priori. They also do suffer from the difficulty in monitoring resource levels.

The capabilities of the unit-specific event-based modelling approach have been well established in the literature [17, 24-25] with a fewer number of event points and smaller model size, which often lead to smaller computational expenses. In most mathematical models developed using the unit-specific event-based modelling approach, the timing variables are defined based on tasks, not on units. In other words, the independent or separate event points are used for tasks, not for units. Therefore, we call them as task-specific event-based models in this work. Most of these task-specific event-based models still require a high number of event points to generate optimal schedules, leading to large model sizes and computational time. This is because most of these models do not allow consumption and production tasks related to the same states to take place at the same event points, unnecessarily increasing the number of event points required. Recently, Shaik and Vooradi [26] proposed a task-specific event-based model for scheduling of multipurpose batch processes, allowing production and consumption tasks related to the same states to take place at the same event points. However, their model is only applicable to the batch process without any recycling loop.

In this work, we develop two novel mathematical models using the unit-specific event-based modelling approach in which related consumption and production tasks are allowed to take place at the same event points. While we define timing variables based on units in one model (called unit-specific event-based model), the timing variables are defined based on tasks in the other model, (called task-specific event-based model). Both formulations are developed based on the STN representation. To make our models applicable for any batch processes, we introduce a definition of recycling tasks slightly different than the definition of recycling tasks of Li et al. [5]. We only allow non-recycling production and consumption tasks to take place at the same event points to avoid suboptimality. The computational results demonstrate that both proposed mathematical models are very general and can be applied for all batch processes even those with recycling loop and reduce the number of event points required. The proposed unit-specific event-based model is the most efficient since it both requires a smaller number of event points and significantly less computational time in most cases, especially for those examples which are computationally expensive from existing models.

2 Problem description

A general multipurpose batch process facility including J ($j = 1, 2, \ldots, J$) processing units such as reactors, separators and heaters. The STN representation of a multipurpose batch process facility is presented in Figure 1. These units are used to produce P ($p = 1, 2, \ldots, P$) final products using F ($f = 1, 2, \ldots, F$) feeds. I ($i = 1, 2, \ldots, I$) tasks will be processed in the processing units. Each processing unit can process $\mathbf{I}_j$ tasks. At each time, at most one task can be processed in a processing unit. Besides final products, intermediate states are also produced. There are total S ($s = 1, 2, \ldots, S$) states including feeds, intermediate states, and final products. The feeds are denoted as $\mathbf{S}^{R}$, the intermediate states are denoted as $\mathbf{S}^{IN}$, and the final products are included in the set of $\mathbf{S}^{FP}$. The proportion of each state *s* produced or consumed by a task *i* in a unit *j* is denoted by $\rho_{i,j,s}$. While positive values of $\rho_{i,j,s}$ denote production of state *s* during the processing of task *i* in unit *j*, negative values of $\rho_{i,j,s}$ denote consumption of state *s* during the processing of task *i* in unit *j*. After production, each batch is allowed to be mixed with other batches or split into several batches for further processing. Some intermediate states are also allowed to be recycled back if necessary. Each intermediate state has its dedicated storage. If the storage capacity for an intermediate state is unlimited, then it is called unlimited intermediate storage (UIS) policy. If the storage capacity is limited or finite, then it is called finite intermediate storage (FIS) policy. If there is no intermediate storage, then it is called no intermediate storage (NIS) policy. In this paper, we assume UIS for all states, including intermediate states, feeds and final products. After production in a processing unit, an intermediate state may or may not be allowed to remain in this processing unit. If an intermediate state has to be transferred immediately to storage or other processing units after production, it is called zero wait (ZW) policy. If an intermediate state is allowed to remain in a processing unit with unlimited time, then it is called unlimited wait (UW) policy. If an intermediate state is allowed to be held in a processing unit with a certain time, then it is called limited wait (LW) policy. In this paper, we also assume UW policy for all intermediate states. By introducing this, the scheduling problem can be stated as follows,

Given:

1) STN representation of a multipurpose batch facility;
2) J units, unit capacities, suitable tasks and their processing times;
3) S states, the portion of states produced or consumed from a task in a processing unit;
4) Product prices;
5) Scheduling horizon.

Determine:

1) Optimal production schedule involving task allocations, start and end timings, sequences and batch sizes;

2) Inventory profiles.

Operating rules:

1) At most one task can be processed in a processing unit at any time;
2) Batch mixing and splitting is allowed.

Assumptions:

1) All parameters are deterministic;
2) The processing time of a task in a processing unit depends on a fixed processing time (denoted as α_{ij} plus a variable process time based on the batch sizes, which is denoted as $(\beta_{ij} \cdot b_{ij})$;
3) Unlimited feed materials are available;
4) Unlimited storage policy for all states;
5) Unlimited resources where required are available;
6) Unlimited wait policy for intermediate states.

The objective is to maximize productivity or minimize makespan. The makespan is defined as the time required to produce a specified demand.

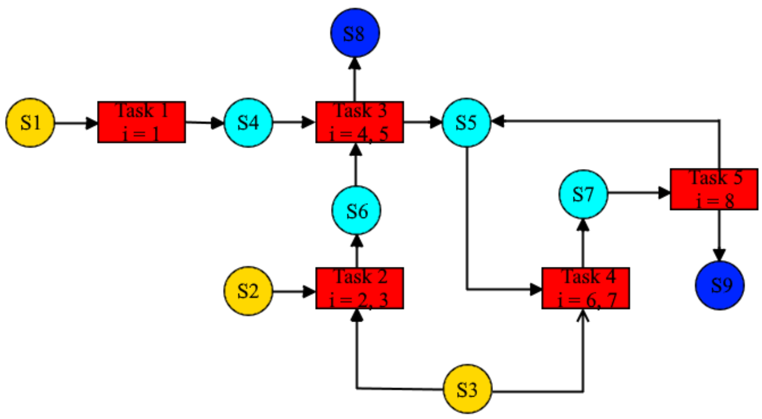

Fig. 1 STN representation of a multipurpose batch process facility (Example 2)

3 Motivating example

Let consider an example whose STN representation is depicted in Figure 2. In this example, a raw material S1 is converted into a final product S3 through two tasks (i.e., I1 and I2) in two processing units (J1 and J2). The scheduling horizon is 9 h. The objective is to maximize the productivity of product S3. All relevant data for this example are given in Table 1. We use the mathematical model of Shaik and Floudas [25] to solve this example. We obtain the optimal solution of 500.00 cu using 2 events. The optimal schedule is illustrated in Figure 3. As seen from Figure 3, the task I1, processed in unit J1, produces 100 cu of S2 at event point N1, which is further processed at task I2 in unit J2 to produce final product S3 with 100 cu at event point N2. This is because the task I2 in unit J2 is a consuming task of S2 and the task I1 in unit J1 is a production task for S2. Therefore, the task I2 must always start at event point N2 since the production task I1 take place at event point N1 based on the model of Shaik and Floudas [25]. However, we can use one event point for the optimal schedule through analysis. Figure 4 illustrates the optimal schedule with only one event point. From Figure 4, it can be observed that the consuming task I2 takes place at the same event point as the production task I1, but not in real time. In real time, task I2 still takes place after I1 is completed. By doing this, we can reduce one event point required for generating the optimal solution. As discussed previously, the model size and computational performance largely depend on the number of event points required. This motivates us to develop new mathematical formulations for scheduling of multipurpose batch facilities by allowing consuming and production tasks related to the same states take place at the same event points to reduce the number of event points required.

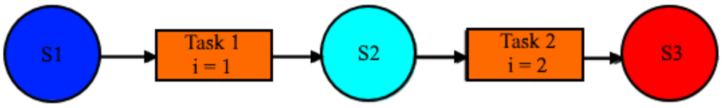

Fig. 2 STN representation of the motivating example

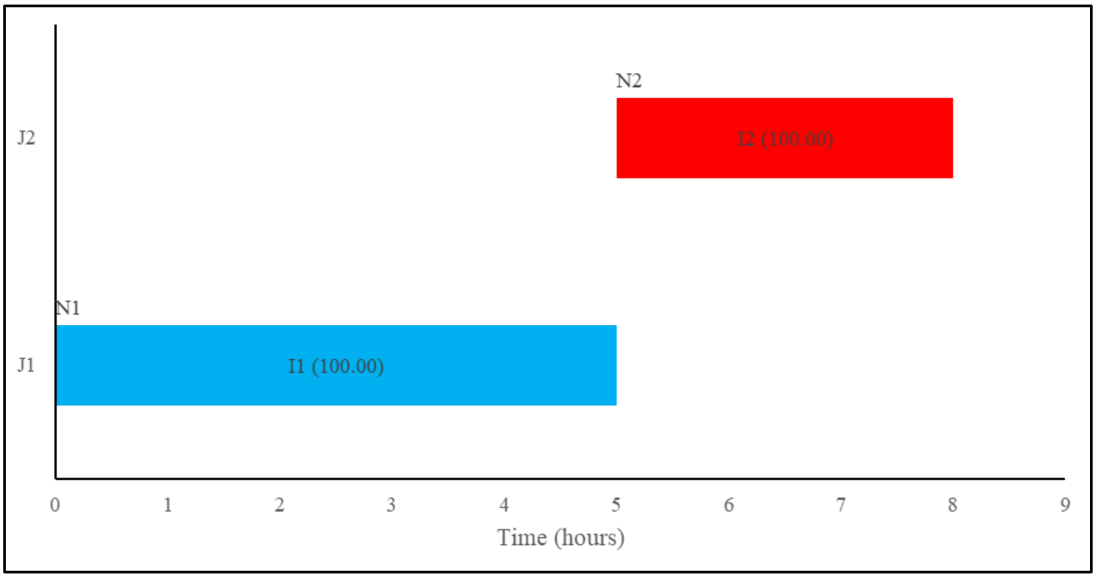

Fig. 3 Optimal schedule for the motivating example using two event points from the model of Shaik and Floudas [25]

Table 1 Data for motivating example

Unit	Maximum capacity (mu)	Minimum capacity (mu)	α_i (h)	β_i (h)
J1	100	0	3	0.02
J2	100	0	2	0.01

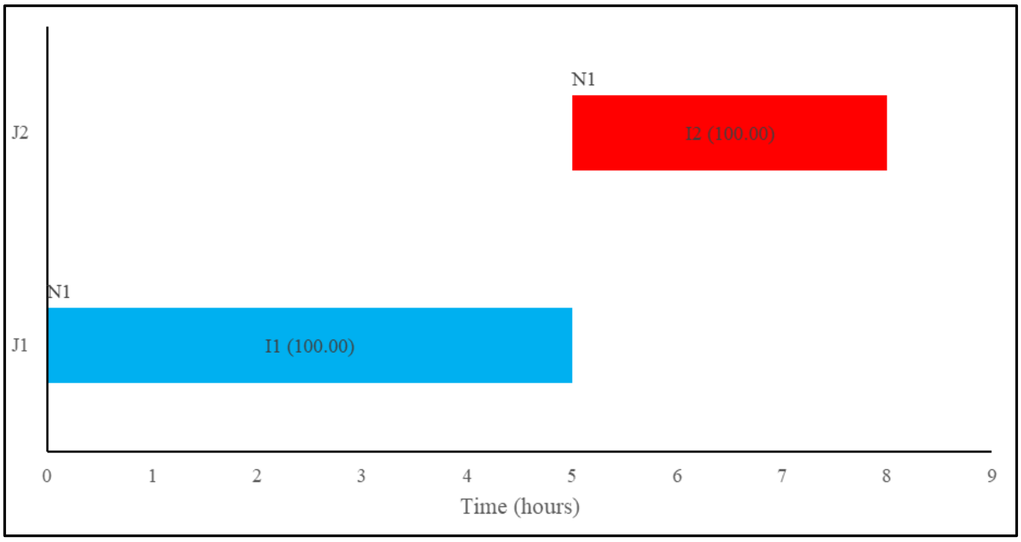

Fig. 4 Optimal schedule for the motivating example using one event point

4 Definition of recycling tasks

Despite the fact that allowing all related production and consumption tasks take place at the same event points can potentially reduce the number of event points and increase computational efficiency, we could obtain suboptimal solutions in some cases by allowing all production and consumption tasks related to the same states to take place at the same event points. Consider the following example, which is depicted in Figure 5. If production and consumption tasks related to the same states are not allowed to take place at the same event points, the optimal productivity of 1656 cu is generated with four event points from the model of Shaik and Floudas [25]. However, if all production and consumption tasks related to the same states are allowed to take place at the same event points, then the suboptimum productivity of 1511 cu is generated. This occurs from tasks I3, I4 and I5. Note that tasks I4 and I5 produce two states S2 and S4 and task I3 consumes state S2 and produces state S3. If these tasks (i.e., tasks I3, I4, and I5) are allowed to take place at the same event points, it is not possible for these tasks to take place at the same time in real time, which leads to suboptimum solutions.

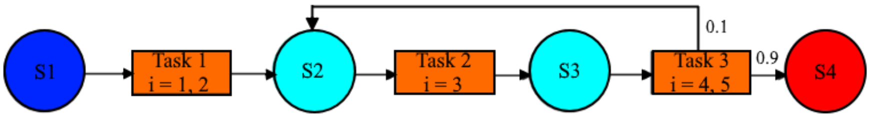

Fig. 5 STN representation of motivating example 2

Table 2 Results for motivating example 2

Example	Model	Event points	CPU time (s)	RMILP (cu)	MILP (cu)	Discrete Variables	Continuous Variables	Equations
1	SF	4	0.094	1800.00	1656.16	20	78	115
(H=8h)	T-S	4	0.156	3300.83	1511.66	20	78	121

SF: the model of Shaik and Floudas [25]. T-S: the revised model of Shaik and Floudas [25] allowing all production and consumption tasks related to the same states take place at the same event points.

To avoid suboptimality in such cases, a new definition slightly different from that of recycling tasks of Li et al [5] is introduced. We define a recycling task in a processing unit if it produces a state that can be consumed either by a task in its upstream processing units or by other tasks in the same processing unit. The recycling tasks are included in the set $\mathbf{I}^R$. In Figure 6, there are four tasks (I1-I4), two processing units (J1-J2) and three states (S1-S3). While tasks I1 and I3 can be processed in unit J1, tasks I2 and I4 can be processed in unit J2. Tasks I1 and I2 consume S1 and produce S2, whilst tasks I3 and I4 consume S2 and produce S1 and S3. Based on this new definition, task I1 is considered as a recycling task because it produces S2 that can be used by task I3 as a raw material in the same unit (i.e., J1). Similarly, tasks I2-I4 are also recycling tasks. Consequently, all tasks in the example depicted in Figure 6 are considered as recycling tasks.

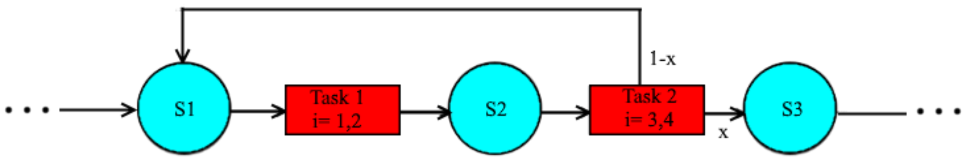

Fig. 6 Illustration of recycling tasks where all tasks are recycling tasks

5 Mathematical formulation

5.1 Time representation

As discussed before, the advantages of the unit-specific event-based modelling approach have been well established in the literature. This modelling approach is used to develop our new models, which are presented below.

5.2 Model M1

In this model, the timing variables are defined based on units. Therefore, this proposed model is called the unit-specific event-based model.

5.2.1 Allocation constraints

To assign tasks to units, we define binary variables $w_{i,j,n,n'}$ to denote if a task *i* is processed in a unit *j* from event point *n* to event point *n′*. We allow a task in a unit to span over Δ*n* event points to make the model general where Δ*n* is a parameter that could be used to control the number of event points that a task can span across. At most one task is allowed to take place in a unit during a time as specified by constraint (1). If tasks are allowed to span over more than one events (Δ*n* > 0), constraint (1) allows at most one task to be active from event point *n* to event point *n′*.

$$\sum_{i \in \mathbf{I}_j} \sum_{n-\Delta n \le n' \le n} \sum_{n \le n'' \le n'+\Delta n} w_{i,j,n',n''} \le 1 \qquad \forall j, n \qquad (1)$$

5.2.2 Capacity constraints

We define variables $b_{i,j,n,n'}$ to denote the amount of materials (i.e., batch size) processed by a task *i* in a unit *j* from event point *n* to event point *n′*. If a unit *j* processes a task *i* from event point *n* to event point *n′*, then the material processed in this unit should be constrained by the minimum ($B_{i,j}^{min}$) and maximum ($B_{i,j}^{max}$) capacity limits.

$$B_{i,j}^{min} w_{i,j,n,n'} \le b_{i,j,n,n'} \le B_{i,j}^{max} w_{i,j,n,n'} \qquad \forall j,\ i \in \mathbf{I}_j,\ n \le n' \le n+\Delta n \qquad (2)$$

5.2.3 Material balance

We define $ST_{s,n}$ to denote the amount of material *s* at event point *n*, which is used to monitor inventory of the materials in storage and ensure no storage capacity violation. Since we allow non-recycling tasks to take place at the same event points as the related consumption tasks, these non-recycling tasks produce materials at event point *n*. However, recycling tasks have to produce materials at event point (*n*-1). With this, the amount of material in a storage at an event point *n* should be equal to the amount of materials at the previous event point (*n*–1) plus the material produced from non-recycling tasks at event point n and recycling tasks at event point (n–1) minus the material consumed by consumption tasks at event point n, as indicated in constraints (3) and (4).

$$ST_{s,n} = ST0_s + \sum_{j} \sum_{i \in (\boldsymbol{I}_j \cap \boldsymbol{I}_S^P), i \notin \boldsymbol{I}^R} \rho_{i,j,s} \sum_{n-\Delta n \le n' \le n} b_{i,j,n',n} +$$

$$+ \sum_{j} \sum_{i \in (\boldsymbol{I}_j \cap \boldsymbol{I}_S^C)} \rho_{i,j,s} \sum_{n \le n' \le n+\Delta n} b_{i,j,n,n'}$$

$$\forall s,\ n = 1 \quad (3)$$

$$ST_{s,n} = ST_{s,n-1} + \sum_{j} \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_S^P), i \notin \mathbf{I}^R} \rho_{i,j,s} \sum_{n-\Delta n \le n' \le n} b_{i,j,n',n} +$$

$$+ \sum_{j} \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_S^P \cap \mathbf{I}^R)} \rho_{i,j,s} \sum_{n-1-\Delta n \le n' \le n-1} b_{i,j,n',n-1} + \sum_{j} \sum_{i \in (\boldsymbol{I}_j \cap \boldsymbol{I}_S^C)} \rho_{i,j,s} \sum_{n \le n' \le n+\Delta n} b_{i,j,n,n'}$$

$$\forall s,\ n > 1 \quad (4)$$

5.2.4 Processing duration constraints

Once a batch is processed on a unit, then it must be processed for some duration. A unit is also allowed to be idle after processing. We define $T_{j,n}^{s}$ and $T_{j,n}^{\mathrm{f}}$ to denote the start and end times of a processing unit j at event point n. The end time of a unit j at event n must be greater than the total processing time, consisting of a fixed term and a variable term depending on the batch size, as indicated in the constraint (5).

$$T_{j,n}^{\mathrm{f}} \ge T_{j,n}^{\mathrm{s}} + \sum_{i \in \mathbf{I}_j} \sum_{n \le n' \le n+\Delta n} \left(\alpha_{ij} \cdot w_{i,j,n,n'} + \beta_{ij} \cdot b_{i,j,n,n'} \right)$$

$$\forall j,\ i \in \mathbf{I}_j,\ n \le n' \le n+\Delta n \qquad (5)$$

Note that we do not force the finish time to be equal to the start time plus the total processing time to allow materials produced temporally stored in unit or a unit to be idle after processing, which may lead to a smaller number of event points that are required to generate the optimum solution as claimed by Li and Floudas [17].

5.2.5 Sequencing constraints

Same or different tasks in the same unit

An event point *n* on a processing unit *j* must always start after its previous event point on the same unit finishes.

$$T^{\mathrm{s}}_{j,n+1} \geq T^{\mathrm{f}}_{j,n} \qquad \forall j,\, n < N \qquad (6)$$

Different tasks in different units

We need to sequence consuming and production tasks related to the same states where the consuming and production tasks are different tasks in different units. Although a consuming task is allowed to take place at the same event points with its related production tasks which are non-recycling tasks, this consuming task must always start after its related production tasks finish in real time. We introduce a new continuous variable $T_{s,n}$, which denotes the time that state *s* is available to be consumed at event point *n*. Then we have:

$$T_{s,n} \leq T_{s,n+1} \qquad \forall s \in \mathbf{S}^{IN},\, n \qquad (7)$$

The finish time of unit *j*, which is related with the production of state *s* should be before the time that state *s* is available.

$$T_{s,n} \geq T^{\mathrm{f}}_{j,n} - M\left(1 - \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}^P_S)} \sum_{n-\Delta n \leq n' \leq n} w_{i,j,n',n}\right) \qquad \forall s \in \mathbf{S}^{IN},\, j,\, \sum_{i \in \mathbf{I}_j} \rho_{s,i} > 0,\, n \qquad (8)$$

The start time of unit *j* at event point *n*, which is related with the consumption of state *s* should be after the time that the state is available if it was produced by a non-recycling task *i′*.

$$T_{s,n} \leq T^{s}_{j,n} + M \cdot \left(1 - \sum_{i\in(\mathbf{I}_j\cap\mathbf{I}_S^C)} \sum_{n\leq n'\leq n+\Delta n} w_{i,j,n,n'}\right)$$

$$\forall s\in \mathbf{S}^{IN}, j, \sum_{j'} \sum_{i'\in(\mathbf{I}_{j'}\cap\mathbf{I}_S^P), i\notin\mathbf{I}^R} \rho_{s,i'} > 0, n \qquad (9)$$

If state *s* is produced by a recycling task *i′*, the end time of unit *j* at event point *n*+1, which is related with the consumption of state *s* should be after the time that the state is available instead.

$$T_{s,n} \leq T^{s}_{j,n+1} + M \cdot \left(1 - \sum_{i\in(\mathbf{I}_j\cap\mathbf{I}_S^C)} \sum_{n+1\leq n'\leq n+1+\Delta n} w_{i,j,n+1,n'}\right)$$

$$\forall s\in \mathbf{S}^{IN}, j, \sum_{j'} \sum_{i'\in(\mathbf{I}_{j'}\cap\mathbf{I}_S^P\cap\mathbf{I}^R)} \rho_{s,i'} > 0, n \qquad (10)$$

5.2.6 Objectives

We consider two different objectives. In the first objective, the productivity of a given facility is maximized for a specified scheduling horizon.

$$z = \sum_{s} p_s \sum_{j} \sum_{i\in(\mathbf{I}_j\cap\mathbf{I}_S^P)} \sum_{n} \sum_{n\leq n'\leq n+\Delta n} \rho_{i,j,s} \cdot b_{i,j,n,n'} \qquad (11)$$

The other objective is to minimize makespan (denoted as *MS*), which is considered as following,

$$MS \geq T^{\mathrm{f}}_{j,n}$$

$$\forall j, n = N \qquad (12)$$

In the minimization of makespan problem, it should be also ensured that the total demand is satisfied.

$$ST_{s,n} + \sum_{j} \sum_{i\in(I_S^P\cap I^R)} \rho_{i,j,s} \sum_{n-\Delta n\leq n'\leq n} b_{i,j,n',n} \geq D_s$$

$$\forall s\in\mathbf{S}^{\mathrm{p}}, n=N \qquad (13)$$

We complete our model **M1** which comprises of eqs. (1) – (11) if maximization of productivity is considered as objective and eqs. (1)-(10), (12), (13) if minimization of makespan is considered.

5.3 Model M2

In this model, the timing variables are defined based on tasks. The same tasks that can be processed in different units have to be divided into two different tasks. We call this model task-specific event-based model. Production and consumption tasks related to the same states are also allowed to take place at the same event points in this model.

5.3.1 Allocation constraints

Similar to the model **M1**, at most one task is allowed to take place at each event point.

$$\sum_{i \in \mathbf{I}_j} \sum_{n-\Delta n \le n' \le n} \sum_{n \le n'' \le n'+\Delta n} w_{i,n',n''} \le 1$$

$$\forall j, n \qquad (14)$$

5.3.2 Capacity constraints

The batch size of task *i* from event point *n* to event point *n*′ should be constrained by the maximum and minimum capacities if the task is active. Otherwise, it should be equal to zero. This constraint (15) is the same as that of Shaik and Floudas [25].

$$B_i^{\min} w_{i,n,n'} \le b_{i,n,n'} \le B_i^{\max} w_{i,n,n'}$$

$$\forall j,\ i \in \mathbf{I}_j,\ n \le n' \le n + \Delta n \qquad (15)$$

5.3.3 Material balance constraints

Similar to the model **M1**, we allow materials that are produced by non-recycling tasks to be consumed by their related consumption tasks at the same event point. However, if the materials are produced by recycling tasks, then they have to be consumed by their related consumption tasks at the next event point.

$$ST_{s,n} = ST0_s + \sum_{i \in \mathbf{I}_S^P, i \notin \mathbf{I}^{\mathrm{R}}} \rho_{i,s} \sum_{n-\Delta n \le n' \le n} b_{i,n',n} + \sum_{i \in \mathbf{I}_S^C} \rho_{i,s} \sum_{n \le n' \le n+\Delta n} b_{i,n,n'}$$

$$\forall s,\ n = 1 \qquad (16)$$

$$ST_{s,n} = ST_{s,n-1} + \sum_{i\in\mathbf{I}_S^P, i\notin\mathbf{I}^{\mathrm{R}}} \rho_{i,s} \sum_{n-\Delta n\le n'\le n} b_{i,n',n} + \sum_{i\in(\mathbf{I}_S^P\cap\mathbf{I}^R)} \rho_{i,s} \sum_{n-1-\Delta n\le n'\le n-1} b_{i,n',n-1} +$$

$$+ \sum_{i\in\mathbf{I}_S^C} \rho_{i,s} \sum_{n\le n'\le n+\Delta n} b_{i,n,n'}$$

$\forall s, n > 1$ (17)

5.3.4 Processing duration constraints

The finish time of a task *i* should always be greater than the start time of the same task. It should be also greater than the start time of the task plus the total processing time if the task is active.

$$T^{\mathrm{f}}_{i,n'} \ge T^{\mathrm{s}}_{i,n} + \alpha_i \cdot w_{i,n,n'} + \beta_i \cdot b_{i,n,n'}$$

$\forall\ i\in\mathbf{I}_j, n \le n' \le n+\Delta n$ (18)

5.3.5 Sequencing constraints

Same tasks in the same units

A task *i* taking place at event point $n+1$ should always start after it finishes at event point *n*. This constraint (19) is the same as those of Shaik and Floudas [25].

$$T^{\mathrm{s}}_{i,n+1} \ge T^{\mathrm{f}}_{i,n}$$

$\forall i\in\mathbf{I}_j, n$ (19)

Different tasks in the same units

A task *i* taking place at event point ($n+1$) should always start after all other tasks that can be processed in the same unit finish at event point *n*.

$$T^{\mathrm{s}}_{i,n+1} \ge T^{\mathrm{f}}_{i',n}$$

$\forall j, i\in\mathbf{I}_j, i'\in\mathbf{I}_j, i \ne i', n < N$ (20)

Different tasks in different units

Similar to the model **M1**, two different sets of constraints are introduced based on whether the production task is a recycling or a non-recycling task.

$$T_{i,n}^{\mathrm{s}} \geq T_{i',n}^{\mathrm{f}} - M \cdot \left(1 - \sum_{n-\Delta n \leq n' \leq n} w_{i',n',n}\right)$$

$$\forall j \neq j', i \in \left(\mathbf{I}_S^C \cap \mathbf{I}_j\right), i' \in \left(\mathbf{I}_S^P \cap \mathbf{I}_{j'}\right), i' \notin \mathbf{I}^R, n \qquad (21)$$

$$T_{i,n+1}^{\mathrm{s}} \geq T_{i',n}^{\mathrm{f}} - M \cdot \left(1 - \sum_{n-\Delta n \leq n' \leq n} w_{i',n',n}\right)$$

$$\forall j \neq j', i \in \left(\mathbf{I}_S^C \cap \mathbf{I}_j\right), i' \in \left(\mathbf{I}_S^P \cap \mathbf{I}_{j'} \cap \mathbf{I}^R\right), n < N \qquad (22)$$

5.3.6 Tightening constraint

The duration of all tasks performed in a unit *j* must not exceed the scheduling horizon.

$$\sum_{i \in \mathbf{I}_j} \sum_{n} \sum_{n \leq n' \leq n+\Delta n} \left(\alpha_i \cdot w_{i,n,n'} + \beta_i \cdot b_{i,n,n'}\right) \leq H$$

$$\forall j \qquad (23)$$

5.3.7 Objectives

Similar to the model **M1**, two objectives were also considered in this model **M2**. In the first objective, the productivity of a given facility is maximized for a specified scheduling horizon.

$$z = \sum_{s} p_s \sum_{i \in \mathbf{I}_S^P} \sum_{n} \sum_{n \leq n' \leq n+\Delta n} \rho_{i,s} b_{i,n,n'}$$

$$(24)$$

The second objective is to minimize makespan.

$$MS \geq T_{i,n}^{\mathrm{f}}$$

$$\forall i, n = N \qquad (25)$$

Finally, in the case of minimization of makespan, the total demand should be satisfied.

$$ST_{s,n} + \sum_{i \in (\mathbf{I}_S^P \cap \mathbf{I}^R)} \rho_{i,s} \sum_{n-\Delta n \le n' \le n} b_{i,n',n} \ge D_s$$

$$\forall s \in \mathbf{S}^p,\ n = N \qquad (26)$$

We complete our model **M2** which comprises eq. (14)-(24) if the maximization of productivity is considered as objective and eqs. (14)-(23), (25), (26) if the minimization of makespan is considered as objective.

6. Computational studies

We solve twelve examples to illustrate the capability of the proposed models **M1** and **M2**. The data for all examples are given in Tables 3-14. The STN representation of these examples are illustrated in Figures 1 and 7-16. Note that the STN representation of Example 2 is illustrated in Figure 1. Among these twelve examples, Examples 1-3 and 8-12 are well-established examples from the literature [1, 17, 25]. These twelve examples have varying tasks, units, recipe structures, processing times, and scheduling horizons. All examples are solved to zero optimality gap using CPLEX 12/GAMS 24.6.1. on a desktop computer with Intel® Core™ i5-2500 3.3 GHz and 8 GB RAM running Windows 7. The maximum computational time is set as one hour for all examples.

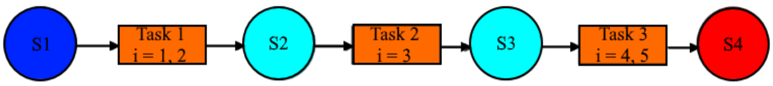

Fig. 7 STN representation of Example 1

Table 3 Data for Example 1

Task	Processing Unit	α_i	β_i	B_i^{min}	B_i^{max}
1	1	1.333	0.01333	0	100
2	2	1.333	0.01333	0	150
3	3	1.000	0.00500	0	200
4	4	0.667	0.00445	0	150
5	5	0.667	0.00445	0	150

Table 4 Data for Example 2

Task	Processing Unit	α_i	β_i	B_i^{min}	B_i^{max}
1	1	0.667	0.00667	0	100
2	2	1.334	0.02664	0	50
3	3	1.334	0.01665	0	80
4	2	1.334	0.02664	0	50
5	3	1.334	0.01665	0	80
6	2	0.667	0.01332	0	50
7	3	0.667	0.008325	0	80
8	4	1.334	0.00666	0	200

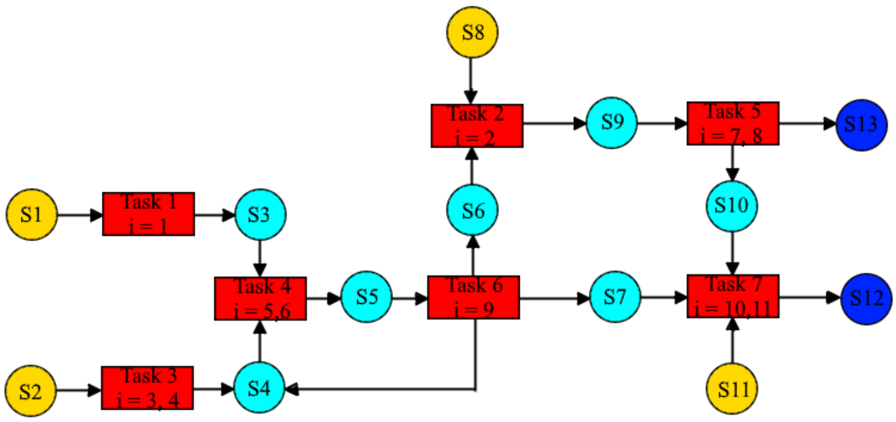

Fig. 8 STN representation of Example 3

Table 5 Data for Example 3

Task	Processing Unit	α_i	β_i	B_i^{min}	B_i^{max}
1	1	0.667	0.00667	0	100
2	1	1.000	0.01000	0	100
3	2	1.333	0.01333	0	100
4	3	1.333	0.00889	0	150
5	2	0.667	0.00667	0	100
6	3	0.667	0.00445	0	150
7	2	1.333	0.01330	0	100
8	3	1.333	0.00889	0	150
9	4	2.000	0.00667	0	300
10	5	1.333	0.00667	20	200
11	6	1.333	0.00667	20	200

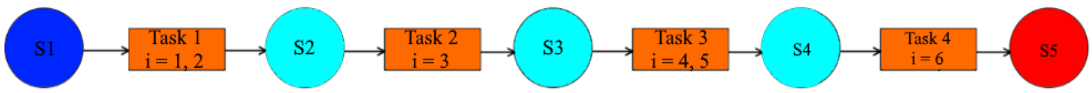

Fig. 9 STN representation of Example 4

Table 6 Data for Example 4

Task	Processing Unit	α_i	β_i	B_i^{min}	B_i^{max}
1	1	1.333	0.01333	0	100
2	2	1.333	0.01333	0	150
3	3	1.000	0.00500	0	200
4	4	0.667	0.00445	0	150
5	5	0.667	0.00445	0	150
6	6	1.000	0.00500	0	200

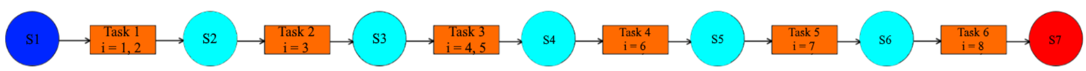

Fig. 10 STN representation of Example 5

Table 7 Data for Example 5

Task	Processing Unit	α_i	β_i	B_i^{min}	B_i^{max}
1	1	1.333	0.01333	0	100
2	2	1.333	0.01333	0	150
3	3	1.000	0.00500	0	200
4	4	0.667	0.00445	0	150
5	5	0.667	0.00445	0	150
6	6	1.000	0.00500	0	200
7	7	1.333	0.01333	0	100
8	8	1.333	0.01333	0	150

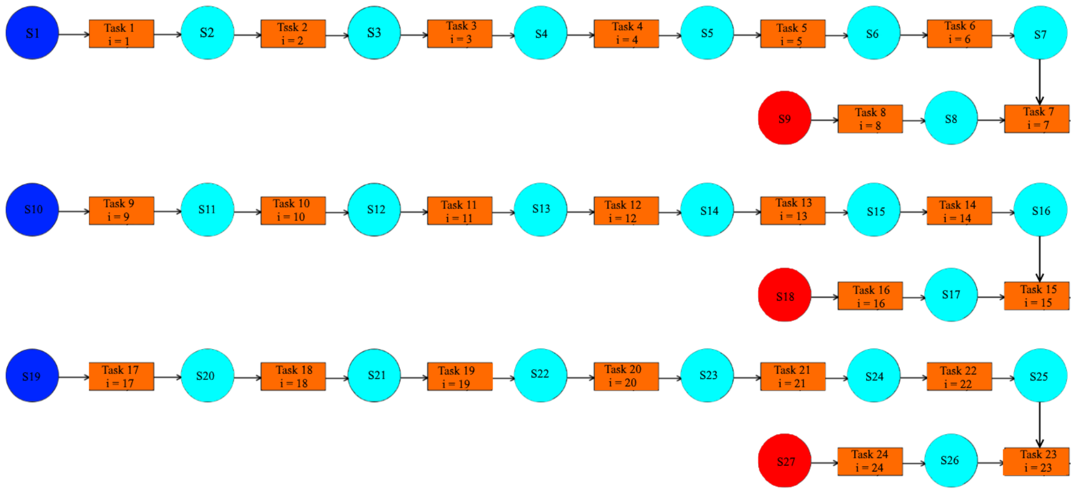

Fig. 11 STN representation of Example 6

Table 8 Data for Example 6

Task	Processing Unit	α_i	β_i	B_i^{min}	B_i^{max}
1-3	1	1.333	0.01333	0	100
4-6	2	1.333	0.01333	0	150
7-9	3	1.000	0.00500	0	200
10-12	4	0.667	0.00445	0	150
13-15	5	0.667	0.00445	0	150
16-18	6	1.000	0.00500	0	200
19-21	7	1.333	0.01333	0	100
22-24	8	1.333	0.01333	0	150

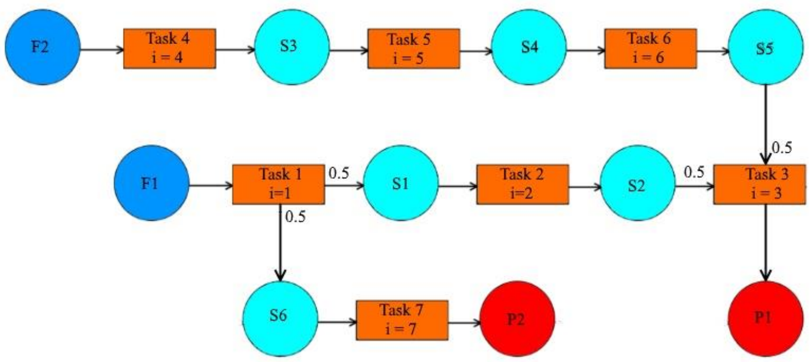

Fig. 12 STN representation of Example 7

Table 9 Data for Example 7

Task	Processing Unit	α_i	β_i	B_i^{min}	B_i^{max}
1	1	6.000	0	0	200
2	2	5.000	0	0	100
3	3	9.000	0	0	100
4	4	2.000	0	0	50
5	5	3.000	0	0	50
6	6	4.000	0	0	50
7	7	2.000	0	0	100

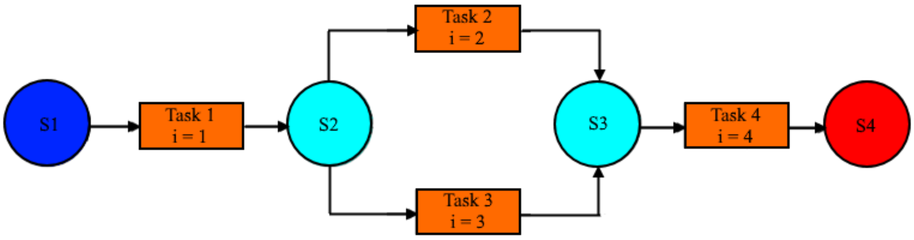

Fig. 13 STN representation of Example 8

Table 10 Data for Example 8

Task	Processing Unit	α_i	β_i	B_i^{min}	B_i^{max}
1	1	1.000	0	0	10
2	2	3.000	0	0	4
3	3	1.000	0	0	2
4	4	2.000	0	0	10

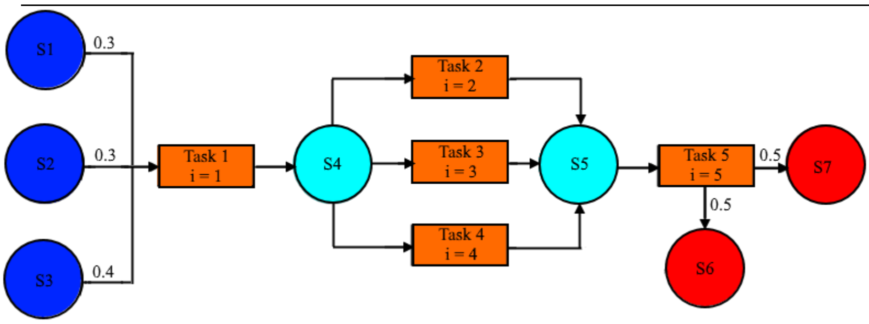

Fig. 14 STN representation of Example 9

Table 11 Data for Example 9

Task	Processing Unit	α_i	β_i	B_i^{min}	B_i^{max}
1	1	1.500	0	0	150
2	2	4.500	0	0	60
3	3	1.500	0	0	30
4	4	1.500	0	0	30
5	5	3.000	0	0	150

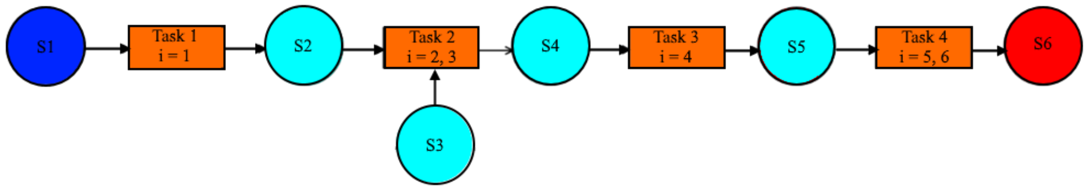

Fig. 15 STN representation of Example 10

Table 12 Data for Example 10

Task	Processing Unit	α_i	β_i	B_i^{min}	B_i^{max}
1	1	17.3333	0.866	0	20
2	2	2.667	0.133	0	20
3	3	2.667	0.133	0	20
4	4	4.000	0.200	0	20
5	5	5.333	0.266	0	20
6	6	5.333	0.266	0	20

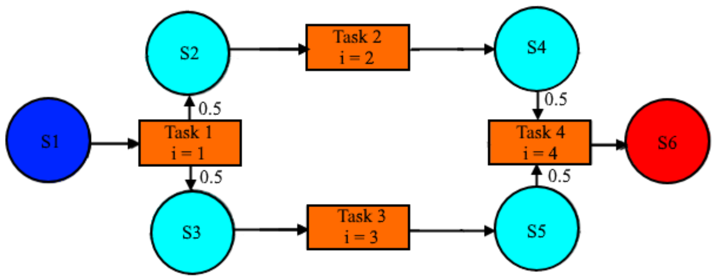

Fig. 16 STN representation of Examples 11 and 12

Table 13 Data for Example 11

Task	Processing Unit	α_i	β_i	B_i^{min}	B_i^{max}
1	1	1.666	0.03335	0	40
2	2	2.333	0.08335	0	20
3	3	0.667	0.06600	0	5
4	4	2.667	0.008325	0	40

Table 14 Data for Example 12

Task	Processing Unit	α_i	β_i	B_i^{min}	B_i^{max}
1	1	1.666	0.03335	0	40
2	2	2.333	0.08335	0	20
3	3	0.333	0.06800	0	2.5
4	4	2.667	0.008325	0	40

The computational results from **M1** and **M2** are presented in Tables 15-20. While Tables 15-19 present the results from **M1** and **M2** with maximization of productivity as the objective, Table 20 presents the results from both models with minimization of makespan as the objective. From Tables 15-20, it can be observed that both **M1** and **M2** models are able to generate optimum solutions using less number of event points, which leads to smaller model sizes and less computational time. For instance, both **M1** and **M2** models require two event points less than the model of Shaik and Floudas [25] to generate the optimal solutions in all instances in Example 1 (see Table 15). More specifically, in Example 1d, the model of Shaik and Floudas [25] require 9 event points, whereas both models **M1** and **M2** require 7 event points, resulting in a reduction in binary variables by 20% (45 vs. 35). The optimal schedule for Example 1d using the model **M1** is illustrated in Figure 17. From Figure 17, it can be observed that the related production and consumption tasks take place at the same event points because all production tasks in this example are treated as non-recycling tasks. For instance, task I1 processed in unit J1 is a non-recycling task, which takes place at event point N1. Its related consumption task is task I3 processed in unit J3 since this task I3 consumes state S2 produced from task I1, which also takes place at the same event point N1.

Similarly, from Table 18, both mathematical models require 7 event points to generate the optimal solution for Example 4, while the model of Shaik and Floudas [25] requires 10 event points. This leads to 30% reduction in the number of binary variables (60 vs. 42) using the proposed mathematical models. From the optimal schedule for Example 4 using the model **M1**, which is depicted in Figure 18, it can be again confirmed that the reduction in the total event points required is due to the fact that all related production and consumption tasks are allowed to take place at the same event points. More specifically, from Figure 18, it seems that task I4 processed in unit J4, which is a non-recycling, task takes place at event point N3, while its related consumption task is task I6 processed in unit J6 which also takes place at the same event point N3. Briefly, it can be concluded that the proposed models **M1** and **M2** can be applied to any batch processes even those with recycling loops such as the batch processes in Figure 1 and Figure 8.

Table 15 Computational results for Example 1 using maximization of productivity as objective

Example	Model	Event points	CPU time (s)	RMILP (cu)	MILP (cu)	Disc. Var.	Cont. Var.	Constr.
1a (H=8h)	SF	4	0.078	2000.00	1840.18	20	78	109
	M2	2	0.125	2000.00	1840.18	10	40	57
	M1	2	0.062	2000.00	1840.18	10	47	58
1b (H=10h)	SF	5	0.094	3000.00	2628.19	25	97	137
	M2	3	0.109	3000.00	2628.19	15	59	85
	M1	3	0.047	3000.00	2628.19	15	68	90
1c (H=12h)	SF	6	0.109	4000.00	3463.62	30	116	165
	M2	4	0.124	4000.00	3463.62	20	78	113
	M1	4	0.078	4000.00	3463.62	20	89	122
1d (H=16h)	SF	9	1.29	6601.65	5038.05	45	173	249
	M2	7	1.54	6601.65	5038.05	35	135	197
	M1	7	1.37	6601.65	5038.05	35	152	218

Note Δn = 0 for all cases

Table 16 Computational results for Example 2 using maximization of productivity as objective

Example	Model	Event points	CPU time (s)	RMILP (cu)	MILP (cu)	Disc. Var.	Cont. Var.	Constr.
2a (H=8h)	SF	4 (Δn=0)	0.078	1730.87	1498.57	32	136	211
	M2	4 (Δn=0)	0.141	1730.87	1498.57	32	136	213
	M1	4 (Δn=0)	0.062	1730.87	1498.57	32	126	180
2b (H=10h)	SF	6 (Δn=0)	0.889	2730.66	1943.17	48	202	331
	M2	6 (Δn=0)	0.889	2730.66	1943.17	48	202	331
	M1	6 (Δn=0)	0.827	2730.66	1943.17	48	184	276
	SF	6 (Δn=1)	5.41	2730.66	1962.69	88	242	737
	M2	6 (Δn=1)	5.10	2730.66	1962.69	88	242	739
	M1	6 (Δn=1)	2.78	2730.66	1962.69	88	224	316
2c (H=12h)	SF	7 (Δn=0)	2.39	3301.03	2658.52	56	235	388
	M2	7 (Δn=0)	2.78	3301.03	2658.52	56	235	390
	M1	7 (Δn=0)	2.86	3301.03	2658.52	56	213	324
2d (H=16h)	SF	8 (Δn=0)	5.97	4291.68	3738.38	64	268	447
	M2	8 (Δn=0)	6.30	4291.68	3738.38	64	268	449
	M1	8 (Δn=0)	3.94	4291.68	3738.38	64	242	372

Table 17 Computational results for Example 3 using maximization of productivity as objective

Example	Model	Event points	CPU time (s)	RMILP (cu)	MILP (cu)	Disc. Var.	Cont. Var.	Constr.
3a (H=8h)	SF	5 (Δn=0)	0.218	2100.00	1583.44	55	235	390
	M2	5 (Δn=0)	0.343	2100.00	1583.44	55	235	390
	M1	5 (Δn=0)	0.358	2100.00	1583.44	55	229	346
3b (H=10h)	SF	7 (Δn=0)	6.24	3369.69	2305.55	77	327	560
	M2	7 (Δn=0)	6.42	3369.69	2305.55	77	327	560
	M1	7 (Δn=0)	10.23	3369.69	2293.46	77	315	494
	SF	8 (Δn=1)	3159	3618.64	2358.20	165	450	1433
	M2	8 (Δn=1)	3141	3618.64	2358.20	165	450	1433
	M1	8 (Δn=1)	892	3618.64	2358.20	165	435	659
3c (H=12h)	SF	7 (Δn=0)	0.437	3465.63	3041.27	77	327	560
	M2	7 (Δn=0)	0.406	3465.63	3041.27	77	327	560
	M1	7 (Δn=0)	0.483	3465.63	3041.27	77	315	494
3d (H=16h)	SF	10 (Δn=0)	7.80	5225.86	4262.80	110	465	815
	M2	10 (Δn=0)	6.99	5225.86	4262.80	110	465	715
	M1	10 (Δn=0)	8.81	5225.86	4262.80	110	444	716

From Tables 16-17, we can observe that the proposed formulations **M1** and **M2** require the same number of event points with the model of Shaik and Floudas [25] for Examples 2-3. For these examples, **M1** and **M2** do not reduce the computational time too much because of the same number of event points required. It should be noted though that when tasks have to span over multiple event points, then model **M1** can significantly reduce the computational time. This main reason may come from the constraint (5) which is tighter than those in Shaik and Floudas [25] when a task has to span over multiple event points. For instance, **M1** requires 49% less computational time than the model of Shaik and Floudas [25] (5.41 s vs 2.78 s) to solve Example 2b and 72% less computational time (3159 s vs 892 s) for Example 3b. On the other hand, even though **M2** require slightly less computational time than the model of Shaik and Floudas [25] for both Example 2b (5.41 s vs 5.10 s) and 3b (3159 s vs 3141 s), it requires 46% more computational time than the model **M1** (2.78 s vs 5.10 s) to solve Example 2b and 72% more computational time (892 s vs 3141 s) to solve Example 3b. Therefore, it can be concluded that the proposed model **M1** is the most efficient.

Table 18 Computational results for Examples 4-7 using maximization of productivity as objective

Example	Model	Event points	CPU time (s)	RMILP (cu)	MILP (cu)	Disc. Var.	Cont. Var.	Constr.
4	SF	10	24.18	6601.65	4305.46	60	232	345
(H=16 h)	M2	7	27.63	6601.65	4305.46	42	163	246
	M1	7	35.88	6601.65	4305.46	42	188	279
5a	SF	8	0.141	1500.00	1414.18	64	250	369
(H=16 h)	M2	3	0.156	1500.00	1414.18	24	95	142
	M1	3	0.156	1500.00	1414.18	24	116	159
5b	SF	14	0.343	4500.00	4414.80	112	436	651
(H=32 h)	M2	9	0.250	4500.00	4414.80	72	281	424
	M1	9	0.296	4500.00	4414.80	72	332	501
6a	SF	57	1.61	25000.00	24927.50	570	2225	3354
(H=144 h)	M2	50	6.13	25000.00	24927.50	500	1952	2951
	M1	50	1.90	25000.00	24927.50	500	2310	3634
6b	SF	111	13.84	52000.00	51933.10	1110	4331	6540
(H=288 h)	M2	104	25.72	52000.00	51933.10	1040	4058	6137
	M1	104	12.14	52000.00	51933.10	1040	4794	7576
6c	SF	219	40.82	106000.00	105944.00	2190	8543	12912
(H=576 h)	M2	212	8.30	106000.00	105944.00	2120	8270	12509
	M1	212	27.97	106000.00	105944.00	2120	9762	15460
7	SF	49	33.81	21000.00	20935.30	1176	4853	8540
(H=128 h)	M2	42	43.54	21000.00	20935.30	1008	4160	7329
	M1	42	33.59	21000.00	20935.30	1008	3724	5768

Note $\Delta n = 0$ for all cases

Table 19 Computational results for Examples 8-12 using maximization of productivity as objective

Example	Model	Event points	CPU time (s)	RMILP (cu)	MILP (cu)	Disc. Var.	Cont. Var.	Constr.
8	SF	5	0.109	14.00	10.00	20	82	117
(H=6h)	M2	3	0.078	14.00	10.00	12	40	73
	M1	3	0.062	14.00	10.00	12	46	79
9	SF	5	0.125	300.00	210.00	25	114	160
(H=9h)	M2	3	0.109	300.00	210.00	15	60	100
	M1	3	0.062	300.00	210.00	15	80	105
10	SF	5	0.109	80.00	58.99	30	123	175
(H=76h)	M2	2	0.125	80.00	58.99	12	51	73
	M1	2	0.046	80.00	58.99	12	61	76
11	SF	6	0.109	400.00	400.00	24	110	153
(H=10h)	M2	4	0.109	400.00	400.00	16	74	105
	M1	4	0.093	400.00	400.00	16	95	129
12	SF	10	0.203	400.00	400.00	40	182	257
(H=5h)	M2	8	0.093	400.00	400.00	32	146	209
	M1	8	0.093	400.00	400.00	32	183	265

Note $\Delta n = 0$ for all cases

However, even though the proposed models, especially model **M1**, are more efficient for most of the examples, it seems that in some special cases they require a bit larger CPU time. For instance, in Example 4 depicted in Table 18 the model of Shaik and Floudas [25] is able to generate the optimum solution in 24.18 s, while models **M1** and **M2** require 35.88 s and 27.63 s respectively. Nevertheless, the difference is not large, which is in the same magnitude. The main possible reason is that the proposed models **M1** and **M2** require more CPU time to prove optimality for this example due to different nodes investigated using the branch and bound algorithm. It should also be noted that in Tables 15-20 only the computational time required to generate the optimal solution using the optimum number of event points and Δn is reported. In practice, an iterative procedure is often used to find the optimal number of event points, Δn and the optimal solution. In this iterative procedure, the problem is solved starting from the minimum number of event points. The number of event points is increased by one until there is no change in the obtained solution. In the iterative procedure, it should be also examined whether allowing one a task to span for more than one event point can lead to the optimal solution. We use the iterative procedure to solve Example 4 with the model of Shaik and Floudas [25] and the proposed model **M1**. The computational results are given in Table 21. From Table 21 it seems that the model **M1** requires less total computational time to locate the optimal solution using the iterative procedure. More specifically for model **M1** 1281 s are required to prove that the optimal solution is generated by using 7 event points while for Shaik and Floudas [25] significantly more time (2771 s) is required to prove that the optimal solution is generated by using 10 event points.

Table 20 Computational results for Examples 1-3 using minimization of makespan as objective

Example	Model	Event points	CPU time (s)	RMILP (h)	MILP (h)	Disc. Var.	Cont. Var.	Constr.
1a (H=50 h)	SF	14	11.45	24.24	27.88	70	268	394
$D_{s4} = 2000$	M2	12	5.30	25.36	27.88	60	230	342
	M1	12	6.96	24.24	27.88	60	254	383
1b (H=100 h)	SF	23	7.50	48.47	52.07	115	439	646
$D_{s4} = 4000$	M2	21	5.70	50.06	52.07	105	401	594
	M1	21	4.81	48.47	52.07	105	443	671
2a(H=50 h)	SF	9	96.00	10.78	19.34	72	301	515
$D_{s8} = 200$	M2	9	28.78	10.78	19.34	72	301	515
$D_{s9} = 200$	M1	9	46.58	18.68	19.34	72	265	425

2b(H=100 h)	SF	19	3600[a]	45.57	46.31	152	631	1105
$D_{s8} = 500$	M2	19	3600[b]	45.57	46.31	152	631	1107
$D_{s9} = 400$	M1	19	3600[c]	45.57	46.31	152	555	905
3a (H=50 h)	SF	7	0.187	11.07	13.37	77	327	572
$D_{s12} = 100$	M2	7	0.250	11.07	13.37	77	327	572
$D_{s13} = 200$	M1	7	0.374	11.25	13.37	77	306	501
3b (H=50 h)	SF	10	0.515	12.50	17.03	110	465	827
$D_{s12} = 250$	M2	10	0.374	12.76	17.03	110	465	827
$D_{s13} = 250$	M1	10	0.359	14.27	17.03	110	435	723

[a] Relative gap 1.10%. [b] Relative gap 1.43%. [c] Relative gap 1.58%. Note Δn = 0 for all cases

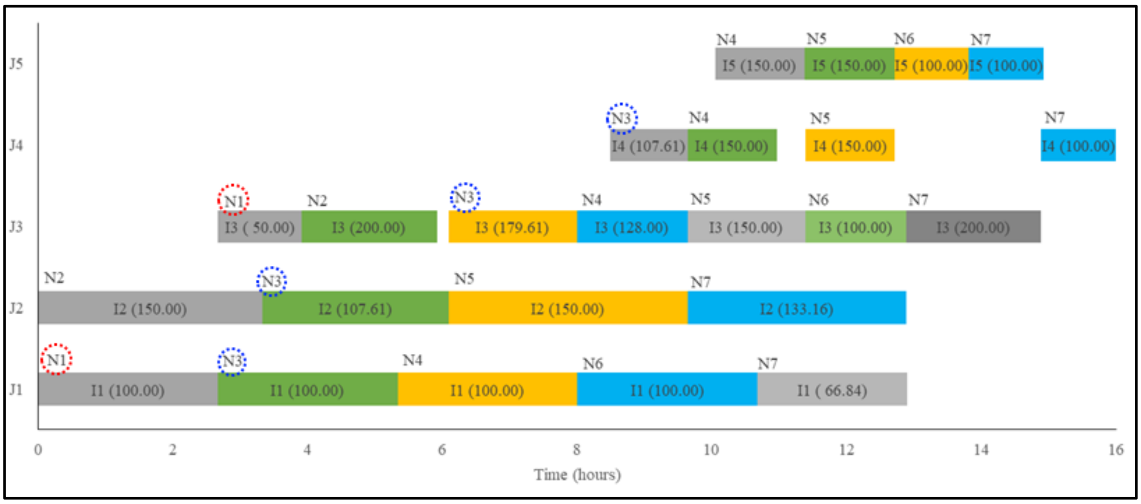

Fig. 17 Optimal schedule of Example 1d using the model **M1**, with maximization of productivity as the objective

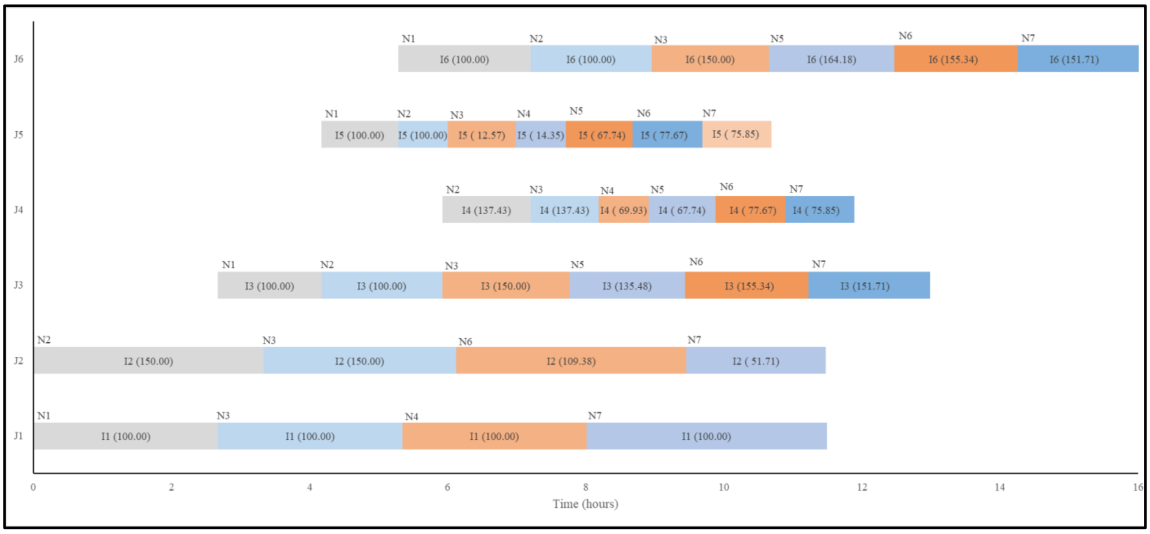

Fig. 18 Optimal schedule for Example 4 using the model **M1** with maximization of productivity as the objective

Table 21 Computational results for Example 4 using the iterative procedure (maximization of productivity)

	CPU time (s)			
	SF		**M1**	
Event Point	(Δn=0)	(Δn=1)	(Δn=0)	(Δn=1)
n=1	-	-	0.093	0.078
n=2	-	-	0.062	0.078
n=3	-	-	0.078	0.031
n=4	0.031	0.062	0.078	0.187
n=5	0.062	0.094	0.218	0.405
n=6	0.078	0.078	1.36	9.70
n=7	0.046	0.188	**35.88**	700
n=8	0.250	0.421	532.5	-
n=9	1.20	19.6	-	-
n=10	**24.18**	2027	-	-
n=11	698	-	-	-
Total	2771		1281	

7 Conclusions

In this paper, we proposed two novel mathematical formulations **M1** and **M2** using the unit-specific event-based modelling approach. While timing variables in **M1** were defined based on units, they were defined based on tasks in **M2**. In both models, production and consuming tasks related to the same states were allowed to take place at the same event points. To avoid suboptimality in some cases, we proposed a new definition of recycling and non-recycling tasks. Only the non-recycling production tasks and related consuming tasks are allowed to take place at the same event points. The computational results demonstrate that the proposed models **M1** and **M2** generated optimal solutions for all examples and reduced the number of event points required, leading to smaller model sizes. Both models are applicable to any batch processes even those with recycling loops. Furthermore, the proposed model **M1** is the most efficient since it requires the least possible computational time which can reach up to one magnitude in most cases. In the future, we will extend the proposed models **M1** and **M2** to solve other more complex intermediate storage policies such as FIS and NIS.

Acknowledgements

Nikolaos Rakovitis would like to acknowledge financial support from the postgraduate award by The University of Manchester. Liping Zhang appreciates financial support from the National Natural Science Foundation of China (No. 51875420).

References

1. 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

2. Pantelides C. Unified frameworks for optimal process planning and scheduling. Proc. Second Conf. on Foundations of Computer Aided Operations, 1994, 253-274

3. Floudas C A, Lin X. Continuous-time versus discrete-time approaches for scheduling of chemical processes: a review. Computers and Chemical engineering, 2004, 28(11): 2109-2129

4. Méndez C A, Cerdá J, Grossmann I E, Harjukoski I, Fahl M. State-of-the-art review of optimization methods for short-term scheduling of batch processes. Computers and Chemical Engineering, 2006, 30(6-7): 913-946

5. Li J, Susarla N, Karimi I A, Shaik M A, Floudas C A. An analisis of some unit-specific event-based models for the short-term scheduling of noncontinuous processes. Industrial and Engineering Chemistry research, 2010, 49(2): 633-647

6. Maravelias C T. General framework and modeling approach classification for chemical production scheduling. AIChE journal, 2012, 58(6): 1812-1828

7. Harjunkoski I, Maravelias C, Bongers P, Castro P, Engell S, Grossmann I, Hooker J, Méndez C, Sand G, Wassick J. Scope for industrial application of production scheduling models and solution methods. Computers and Chemical Engineering, 2014, 62(5): 161-193

8. Velez S, Maravelias C T. Multiple and nonuniform time grids in discrete-time MIP models for chemical production scheduling. Computers and Chemical Engineering, 2013, 53(11): 70-85

9. Lee H, Maravelias C T. Discrete-time mixed integer programming models for short-term scheduling in multipurpose environments. Computers and Chemical Engineering, 2017, 107: 171-183

10. Pinto J M, Grossmann I E. A continuous time mixed integer linear programming model for short-term scheduling of multistage batch plants. Industrial and Engineering Chemistry research, 1995, 34(9): 3037-3051

11. 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

12. Susarla N, Li J, Karimi I. A Novel Approach to Scheduling Multipurpose Batch Plants Using Unit-Slots. AlChe Journal, 2010, 56(7): 1859-1879

13. Zhang X, Sargent R W H. The optimal operation of mixed production facilities-A general formulation and some approaches for the solution. Computers and Chemical Engineering, 1996, 20(6-7): 897-904

14. Castro P, Barbosa-Póvoa A P F D, Matos H. An improved RTN continuous-time formulation for the short-term scheduling of multipurpose batch plants. Industrial and Engineering Chemistry research, 2001, 40(9): 2059-2068

15. Maravelias C T, Grossmann I E. New General Continuous-Time State-Task Network Formulation for Short-Term Scheduling of Multipurpose Batch Plants. Industrial Engineering and Chemistry research, 2003, 42(13): .3056-3074

16. Ierapetritou M G, Floudas C A. Effective continuous-time formulation for short-term scheduling. 1. Multipurpose batch processes. Industrial & Engineering Chemistry, 1998, 37(11): 4341-4359

17. Li J, Floudas C. Optimal event point determination for short-term scheduling of multipurpose batch plants via unit-specific event-based continuous-time approaches. Industrial & Engineering Chemistry Research, 2010, 49(16): 7446-7469.

18. Tang Q H, Li J, Floudas C A, Deng M X, Yan Y B, Xi Z H, Chen P H, Kong J Y. Optimization framework for process scheduling of operation-dependent automobile assembly lines. Optimization Letters, 2012,6(4): 797-824

19. Li J, Xiao X, Floudas C A. Integrated gasoline blending and order delivery operations: part I. short-term scheduling and global optimization for single and multi-period operations. AIChE journal, 2016, 62(6): 2043-2070

20. Méndez C A, Cerdá J. Optimal scheduling of a resource-constrained multiproduct batch plant supplying intermediates to nearby end-product facilities. 2000, 24(2-7): 369-376

21. Hui C, Gupta A, van der Meulen H A J. A novel MILP formulation for short-term scheduling of multi-stage multi-product batch plants with sequence-dependent constraints. Computers and Chemical Engineering, 2000, 24(12): 2705 – 271

22. Méndez C A, Cerdá 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

23. Li J, Karimi I A. Scheduling gasoline blending operations from recipe determination to shipping using unit slots. Industrial & Engineering Chemistry Research, 2011, 50(15): 9156-9174

24. Shaik M. A, Janak S L, Floudas C A. Continuous-time models for short-term scheduling of multipurpose batch plants: a comparative study. Industrial & Engineering Chemistry Research, 2006, 45(18): 6190-6209

25. Shaik M, Floudas C. Novel Unified Modeling Approach for Short-Term Scheduling. Industrial & Engineering Chemistry *Research,* 2009, 48(6): 2947-2964

26. Shaik M, Vooradi R. Short-term scheduling of batch plants: Reformulation for handling material transfer at the same event, Industrial & Engineering Chemistry 2017, 56(39): 11175-11185

Nomenclature

Task-specific event-based model

Indices

$i,\ i'$: tasks

$j,\ j'$: units

$n,\ n', n''$: event points

s : states

Sets

I : tasks

$\mathbf{I}_j$: tasks that can be performed in unit j

$\mathbf{I}_s^c$: tasks that consume state s

$\mathbf{I}_s^P$: tasks that produce state s

$\mathbf{I}^R$: tasks considered as recycling tasks

J : units

N : event points

S : states

$\mathbf{S}^{FP}$: states that are final products

$\mathbf{S}^{IN}$: states that are intermediate products

$\mathbf{S}^R$: states that are raw materials

Parameters

B_i^{max}: maximum batch size that can be processed in task i

B_i^{min} : minimum batch size that can be processed in task i

D_s: demand of state s

H : scheduling horizon

p_s : price of state s

α_i : coefficient of constant term of processing time of task i

β_i : coefficient of variable term of processing time of task i

Δn : maximum number of event points that task i is allowed to be active

$\rho_{i,s}$: portion of state s consumed/produced by task i

Binary Variables

$w_{i,n,n'}$: binary variable which takes the value 1 if task i starts at time event point n and finishes at time event point $n' \geq n$.

Continuous Variables

$b_{i,n,n'}$: batch size of task i that is active from time event point n to time event point $n' \geq n$

$ST0_s$: initial amount of state s ($s \in \mathbf{S}^R$)

$ST_{s,n}$: excess amount of state s that needs to be stored at time event point n

$T^{\mathrm{f}}_{i,n}$: finish time of task i at time event point n

$T^{\mathrm{s}}_{i,n}$: start time of task i at time event point n

Unit-specific event-based model

Indices

i, i': tasks

j, j': units

n, n', n'': event points

s : states

Sets

I : tasks

$\mathbf{I}_j$: tasks that can be performed in unit j

$\mathbf{I}_s^c$: tasks that consume state s

$\mathbf{I}_s^P$: tasks that produce state s

$\mathbf{I}^R$: tasks considered as recycling tasks

J : units

N : event points

S : states

$\mathbf{S}^{FP}$: states that are final products

$\mathbf{S}^{IN}$: states that are intermediate products

$\mathbf{S}^{R}$: states that are raw materials

Parameters

$B_{i,j}^{max}$: maximum batch size of task i processed in unit j

B_{i}^{min} : minimum batch size of task i processed in unit j

D_s: demand of state s

H : scheduling horizon

p_s : price of state s

$\alpha_{i,j}$: coefficient of constant term of processing time of task i in unit j

$\beta_{i,j}$: coefficient of variable term of processing time of task i in unit j

Δn : maximum number of event points that task i is allowed to be active

$\rho_{i,j,s}$: portion of state s consumed/produced by task i processed in unit j

Binary variables

$w_{i,j,n,n'}$: binary variable which takes the value 1 if task i is processed in unit j from time event point n to time event point $n' \geq n$

Continuous variables

$b_{i,j,n,n'}$: amount of materials that are processed in unit j processing task i from time event point n to time event point $n' \geq n$

$ST_{s,n}$: amount of state s that has to be stored at time event point n

$T^{\mathrm{s}}_{j,n}$: start time of unit j at time event point n

$T^{\mathrm{f}}_{j,n}$: end time of unit j at time event point n

$T^{\mathrm{s}}_{i,j,n}$: start time of task i in unit j at time event point n

$T^{\mathrm{f}}_{i,j,n}$: end time of task i in unit j at time event point n

Chapter 4: Generic mathematical formulations for scheduling of multipurpose batch plants

4.1 Introduction

In Chapter 3, it is presented that allowing related production and consumption tasks can reduce the number of event points and, as a result, it can reduce the model size and the computational time required to generate the optimal solution. However, there are more cases where existing mathematical models lead to more time slots/event points or even to a suboptimum solution. An issue, for instance, is that a consumption task can only start after all related production tasks finish within the same event point (or the previous event point if those production and consumption tasks are not allowed to the place at the same event point). Such constraint still holds, even if the consumption task consumes materials from the storage tank or a related production task that finishes earlier. In both cases, a mathematical model requires additional event points to generate an optimal solution.

Existing mathematical models also require additional event points to generate the optimal solution for problems with limited storage policies. By carefully examining the results generated using existing formulations for examples with both unlimited and limited storage policies, it seems that the latter requires more event points, even if the units process the same number of batches in both cases. The main issue that leads in such an increase is the fact that the start time of a consumption task at event point (n + 1) (or at event point n if related processes can take place at the same event point) must always be equal to the finish time of all related production tasks at event point n if the consumption task consumes a state with a limited storage policy. Such constraint is introduced in formulations to ensure that there is no storage violation in the generated schedule. However, if there is enough storage available or the processing units can store the producing materials, then related production and consumption tasks should not align. Another issue is that most existing formulations for limited storage capacity only allow materials to remain in a producing processing unit for the current time slot/event point. For the next time slot/event point, a storage tank or another processing unit should store or process those materials. However, not allowing a task to store materials at the next event point can either increase the number of event points or even to lead to a suboptimum solution.

In the literature, a few works are dealing with those issues (Seid and Majozi 2012; Vooradi and Shaik 2013). Such models even though they can reduce the number of event points required, they fail to handle all these cases simultaneously. Furthermore, in some cases, those models can generate schedules with storage violations (Seid and Majozi 2012) or suboptimum solutions (Vooradi and Shaik 2013). In this chapter, two generic formulations for scheduling of multipurpose batch processes are developed. While the first model allows relating production and consumption tasks to take place at the same event point, the second model does not. Both models conditionally sequence and align related production and consumption tasks if there is an indirect or direct material transfer between units that process those tasks. Additionally, processing units are allowed to store materials for multiple event points by avoiding schedules with a real-time violation. By using such an approach, the aim is to generate the optimum solution in all cases by using the least number of event points.

4.2 Research contribution 2

Rakovitis, N., Pan Y, Zhang, N., Li, J. Kopanos, G. Generic mathematical formulations for scheduling of multipurpose batch plants, AIChE journal, submitted

Blank Page

Generic mathematical formulations for scheduling of multipurpose batch plants

Nikolaos Rakovitis,[1] Yueting Pan,[1] Nan Zhang,[1] Jie Li,[1,*] and Giorgos Kopanos[2]

[1]Centre for Process Integration, Department of Chemical Engineering and Analytical Science, The University of Manchester, Manchester, M13 9PL, United Kingdom

[2]Flexciton Limited, London, 145 City Rd, Hoxton, London EC1V 1AZ

Abstract

In this work, we develop two generic mixed-integer linear programming formulations for scheduling of multipurpose batch plants using the unit-specific event-based modelling approach. While related non-recycling production and consumption tasks are allowed to take place at the same event points but in different actual time in the first model, they are not allowed in the second model. We also introduce the concept of indirect and direct material transfer, which conditionally aligns the operational sequence of related production and consumption tasks. In these models, processing units can hold materials previously produced over multiple event points. The computational results demonstrate that the proposed models do not require a task to span over different event points and, as a result, they can generate the same or better solutions with up to one order of magnitude less computational time compared to the existing models.

Keywords: Scheduling, Multipurpose batch processes, mixed-integer linear programming, unit-specific event-based approach

[*] To whom correspondence should be addressed. Email: jie.li-2@manchester.ac.uk. Tel: +44 (0) 161 306 8622

1 Introduction

Multipurpose batch plants widely exist in the chemical industry for the production of a large number of low-volume, high-value products. To achieve higher utilization of resources, lower inventory costs and better responsiveness to a fluctuating manufacturing environment, optimal scheduling of the multipurpose facilities is desirable and has attracted much interest of both academia and industry in the past decades. Many mathematical formulations attempt to solve this problem by either using the State Task Network (STN) representation[1] or the Resource Task Network (RTN) representation[2]. These models are classified based on the time representation of the scheduling horizon into discrete-time and continuous-time representations. The discrete-time representation divides the scheduling horizon into time intervals with fixed and known length. The time intervals can either be uniform or non-uniform[3] within the scheduling horizon, and a task or activity can only start and finish at these time intervals. The continuous-time representation uses time points, slots, or event points to divide the scheduling horizon with a variable and unknown length. It can be further classified into global event-based[4-6], slot-based including process-slot based[7-8] and unit-slot based[8-9], unit-specific event-based[10-17] and sequence-based[18-20] time representations. These mathematical models are also classified into single- and multiple- time grid mathematical models[21]. For more details about these time representations, the reader can refer to[22-24], which provide excellent reviews for scheduling in chemical industries.

All existing time-grid mathematical models divide the scheduling horizon using time points/slots/event points on which a task or activity can both start and finish. Therefore, the number of time points/slots/event points required directly affects the efficiency of the existing mathematical models. More specifically, an additional time point/slot/event point can lead to an exponential increase in the number of binary variables, continuous variables and constraints, which can potentially increase the computational time required to generate the optimal solution by even one order of magnitude. Some task must be allowed to span over multiple time points/slots/event points to provide the optimal solution, which further increases the computational burden. The capabilities of the unit-specific event-based formulations are well established in the literature[11-12, 17]. However, they still require excessive computational time for industrial-scale problems due to the introduction of additional event points to generate the optimal solution. The main possible reason is that most existing unit-specific event-based formulations unconditionally impose that a consumption task starts after its related production tasks (with the same states) even if the consumption task does not consume materials from the related production tasks Additionally, in some cases, this task should start immediately after its related production tasks finish, even if there is enough storage available.

Two works[14-15] in the literature have attempted to relax such unconditional sequencing and alignment. Seid and Majozi[14] investigated whether consumption tasks consume materials from storage tanks and whether producing materials can be stored in the storage tanks. For the former case, if there are not enough materials in the storage tanks for all consumption tasks, then the unconditional sequencing of all related production and consumption tasks are imposed. In the latter case, if a production task produces materials that the storage tanks cannot store then all related production and consumption tasks should be unconditionally aligned. However, the problem of unconditional sequencing of related production and consumption tasks even if the consumption task does not consume any materials from the production task was not addressed. They also did not consider to conditionally align a production task with a related consumption task, if in storage tanks can store the producing materials. Another issue of their formulation is that it can generate schedules with a real-time violation, as demonstrated by Vooradi and Shaik[15]. To address all those issues, Vooradi and Shaik[15] explicitly examined if a consumption task consumes materials from a specifically related production task or if there is enough storage for materials produced by a specific production task. They sequenced a production task with a related consumption task only if the consumption task consumes materials from the production task, while they aligned a production task with a related consumption task only if storage tanks cannot store the materials from a specific production task. With this approach, they have managed to further reduce the number of event points in comparison to the model of Seid and Majozi[14], while they avoided generating a solution with a real-time violation. However, Vooradi and Shaik[15] used an increased number of binary variable sets to denote whether tasks have to be sequenced or aligned during an event point in their model, leading to computational inefficiency. Most of the existing models fail to generate the optimal solution in some cases, especially when the materials have to be temporarily stored in processing units, as illustrated later.

In this work, we develop two generic mixed-integer linear programming formulations for scheduling of multipurpose batch plants using the unit-specific event-based modelling approach. While we follow the methodology of Rakovitis et al.[17] where all related non-recycling production and consumption tasks can take place at the same event points but in different real times in the first model, we do not allow all related non-recycling production and consumption tasks to take place at the same event points in the second model. We also introduce the concept of indirect and direct material transfer, which allows us to conditionally and unconditionally align the operational sequences of related production and consumption tasks. More specifically, we sequence production and consumption tasks related to the same state if there is an indirect material transfer between the units that are processing these tasks, while we align them if there is a direct material transfer between these units. Additionally, we allow the processing units to hold materials previously produced from these units over multiple event points. Nonsimultaneous material transfer[8] can also take place in both models. We solve several well-established examples in the literature to illustrate the capability of the proposed formulations. The computational results demonstrate that both models require a smaller number of binary variables in most cases, especially in the cases where a processing unit can process multiple tasks, compared to the existing mathematical formulation[15]. It is interesting to note that the proposed models do not need to allow a task to span over several event points to generate the optimal solution. As a result, the computational time is significantly reduced by one order of magnitude in most cases. More importantly, the proposed models can generate better solutions than the existing models such as Vooradi and Shaik[15] and Mostafaei and Harjunkoski[21]. The first model, which allows related non-recycling production and consumption tasks to take place at the same event points is slightly more efficient than the second one. Finally, we use the proposed model to solve a large-scale industrial batch plant scheduling problem from Janak et al.[25] using the rolling-horizon decomposition algorithm. The results demonstrate that the proposed model can improve productivity by 26.7% in significantly less computational time compared to that of Janak et al.[25].

2 Problem statement

Figure 1 illustrates a general STN representation of a multipurpose batch plant. There are I (i = 1, 2, 3, …, I) tasks that are processed in total J (j = 1, 2, …, J) processing units. In a batch plant, a task means heating, reaction, separation, and so on. Each unit can process

$\mathbf{I}_j$ available tasks. A processing unit can process multiple tasks. However, at most one task can be processed in a unit at a time. Raw materials, intermediate products, and final products are denoted as states in the STN representation. There are S (s = 1, 2, 3, …, S) states in total. The raw materials are denoted as $\mathbf{S}^R$, intermediate materials are denoted as $\mathbf{S}^{IN}$ and final products are denoted as $\mathbf{S}^P$. A state is consumed or produced by $\mathbf{I}_s$ tasks including $\mathbf{I}_s^C$ consumption tasks and $\mathbf{I}_s^P$ production tasks. The proportion of a state s that a task i in a unit j consumes or produces is known which is denoted by a parameter ρ_{sij}. While this proportion parameter is positive if the state is produced, it is negative if the state is consumed. A task i on unit j processes a batch size (b_{ij}) of material state. The processing time is assumed to be a linear function of the batch size, which is calculated by $\alpha_{ij} + \beta_{ij} \cdot b_{ij}$.

Once a batch is produced in a processing unit, it may be transferred immediately or remain in the processing unit for some limited or unlimited time. This transfer can be into a dedicated storage tank, split into different small batches or mixed with other batches for downstream processing. In other words, batch splitting and mixing are allowed. There are several different storage policies including unlimited intermediate storage (UIS) policy and finite intermediate storage (FIS) policy. With this, the entire scheduling problem can be stated as follows,

Given:

a) J units, suitable $\mathbf{I}_j$ tasks, minimum (b_{ij}^{min}) and maximum (b_{ij}^{max}) capacities and constant processing time coefficients;

b) S states, suitable $\mathbf{I}_s$ tasks including production tasks and consumption tasks, detailed processing paths and recipes, their initial inventories, and minimum and maximum capacities.

c) The production recipe (i.e., the coefficients of processing time for each task, and the consuming or producing proportions of each batch).

The product prices;

d) The scheduling horizon for maximization of productivity problems or the product demand for minimization of makespan problems.

Determine:

a) Optimal production schedule including allocation, sequence, timings of tasks in a unit;

b) The amount of material being processed in each unit at each time;

c) Inventory profiles of all material states through the scheduling horizon.

Operating rules:

a) At most one task can be processed in a unit at a time;

b) Batch mixing and splitting is allowed.

Assumptions:

a) All parameters are deterministic with no batch/unit failures or operational interruptions;

b) The processing time of a task in a processing unit depends on the batch size;

c) Unlimited feed materials are available;

d) Unlimited storage policy for raw materials and final products;

e) Unlimited or Finite storage policy for intermediate products;

f) Unlimited resources are available;

g) Unlimited wait policy for intermediate states.

h) Negligible transfer times between units (i.e., processing units and storage units).

i) Setup or changeover times are lumped into batch processing times.

j) All processing units can hold a batch temporarily before its start and after its end.

k) Each material state has its dedicated storage unit.

We consider two objectives. The first objective is to maximize productivity in the given scheduling horizon. The second objective is to minimize the total time required to fulfil the product demand, which is known as minimization of makespan.

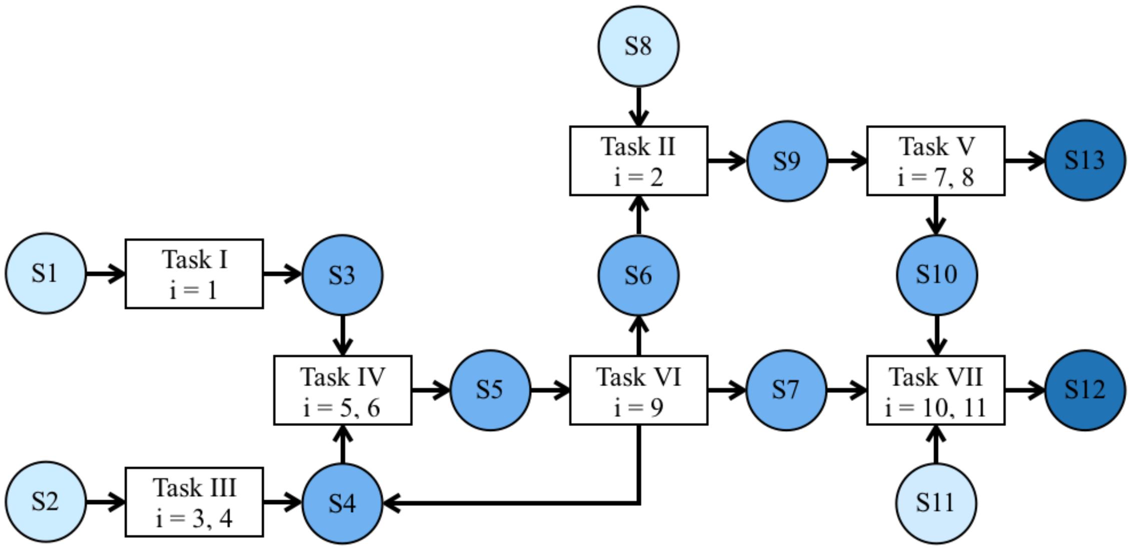

Figure 1 STN representation of a multipurpose batch plant

3 Motivating Example 1

Let consider a motivating example, the data of which is given in Table 1. Figure 2 illustrates the STN of this example. There are two processing units (J1-J2), two tasks (I1-I2) and three states (S1-S3). I1 is processed on unit J1, and I2 is processed on unit J2. There is no initial amount for the intermediate state S2. The maximum storage capacity of state S2 is 10 mu.

Table 1 Data for the Motivating Example

Task	Processing Unit	α_i	β_i	B_i^{min}	B_i^{max}
1	1	3.00	0.02	0	100
2	2	1.00	0.01	0	50

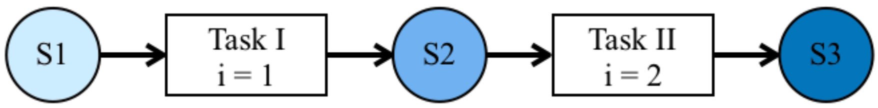

Figure 2 STN representation of the Motivating Example

We use the mathematical models of Li and Floudas[12], Vooradi and Shaik[15] and Mostafaei and Harjunkoski[21] to solve this motivating example. Table 2 provides the computational results. The optimal schedule with a maximum productivity of 300 cu obtained from the existing models[12, 15, 21] is illustrated in Figure 3. In this schedule, 60 cu of S2 is produced by task I1 in the unit J1 at 5 hr. Then, 50 cu of S2 is consumed immediately after production. Storage tanks store 10 cu of S2 , which does not violate the storage capacity, while another 10 cu of S2 are consumed at 7 hr. Finally, product S3 with a total price of 300 cu is produced. From the schedule, it seems that the intermediate S2 is immediately transferred to the storage tank and consumption unit after it is produced. However, we can generate another solution with a maximum productivity of 500 cu through trial and errors, as illustrated in Figure 4, which means that all these existing models generate a suboptimal solution for this example. Through a detailed analysis of the schedules in Figures 3 and 4, the possible reason is that the latter solution allows material S2 produced in the unit J1 to be stored in this unit. Only 50 cu is transferred into unit J2 for further processing after production. Even though the model of Vooradi and Shaik[15] allows a production task to store materials, this can only take place at event point N1. For N2, the materials have to be either consumed by a consumption task or stored in the storage task. However, since there is no storage available, J1 cannot process the same amount of state S2. Instead, it can only produce 60 cu. Therefore, the model of Vooradi and Shaik[15] also fail to generate an optimum solution. This example motivates us to develop a new generic mathematical formulation with consideration of these additional features that can result in a significant increase in the productivity of the batch plant.

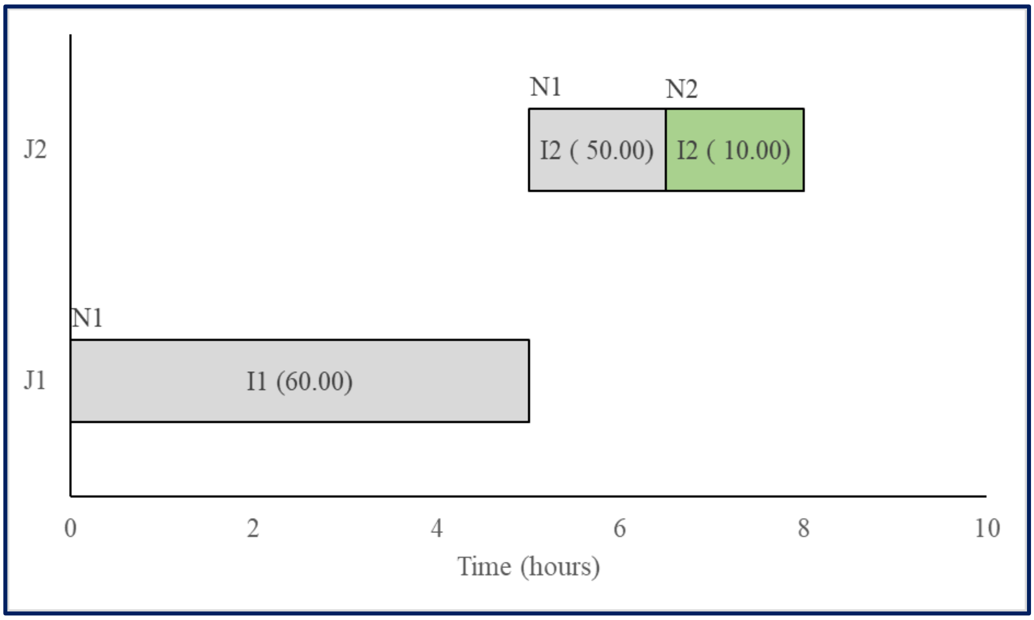

Figure 3 Optimal schedule for the Motivating Example 1 with maximum productivity of 300 cu

Table 2 Computational results for Motivating Example 1 from the models of Li and Floudas[12], Vooradi and Shaik[15] and Mostafaei and Harjunkoski[21]

Example	Model	Number of event points	CPU time (s)	RMILP	MILP (h)	Bin. Var.	Cont. Var.	Constr.
M. E.	LF2010	3	0.11	500.00	300.00	6	29	41
(H = 8 h)	VS2013	3	0.09	500.00	300.00	14	33	72
	MH2019	4 (ΔR=1)	0.08	500.00	300.00	6	34	78

LF2010: Li and Floudas[12] model. VS2013: Vooradi and Shaik[15] model. MH2019: Mostafaei and Harjunkoski[21]

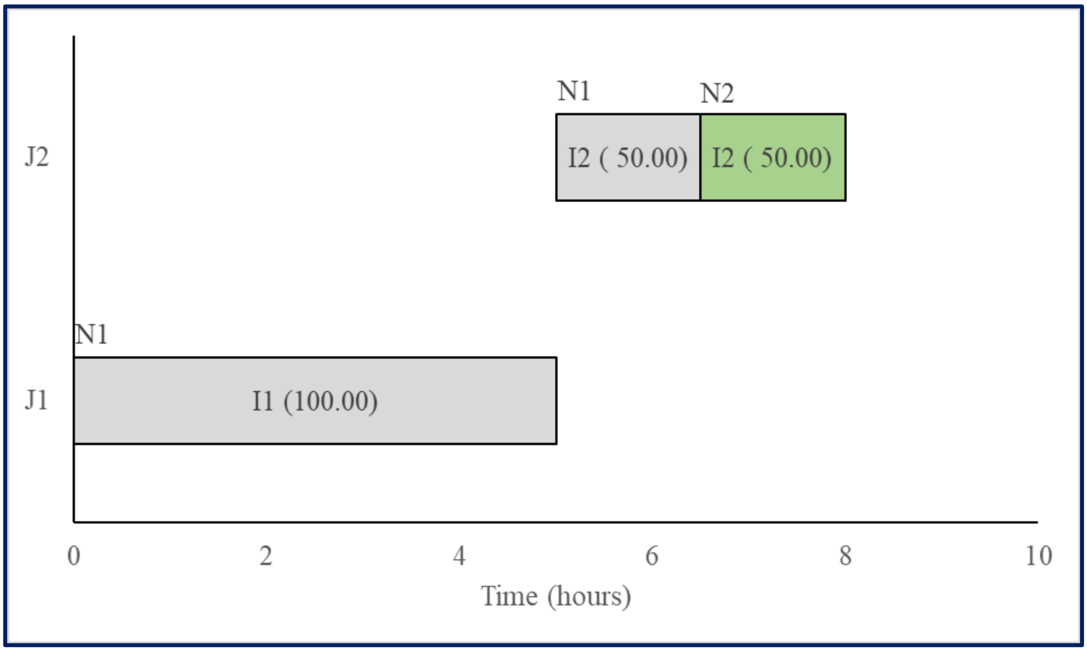

Figure 4 A feasible schedule for the Motivating Example 1 with maximum productivity of 500 cu

4 Generic mathematical formulation

It is of great importance to represent time horizon for scheduling problems before developing a mathematical formulation. Although there are several existing time representations for scheduling problems including discrete-time, slot-based, global event-based, unit-specific event-based, and sequence-based time representations as discussed, the well-established unit-specific event-based time representation is adopted in this work. The reason is that it often leads to smaller model size and less computational effort in comparison to other time representations. The reader can refer for more details about this time representation to the work of Ierapetritou and Floudas[10].

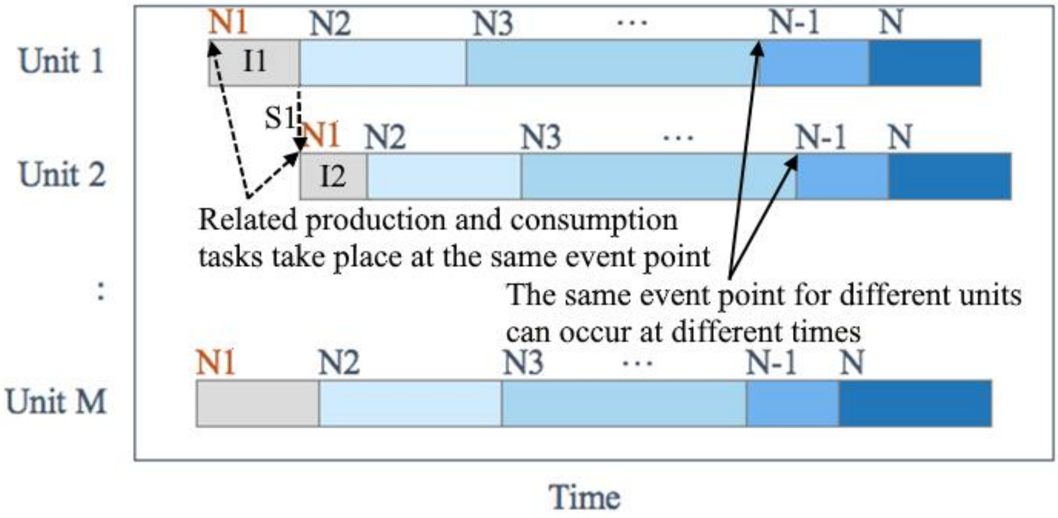

Figure 5 The unit-specific event-based representation where related production and consumptions tasks are allowed to take place at the same event points

4.1 Model M1

In this model **M1**, we allow production and consumption tasks related to the same state to take place at the same event points, which is similar to those of Rakovitis et al.[17]. We also use the definition of recycling tasks presented on Rakovitis et al.,[17] and we only allow non-recycling production and consumption tasks related to the same state to take place at the same event point. Furthermore, the timing variables are defined based on units, not tasks. The unit-specific event-based time representation for model **M1** is illustrated in Figure 5. In Figure 5, task I1 produces S1, which is consumed by task I2. I1 and I2 take place at the same event point N1 but in different actual time.

4.1.1 Allocation constraints

We introduce four-index binary variables $w_{ijnn'}$ to denote the allocation of tasks to units below,

$$w_{ijnn'} = \begin{cases} 1 & \text{if a task } i \text{ is processed in a unit } j \text{ from an event point } n \text{ to } n' \\ 0 & \text{otherwise} \end{cases}$$

where $n \le n' \le n + \Delta n$. The parameter Δn is used to denote the maximum number of event points that a task is allowed to span over.

Based on the operating policy, at most, one task is allowed to be processed in a processing unit at a time.

$$\sum_{i\in\mathbf{I}_j}\sum_{n-\Delta n\le n'\le n}\sum_{n\le n''\le n'+\Delta n} w_{ijn'n''}\le 1$$

$$\forall j, n \qquad (1)$$

4.1.2 Capacity constraints

The materials processed in a unit j should not exceed its minimum (B_{ij}^{min}) and maximum (B_{ij}^{max}) capacities.

$$B_{ij}^{\min}\cdot w_{ijnn'}\le b_{ijnn'}\le B_{ij}^{\max}\cdot w_{ijnn'}$$

$$\forall j,\ i\in\mathbf{I}_j,\ n\le n'\le n+\Delta n \qquad (2)$$

4.1.3 Material balance constraints

The amount of a state s that has to be stored at event point n (ST_{sn}) should be equal to the amount of the state that has been stored at event point $(n-1)$, plus the amount of the state produced by recycling tasks at event point $(n-1)$ and by non-recycling tasks at event point n, minus the amount of the state consumed at event point n. At the first event point, the amount of a state s that has to be stored should be equal to the initial amount of the state ($ST0_s$) plus the amount of the state produced by non-recycling tasks, minus the amount of state s consumed at event point n.

$$ST_{sn}=ST_{s(n-1)}+\sum_{i\in\mathbf{I}_s^P\backslash\mathbf{I}^R}\sum_{j\in(\mathbf{J}_s\cap\mathbf{J}_i)}\sum_{n-\Delta n\le n'\le n}\rho_{sij}b_{ijn'n}+$$

$$+\sum_{i\in(\mathbf{I}_s^P\cap\mathbf{I}^R)}\sum_{j\in(\mathbf{J}_s\cap\mathbf{J}_i)}\sum_{n-1-\Delta n\le n'\le n-1}\rho_{sij}b_{ijn'(n-1)}+\sum_{i\in\boldsymbol{I}_s^C}\sum_{j\in(\mathbf{J}_s\cap\mathbf{J}_i)}\sum_{n\le n'\le n+\Delta n}\rho_{sij}b_{ijnn'}$$

$$\forall s,\ n>1 \qquad (3)$$

$$ST_{sn}=ST0_s+\sum_{i\in\mathbf{I}_s^P\backslash\mathbf{I}^R}\sum_{j\in(\mathbf{J}_s\cap\mathbf{J}_i)}\sum_{n-\Delta n\le n'\le n}\rho_{sij}b_{ijn'n}+\sum_{i\in\boldsymbol{I}_s^C}\sum_{j\in(\mathbf{J}_s\cap\mathbf{J}_i)}\sum_{n\le n'\le n+\Delta n}\rho_{sij}b_{ijnn'}$$

$$\forall s,\ n=1 \qquad (4)$$

where $\mathrm{I}_s^P\backslash\mathrm{I}^R$ means all production tasks except recycling tasks.

4.1.4 Duration constraints

The finish time of a unit *j* at event point *n* must be after its start time plus the processing time of the task i that the unit starts processing at event point *n*.

$$T_{jn}^{\mathrm{f}} \geq T_{jn}^{s} + \sum_{i \in \mathbf{I}_j} \sum_{n \leq n' \leq n + \Delta n} \left(\alpha_{ij} \cdot w_{ijnn'} + \beta_{ij} \cdot b_{ijnn'} \right) \qquad \forall j, n \qquad (5)$$

4.1.5 Material transfer

Material transfer in the batch process is more flexible and complex compared to that in the continuous process. There are several scenarios of material transfer. Figure 6 illustrates all those different scenarios of material transfer. First, materials can be transferred to storage or downstream processing units immediately after production (e.g. material transfer MT1 in Figure 6). Second, materials can be held in the production units after production and then transferred to storage or downstream processing units (e.g. material transfer MT2 in Figure 6). If the storage capacity is large enough, then the material can be first transferred to storage and then transferred to the downstream processing units (e.g. material transfer MT3 in Figure 6). If the storage capacity is not large enough, then some material has to be transferred directly to the downstream processing units (e.g. material transfer MT4 in Figure 6). Besides, materials produced from several production units can be transferred at the same time to storage or downstream processing units, which is called simultaneous material transfer. Alternatively, material produced from several production units can be transferred to storage or downstream processing units at different times, which is called nonsimultaneous material transfer. We generally classify the material transfer as indirect and direct material transfer. If all material is transferred to storage tank first and then to downstream processing units, then it is indirect material transfer. Otherwise, it is direct material transfer.

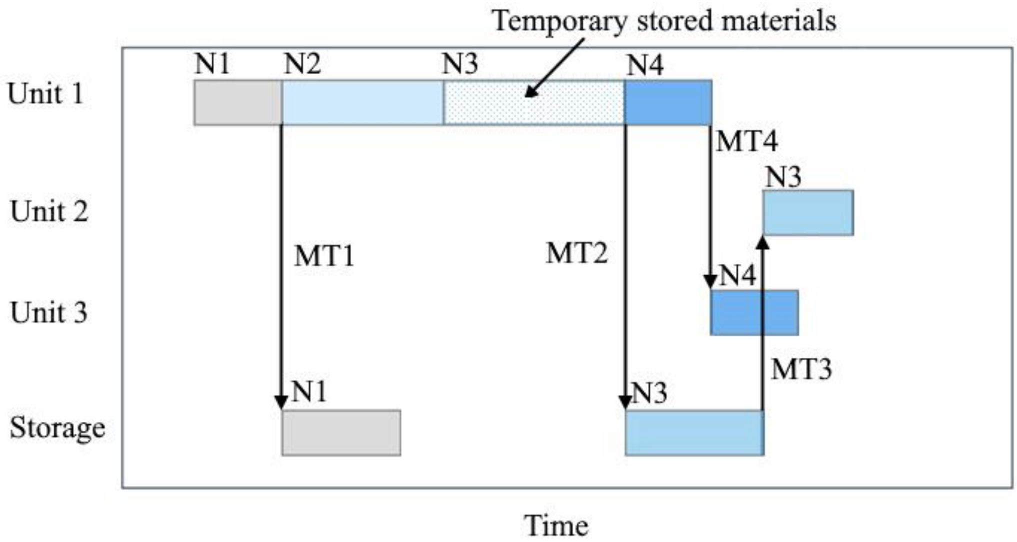

Figure 6 Different scenarios of material transfer

Indirect material transfer

In this scenario, the storage capacity is usually large enough. As a result, materials produced can always be transferred to the storage tank first and then to the downstream processing units from the storage. Such transfer is an indirect material transfer from the production units to the downstream consumption units. To model this indirect material transfer, we define an additional binary variable $zI_{jj'n}$ as follows,

$$zI_{jj'n} = \begin{cases} 1 & \text{if material transfer happens between units } j \text{ and } j' \text{ at event point } n \\ 0 & \text{otherwise} \end{cases} \qquad \forall\, j \neq j', n$$

We also define continuous variables $bTi_{iji'j'n}$ to denote the amount of material transferred from a production task i in unit j to a consumption task i' in unit j' at event point n. Note that the material is first transferred from the production task i to the storage tank and then it is transferred to a consumption task i'. Therefore, it is an indirect material transfer from the production task i to the consumption task i'. The total amount of materials through indirect transfer from a production task i should not exceed the amount produced.

$$\rho_{sij} \cdot \sum_{n-\Delta n \le n' \le n} b_{ijn'n} \ge \sum_{j' \in \mathbf{J}_s} \sum_{i' \in \left(\mathbf{I}_{j'} \cap \mathbf{I}_s^C\right)} bTi_{iji'j'n}$$

$$\forall s \in \mathbf{S}^{IN}, j \in \mathbf{J}_s, i \in (\mathbf{I}_j \cap \mathbf{I}_s^P) \backslash \mathbf{I}^R, n \qquad (6)$$

$$\rho_{sij} \cdot \sum_{n-1-\Delta n \le n' \le n-1} b_{ijn'(n-1)} \ge \sum_{j' \in \mathbf{J}_s} \sum_{i' \in \left(\mathbf{I}_{j'} \cap \mathbf{I}_s^C\right)} bTi_{iji'j'n}$$

$$\forall s \in \mathbf{S}^{IN}, j \in \mathbf{J}_s, i \in (\mathbf{I}_j \cap \mathbf{I}_s^P \cap \mathbf{I}^R), n > 1 \qquad (7)$$

While constraint (6) is used for non-recycling tasks, constraint (7) is proposed for recycling tasks only.

Similarly, the amount of materials through indirect transfer to a consumption task i' at a time should not exceed the amount of materials consumed by this consumption task at event point n.

$$-\rho_{si'j'} \cdot \sum_{n \le n' \le n+\Delta n} b_{i'j'nn'} \ge \sum_{j \in \mathbf{J}_s} \sum_{i \in \left(\mathbf{I}_j \cap \mathbf{I}_s^P\right)} bTi_{iji'j'n}$$

$$\forall s \in \mathbf{S}^{IN}, j' \in \mathbf{J}_s, i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C), n \qquad (8)$$

The total amount of materials consumed at event point n should not exceed the material stored at the previous event point $(n-1)$ plus the amount of materials through indirect transfer.

$$\sum_{j' \in \mathbf{J}_s} \sum_{i' \in \left(\mathbf{I}_s^C \cap \mathbf{I}_{j'}\right)} \left(-\rho_{si'j'} \cdot \sum_{n \le n' \le n+\Delta n} b_{i'j'nn'}\right) \le ST_{s(n-1)} +$$

$$+\sum_{j \in \mathbf{J}_s} \sum_{j' \in \mathbf{J}_s} \sum_{i \in \left(\mathbf{I}_s^P \cap \mathbf{I}_j\right)} \sum_{i' \in \left(\mathbf{I}_s^C \cap \mathbf{I}_{j'}\right)} bTi_{iji'j'n}$$

$$\forall s \in \mathbf{S}^{IN}, n \qquad (9)$$

When there is no indirect material transfer between two processing units, the amount through this indirect transfer should be zero.

$$\sum_{i \in \left(\mathbf{I}_s^P \cap \mathbf{I}_j\right)} \sum_{i' \in \left(\mathbf{I}_s^C \cap \mathbf{I}_{j'}\right)} bTi_{iji'j'n} \le \min\left[B_j^{\max}, B_{j'}^{\max}\right] \cdot zI_{jj'n}$$

$$\forall s \in \mathbf{S}^{IN}, j \ne j', j \in \mathbf{J}_s, j' \in \mathbf{J}_s, n \qquad (10)$$

where $B_j^{max} = \max_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P)} [B_{ij}^{max}]$ and $B_{j'}^{max} = \max_{i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C)} [B_{ij'}^{max}]$.

Direct material transfer

For states with FIS policy, if there is no storage available, then these states cannot be transferred to a storage tank. Instead, they must be transferred directly from the production task i to a consumption task i'. For such a direct material transfer, we introduce an additional binary variable $zD_{jj'n}$ as follows,

$$zD_{jj'n} = \begin{cases} 1 & \text{if there is a direct material transfer between units } j \text{ and } j' \text{ at event point } n \\ 0 & \text{otherwise} \end{cases}$$

$$\forall\, j \neq j', n$$

Similar to indirect material transfer, we also define continuous variables $bTd_{iji'j'n}$ to denote the amount of material directly transferred from a production task i in unit j to a consumption task i' in unit j' at event point n. The amount of materials directly transferred from between processing a production task i in unit j and a consumption task i' in unit j' must not exceed the amount of state produced from production task i. Constraints (11) and (12) are used for non-recycling tasks and recycling tasks, respectively.

$$\rho_{sij} \cdot \sum_{n-\Delta n \leq n' \leq n} b_{ijn'n} + bs_{ijn} \geq \sum_{j' \in \mathbf{J}_s} \sum_{i' \in \left(\mathbf{I}_s^C \cap \mathbf{I}_{j'}\right)} bTd_{iji'j'n}$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}), j \in \mathbf{J}_s, i \in (\mathbf{I}_j \cap \mathbf{I}_s^P) \backslash \mathbf{I}^R, n \qquad (11)$$

$$\rho_{sij} \cdot \sum_{n-1-\Delta n \leq n' \leq n-1} b_{ijn'(n-1)} + bs_{ij(n-1)} \geq \sum_{j' \in \mathbf{J}_s} \sum_{i' \in \left(\mathbf{I}_s^C \cap \mathbf{I}_{j'}\right)} bTd_{iji'j'n}$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}), j \in \mathbf{J}_s, i \in (\mathbf{I}_j \cap \mathbf{I}_s^P \cap \mathbf{I}^R), n > 1 \qquad (12)$$

The amount of materials through direct transfer to a consumption task i' at a time should not exceed the amount of materials consumed by this consumption task at event point n.

$$-\rho_{si'j'} \cdot \sum_{n \leq n' \leq n+\Delta n} b_{i'j'nn'} \geq \sum_{j \in \mathbf{J}_s} \sum_{i \in \left(\mathbf{I}_s^P \cap \mathbf{I}_j\right)} bTd_{iji'j'n}$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}), j' \in \mathbf{J}_s, i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C), n \qquad (13)$$

A direct material transfer between a production task i in unit j and a consumption task i' in unit j' takes place only if the amount of state s produced at event point n for recycling tasks or at event point $(n-1)$ for non-recycling tasks, plus the amount of state s stored at event point $(n-1)$ exceeds the maximum storage capacity, plus the amount of materials stored in processing units. In this case, there are no storage tanks or processing units to temporary store the materials produced.

$$\sum_{j\in\mathbf{J}_s}\sum_{i\in(\mathbf{I}_j\cap\mathbf{I}_s^P)\setminus\mathbf{I}^{\mathrm{R}}}\left(\rho_{sij}\sum_{n-\Delta n\le n'\le n} b_{ijn'n}\right)+ST_{s(n-1)}\le ST_s^{\max}+$$

$$+\sum_{j\in\mathbf{J}_s}\sum_{j'\in\mathbf{J}_s}\sum_{i\in(\mathbf{I}_j\cap\mathbf{I}_s^P)\setminus\mathbf{I}^R}\sum_{i'\in\left(\mathbf{I}_{j'}\cap\mathbf{I}_s^C\right)} bTd_{iji'j'n}+\sum_{j\in\mathbf{J}_s}\sum_{i\in(\mathbf{I}_j\cap\mathbf{I}_s^P)\setminus\mathbf{I}^R} bs_{ij(n+1)}$$

$$\forall s\in(\mathbf{S}^{IN}\cap\mathbf{S}^{FIS}),\, n \qquad (14)$$

$$\sum_{j\in\mathbf{J}_s}\sum_{i\in(\mathbf{I}_j\cap\mathbf{I}_s^P\cap\mathbf{I}^{\mathrm{R}})}\left(\rho_{sij}\sum_{n-1-\Delta n\le n'\le n-1} b_{ijn'(n-1)}\right)+ST_{s(n-1)}\le ST_s^{\max}+$$

$$+\sum_{j\in\mathbf{J}_s}\sum_{j'\in\mathbf{J}_s}\sum_{i\in(\mathbf{I}_j\cap\mathbf{I}_s^P\cap\mathbf{I}^R)}\sum_{i'\in\left(\mathbf{I}_{j'}\cap\mathbf{I}_s^C\right)} bTd_{iji'j'n}+\sum_{j\in\mathbf{J}_s}\sum_{i\in(\mathbf{I}_j\cap\mathbf{I}_s^P\cap\mathbf{I}^R)} bs_{ijn}$$

$$\forall s\in(\mathbf{S}^{IN}\cap\mathbf{S}^{FIS}),\, n>1 \qquad (15)$$

where variable $bs_{i,j,n}$ denotes the amount of materials stored in a unit j at event point n, previously produced by task i in this unit, which will be explained later.

When there is no direct material transfer between two related processing units, the amount through this direct transfer should be zero, similar to the indirect material transfer.

$$\sum_{i\in(\mathbf{I}_j\cap\mathbf{I}_s^P)}\sum_{i'\in\left(\mathbf{I}_{j'}\cap\mathbf{I}_s^C\right)} bTd_{iji'j'n}\le\min\left[B_j^{\max},B_{j'}^{\max}\right]\cdot zD_{jj'n}$$

$$\forall s\in(\mathbf{S}^{IN}\cap\mathbf{S}^{FIS}),\, j\neq j',\, j\in\mathbf{J}_s,\, j'\in\mathbf{J}_s,\, n \qquad (16)$$

where $B_j^{max}=\max_{i\in(\mathbf{I}_j\cap\mathbf{I}_s^P)}[B_{ij}^{max}]$ and $B_{j'}^{max}=\max_{i'\in(\mathbf{I}_{j'}\cap\mathbf{I}_s^C)}[B_{ij'}^{max}]$.

4.1.6 Sequencing constraints

Different tasks in the same unit

The start time of a unit j at event point $(n + 1)$ must always be after its end time at the previous event point n.

$$T^{\mathrm{f}}_{j(n+1)} \geq T^{\mathrm{s}}_{jn}$$

$$\forall j,\ n < N \qquad (17)$$

Different task in different unit

To make sure that correct operational sequences between production and consumption tasks in different processing units, we define continuous variables T_{sjn} to denote the time when a state s produced by a unit j is available to be transferred (i.e., consumed or stored) at event point n. Then we require that the time when a state s produced by a unit j is available to be consumed at event point $(n+1)$ is always after the time when the state is available at the previous event point n.

$$T_{sj(n+1)} \geq T_{sjn}$$

$$\forall s \in \mathbf{S}^{IN},\ j \in \mathbf{J}_s,\ n < N \qquad (18)$$

When a state s produced by a unit j is available at event point n, the production of this state in the same unit j must be completed at this event point n. In other words,

$$T_{sjn} \geq T^{\mathrm{f}}_{jn} - M\left(1 - \sum_{i \in \left(\mathbf{I}_j \cap \mathbf{I}^P_s\right)} \sum_{n-\Delta n \leq n' \leq n} w_{ijn'n}\right)$$

$$\forall s \in \mathbf{S}^{IN},\ j \in \mathbf{J}_s,\ \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}^P_s)} \rho_{sij} > 0,\ n \qquad (19)$$

If a unit j' processes a task i', which consumes state s at event point n and also receives materials from unit j, then this unit should start after the time that state s, which was produced by unit j from a non-recycling task at event point n, is available.

$$T_{sjn} \leq T^{\mathrm{s}}_{j'n} + M\left(1 - zI_{jj'n}\right)$$

$$\forall s \in \mathbf{S}^{IN},\ j, j' \in \mathbf{J}_s,\ j \neq j',\ \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}^P_s) \setminus \mathbf{I}^R} \rho_{sij} > 0,\ \sum_{i' \in (\mathbf{I}_{j'} \cap \mathbf{I}^C_s)} \rho_{si'j'} < 0,\ n \qquad (20)$$

Similarly, if a unit j' process a task i' at event point $(n + 1)$ and also receives materials from task j then the start time of this unit should be after the time that state s, which was produced by a unit j from a recycling task at event point n, is available.

$$T_{sjn} \leq T^{s}_{j'(n+1)} + M\left(1 - zI_{jj'n}\right)$$

$$\forall s \in \mathbf{S}^{IN}, j, j' \in \mathbf{J}_s,\ j \neq j',\ \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P \cap \mathbf{I}^R)} \rho_{sij} > 0,\ \sum_{i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C)} \rho_{si'j'} < 0,\ n < N \tag{21}$$

If the materials produced by a non-recycling task in a processing unit at event point *n* are not transferred to a consumption task in a processing unit at the same event point *n*, then all material should be stored in its dedicated storage tank, before another production task is processed in the unit. The start time of this consumption task at event point $(n+1)$ should always exceed the time that the state is available at event point *n*.

$$T_{sjn} \leq T^{s}_{j'(n+1)} + M\left(1 - \sum_{i' \in \left(\mathbf{I}_{j'} \cap \mathbf{I}_s^C\right)} \sum_{n+1 \leq n' \leq n+1+\Delta n} w_{i'j'(n+1)n'}\right)$$

$$\forall s \in \mathbf{S}^{IN}, j, j' \in \mathbf{J}_s,\ j \neq j',\ \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P) \setminus \mathbf{I}^R} \rho_{sij} > 0,\ \sum_{i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C)} \rho_{si'j'} < 0,\ n < N \tag{22}$$

In other words, a unit that processes a consumption task at event point (n + 1) are unconditionally sequenced with the units that process a related non-recycling production task at event point *n*. The units that are processing a consumption task at event point (n + 2) are unconditionally sequenced with units that process a related recycling production task at event point *n*.

$$T_{sjn} \leq T^{s}_{j'(n+2)} + M\left(1 - \sum_{i' \in \left(\mathbf{I}_{j'} \cap \mathbf{I}_s^C\right)} \sum_{n+2 \leq n' \leq n+2+\Delta n} w_{i'j'(n+2)n'}\right)$$

$$\forall s \in \mathbf{S}^{IN}, j, j' \in \mathbf{J}_s,\ j \neq j',\ \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P \cap \mathbf{I}^R)} \rho_{sij} > 0,\ \sum_{i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C)} \rho_{si'j'} < 0,\ n < N-1 \tag{23}$$

If there is a direct material transfer at event point *n* from a unit *j* that processes a non-recycling production task *i*, to a unit *j′* that processes a related consumption task *i′*, then the finish time of the unit *j′* at the previous event point (*n* – 1) must be before the finish time of the unit *j*.

$$T^{f}_{j'(n-1)} \leq T^{f}_{jn} + M\left(1 - zD_{jj'n}\right)$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}), j, j' \in \mathbf{J}_s,\ j \neq j',\ \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P) \setminus \mathbf{I}^R} \rho_{sij} > 0,\ \sum_{i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C)} \rho_{si'j'} < 0,\ n > 1 \tag{24}$$

If there is a material direct transfer at event point *n* from unit *j*, that process a recycling production task *i*, to unit *j′*, that process a related consumption task *i′*, then the finish time of unit *j′* at event point *n* must be before the finish time of unit *j*.

$$T^{\mathrm{f}}_{j'n} \le T^{\mathrm{f}}_{jn} + M\left(1 - zD_{jj'(n+1)}\right)$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}),\ j, j' \in \mathbf{J}_s,\ j \ne j',\ \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P \cap \mathbf{I}^R)} \rho_{sij} > 0,\ \sum_{i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C)} \rho_{si'j'} < 0,\ n < N \quad (25)$$

Finally, to avoid real time violations, between production and consumption tasks occurring at the same event for recycling tasks or at the previous event for non-recycling tasks the following constraints are introduced.

$$T^{\mathrm{f}}_{jn} \ge T^{\mathrm{s}}_{j'(n-1)} - M\left(1 - \textstyle\sum_{n-\Delta n \le n' \le n} w_{ijn'n}\right)$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}),\ j, j' \in \mathbf{J}_s,\ j \ne j',\ \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P) \setminus \mathbf{I}^R} \rho_{sij} > 0,\ \sum_{i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C)} \rho_{si'j'} < 0,\ n > 1 \quad (26)$$

$$T^{\mathrm{f}}_{jn} \ge T^{\mathrm{s}}_{j'n} - M\left(1 - \textstyle\sum_{n-\Delta n \le n' \le n} w_{ijn'n}\right)$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}),\ j, j' \in \mathbf{J}_s,\ j \ne j',\ \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P \cap \mathbf{I}^R)} \rho_{sij} > 0,\ \sum_{i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C)} \rho_{si'j'} < 0,\ n \quad (27)$$

4.1.7 Allowing processing units to store materials

In this work, we allow processing units to store materials for multiple event points. Generally, most existing mathematical models even though they allow processing units to store materials, they only allow these materials to be stored at the event point that they were produced. At the next event point, these materials should be either consumed by another task or transferred to the storage tanks. To avoid this case, we introduce an additional binary variable $ys_{i,j,n}$ as follows,

$$ys_{ijn} = \begin{cases} 1 & \text{if unit } j \text{ stores materials at event point } n\text{, previously produced by task } i \\ 0 & \text{otherwise} \end{cases}$$

We also introduce a new continuous variable $bs_{i,j,n}$ which denotes the amount of materials stored in a unit *j* at event point n, previously produced by task i in this unit. The amount of materials stored in a unit *j* cannot exceed its maximum capacity.

$$bs_{ijn} \le B^{\max}_{ij} \cdot ys_{ijn}$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}),\ j \in \mathbf{J}_s,\ i \in (\mathbf{I}_j \cap \mathbf{I}_S^P),\ n \quad (28)$$

Additionally, the amount of materials stored in a unit *j* at event point *n* cannot exceed the amount produced or stored at the previous event point (*n*–1).

$$bs_{ijn} \leq \sum_{n-1-\Delta n \leq n' \leq n} \left(\rho_{sij} \cdot b_{ijn'(n-1)}\right) + bs_{ij(n-1)}$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}), j \in \mathbf{J}_s, i \in (\mathbf{I}_j \cap \mathbf{I}_s^P), n > 1 \quad (29)$$

Materials stored in a processing unit can only be directly transferred to another unit that process a consumption task. Constraint (30) is used if materials are produced by non-recycling tasks, while constraint (31) is used if materials are produced by recycling tasks.

$$bs_{ijn} \geq bs_{ij(n-1)} - \sum_{j' \in \mathbf{J}_s} \sum_{i' \in \left(\mathbf{I}_{j'} \cap \mathbf{I}_s^C\right)} bTd_{iji'j'n}$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}), j \in \mathbf{J}_s, i \in (\mathbf{I}_j \cap \mathbf{I}_s^P) \backslash \mathbf{I}^R, n > 1 \quad (30)$$

$$bs_{ijn} \geq bs_{ij(n-1)} - \sum_{j' \in \mathbf{J}_s} \sum_{i' \in \left(\mathbf{I}_{j'} \cap \mathbf{I}_s^C\right)} bTd_{iji'j'(n+1)}$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}), j \in \mathbf{J}_s, i \in (\mathbf{I}_j \cap \mathbf{I}_s^P \cap \mathbf{I}^R), 1 < n < N \quad (31)$$

Finally, if a unit *j* holds some material at event point n, then it cannot process any task at this event point n.

$$\sum_{i \in \mathbf{I}_j} ys_{ijn} \leq 1 - \sum_{i \in \left(\mathbf{I}_j \cap \mathbf{I}_s^P\right)} \sum_{n-\Delta n \leq n' \leq n} \sum_{n \leq n'' \leq n'+\Delta n} w_{ijn'n''}$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}), j \in \mathbf{J}_s, n \quad (32)$$

4.1.8 Additional constraints

Several additional constraints are introduced to improve the performance of the proposed model. Constraints (33)-(36) relate $w_{ijnn'}$ with $zI_{jj'n}$. More specifically, if a unit *j′* process a consumption task *i′*, and there is indirect material transfer between units *j* and *j′* then unit *j* must process the related production task *i* according to (33). Similarly, if a unit *j* processes a production task *i*, and there is an indirect material transfer between units *j* and *j′* then unit *j′* must process the related consumption task *i′* according to (34). While (33) and (34) are used for non-recycling production tasks, constraints (35) and (36) are for recycling production tasks.

$$\sum_{n-\Delta n \leq n' \leq n} w_{ijn'n} \geq \sum_{n \leq n' \leq n+\Delta n} w_{i'j'nn'} + zI_{jj'n} - 1$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{UIS}), j, j' \in \mathbf{J}_s, j \neq j', i \in (\mathbf{I}_j \cap \mathbf{I}_s^P) \backslash \mathbf{I}^R, i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C), n \quad (33)$$

$$\sum_{n \le n' \le n+\Delta n} w_{i'j'nn'} \ge \sum_{n-\Delta n \le n' \le n} w_{ijn'n} + zI_{jj'n} - 1$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{UIS}),\ j, j' \in \mathbf{J}_s,\ j \neq j',\ i \in (\mathbf{I}_j \cap \mathbf{I}_s^P) \backslash \mathbf{I}^R,\ i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C),\ n \quad (34)$$

$$\sum_{n-\Delta n \le n' \le n} w_{ijn'n} \ge \sum_{n+1 \le n' \le n+1+\Delta n} w_{i'j'(n+1)n'} + zI_{jj'(n+1)} - 1$$

$$\forall\ s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{UIS}),\ j, j' \in \mathbf{J}_s,\ j \neq j',\ i \in (\mathbf{I}_j \cap \mathbf{I}_s^P \cap \mathbf{I}^R),\ i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C),\ n < N \quad (35)$$

$$\sum_{n+1 \le n' \le n+1+\Delta n} w_{i'j'(n+1)n'} \ge \sum_{n-\Delta n \le n' \le n} w_{ijn'n} + zI_{jj'(n+1)} - 1$$

$$\forall\ s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{UIS}),\ j, j' \in \mathbf{J}_s,\ j \neq j',\ i \in (\mathbf{I}_j \cap \mathbf{I}_s^P \cap \mathbf{I}^R),\ i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C),\ n < N \quad (36)$$

If an intermediate state s has a FIS policy, then a unit j that transfers materials at unit j', then unit j can either process a production task or store materials at event point n. Constraints (37) and (38) handle cases with non-recycling production tasks, while (39) and (40) handle cases with recycling tasks.

$$\sum_{n-\Delta n \le n' \le n} w_{ijn'n} + ys_{ijn} \ge \sum_{n \le n' \le n+\Delta n} w_{i'j'nn'} + zI_{jj'n} - 1$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}),\ j, j' \in \mathbf{J}_s,\ j \neq j',\ i \in (\mathbf{I}_j \cap \mathbf{I}_s^P) \backslash \mathbf{I}^R,\ i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C),\ n \quad (37)$$

$$\sum_{n \le n' \le n+\Delta n} w_{i'j'nn'} \ge \sum_{n-\Delta n \le n' \le n} w_{ijn'n} + ys_{ijn} + zI_{jj'n} - 1$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}),\ j, j' \in \mathbf{J}_s,\ j \neq j',\ i \in (\mathbf{I}_j \cap \mathbf{I}_s^P) \backslash \mathbf{I}^R,\ i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C),\ n \quad (38)$$

$$\sum_{n-\Delta n \le n' \le n} w_{ijn'n} + ys_{ijn} \ge \sum_{n+1 \le n' \le n+1+\Delta n} w_{i'j'(n+1)n'} + zI_{jj'(n+1)} - 1$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}),\ j, j' \in \mathbf{J}_s,\ j \neq j',\ i \in (\mathbf{I}_j \cap \mathbf{I}_s^P \cap \mathbf{I}^R),\ i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C),\ n < N \quad (39)$$

$$\sum_{n+1 \le n' \le n+1+\Delta n} w_{i'j'(n+1)n'} \ge \sum_{n-\Delta n \le n' \le n} w_{ijn'n} + ys_{ijn} + zI_{jj'(n+1)} - 1$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}),\ j, j' \in \mathbf{J}_s,\ j \neq j',\ i \in (\mathbf{I}_j \cap \mathbf{I}_s^P \cap \mathbf{I}^R),\ i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C),\ n < N \quad (40)$$

In the same manner we relate $w_{ijnn'}$ and ys_{ijn} with $zD_{jj'n}$.

$$\sum_{n-\Delta n \le n' \le n} w_{ijn'n} + ys_{ijn} \ge \sum_{n \le n' \le n+\Delta n} w_{i'j'nn'} + zD_{jj'n} - 1$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}), j, j' \in \mathbf{J}_s, j \neq j', i \in (\mathbf{I}_j \cap \mathbf{I}_s^P) \backslash \mathbf{I}^R, i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C), n \quad (41)$$

$$\sum_{n \le n' \le n+\Delta n} w_{i'j'nn'} \ge \sum_{n-\Delta n \le n' \le n} w_{ijn'n} + ys_{ijn} + zD_{jj'n} - 1$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}), j, j' \in \mathbf{J}_s, j \neq j', i \in (\mathbf{I}_j \cap \mathbf{I}_s^P) \backslash \mathbf{I}^R, j' \in \mathbf{J}_s, i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C), n \quad (42)$$

$$\sum_{n-\Delta n \le n' \le n} w_{ijn'n} + ys_{ijn} \ge \sum_{n+1 \le n' \le n+1+\Delta n} w_{i'j'nn'} + zD_{jj'(n+1)} - 1$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}), j, j' \in \mathbf{J}_s, j \neq j', i \in (\mathbf{I}_j \cap \mathbf{I}_s^P \cap \mathbf{I}^R), i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C), n < N \quad (43)$$

$$\sum_{n+1 \le n' \le n+1+\Delta n} w_{i'j'(n+1)n'} \ge \sum_{n-\Delta n \le n' \le n} w_{ijn'n} + ys_{ijn} + zD_{jj'(n+1)} - 1$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}), j, j' \in \mathbf{J}_s, j \neq j', i \in (\mathbf{I}_j \cap \mathbf{I}_s^P \cap \mathbf{I}^R), i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C), n < N \quad (44)$$

Objective functions

As already discussed, two objectives have been considered. While constraint (45) is the objective for maximization of productivity, constraint (46) handles the case of minimization of makespan.

$$z = \sum_s p_s \sum_{j \in \mathbf{J}_s} \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_S^P)} \sum_n \sum_{n \le n' \le n+\Delta n} \rho_{ijs} \cdot b_{ijnn'} \quad (45)$$

$$MS \ge T_{jn}^{\mathrm{f}} \qquad \forall j, n = N \quad (46)$$

In the minimization of makespan problem, the total demand should be satisfied.

$$ST_{s,n''} + \sum_{i \in (\mathbf{I}_S^P \cap \mathbf{I}^R)} \rho_{i,s} \sum_{n-\Delta n \le n' \le n} b_{i,n',n} \ge D_s \qquad \forall s \in \mathbf{S}^p, n'' = N \quad (47)$$

Finally, (48) and (49) denote all the continuous and binary variables of the model, respectively

$$b_{ijnn'}, bs_{ijn}, bTi_{iji'j'n}, bTd_{iji'j'n}, MS, ST_{sn}, T_{sjn}, T_{jn}^{\mathrm{s}}, T_{jn}^{\mathrm{f}} \ge 0 \quad (48)$$

$w_{ijnn'}, ys_{ijn}, zD_{jj'n}, zI_{jj'n} \in \{0, 1\}$ (49)

We complete the mathematical model **M1**, which consists of constraints (1)-(45) and (48-49) for maximization of productivity, and (1)-(44) and (46)-(49) for minimization of makespan. We consider two different variations of this model.

4.2 Model M2

In the mathematical model **M2**, we also use the unit-specific event-based approach with timing variables based on units. The main difference from the mathematical model **M1** is that related production and consumption tasks are not allowed to take place at the same event point. Therefore, we use the following material balance constraints instead.

$$ST_{sn} = ST_{s(n-1)} + \sum_{i \in (\mathbf{I}_s^P \cap \mathbf{I}^R)} \sum_{j \in (\mathbf{J}_s \cap \mathbf{J}_i)} \sum_{n-1-\Delta n \le n' \le n-1} \rho_{sij} b_{ijn'(n-1)} +$$

$$+ \sum_{i \in \boldsymbol{I}_s^C} \sum_{j \in (\mathbf{J}_s \cap \mathbf{J}_i)} \sum_{n \le n' \le n+\Delta n} \rho_{sij} b_{ijnn'}$$

$$\forall s, n > 1 \qquad (50)$$

$$ST_{sn} = ST0_s + \sum_{i \in \boldsymbol{I}_s^C} \sum_{j \in (\mathbf{J}_s \cap \mathbf{J}_i)} \sum_{n \le n' \le n+\Delta n} \rho_{sij} b_{ijnn'}$$

$$\forall s, n = 1 \qquad (51)$$

The mathematical model **M2** consists of constraints (1)-(2), (5)-(7), (8)-(13), (15), (17), (19)-(20), (22)-(23), (25), (27)-(29), (31)-(32), (35)-(36), (39)-(40), (43)-(45), (48)-(49) and (50)-(51) for maximization of productivity and (1)-(2), (5)-(7), (8)-(13), (15), (17), (19)-(20), (22)-(23), (25), (27)-(29), (31)-(32), (35)-(36), (39)-(40), (43)-(44) and (46)-(47), (48)-(49), (50)-(51) for minimization of makespan.

5 Computational studies

To examine the performance of the proposed mathematical models **M1** and **M2**, we revisit the motivating example 1 and solve additional three motivating examples. The maximum computational time is one hour for all examples. The optimality gap is set to zero. All examples are solved using CPLEX 12/GAMS 24.6.1. on a desktop computer with Intel® Core™ i5-2500 3.3 GHz and 8 GB RAM running Windows 7.

Revisit of Motivating Example 1

We use the proposed models **M1** and **M2** to solve the motivating example 1. The optimal solution of 500.00 cu is generated in less than 0.1 CPU s for both models. The model statistics are provided in Table 3. It involves 12 binary variables, 28 continuous variables, and 60 constraints for model **M1** and 17 binary variables, 39 continuous variables, and 86 constraints for model **M2**. The optimal schedule is the same as that illustrated in Figure 4. As discussed before, intermediate state S2 is held in unit J2 after production because of small storage capacity.

Table 3 Computational results for motivating examples 1-3

Motivating Example	Model	Number of event points	CPU time (s)	RMILP	MILP (h)	Bin. Var.	Cont. Var.	Constr.
1 (H = 8 h)	**LF2010**[a]	3	0.11	500.00	300.00	6	29	41
	VS2013[b]	3	0.09	500.00	300.00	14	33	72
	MH2019[c]	4 (ΔR=1)	0.08	500.00	300.00	6	34	78
	M1	2	0.02	500.00	500.00	12	28	60
	M2	3	0.03	500.00	500.00	17	39	86
2 (H=12 h)	**LF2010**	7	5.4	3281.50	2385.32	56	235	518
	VS2013	7	29.5	3281.50	2392.46	256	403	1268
	MH2019	9 (ΔR=2)	30.8	3332.63	2385.32	120	361	962
	M1	7	39.2	3281.50	2433.16	218	463	1442
	M2	7	38.0	3281.50	2433.16	216	459	1430
3 (H=12 h)	**LF2010**	9 (Δn=1)	58.4	3879.34	887.68	187	507	1727
	VS2013	9	5.6	3879.34	989.03	541	723	2549
	MH2019	9 (ΔR=2)	0.2	887.68	887.68	165	506	1424
	M1	9	10.4	3879.34	1033.60	453	813	2935
	M2	9	10.5	3879.84	1033.60	453	813	2935
4 (H=12 h)	**LF2010**	9	-	-	Infeasible	99	419	1019
	VS2013	9	-	-	infeasible	541	723	2549
	MH2019	9 (ΔR=2)	-	-	infeasible	165	510	1424
	M1	9	27.9	4297.11	2503.15	453	813	2935
	M2	9	27.3	4297.11	2503.15	453	813	2935

[a] Li and Floudas[12] model. [b] Vooradi and Shaik[15] model. [c] Mostafaei and Harjunkoski[21] model

As illustrated in Table 3, it is possible to generate the optimum solution for the motivating example using the proposed models **M1** and **M2** as both of them allow production units to store materials over multiple event points. As already discussed, even though the model of Vooradi and Shaik[15] allows materials to be stored in the processing unit during an event point *n*, these materials cannot be stored to the processing unit for the next event points. Similarly, Li and Floudas[12] and Mostafaei and Harjunkoski[21] do not allow processing units to store materials for the successive event points. As a result, both proposed models **M1** and **M2** can generate a significantly better solution.

Motivating Example 2

This example is very similar to Example 2c from Li *et al.*[16] but a modified maximum capacity of state S7 of 10 mu. The objective is to maximize productivity. Similarly to the Motivating Example 1, we use the model of Li and Floudas[12], Vooradi and Shaik[15], Mostafaei and Harjunkoski[21] and the proposed model **M1** and **M2** to solve this motivating example. Table 3 provides the computational results. From Table 3, it seems that both proposed mathematical models **M1** and **M2** can generate a solution of 2433.16 mu, whilst the model of Li and Floudas[12], Vooradi and Shaik[15] and Mostafaei and Harjunkoski[21] are only able to provide a suboptimum solution (2392.46 mu and 2385.32 mu respectively). Such a difference is mainly because the proposed models **M1** and **M2** allow production units to storage materials over multiple event points. The optimal schedule from model **M1** is illustrated in Figure 7. As seen from Figure 7, unit J2 produces 50 mu from 6.3h to 8.9h by processing task I2 at event point N4. Those materials can be stored in the processing unit J2 and processed in the same unit at event point N7. However, this is not possible with the model of Vooradi and Shaik,[15] and as a result, less materials can be produced during the same period as depicted in Figure 8 (2.80 mu by processing task I2 at event point N4 and 39.03 mu by processing task I2 at event point N6), which leads to less productivity and as a result a suboptimum solution.

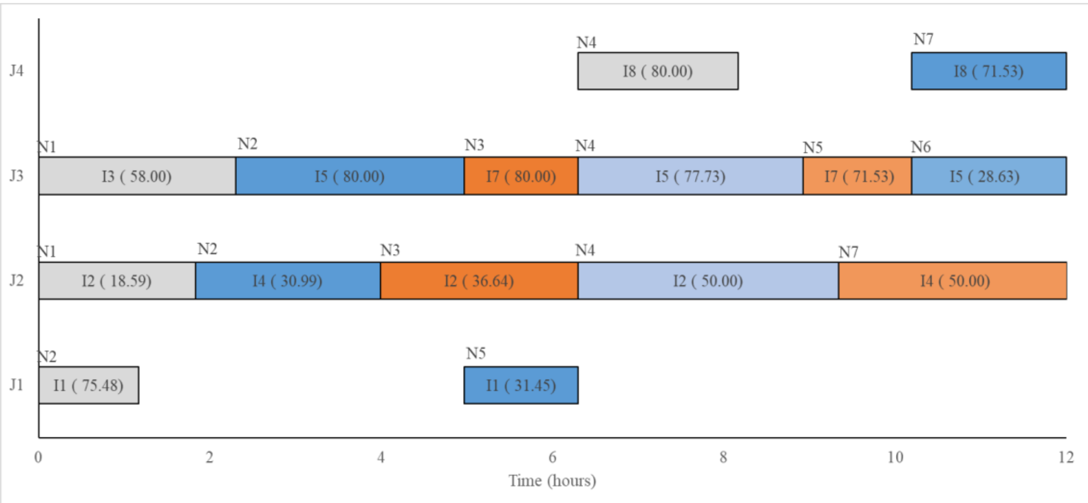

Figure 7 Optimal schedule for Motivating Example 2 using model M1

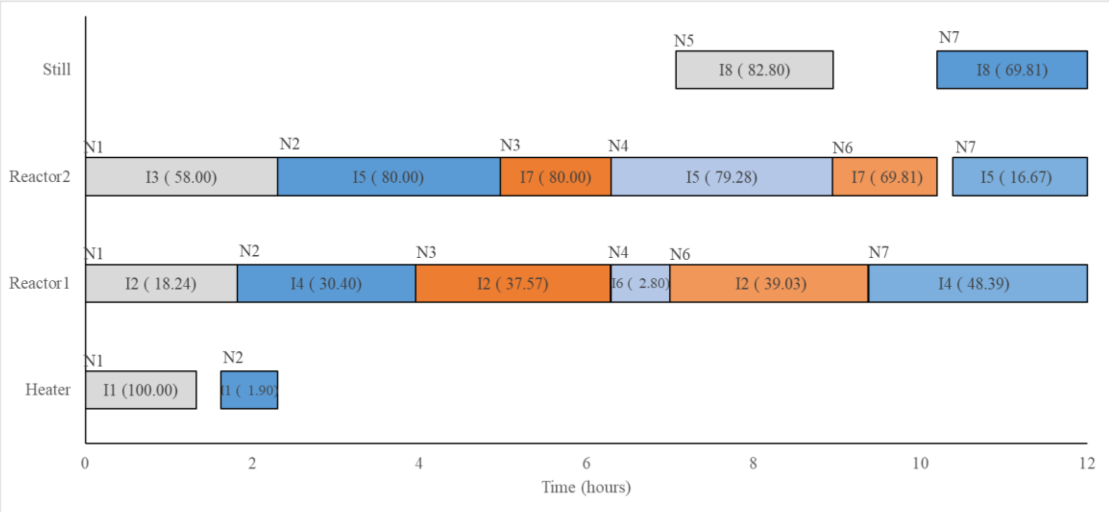

Figure 8 Schedule for Motivating Example 2 using the model of Vooradi and Shaik[15]

Motivating Example 3

Motivating example 3 is quite similar to the Example 3c from Li *et al.*[16]. The maximum capacity for states S5, S6 and S7 has changed to 10 mu. Additionally, the initial amount of materials for states S6 and S7 is changed to 0 mu. Similar to Motivating Example 2, we use the model of Li and Floudas[12], Vooradi and Shaik[15], Mostafaei and Harjunkoski[21] and the proposed model **M1** and **M2** to solve this motivating example. The computational results are provided in Table 3. From Table 3, both proposed models **M1** and **M2** can generate a better solution than the models from the literature (1033.60 cu). Such difference in the solution can be explained by examining the optimal schedule generated by using model **M1** (see Figure 9) and the model of Vooradi and Shaik[15] (see Figure 10). Since there is small storage capacity for states S6 and S7, unit J4 can process batches with small sizes with the model of Vooradi and Shaik[15]. More specifically, I9 is processed in unit J4 in event points N3, N5 and N7 with batch sizes of 20.00 mu, 38.40 mu and 53.73 mu respectively. On the other hand, model **M1** can produce significantly higher amounts of states S6 and S7, since those excessive amounts can be temporarily stored in the processing units before transferred to another one. For instance, with model **M1** unit J4 also processes three batches of I9 at event points N3, N6 and N8 with batch size 25.00 mu, 43.00 mu and 90.00 mu respectively.

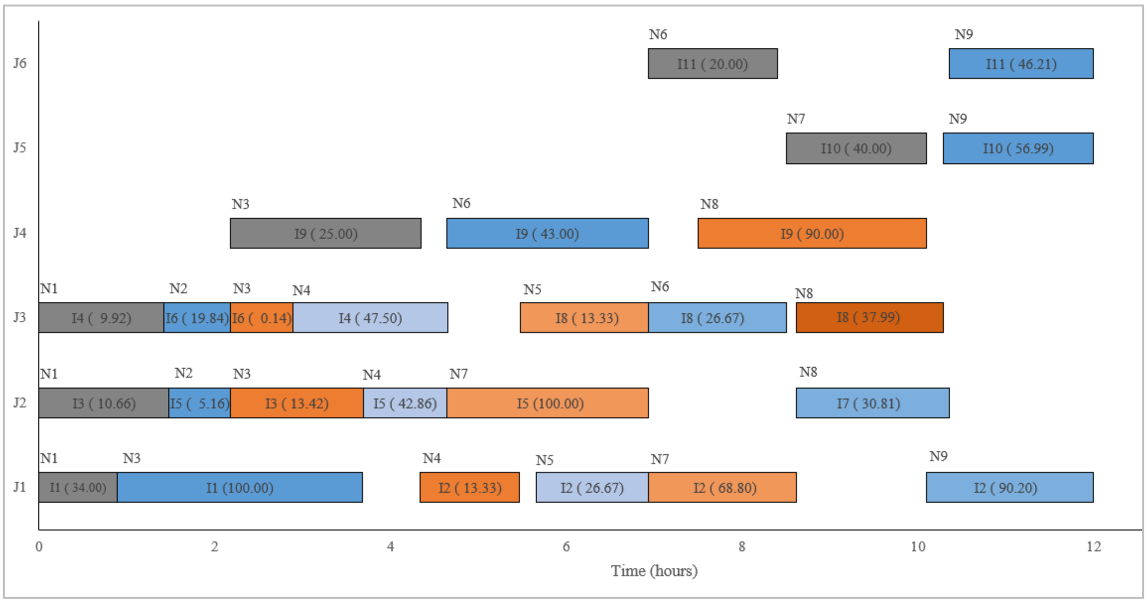

Figure 9 Optimal schedule for Motivating Example 3 using model M1

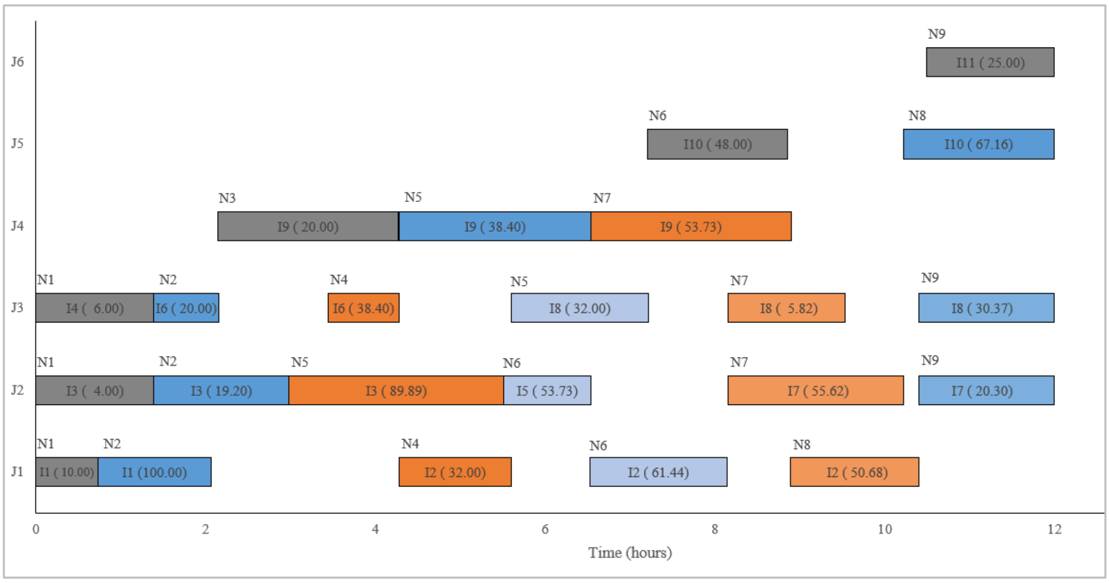

Figure 10 Optimal schedule for Motivating Example 3 using the model of Vooradi and Shaik[15]

Motivating Example 4

This example is also quite similar to the Example 3c from Li *et al.*[16]. The maximum capacity for state S7 is 10 mu. Additionally, in the first event point, 40 mu of S7 are stored in unit J4 at the first event point. Since the models of Vooradi and Shaik[15] and Mostafaei and Harjunkoski[21] do not allow materials to be stored for multiple event points, they fail to generate a feasible solution. On the other hand, in the proposed models **M1** and **M2**, allows materials to be stored in processing units for multiple event points and, as a result, they can generate the optimum solution of 2503.15 mu in less than 30 s.

Benchmark Examples

To further examine the performance of the proposed mathematical models **M1** and **M2**, we solve in total nine examples from the literature[1, 8, 16]. The data, as well as the STN representations for all those examples, are presented in the **Supplementary Material**. The maximum computational time is one hour, and the optimality gap is zero. All cases are solved using CPLEX 12/GAMS 24.6.1. on a desktop computer with Intel® Core™ i5-2500 3.3 GHz and 8 GB RAM running Windows 7. It should also be noted that we only compare our models with the model of Vooradi and Shaik[15] (denoted as **VS2013**) since they incorporate similar features. The model of Mostafaei and Harjunkoski[21] is very similar to the model of Shaik and Floudas,[11] which requires more event points in some examples, as demonstrated in the Motivating Example 1 and Vooradi and Shaik[15]. Detailed comparison of our models with Shaik and Floudas[11], Li and Floudas[12], Susarla et al.[8], and Mostafaei and Harjunkoski[21] will be presented in our next contribution.

The computational results for Examples 1-9 with UIS policy for maximization of productivity, are presented in Tables 4 and 5. From Tables 4 and 5, it seems that both the model of Vooradi and Shaik[15] and the model **M2** require the same number of event points since both models do not allow related production and consumption tasks to take place at the same event point. Nevertheless, it seems that model **M2** requires fewer binary variables in some cases. For instance, in Example 3d, the model of Vooradi and Shaik[15] requires 263 binary variables, while **M2** requires 245 binary variables. The reason is that **M2** only examines if there is a material transfer between processing units, whilst the model of Vooradi and Shaik[15] tests if there is a material transfer from a production task to a related consumption task. In a multipurpose batch process facility, a processing unit can process more than one tasks. Therefore, two processing units can process two or more tasks which are related to the same state. In such a case, the **M2** only requires one binary decision variable, while the model of Vooradi and Shaik[15] requires two or more binary decision variables. As a result, **M2** can lead to smaller model size and less computational time. For instance, **M2** requires 36% (15.1 s vs 23.6 s), 87.7% (146.4 s vs 1191 s) and 62.2% (41.7 s vs 110.3 s) less computational time for Examples 2d, 3b and 3d than the model of Vooradi and Shaik,[15] respectively. Additional constraints (33)-(44) can also improve the performance of the proposed models. For instance, even though both models **M2** and the model of Vooradi and Shaik[15] lead to the same model size for Example 1d, **M2** requires 51.8% less computational time (10.5s vs 21.8 s).

Table 4 Computational results for Examples 1-3 with maximization of productivity (UIS policy)

Example	Model	Number of event points	CPU time (s)	RMILP	MILP (h)	Bin. Var.	Cont. Var.	Constr.
Ex1a	**VS2013**	4	**0.125**	2000.00	1840.17	32	90	177
(H = 8 h)	**M1**	2	**0.031**	2000.00	1840.17	18	54	105
	M2	4	**0.031**	2000.00	1840.17	32	102	198
Ex1b	**VS2013**	5	**0.125**	3000.00	2628.19	41	113	226
(H = 10h)	**M1**	3	**0.046**	3000.00	2628.19	27	80	163
	M2	5	**0.047**	3000.00	2628.19	41	128	256
Ex1c	**VS2013**	6	**0.250**	4000.00	3463.62	50	136	275
(H = 12h)	**M1**	4	**0.062**	4000.00	3463.62	36	106	221
	M2	6	**0.109**	4000.00	3463.62	50	154	314
Ex1d	**VS2013**	9	21.8	6601.65	5038.05	77	205	422
(H = 16h)	**M1**	7	12.9	6601.65	5038.05	63	184	395
	M2	9	10.5	6601.65	5038.05	77	232	488
Ex2a	**VS2013**	4	**0.125**	1730.87	1498.57	62	178	384
(H = 8 h)	**M1**	4	**0.063**	1730.87	1498.57	64	180	396
	M2	4	**0.078**	1730.87	1498.57	56	178	377
Ex2b	**VS2013**	5	0.17	2436.69	1962.69	80	225	496
(H = 10h)	**M1**	5	0.22	2436.69	1962.69	80	227	510
	M2	5	0.20	2436.69	1962.68	72	225	491
Ex2c	**VS2013**	6	0.48	3076.62	2658.52	98	272	608
(H = 12h)	**M1**	6	0.42	3076.62	2658.52	96	274	624
	M2	6	0.42	3076.62	2658.52	88	272	605
Ex2d	**VS2013**	8	23.6	4291.67	3738.38	134	366	832
(H = 16h)	**M1**	8	15.6	4291.67	3738.38	128	368	852
	M2	8	15.1	4291.67	3738.38	120	366	833
Ex3a	**VS2013**	5	**1.92**	2100.00	1583.44	123	311	741
(H = 8h)	**M1**	5	**0.90**	2100.00	1583.44	130	316	793
	M2	5	**0.86**	2100.00	1583.44	115	316	776
Ex3b	**VS2013**	7	**1191**	3369.69	2358.20	179	441	1077
(H = 10h)	**M1**	7	**146.4**	3369.69	2358.20	182	448	1155
	M2	7	**157.0**	3369.69	2358.20	167	448	1138
Ex3c	**VS2013**	7	1.31	3465.63	3041.27	179	441	1077
(H = 12h)	**M1**	7	1.22	3465.63	3041.27	182	448	1155
	M2	7	1.14	3465.63	3041.27	167	448	1138
Ex3d	**VS2013**	10	**110.3**	5225.86	4262.80	263	636	1581
(H = 16h)	**M1**	10	**42.8**	5225.86	4262.80	260	646	1698
	M2	10	**41.7**	5225.86	4262.80	245	646	1681

Note. Δn = 0 for all examples. VS2013: Vooradi and Shaik[15] model.

Table 5 Computational results for Examples 4-9 with maximization of productivity (UIS policy)

Example	Model	Number of event points	CPU time (s)	RMILP	MILP (h)	Bin. Var.	Cont. Var.	Constr.
Ex4	**VS2013**	6	**0.124**	7.5000	5.3225	65	157	358
(H=15 h)	**M1**	4	**0.078**	7.5000	5.3225	48	127	298
	M2	6	**0.078**	7.5000	5.3225	65	187	434
Ex5	**VS2013**	5	**0.109**	14.00	10.00	36	98	201
(H=6 h)	**M1**	3	**0.032**	14.00	10.00	24	71	152
	M2	5	**0.031**	14.00	10.00	36	113	237
Ex6	**VS2013**	5	**0.141**	300.00	210.00	49	138	283
(H=9 h)	**M1**	3	**0.047**	300.00	210.00	33	100	211
	M2	5	**0.031**	300.00	210.00	49	158	327
Ex7	**VS2013**	5	**0.125**	80.00	58.99	54	147	301
(H=76 h)	**M1**	2	**0.031**	80.00	58.99	24	71	145
	M2	5	**0.046**	80.00	58.99	54	167	351
Ex8	**VS2013**	6	**0.093**	400.00	400.00	44	130	289
(H=10 h)	**M1**	4	**0.032**	400.00	400.00	32	106	237
	M2	6	**0.047**	400.00	400.00	44	154	337
Ex9	**VS2013**	10	**0.109**	400.00	400.00	76	218	497
(H=10 h)	**M1**	8	**0.062**	400.00	400.00	64	210	485
	M2	10	**0.032**	400.00	400.00	76	258	585

Note. $\Delta n = 0$ for all examples. VS2013: Vooradi and Shaik[15] model.

Mathematical model **M1** requires a smaller number of event points in most cases since related production and consumption tasks are allowed to take place at the same event point. For instance, the model **M1** requires two event points less than the models **M2** and the model of Vooradi and Shaik[15]for Examples 1a-d, 8 and 9. As a result, model **M1** leads to the smallest model size with less number of binary variables, continuous variables and constraints, which makes it more efficient than the mathematical model Vooradi and Shaik[15]. Nevertheless, it seems that both mathematical models **M1** and **M2** require similar computational time to generate the optimal solution, mainly because they both models can solve all examples in less than three minutes. From Tables 4 and 5, it can be concluded that the models **M1** and **M2** reduced the computational time by one order of magnitude for most examples in comparison to **VS2013**.

Table 6 Computational results for Examples 1-3 with maximization of productivity (FIS policy)

Example	Model	Number of event points	CPU time (s)	RMILP	MILP (h)	Bin. Var.	Cont. Var.	Constr.
Ex1a (H = 8 h)	**VS2013**	4	0.094	2000.00	1840.17	64	102	273
	M1	2	0.031	2000.00	1840.17	38	72	175
	M2	4	0.047	2000.00	1840.17	74	126	358
Ex1b (H = 10h)	**VS2013**	5	**0.234**	3000.00	2628.19	81	129	352
	M1	3	**0.046**	3000.00	2628.19	58	107	279
	M2	5	**0.062**	3000.00	2628.19	94	159	462
Ex1c (H = 12h)	**VS2013**	6	0.23	4000.00	3463.62	98	156	431
	M1	4	0.17	4000.00	3463.62	78	142	383
	M2	6	0.25	4000.00	3463.62	114	192	566
Ex1d (H = 16h)	**VS2013**	9	45.3	6601.65	5038.05	149	240	668
	M1	7	40.3	6601.65	5038.05	138	247	695
	M2	9	43.5	6601.65	5038.05	174	291	878
Ex2a (H = 8 h)	**VS2013**	4	**0.125**	1730.87	1498.57	142	220	665
	M1	4	**0.078**	1730.87	1498.57	122	256	774
	M2	4	**0.078**	1730.87	1498.57	120	252	755
Ex2b (H = 10h)	**VS2013**	5	0.45	2436.69	1962.69	180	281	866
	M1	5	0.30	2436.69	1962.69	154	325	999
	M2	5	0.20	2436.69	1962.69	152	321	980
Ex2c (H = 12h)	**VS2013**	6	0.66	3076.62	2658.52	218	342	1067
	M1	6	0.50	3076.62	2658.52	186	394	1224
	M2	6	0.47	3076.62	2658.52	184	390	1205
Ex2d (H = 16h)	**VS2013**	8	34.6	4291.67	3738.38	294	464	1469
	M1	8	22.2	4291.67	3738.38	250	532	1674
	M2	8	23.7	4291.67	3738.38	248	528	1655
Ex3a (H = 8h)	**VS2013**	5	3.32	2100.00	1583.44	293	387	1321
	M1	5	1.80	2100.00	1583.44	245	437	1542
	M2	5	1.92	2100.00	1583.44	245	437	1531
Ex3b (H = 10h)	**VS2013**	7	**976.3**	3369.69	2358.20	417	555	1935
	M1	7	**383.8**	3369.69	2358.20	349	625	2233
	M2	7	**364.9**	3369.69	2358.20	349	625	2233
Ex3c (H = 12h)	**VS2013**	7	2.90	3465.63	3041.27	417	555	1935
	M1	7	1.47	3465.63	3041.27	349	625	2244
	M2	7	1.42	3465.63	3041.27	349	625	2233
Ex3d (H = 16h)	**VS2013**	10	**155.6**	5225.86	4262.80	603	807	2856
	M1	10	**85.5**	5225.86	4262.80	505	907	3297
	M2	10	**82.2**	5225.86	4262.80	505	907	3286

$\Delta n = 0$ for all examples. VS2013: Vooradi and Shaik[15] model.

Tables 6 and 7 present the computational results for Examples 1-10 with FIS policy for maximization of productivity. Both mathematical models **M2** and the model of Vooradi and Shaik[15] require the same number of event points for all examples to generate the optimal solution. As we introduce additional binary variables to allow processing units to store materials for multiple event points, model **M2** leads to a big larger model size for Examples 1a-1d, 4, 5, and 7 where the model of Vooradi and Shaik[15] does not need to allow tasks to span over multiple event points (i.e., $\Delta n = 0$) to generate the optimal solution. However, model **M2** requires similar computational time as the model of Vooradi and Shaik[15] for these examples. On the other hand, model **M2** requires 15.5%-16.5% fewer binary variables for Examples 2a-2d, 3a-3d due to fact that the proposed model only uses binary variables to examine whether there is a material transfer between two units. More importantly, both models **M2** and **M1** do not require to allow tasks to span over multiple event points in any case due to allowing processing units to store materials over multiple event points. Therefore, both proposed models lead to significantly smaller model size with less binary and continuous variables and constraints in Examples 6, 8 and 9. For instance, both models **M2** and **M1** require 61.9% (128 vs 336) and 53.5% (156 vs 336) fewer binary variables than the model of Vooradi and Shaik[15] to generate the optimal solution for Example 9 respectively. Such reduction in the model size leads to one magnitude less computational time required for both proposed models **M1** and **M2** in comparison to the model of Vooradi and Shaik[15].

Table 7 Computational results for Examples 4-10. Maximization of productivity (FIS)

Example	Model	Number of event points	CPU time (s)	RMILP	MILP (h)	Bin. Var.	Cont. Var.	Constr.
Ex4	**VS2013**	6 (Δn=0)	0.312	7.5000	5.3225	149	341	618
(H=15 h)	**M1**	4 (Δn=0)	0.218	7.5000	5.3225	105	168	560
	M2	6 (Δn=0)	0.172	7.5000	5.3225	157	246	845
Ex5	**VS2013**	5 (Δn=0)	**0.141**	14.00	10.00	76	114	327
(H=6 h)	**M1**	3 (Δn=0)	**0.062**	14.00	10.00	52	92	265
	M2	5 (Δn=0)	**0.047**	14.00	10.00	84	144	438
Ex6	**VS2013**	5 (Δn=1)	**0.265**	300.00	210.00	144	182	547
(H=9 h)	**M1**	3 (Δn=0)	**0.078**	300.00	210.00	78	130	380
	M2	5 (Δn=0)	**0.078**	300.00	210.00	128	202	636
Ex7	**VS2013**	5 (Δn=0)	**0.125**	80.00	58.99	114	171	490
(H=76 h)	**M1**	2 (Δn=0)	**0.047**	80.00	58.99	50	91	243
	M2	5 (Δn=0)	**0.062**	80.00	58.99	122	211	638
Ex8	**VS2013**	6 (Δn=3)	**0.343**	400.00	400.00	152	198	665
(H=10 h)	**M1**	4 (Δn=0)	**0.047**	400.00	400.00	64	134	409
	M2	6 (Δn=0)	**0.062**	400.00	400.00	92	192	597
Ex9	**VS2013**	10 (Δn=7)	**2.10**	400.00	400.00	336	422	1453
(H=10 h)	**M1**	8 (Δn=0)	**0.23**	400.00	400.00	128	266	849
	M2	10 (Δn=0)	**0.11**	400.00	400.00	156	324	1037

VS2013: Vooradi and Shaik[15] model.

Table 8 Computational results for Examples 1-3 with minimization of makespan (UIS policy)

Example	Model	Number of event points	CPU time (s)	RMILP	MILP (h)	Bin. Var.	Cont. Var.	Constr.
Ex1a	**VS2013**	14	**>3600[a]**	24.24	27.88	122	320	672
(D_{S4}=2000 cu)	**M1**	12	**412**	24.24	27.88	108	314	690
	M2	14	**639**	24.24	27.88	122	362	783
Ex1b	**VS2013**	23	**>3600[b]**	48.47	52.07	203	527	1113
(D_{S4}=4000 cu)	**M1**	21	**1004**	48.47	52.07	189	548	1212
	M2	23	**1978**	48.47	52.07	203	596	1305
Ex2a	**VS2013**	9	173.4	10.78	19.34	152	413	953
(D_{S8}=200 cu)	**M1**	9	99.6	18.68	19.34	138	415	963
(D_{S9}=200 cu)	**M2**	9	106.1	18.68	19.34	136	413	952
Ex2b	**VS2013**	20	>3600[c]	26.12	46.11	350	930	2185
(D_{S8}=500 cu)	**M1**	20	>3600[d]	45.57	46.11	312	930	2206
(D_{S9}=400 cu)	**M2**	20	>3600[e]	45.57	46.11	314	932	2217
Ex3a	**VS2013**	7	0.578	10.00	13.37	179	441	1089
(D_{S12}=100 cu)	**M1**	7	0.546	11.25	13.37	167	448	1145
(D_{S13}=200 cu)	**M2**	7	0.702	11.25	13.37	167	448	1145
Ex3b	**VS2013**	10	0.889	12.50	17.02	263	636	1593
(D_{S12}=250 cu)	**M1**	10	0.873	14.27	17.02	245	646	1688
(D_{S13}=250 cu)	**M2**	10	0.874	14.27	17.02	245	646	1688

Note that Δn = 0 in all cases. [a] Relative Gap 0.19%. [b] Relative Gap 0.01%. [c] Relative Gap 17.3%. [d] Relative Gap 1.17% [e] Relative Gap 1.17%. VS2013: Vooradi and Shaik[15] model.

The computational results for examples using minimization of makespan as objective are presented in Tables 8 and 9. While Table 8 depicts the results with UIS policy, Table 9 gives the results with FIS policy. From Table 8, it seems that mathematical models **M1** and **M2** both lead to tighter MILP relaxation and smaller model sizes. For instance, the MILP relaxation from both M1 and M2 are 18.68 h for Example 2a, which is improved by 73.2% compared to 10.78 from the model of Vooradi and Shaik[15]. The number of binary variables is reduced from 152 to 138 by 9%. As a result, they can successfully solve all examples except Example 2b to global optimality within one hour. On the other hand, the model of Vooradi and Shaik[15] can only solve for Examples 2a, 3a and 3b to optimality, whilst both models **M1** and **M2** require similar or less computational time to solve Examples 2a, 3a and 3b to optimality. The maximum reduction in the computational time can reach 43% for Example 2a (174 vs. 99 and 174 vs. 106). By comparing models **M1** and **M2** in Table 8, it seems that allowing related production and consumption tasks at the same event point can also lead to less computational times. For instance, model **M1** requires 34.8% for Example 1a (412 s vs 639 s) and 49.2% (1004 s vs 1978 s) less computational time for Example 1b compared to model **M2**.

Table 9 Computational results for Examples 1-3 with minimization of makespan (FIS policy)

Example	Model	Number of event points	CPU time (s)	RMILP	MILP (h)	Bin. Var.	Cont. Var.	Constr.
Ex1a	**VS2013**	14	>3600[a]	24.24	27.88	234	372	1068
(D_{S4}=2000 cu)	**M1**	12	>3600[b]	24.24	27.88	214	398	1196
	M2	14	>3600[c]	24.24	27.88	242	456	1371
Ex1b	**VS2013**	23	>3600[d]	48.47	**52.23**	387	615	1779
(D_{S4}=4000 cu)	**M1**	21	>3600[e]	48.47	**52.07**	376	695	2114
	M2	23	>3600[f]	48.47	**52.07**	404	753	2289
Ex2a	**VS2013**	9	241.7	10.78	19.34	332	525	1679
(D_{S8}=200 cu)	**M1**	9	125.3	18.68	19.34	276	601	1889
(D_{S9}=200 cu)	**M2**	9	142.7	18.68	19.34	272	597	1875
Ex2b	**VS2013**	21	>3600[g]	26.40	47.68	788	1257	4091
(D_{S8}=500 cu)	**M1**	21	>3600[h]	45.57	47.68	660	1429	4589
(D_{S9}=400 cu)	**M2**	21	>3600[i]	45.57	47.68	656	1425	4575
Ex3a	**VS2013**	7	0.780	10.00	13.37	417	555	1947
(D_{S12}=100 cu)	**M1**	7	1.841	11.25	13.37	320	625	2209
(D_{S13}=200 cu)	**M2**	7	1.311	11.25	13.37	320	625	2209
Ex3b	**VS2013**	10	1.545	12.50	17.02	603	807	2868
(D_{S12}=250 cu)	**M1**	10	1.092	14.27	17.02	470	907	3256
(D_{S13}=250 cu)	**M2**	10	1.513	14.27	17.02	470	907	3256

Note Δn = 0 in all cases. [a] Relative Gap 1.75%. [b] Relative Gap 1.40%. [c] Relative Gap 1.67%. [d] Relative Gap 0.39%. [e] Relative Gap 0.15%. [f] Relative Gap 0.08%. [g] Relative Gap 19.4%. [h] Relative Gap 0.64%. [i] Relative Gap 1.07%. VS2013: Vooradi and Shaik[15] model.

From Table 9, we can observe that models **M1** and **M2** lead to tighter MILP relaxation and smaller model size. For instance, the MILP relaxation from both **M1** and **M2** are 45.57 h for Example 2b, which is improved by 73% compared to 26.40 from the model of Vooradi and Shaik[15]. The number of binary variables is reduced by 16.2% (788 vs. 660). As a result, the models **M1** and **M2** can solve Examples 2a, 3a and 3b to optimality within 1 hour and solve Examples 1a, 1b, and 2b with smaller optimality gap within 1 hour compared to the model of Vooradi and Shaik[15]. It should also be noted that models **M1** and **M2** find a better solution of 52.07 within 1 hour compared to the model of Vooradi and Shaik[15] (52.07 vs. 52.23), which has not been found in the literature. By comparing models **M1** and **M2** in Table 9**,** it seems that allowing related production and consumption tasks at the same event point can also lead to less computational times. For instance, model **M1** requires 12.5% (125 s vs 143 s) less computational time for Example

2a. In brief, we can conclude that the mathematical model **M1** is the most efficient for makespan minimization.

Large-scale example

We also solve a large-scale industrial batch plant example from Janak et al.[25] to further illustrate the capabilities of models **M1** as model **M1** performs slightly better than **M2** based on the above computational results. Figure 11 depicts the STN representation of this batch plant. The facility produces 87 different products by processing 17 raw materials in 8 different processing paths, and there is a total of 6 different types of processing tasks. Twenty processing units are available to process these tasks, and each processing unit can only process one group of them. The batch plant has to fulfil 402 orders within 19 days. The work Janak et al.[25] contains more information for this example.

We first use the proposed model **M1** to solve this problem directly. It fails to generate a feasible schedule within 12 hours due to intractable problem size. We then employ the rolling horizon decomposition approach of Janak et al.[25] with the proposed model **M1** as the short-term scheduling model to solve this problem, denoted as **RH-M1**. We provide the level-1 model and the modified short-term scheduling model **M1** in the **Supplementary Material**. Each subproblem is solved to zero optimality gap using CPLEX 12/GAMS 24.6.1. on a desktop computer with Intel® Core™ i5-2500 3.3 GHz and 8 GB RAM running Windows 7. The maximum computational time is 3 hours for each level, while the integer solution limit is forty.

Table 10 provides the computational results . From Table 10, **RH-M1** can generate a better solution with the productivity of 6880.2 mu, which is increased by 26.7% in comparison to the 5427.8 mu from the model of Janak et al.[25]. More interestingly, **RH-M1** requires 11.5 h to generate such an improved solution, which is approximately half of the CPU time of the model of Janak et al.[25] (22.4 h). Since both cases use the same rolling horizon decomposition approach, such improvement solely derives from the improved efficiency of the short-term model.

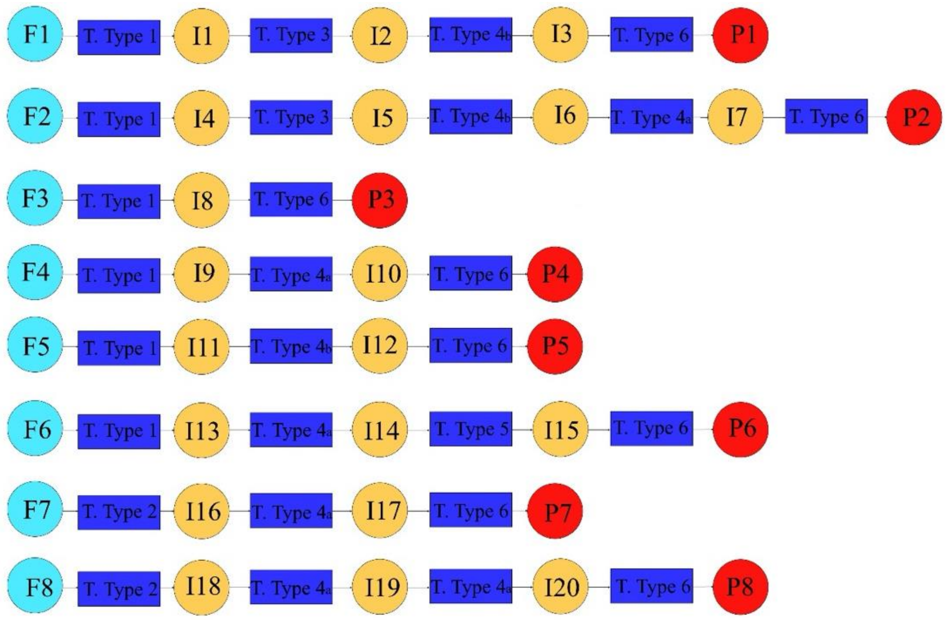

Figure 11 STN representation of large-scale industrial plant example

Table 10 Computational results for the industrial plant example

Model	Total production (mu)	Total CPU time (h)
RH-JF	5427.8	22.4
RH-M1	6880.2	11.5

Table 11 Computational results for each subproblem for industrial plant example

Sub-problem	Model	Days	Production (mu)	CPU time (s)	Bin. Var.	Cont. Var.	Constr.
1	JF	0-2	857.7	3315	4880	35384	187833
	M1	0-2	853.5	1972	15465	60213	127010
2	JF	3-4	758.5	7202	3834	27053	135916
	M1	3-4	790.1	10200	11021	42382	94200
3	JF	5-6	697.0	9878	5406	30545	248663
	M1-J	5-6	777.3	3899	15388	48876	160812
4	JF	7-8	788.8	10329	5526	30729	276612
	M1-J	7-8	994.9	1355	15781	48915	172265
5	JF	9-10	634.7	6945	5406	30465	271764
	M1-J	9-10	853.5	706	15855	49310	172930
6	JF	11-12	517.9	10800	6222	32280	354410
	M1-J	11-12	779.0	10800	17323	53395	198433
7	JF	13-14	532.3	10800	6252	32318	359365
	M1-J	13-14	1114.6	1541	17228	53642	199952
8	JF	15-16	315.7	10800	6156	32085	354649
	M1-J	15-18	717.3	10800	28614	90455	350605
9	JF	17-18	335.3	10800	5976	31664	344065

The computational results for each subproblem from **RH-M1** and **RH-JF** are depicted in Table 11. While **RH-JF** divides the entire scheduling problem into nine subproblems, **RH-M1** divides into eight subproblems. **RH-M1** can solve all subproblems except the subproblems 6 and 8 to optimality within 3 hours. However, **RH-JF** reaches the maximum time of 3 hours for four subproblems out of nine. **RH-M1** leads to higher productivity in comparison to **RH-JF** for all subproblems except the subproblem 1. The difference in productivity for the subproblem 1 between **RH-M1** and **RH-JF** is 0.5% only. Since processing units overproduce some materials in **RH-M1**, which do not fulfil any order at the current scheduling horizon, they can be stored and used for order delivery directly at a later sub-problem without the need of using the facility to produce. As a result, processing units require to process fewer tasks in the successive sub-problems. Therefore, **RH-M1** can successfully generate the schedule of subproblem 8, which contains days 15-18 without the need of further dividing into smaller sub-problems. On the other hand, **RH-JF** needs to produce significantly more materials to fulfil the demand within days 15-18. Therefore, **RH-JF** divides this sub-horizon into sub-problem 8 with days 15-16, and sub-problem 9 with days 17-18 to successfully develop a schedule for this period.

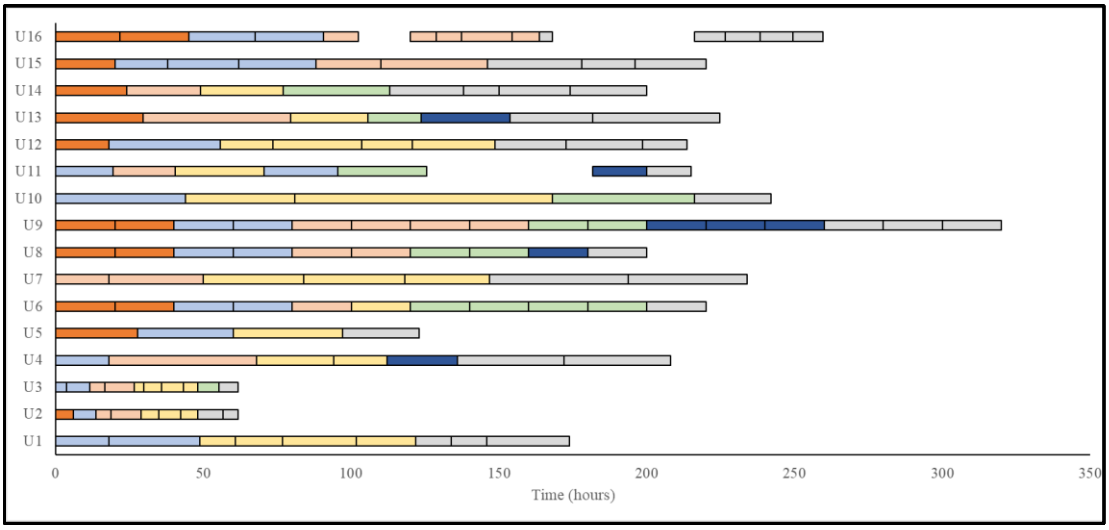

Figure 12 Optimal schedule for the large-scale industrial plant example using **RH-M1**

Table 12 Utilisation efficiency of processing units from RH-M1 and RH-JF

	RH-M1			RH-JF		
	Time used	Time left	% utilised	Time used	Time left	% utilised
U1	173.8	282.2	40.2	187.6	268.4	41.1
U2	61.8	394.2	14.3	92.4	363.6	20.3
U3	61.6	394.4	14.3	118.0	338.0	25.9
U4	208.0	248.0	48.1	200.0	256.0	43.9
U5	123.0	333.0	28.5	49.6	406.4	10.9
U6	123.0	333.0	28.5	296.2	159.8	65.0
U7	233.8	222.2	54.1	174.1	281.9	38.2
U8	200.0	256.0	46.3	233.0	223.0	51.1
U9	340.0	116.0	78.7	311.4	144.6	68.3
U10	242.0	214.0	56.0	170.4	285.6	37.4
U11	158.9	297.1	36.8	90.0	366.0	19.7
U12	213.6	242.4	49.4	129.2	326.8	28.3
U13	224.6	231.4	52.0	185.5	270.5	40.7
U14	200.0	256.0	46.3	162.0	294.0	35.5
U15	220.0	236.0	50.9	129.7	326.3	28.4
U16	194.0	262.0	44.9	451.9	4.1	99.1
U17	-	-	-	12.0	444.0	2.6

The feasible schedule from RH-M1 is illustrated in Figure 12. Table 12 depicts the utilization efficiency for all processing units for both models. **RH-M1** utilizes most of the processing unit for larger periods in order to produce a larger amount of materials and fulfill more orders than **RH-JF**. Additionally, **RH-M1** utilizes one processing unit less during the whole scheduling horizon. In other words, **RH-M1** utilizes the processing units more efficiently.

6 Conclusions

In this work, we presented two generic unit-specific event-based models for scheduling of multipurpose batch processes using the unit-specific event-based modelling approach. While we followed the methodology of Rakovitis et al.[17] to allow all related production and consumption tasks to take place at the same event points but in different real times in the first model, we did not in the second model. We introduced the concept of indirect and direct material transfer, which allows us to conditionally align the operational sequence of related production and consumption tasks. The processing units were allowed to hold materials they previously produced over multiple event points. Both models also consider the nonsimultaneous material transfer[8]. The computational results demonstrated that both models require a smaller number of binary variables in most cases, especially in the cases where a processing unit can process multiple tasks, compared to the existing mathematical formulation[15]. The proposed models did not need to allow a task to span over multiple event points to generate the optimal solution, which resulted in a significant reduction in the computational time by up to one order of magnitude in most cases. More importantly, the proposed models were able to generate better solutions than Vooradi and Shaik[15] and Mostafaei and Harjunkoski[21]. Additionally, the first model allowing related production and consumption tasks to take place at the same event points was slightly more efficient than the second one. Finally, we used the proposed model to solve a large-scale industrial batch plant scheduling problem from Janak *et al.*[25] using the rolling-horizon decomposition algorithm. The results demonstrated that the proposed model can improves productivity by 26.7% in significantly less computational time compared to that from Janak et al.[25]. The future work will extend the proposed models to consider other intermediate storage policy and unit wait policy. A Detailed comparison with all existing models in the literature, especially the model of Mostafaei and Harjunkoski,[21] will also be conducted.

Acknowledgments

Nikolaos Rakovitis would like to acknowledge financial support from the postgraduate award by The University of Manchester.

Literature Cited

1. Kondili E, Pantelides CC, Sargent RWH. A general algorithm for short-term scheduling of batch operations-I MILP formulation. *Comput Chem Eng*. 1993;17(2):211-227. https://doi.org/10.1016/0098-1354(93)80015-F.

2. Pantelides C. Unified frameworks for optimal process planning and scheduling. *Proceedings of the Second Conference on Foundations of Computer Aided Operations*. 1994:253-274.

3. Lee H. Maravelias CT. Discrete-time mixed integer programming models for short-term scheduling in multipurpose environments. *Comput Chem Eng*. 2017;107:171-183. https://doi.org/10.1016/j.compchemeng.2017.06.013.

4. Zhang X. Sargent RWH. 1996. The optimal operation of mixed production facilities-A general formulation and some approaches for the solution. *Comput Chem Eng*. 1996;20(6-7):897-904. https://doi.org/10.1016/0098-1354(95)00186-7.

5. Castro P. Barbosa-Póvoa APFD. Matos H. An improved RTN continuous-time formulation for the short-term scheduling of multipurpose batch plants. *Ind Eng Chem Res*. 2001;40(9):2059-2068. https://doi.org/10.1021/ie000683r.

6. Maravelias CT. Grossmann IE. New General Continuous-Time State-Task Network Formulation for Short-Term Scheduling of Multipurpose Batch Plants. *Ind Eng Chem Res*. 2003;42(13):3056-3074. https://doi.org/10.1021/ie020923y.

7. Sundaramoorthy A, Karimi IA. A simpler better slot-based continuous-time formulation for short-term scheduling in multipurpose batch plants. *Chem Eng Sci*. 2005;60(10):2679-2702. https://doi.org/10.1016/j.ces.2004.12.023.

8. Susarla N, Li J, Karimi I. A. Novel approach to scheduling multipurpose batch plants using unit-slots. *AIChE J*. 2010;56(7):1859-1879. https://doi.org/10.1002/aic.12120.

9. Li J. Karimi IA. Scheduling gasoline blending operations from recipe determination to shipping using unit slots. *Ind Eng Chem Res*. 2011;50(15):9156-9174. https://doi.org/10.1021/acs.iecr.6b01930.

10. Ierapetritou MG. Floudas CA. Effective continuous-time formulation for short-term scheduling. 1. Multipurpose batch processes. *Ind Eng Chem Res*. 1998;37(11):4341-4359. https://doi.org/10.1021/ie970927g.

11. Shaik MA. Floudas CA. Novel Unified Modeling Approach for Short-Term Scheduling. *Ind Eng Chem Res*. 2009;48(6):2947-2964. https://doi.org/10.1021/ie8010726.

12. Li J. Floudas CA. Optimal Event Point Determination for Short-Term Scheduling of

Multipurpose Batch Plants via Unit-Specific Event-Based Continuous-Time Approaches, *Ind Eng Chem Res.* 2010;49(16):7446-7469. https://doi.org/10.1021/ie901842k.

13. Tang QH. Li J. Floudas CA. et al. Optimization framework for process scheduling of operation-dependent automobile assembly lines. *Optim Lett*. 2012;6(4):797-824. https://doi.org/10.1007/s11590-011-0303-5.

14. Seid R. Majozi T. A robust mathematical formulation for multipurpose batch plants. *Chem Eng Sci*. 2012;68(1):36-53. https://doi.org/10.1016/j.ces.2011.08.050.

15. Vooradi R. Shaik MA. Rigorous unit-specific event-based model for short term scheduling of batch plants using conditional sequencing and unit-wait times. *Ind Eng Chem Res*. 2013;52(36):12950-12792. https://doi.org/ 10.1021/ie303294k.

16. Li J. Xiao X. Floudas CA. Integrated gasoline blending and order delivery operations: Part I. short-term scheduling and global optimization for single and multi-period operations, *AIChE J*. 2016;62(6):2043-2070. https://doi.org/10.1002/aic.15168.

17. Rakovitis N. Zhang N. Li J. Zhang L. A new approach for scheduling of multipurpose batch processes with unlimited intermediate storage policy. *Front Chem Sci Eng*. 2019;13:784-802. https://doi.org/10.1007/s11705-019-1858-4.

18. Méndez CA. Cerdá J. Optimal scheduling of a resource-constrained multiproduct batch plant supplying intermediates to nearby end-product facilities. *Comput Chem Eng*. 2000;24(2-7):369-376. https://doi.org/10.1016/S0098-1354(00)00482-8.

19. Hui C. Gupta A. van der Meulen HAJ. A novel MILP formulation for short-term scheduling of multi-stage multi-product batch plants with sequence-dependent constraints. *Comput Chem Eng.* 2000;24(12):2705 – 2717. https://doi.org/10.1016/S0098-1354(00)00623-2.

20. Méndez CA. Cerdá J. An MILP continuous-time framework for short-term scheduling of multipurpose batch processes under different operation strategies. *Optim Eng*. 2003;4(1-2):7-22. https://doi.org/10.1023/A:1021856229236.

21. Mostafaei H, Harjunkoski I. Continuous-time scheduling formulation for multipurpose batch plants. AIChE J. 2020;66:e16804. https://doi.org/10.1002/aic.16804.

22. Floudas CA. Lin X. Continuous-time versus discrete-time approaches for scheduling of chemical processes: a review. *Comput Chem Eng.* 2004;28(11):2109-2129. https://doi.org/10.1016/j.compchemeng.2004.05.002.

23. Méndez CA. Cerdá, J. Grossmann IE. Harjukoski I. Fahl M. State-of-the-art review of optimization methods for short-term scheduling of batch processes. *Comput Chem Eng*. 2006;30(6-7):913-946. https://doi.org/10.1016/j.compchemeng.2006.02.008.

24. Harjunkoski I. Maravelias CT. Bongers P. et al. Scope for industrial application of

production scheduling models and solution methods. *Comput Chem Eng*. 2014;62(5):161-193. https://doi.org/10.1016/j.compchemeng.2013.12.001.
25. Janak SL. Floudas CA. Kallrath J. Vormbrock N. Production scheduling of a large-scale industrial batch plant. I. Short-term and Medium-term scheduling. *Ind Eng Chem Res*. 2006;45(25):8234-8252. https://doi.org/10.1021/ie0600588.

Nomenclature

Indices

i, i': tasks

j, j': units

n, n', n'': event points

s: states

Sets

I: tasks

$\mathbf{I}_j$: tasks that can be performed in unit j

$\mathbf{I}_s$: tasks that produce/consume state s

$\mathbf{I}_s^c$: tasks that consume state s

$\mathbf{I}_s^P$: tasks that produce state s

$\mathbf{I}^R$: tasks considered as recycling tasks

J: units

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

$\mathbf{J}_s$: units that produce/consume state s

N: event points

S: states

$\mathbf{S}^{FIS}$: states with unlimited intermediate storage policy

$\mathbf{S}^P$: states that are final products

$\mathbf{S}^{IN}$: states that are intermediate products

$\mathbf{S}^{\mathrm{R}}$: states that are raw materials

$\mathbf{S}^{UIS}$: states with unlimited intermediate storage policy

Parameters

B_{ij}^{max}: maximum batch size of task i processed in unit j

B_{ij}^{min} : minimum batch size of task i processed in unit j

D_s: demand of state s

H: scheduling horizon

M: big-M value

P_s: price of state s

$ST0_s$: initial amount of state s

$ST_s^{\max}$: maximum capacity of state s (for states with FIS policy)

α_{ij}: coefficient of constant term of processing time of task i in unit j

β_{ij}: coefficient of variable term of processing time of task i in unit j

Δn: maximum number of event points that task i is allowed to be active

ρ_{sij}: portion of state s consumed/produced by task i processed in unit j

Binary variables

$w_{ijnn'}$: binary variable which takes the value 1 if task i is processed in unit j from event point n to $n' \geq n$

ys_{ijn}: binary variable which takes the value 1 if there is any amount of materials stored in unit j at event point n, which were previously produced by task i processed in unit j at event point $n' < n$

$zI_{jj'\mathrm{n}}$: binary variable which takes the value 1 if there is indirect material transfer between unit j and j'

$zD_{jj'\mathrm{n}}$: binary variable which takes the value 1 if there is indirect material transfer between unit j and j'

Continuous variables

$b_{ijnn'}$: amount of materials that are processed in unit j processing task i from time event point n to time event point $n' \geq n$

bs_{ijn}: amount of materials stored in unit j at event point n, which were previously produced by task i processed in unit j at event point $n' < n$

$bTi_{iji'j'n}$: amount of materials, which produced by task i processed in unit j, were indirectly transferred to unit j' which consumes task i' at event point n

$bTd_{iji'j'n}$: amount of materials, which produced by task i processed in unit j, were indirectly transferred to unit j' which consumes task i' at event point n

ST_{sn}: amount of state s that has to be stored at time event point n

T_{sjn}: time that state s produced in unit j is available to be consumed at event point n

T_{jn}^{s}: start time of unit j at time event point n

T_{jn}^{f}: end time of unit j at time event point n

Blank page

Chapter 5: Scheduling of continuous processes

5.1 Introduction

The framework developed and implemented in Chapter 4 is only implemented for scheduling of batch processes. The continuous processes have significant differences from batch processes, and as a result, it is not possible to directly implement mathematical models for scheduling of batch processes for this problem. More specifically, in continuous processes, the processing time is not predefined as in batch processes. Instead, the processing time of each task processed in a unit is a variable that needs to be optimized. The only limitation, in this case, is that if a unit starts processing a task, then it should process it for a minimum or both minimum and maximum time. Furthermore, a processing unit must continuously receive raw materials, and it extracts final products without interruption in contrast to the batch process where this occurs only at the beginning and the end of the processing, respectively.

In this chapter, the proposed framework is implemented for scheduling of continuous processes. Since continuous processes are different from batch processes, several constraints are slightly modified to handle such type of industry. The mathematical model is extended to handle several cases, including no intermediate storage policy, flexible or swing storage, storage bypass and planned maintenance. Multiple well-established examples are used to investigate the performance of the proposed model.

Blank Page

5.2 Research contribution 3

Rakovitis, N., Hasnuddin, W. M. A. W., Zhang, N., Li, J. A Generic Approach for Scheduling of Semi-continuous and Continuous Processes, to be submitted to Chemical Engineering Science

Blank Page

A Generic Approach for Scheduling of Semi-continuous and Continuous Processes

Nikolaos Rakovitis, Wan Mohd Azril bin Wan Hasnuddin, Nan Zhang and Jie Li†

Centre for Process Integration, School of Chemical Engineering and Analytical Science, The University of Manchester, Manchester, M13 9PL, United Kingdom

Abstract

In this work, we extend our proposed modelling approach for batch processes (Rakovitis et al., 2020) to develop a generic and efficient mathematical formulation for the scheduling of semi-continuous and continuous processes. In this approach, we conditionally sequence or synchronize related production and consumption tasks by using the concept of indirect and direct material transfer. The model also considers different intermediate storage policies, flexible intermediate storage and planned maintenance. We also extend the model to consider the case of not allowing storage bypass where we consider two different scenarios; in the first scenario, a storage tank can receive and deliver materials at the same time, while in the second scenario it cannot. The results demonstrate that the proposed mathematical model requires a smaller number of event points than the model of Omar and Shaik (2019). Additionally, it requires significantly less computational time which can reach up to two magnitudes less computational time.

† To whom correspondence should be addressed. jie.li-2@manchester.ac.uk. Tel: +44 (0) 161 306 8622

1 Introduction

The increasing demand for specific products with the same specifications lead process industry to use semi-continuous or continuous processes. In those processes, one or more raw materials are processed uninterrupted within a period to produce large quantities of a specific product with the same quality. Oil refinery, chemical and steel industry are some examples of continuous process industry. Due to the highly competitive market, such process industries must reduce their operational costs and increase their profit. The more and more strict environmental regulations also lead facilities to examine different alternatives to eliminate their footprint by reducing their raw material and fuel consumption. Scheduling is one of the main managerial tools that can help industries to reduce their costs and fuel consumption.

Process industries developed and implemented several approaches to improve their scheduling decisions, including heuristic rules, spreadsheet-based methods and mathematical programming approach. Even though the first two approaches can generate schedules fast, the quality of the solution depends on the operator's experience. Therefore, they are only capable of generating a feasible solution, which can be significantly far from optimum. On the other hand, mathematical modelling, especially mixed-integer linear programming (MILP), can generate optimal solutions for a given set of operations. As a result, several mathematical models have been developed for this scheduling problem using different time representations including discrete-time (Zhang *et al.* 2016) and continuous-time approaches, such as sequence-based (Kopanos *et al.* 2011), slot-based (Schiling and Pantelides 1996; Karimi and McDonald 1997; Lee et al. 2001), global-event based (Mockus and Reklaitis 1999; Castro et al. 2004) and unit-specific event-based (Ierapetritou and Floudas 1998; Gianelos and Georgiadis 2002; Shaik and Floudas 2007; Shaik *et al.* 2009; Tang *et al.* 2012; Li *et al.* 2012; Omar and Shaik 2018). Floudas and Lin (2004), and Harjunkoski *et al.* (2014) provide an excellent review of different timing approaches used in the process industry.

The capabilities of the unit-specific event-based time representations are well established in the literature (Li *et* al., 2010; Rakovitis *et al.*, 2019; Rakovitis *et al.* 2020). However, some unit-specific event-based mathematical models for scheduling of continuous processes (Ierapetritou and Floudas, 1998; Shaik and Floudas, 2007) can generate schedules with real-time violations (Li et al., 2010). These models can also fail to generate a feasible solution in some cases, even if there is one (Li *et al.* 2010). Shaik and Floudas (2007) developed a mathematical model using unit-specific event-based time representation. In contrast to Ierapetritou and Floudas (1998) model, they considered different intermediate storage policies. However, they unconditionally sequenced and synchronized related production and consumption tasks, which leads to an increase in the number of event points required (Omar and Shaik 2018; Rakovitis *et al.* 2019). Recently, Omar and Shaik (2018) developed a unit-specific event-based mathematical model for scheduling of continuous processes with simultaneously considering planned maintenance under unlimited intermediate storage (UIS) policy. They conditionally aligned related production and consumption tasks in different units only when a consumption task received materials from the related production task. Omar and Shaik (2019) extended the mathematical model of Omar and Shaik (2018) for different intermediate storage policies, such as finite intermediate storage (FIS) and no intermediate storage (NIS). They also conditionally aligned related production and consumption tasks only if the total amount to be stored exceeds the maximum storage capacity. Even though the model of Omar and Shaik (2019) handles the real-time violation issue of the previous model of Shaik and Floudas (2007), it leads to significantly large model sizes and hence it requires excessive computational time even for small examples. Although the proposed model requires a smaller number of event point than the model of Shaik and Floudas (2007) in problems with planned maintenance, they introduce a large number of binary variables which leads to large model-sizes.

In this work, we extend our proposed modelling framework (Rakovitis et al., 2020) to develop a generic and efficient mathematical formulation for the scheduling of semi-continuous and continuous processes. In the formulation, non-recycling tasks are allowed to take place at the same event points. The concept of indirect and direct material transfer (Rakovitis *et al.*, 2020) is employed to conditionally sequence or synchronize related production and consumption tasks in different units. We consider several storage policies, including unlimited, finite, and no intermediate storage policies. Some storage tanks are allowed to hold multiple materials during the scheduling horizon. Materials produced are allowed to directly enter the downstream consumption units, which is called storage bypass policy. The case of not allowing storage bypass is also considered, with two distinct scenarios; while in the first scenario a storage tank can receive and deliver materials at the same time, while in the second scenario it cannot. The model can also handle cases with planned maintenance. The capability of the proposed model is illustrated by solving several well-established examples in the literature (Shaik and Floudas 2007; Li *et al.* 2010; Omar and Shaik 2018; Omar and Shaik 2019). The computational results demonstrate that the proposed model can generate the optimal solution for all examples using a smaller number of event points than the model of Omar and Shaik (2019). It is also more general and efficient than the model of Omar and Shaik (2019) since it can handle cases of flexible intermediate storage and it requires significantly less computational time which can reach in up to two orders of magnitude less computational time.

2 Problem description

Figure 1 illustrates a typical semi-continuous or continuous process facility. This facility contains J ($j = 1, 2, 3, \ldots., J$) processing units which convert several feeds into multiple valuable products. Besides feeds and final products, the processing units also produce some intermediates. We use a set S ($s = 1, 2, 3, \ldots, S$) to denote all material states in the facility including feeds (denoted in set $\mathbf{S}^R$), intermediates (included in set $\mathbf{S}^{IN}$), and products (included in set $\mathbf{S}^P$). There are I ($i = 1, 2, 3, \ldots, I$) tasks in total, which contains processing tasks and storage tasks. We use $\mathbf{I}^p$ to denote processing tasks and $\mathbf{I}^{st}$ to denote storage tasks. Each processing unit j can process $\mathbf{I}_j$ tasks. A task consumes raw materials with different proportions to produce multiple states with different yields. A parameter $\rho_{i,s}$ is used to denote the proportion of state s produced or consumed by a task i. While positive values of $\rho_{i,s}$ denote production of state s during the processing of task i, negative values of $\rho_{i,s}$ indicate consumption of state s by task i. A processing unit can process multiple tasks. When transforming a task to another in a processing unit consecutively, some changeover time is required. The changeover time can be either sequence-dependent or unit-dependent only. We use τ_j to denote unit-dependent changeover time and $\tau_{i,i',j}$ to denote sequence-dependent changeover time.

After production, an intermediate state may be transferred directly to the downstream processing units, which is called storage bypass. It may be also transferred to storage. There are several storage policies, including unlimited intermediate storage (UIS), finite intermediate storage (FIS) and no intermediate storage (NIS) policies. Two types of storage tanks are often used in practice, which includes dedicate and flexible (swing) storage tanks. While the dedicate storage tank can only hold one dedicated intermediate state at any time, the flexible (or swing) storage tank can hold multiple intermediate states during the scheduling horizon. However, at most one intermediate state can be held in such a storage tank at a time. The products may be used to satisfy orders. There are totally O $(o = 1, 2, 3, \dots, O)$ orders. We assume each order involves only one product as one order with multiple products can be divided into multiple orders without loss of generality. Each order has release time (r_o) and due date (d_o). The amount of an order is denoted as T_o. With these, the entire scheduling problem can be stated as follows,

Given:

1) *O* orders, their products, release times and due dates;
2) *J* units, suitable tasks, minimum and maximum capacities, processing rates and planned maintenance period;
3) *S* states, the portion of states produced or consumed from a task;
4) Storage policy and capacity for each state;
5) Product prices;
6) Scheduling horizon.

Determine:

1) Optimal production schedule involving task allocations, start and end timings, and sequences;
2) Inventory profiles.

Operating rules:

1) At most one task can be processed in a processing unit at any time;
2) At most one intermediate state can be stored in a flexible (swing) storage tank at a time

Assumptions:

1) All parameters are deterministic;
2) Unlimited feed materials are available;
3) Unlimited storage policy for all raw materials and products;
4) Unlimited resources where required are available;

The objective is to maximize productivity or minimize total operating cost.

3 Mathematical formulation

We extend the proposed unit-specific event-based modelling approach (Rakovitis et al., 2019; Rakovitis et al., 2020) for this scheduling problem of semi-continuous and continuous processes where timing variables are defined based on units (i.e., $T_{j,n}^{s}$ and $T_{j,n}^{f}$) and a production task is sequenced and/or synchronised with its related consumption tasks if materials are transferred between these tasks.

3.1 Allocation constraints

We define a binary variable $w_{i,j,n,n'}$ to denote if a task i is processed in a unit j from event point n to n' as follows,

$$w_{i,j,n,n'} = \begin{cases} 1 & \text{if a task } i \text{ is processed in a unit } j \text{ from an event point } n \text{ to } n' \\ 0 & \text{otherwise} \end{cases}$$

where $n \leq n' \leq n + \Delta n$. The parameter Δn is used to denote the maximum number of event points that a task is allowed to cross over.

Based on the operating policy, at most one task can be processed in a processing unit j at a time.

$$\sum_{i \in \mathbf{I}_j} \sum_{n-\Delta n \leq n' \leq n} \sum_{n \leq n'' \leq n'+\Delta n} w_{i,j,n',n''} \leq 1 \qquad \forall j \in \mathbf{J}^p, n \tag{1}$$

3.2 Capacity constraints

The amount of materials processed in a processing unit j at an event point n (denoted as $b_{i,j,n}$) is limited by the minimum (R_{ij}^{min}) and the maximum (R_{ij}^{max}) processing rates multiplying processing duration ($L_{i,j,n}$).

$$R_{i,j}^{\min} \cdot L_{i,j,n} \leq b_{i,j,n} \leq R_{i,j}^{\max} \cdot L_{i,j,n} \qquad \forall j \in \mathbf{J}^p, i \in \mathbf{I}_j, n \tag{2a, b}$$

For processes with fixed processing rate (denoted as $R_{i,j}$), the amount of materials produced is proportional to the task duration

$$b_{i,j,n} = R_{i,j} \cdot L_{i,j,n} \qquad \forall j \in \mathbf{J}^p, i \in \mathbf{I}_j, n \tag{3}$$

3.3 Duration constraints

The finish time of a processing unit j at event point n must be after its start time plus the task duration of task i that the unit starts processing at event point n.

$$T_{j,n}^{\mathrm{f}} \geq T_{j,n}^{\mathrm{s}} + \sum_{i \in \mathbf{I}_j} L_{i,j,n} \qquad \forall j \in \mathbf{J}^p, n \tag{4}$$

If a task i is not processed in a unit j during an event point n, then the task duration in the unit during this event point n should be equal to zero.

$$L_{i,j,n} \le H \cdot \left(\sum_{n' \le n} \sum_{n' \le n'' \le n' + \Delta n} w_{i,j,n',n''} - \sum_{n' < n} \sum_{n' - \Delta n \le n'' \le n'} w_{i,j,n',n''} \right)$$

$$\forall j \in \mathbf{J}^p,\ i \in \mathbf{I}_j,\ n \quad (5)$$

3.4 Material balance constraints

The amount of a state s stored at event point n (denoted as ST_{sn}) should be equal to the amount of the state stored at event point $(n-1)$, plus the amount of the state produced at event point n, minus the amount of the state consumed at event point n. At the first event point, the amount of a state s stored should be equal to the initial amount of the state ($ST0_s$) plus the amount of the state produced, minus the amount of state s consumed at event point $n = 1$.

$$ST_{s,n} = ST0_s + \sum_{j \in \mathbf{J}^P} \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P)} \rho_{i,s} \cdot b_{i,j,n} + \sum_{j \in \mathbf{J}^P} \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^C)} \rho_{i,s} \cdot b_{i,j,n}$$

$$\forall s,\ n = 1 \quad (6a)$$

$$ST_{s,n} = ST_{s,n-1} + \sum_{j \in \mathbf{J}^P} \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P)} \rho_{i,s} \cdot b_{i,j,n} + \sum_{j \in \mathbf{J}^P} \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^C)} \rho_{i,s} \cdot b_{i,j,n}$$

$$\forall s,\ n > 1 \quad (6b)$$

where set $\mathbf{I}_s^C$ denotes tasks that consume state s, while $\mathbf{I}_s^P$ denotes tasks that produce state s.

3.5 Material transfer

Material transfer in the semi-continuous or continuous process is simpler compared to that in the batch processes of Rakovitis et al. (2020). A production unit starts to transfer materials to storage or downstream processing units immediately when it starts producing the related production task. Materials are continuously transferred from the production unit until it finishes processing the related production task. We generally classify material transfer as indirect and direct material transfer. If materials are allowed to be transferred to downstream processing units directly, it is a direct material transfer, as illustrated in Figure 1 (denoted as MT1). If materials are transferred to storage tank first and then to downstream processing units, then it is an indirect material transfer (denoted as MT2 in figure 1).

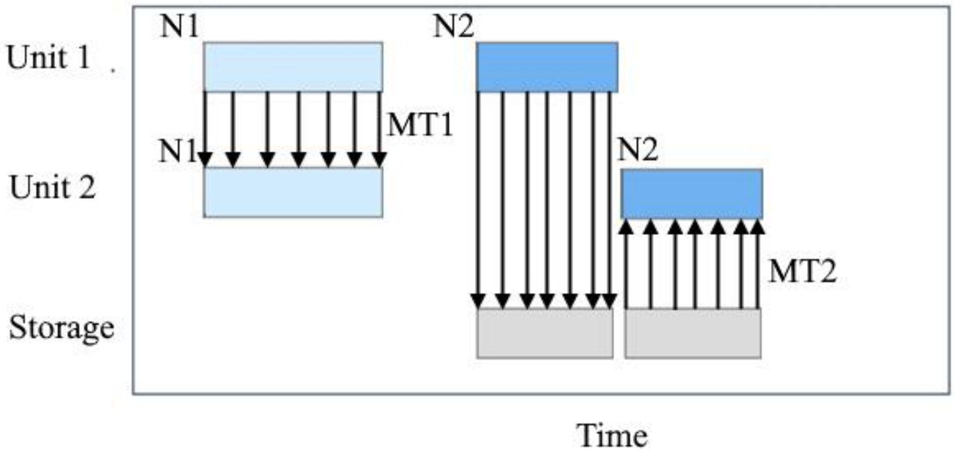

Figure 1 Different scenarios of material transfer

Indirect material transfer

In the indirect material transfer, the storage capacity is large enough to hold all producing materials. As a result, materials produced can always transferred to storage first and then transferred to downstream processing units from storage. To model this indirect material transfer, we define an additional binary variable $zI_{j,j',n}$ as follows,

$$zI_{j,j',n} = \begin{cases} 1 & \text{if material transfer happens between units } j \text{ and } j' \text{ at event point } n \\ 0 & \text{otherwise} \end{cases}$$

$$\forall\, j \neq j', n$$

We also define a continuous variable $bTi_{i,j,i',j',n}$ to denote the amount of materials indirectly transferred from a production task i in unit j to a consumption task i' in unit j' at event point n. The total amount of materials through indirect transfer from a production task i should not exceed that produced from this task i.

$$\rho_{i,s} \cdot b_{i,j,n} \geq \sum_{j' \in (\mathbf{J}_s \cap \mathbf{J}^P)} \sum_{i' \in \left(\mathbf{I}_{j'} \cap \mathbf{I}_s^C\right)} bTi_{i,j,i',j',n}$$

$$\forall s \in \mathbf{S}^{IN}, j \in (\mathbf{J}^p \cap \mathbf{J}_s), i \in (\mathbf{I}_j \cap \mathbf{I}_s^P), n \qquad (7)$$

Similarly, the amount of materials through indirect transfer to a consumption task i' at a time should not exceed the amount of materials consumed by this consumption task at event point n.

$$-\rho_{i',s} \cdot b_{i',j',n} \geq \sum_{j \in (\mathbf{J}_s \cap \mathbf{J}^P)} \sum_{i \in \left(\mathbf{I}_j \cap \mathbf{I}_s^P\right)} bTi_{i,j,i',j',n}$$

$$\forall s \in \mathbf{S}^{IN}, j' \in (\mathbf{J}^p \cap \mathbf{J}_s), i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C), n \qquad (8)$$

The total amount of materials consumed at event point n should not exceed the material stored at previous event point $(n-1)$ plus the amount of materials through indirect transfer.

$$\sum_{j'\in(\mathbf{J}_s\cap\mathbf{J}^P)}\sum_{i'\in\left(\mathbf{I}_{j'}\cap\mathbf{I}_s^C\right)}\left(-\rho_{i',s}\cdot b_{i',j',n}\right)\le ST0_s+$$

$$+\sum_{j\in(\mathbf{J}_s\cap\mathbf{J}^P)}\sum_{j'\in(\mathbf{J}_s\cap\mathbf{J}^P)}\sum_{i\in\left(\mathbf{I}_j\cap\mathbf{I}_s^P\right)}\sum_{i'\in\left(\mathbf{I}_{j'}\cap\mathbf{I}_s^C\right)} bTi_{i,j,i',j',n}$$

$$\forall s\in\mathbf{S}^{IN},\ n=1 \quad (9a)$$

$$\sum_{j'\in(\mathbf{J}_s\cap\mathbf{J}^P)}\sum_{i'\in\left(\mathbf{I}_{j'}\cap\mathbf{I}_s^C\right)}\left(-\rho_{i',s}\cdot b_{i',j',n}\right)\le ST_{s,n-1}+$$

$$+\sum_{j\in(\mathbf{J}_s\cap\mathbf{J}^P)}\sum_{j'\in(\mathbf{J}_s\cap\mathbf{J}^P)}\sum_{i\in\left(\mathbf{I}_j\cap\mathbf{I}_s^P\right)}\sum_{i'\in\left(\mathbf{I}_{j'}\cap\mathbf{I}_s^C\right)} bTi_{i,j,i',j',n}$$

$$\forall s\in\mathbf{S}^{IN},\ n>1 \quad (9b)$$

When there is no indirect material transfer between two processing units, the amount through this indirect material transfer should be zero.

$$\sum_{i\in\left(\mathbf{I}_j\cap\mathbf{I}_s^P\right)}\sum_{i'\in\left(\mathbf{I}_{j'}\cap\mathbf{I}_s^C\right)} bTi_{i,j,i',j',n}\le\min\left\{\max_{i\in\left(\mathbf{I}_j\cap\mathbf{I}_s^P\right)}\left(R_{i,j}^{\max}\cdot H\right),\ \max_{i'\in\left(\mathbf{I}_{j'}\cap\mathbf{I}_s^C\right)}\left(R_{i',j'}^{\max}\cdot H\right),ST_s^{\max}\right\}\cdot zI_{j,j',n}$$

$$\forall s\in\mathbf{S}^{IN},\ j\ne j',\ j\in(\mathbf{J}^p\cap\mathbf{J}_s),\ j'\in(\mathbf{J}^p\cap\mathbf{J}_s),\ n \quad (10)$$

Direct material transfer

For states with FIS policy, if there is no storage available, then these states cannot be transferred to a storage tank. Instead, they must be transferred directly from the production task *i* to a consumption task *i′*. To model such direct material transfer, we introduce an additional binary variable $zD_{j,j',n}$ as follows,

$$zD_{jj'n}=\begin{cases}1 & \text{if there is a direct material transfer between units } j \text{ and } j' \text{ at event point } n\\ 0 & \text{otherwise}\end{cases}$$

$$\forall\ j\ne j',\ n$$

Similar to indirect material transfer, we also define a continuous variable $bTd_{i,j,i',j',n}$ to denote the amount of materials directly transferred from a production task *i* in unit *j* to a consumption task *i′* in unit *j′* at event point *n*. The amount of materials directly transferred between processing a production task *i* in unit *j* and a consumption task *i′* in unit *j′* must not exceed the amount of state produced from production task *i*.

$$\rho_{i,s} \cdot b_{i,j,n} \geq \sum_{j' \in (\mathbf{J}_s \cap \mathbf{J}^p)} \sum_{i' \in \left(\mathbf{I}_{j'} \cap \mathbf{I}_s^C\right)} bTd_{i,j,i',j',n}$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}), j \in (\mathbf{J}^p \cap \mathbf{J}_s), i \in (\mathbf{I}_j \cap \mathbf{I}_s^P), n \quad (11)$$

The amount of materials through direct transfer to a consumption task *i′* at a time should not exceed the amount of materials consumed by this consumption task at event point *n*.

$$-\rho_{i,s} \cdot b_{i',j',n} \geq \sum_{j \in (\mathbf{J}_s \cap \mathbf{J}^p)} \sum_{i \in \left(\mathbf{I}_j \cap \mathbf{I}_s^P\right)} bTd_{i,j,i',j',n}$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}), j' \in (\mathbf{J}^p \cap \mathbf{J}_s), i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C), n \quad (12)$$

A direct material transfer between a production task *i* in unit *j* and a consumption task *i′* in unit *j′* takes place only if the amount of state s produced at event point *n*, plus the amount of materials stored in storage tanks at event point $(n-1)$ exceeds the maximum storage capacity. In this case, there are no storage tanks to temporary store the materials produced. For the first event point, it should be examined whether the amount of state s produced at the first event point, plus the initial amount of materials stored in storage tanks.

$$\sum_{j \in (\mathbf{J}_s \cap \mathbf{J}^P)} \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P)} \left(\rho_{i,s} \cdot b_{i,j,n}\right) + ST0_s \geq ST_s^{\max} + \sum_{j \in (\mathbf{J}_s \cap \mathbf{J}^P)} \sum_{j' \in (\mathbf{J}_s \cap \mathbf{J}^P)} \sum_{i \in \left(\mathbf{I}_j \cap \mathbf{I}_s^P\right)} \sum_{i' \in \left(\mathbf{I}_{j'} \cap \mathbf{I}_s^C\right)} bTd_{i,j,i',j',n}$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}), n = 1 \quad (13a)$$

$$\sum_{j \in (\mathbf{J}_s \cap \mathbf{J}^P)} \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P)} \left(\rho_{i,s} \cdot b_{i,j,n}\right) + ST_{s,n-1} \geq ST_s^{\max} + \sum_{j \in (\mathbf{J}_s \cap \mathbf{J}^P)} \sum_{j' \in (\mathbf{J}_s \cap \mathbf{J}^P)} \sum_{i \in \left(\mathbf{I}_j \cap \mathbf{I}_s^P\right)} \sum_{i' \in \left(\mathbf{I}_{j'} \cap \mathbf{I}_s^C\right)} bTd_{i,j,i',j',n}$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}), n > 1 \quad (13b)$$

When there is no direct material transfer between two related processing units, the amount through this direct transfer should be zero, similar to the indirect material transfer.

$$\sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P)} \sum_{i' \in \left(\mathbf{I}_{j'} \cap \mathbf{I}_s^C\right)} bTd_{i,j,i',j',n} \leq \min\left\{\max_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P)} \left(R_{i,j}^{\max} \cdot H\right), \max_{i' \in \left(\mathbf{I}_{j'} \cap \mathbf{I}_s^C\right)} \left(R_{i',j'}^{\max} \cdot H\right), ST_s^{\max}\right\} \cdot zD_{j,j',n}$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}), j \neq j', j \in (\mathbf{J}^p \cap \mathbf{J}_s), j' \in (\mathbf{J}^p \cap \mathbf{J}_s), n \quad (14)$$

3.6 Sequencing constraints

Different tasks in the same unit

The start time of a unit *j* at event point (*n* + 1) must always be after its finish time at the previous event point *n*.

$$T_{j,n+1}^{\mathrm{s}} \geq T_{j,n}^{\mathrm{f}}$$

$$\forall j \in \mathbf{J}^p, n < N \quad (15)$$

If a task i requires to span over multiple event points (i.e., from event point n' to n''), then the start time of a unit j at event point $(n+1)$ must be equal to its end time at the previous event point n if event point $(n+1)$ is between event points n' and n''.

$$T^{\mathrm{s}}_{j,n+1} \le T^{\mathrm{f}}_{j,n} + H\left(1 - \sum_{i\in\mathbf{I}_j}\sum_{n-\Delta n\le n'\le n}\sum_{n+1\le n''\le n'+\Delta n} w_{i,j,n',n''}\right)$$

$$\forall j\in\mathbf{J}^p,\ n < N,\ \Delta n > 0 \qquad (16)$$

Different task in different unit

In order to make sure correct operational sequences between production and consumption tasks in different processing units, we define two continuous variables $T^{\mathrm{s}}_{s,j,n}$ and $T^{\mathrm{f}}_{s,j,n}$ to denote the start and finish time that a state s produced by a unit j is available to be transferred (i.e., consumed or stored) at event point n. The start time that a state s produced by a unit j is available to be consumed at event point $(n+1)$ should always be after the finish time that state is available at the previous event point n.

$$T^{\mathrm{s}}_{s,j,n+1} \ge T^{\mathrm{f}}_{s,j,n}$$

$$\forall s \in \mathbf{S}^{IN},\ j\in(\mathbf{J}^p \cap \mathbf{J}_s),\ n < N \qquad (17a)$$

The finish time that a state s produced by a unit j is available to be consumed at event point n should always be after the start time a state s produced by a unit j is available to be consumed at the same event point.

$$T^{\mathrm{f}}_{s,j,n} \ge T^{\mathrm{s}}_{s,j,n}$$

$$\forall s \in \mathbf{S}^{IN},\ j\in(\mathbf{J}^p \cap \mathbf{J}_s),\ n < N \qquad (17b)$$

When a state s produced by a unit j is available at event point n, the start and finish of production of this state in the same unit j must be before the start and finish time that the state is available at this event point n respectively. In other words,

$$T^{\mathrm{s}}_{s,j,n} \ge T^{\mathrm{s}}_{j,n} - M\left(1 - \sum_{i\in\left(\mathbf{I}_j\cap\mathbf{I}^P_s\right)}\sum_{n\le n'\le n+\Delta n} w_{i,j,n,n'}\right)$$

$$\forall s \in \mathbf{S}^{IN},\ j\in(\mathbf{J}^p \cap \mathbf{J}_s),\ \sum_{i\in(\mathbf{I}_j\cap\mathbf{I}^P_s)} \rho_{i,s} > 0,\ n \qquad (18a)$$

$$T^{\mathrm{f}}_{s,j,n} \ge T^{\mathrm{f}}_{j,n} - M\left(1 - \sum_{i\in\left(\mathbf{I}_j\cap\mathbf{I}^P_s\right)}\sum_{n-\Delta n\le n'\le n} w_{i,j,n',n}\right)$$

$$\forall s \in \mathbf{S}^{IN},\ j\in(\mathbf{J}^p \cap \mathbf{J}_s),\ \sum_{i\in(\mathbf{I}_j\cap\mathbf{I}^P_s)} \rho_{i,s} > 0,\ n \qquad (18b)$$

The start and finish time of a unit j' should be after the start and finish time of unit j at event point n, if there is an indirect material transfer between units j and j'.

$$T^{s}_{j',n} \geq T^{s}_{j,n} - M\left(1 - zI_{j,j',n}\right)$$
$$\forall s \in \mathbf{S}^{IN}, (j, j') \in (\mathbf{J}^{p} \cap \mathbf{J}_{s}), j \neq j', \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P)} \rho_{i,s} > 0, \sum_{i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C)} \rho_{i',s} < 0, n \qquad (19)$$

$$T^{f}_{j',n} \geq T^{f}_{j,n} - M\left(1 - zI_{j,j',n}\right)$$
$$\forall s \in \mathbf{S}^{IN}, (j, j') \in (\mathbf{J}^{p} \cap \mathbf{J}_{s}), j \neq j', \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P)} \rho_{i,s} > 0, \sum_{i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C)} \rho_{i',s} < 0, n \qquad (20)$$

If the materials produced in a processing unit at event point n is not transferred to a consumption task in a processing unit at the same event point n, then all materials should be stored in its dedicated storage tank, before another production task is processed in this unit. In this case, the start and finish time of this consumption task at an event point $(n + 1)$ should always exceed the time that the state starts and finishes being available at event point n.

$$T^{s}_{s,j,n} \leq T^{s}_{j',n+1} + M\left(1 - \sum_{i' \in \left(\mathbf{I}_{j'} \cap \mathbf{I}_s^C\right)} \sum_{n+1 \leq n' \leq n+1+\Delta n} w_{i',j',n+1,n'}\right)$$
$$\forall s \in \mathbf{S}^{IN}, (j, j') \in (\mathbf{J}^{p} \cap \mathbf{J}_{s}),\ j \neq j', \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P)} \rho_{i,s} > 0, \sum_{i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C)} \rho_{i',s} < 0, n < N \qquad (21a)$$

$$T^{f}_{s,j,n} \leq T^{f}_{j',n+1} + M\left(1 - \sum_{i' \in \left(\mathbf{I}_{j'} \cap \mathbf{I}_s^C\right)} \sum_{n+1-\Delta n \leq n' \leq n+1} w_{i',j',n',n+1}\right)$$
$$\forall s \in \mathbf{S}^{IN}, (j, j') \in (\mathbf{J}^{p} \cap \mathbf{J}_{s}),\ j \neq j', \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P)} \rho_{i,s} > 0, \sum_{i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C)} \rho_{i',s} < 0, n < N \qquad (21b)$$

If there is a direct material transfer at event point n from unit j to unit j' then the start and finish time of unit j' should be after the start and finish time of unit j at this event point similar to other scenario of indirect material transfer.

$$T^{s}_{j',n} \geq T^{s}_{j,n} - M\left(1 - zD_{j,j',n} + zI_{j,j',n}\right)$$
$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}), (j, j') \in (\mathbf{J}^{p} \cap \mathbf{J}_{s}), j \neq j', \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P)} \rho_{i,s} > 0, \sum_{i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C)} \rho_{i',s} < 0, n \qquad (22)$$

$$T^{f}_{j',n} \geq T^{f}_{j,n} - M\left(1 - zD_{j,j',n} + zI_{j,j',n}\right)$$
$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}), (j, j') \in (\mathbf{J}^{p} \cap \mathbf{J}_{s}), j \neq j', \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P)} \rho_{i,s} > 0, \sum_{i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C)} \rho_{i',s} < 0, n \qquad (23)$$

Since materials are transferred from a production unit to a downstream processing unit in direct material transfer, it should be ensured that both processes start and finish at the same time.

$$T^{s}_{j',n} \leq T^{s}_{j,n} + M\left(1 - zD_{j,j',n}\right)$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}),\ (j, j') \in (\mathbf{J}^{p} \cap \mathbf{J}_{s}),\ j \neq j',\ \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P)} \rho_{i,s} > 0\,,\ \sum_{i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C)} \rho_{i',s} < 0\,,\ n \qquad (24)$$

$$T^{f}_{j',n} \leq T^{f}_{j,n} + M\left(1 - zD_{j,j',n}\right)$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}),\ (j, j') \in (\mathbf{J}^{p} \cap \mathbf{J}_{s}),\ j \neq j',\ \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P)} \rho_{i,s} > 0\,,\ \sum_{i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C)} \rho_{i',s} < 0\,,\ n \qquad (25)$$

Finally, the following constraints are introduced to avoid real-time storage violations. In the first set of constraints, it is ensured that the start time of unit *j* processing a producing task *i* at event point (*n* + 1) must be after the time that state s produced by this unit,

$$T^{f}_{s,j,n} \leq T^{s}_{j,n+1} + M\left(1 - \sum_{i \in \left(\mathbf{I}_j \cap \mathbf{I}_s^P\right)} \sum_{n+1 \leq n' \leq n+1+\Delta n} w_{i,j,n+1,n'}\right)$$

$$\forall s \in\ \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}),\ j \in (\mathbf{J}^{p} \cap \mathbf{J}_{s}),\ \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P)} \rho_{i,s} > 0\,,\ n < N \qquad (26a)$$

Additionally, the finish time of unit j' processing a consuming task *i* at event point *n* must be before the time that state s produced by unit *j* finishes being available at event point *n*.

$$T^{f}_{s,j,n} \geq T^{f}_{j',n} - M\left(1 - \sum_{i' \in \left(\mathbf{I}_{j'} \cap \mathbf{I}_s^C\right)} \sum_{n-\Delta n \leq n' \leq n} w_{i,j,n',n}\right)$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}),\ (j, j') \in (\mathbf{J}^{p} \cap \mathbf{J}_{s}),\ j \neq j',\ \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P)} \rho_{i,s} > 0\,,\ \sum_{i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C)} \rho_{i',s} < 0\,,\ n \qquad (26b)$$

3.7 Demand constraints

The quantity of the products produced within the scheduling horizon should fulfill the minimum and maximum market demands. Constraint (27) ensures that the total amount of product state s produced, should be within the demands of this state.

$$D_s^{\min} \leq \sum_{n} \sum_{j} \sum_{i \in \left(\mathbf{I}_j \cap \mathbf{I}_s^P\right)} \rho_{s,i,j} \cdot b_{i,j,n} \leq D_s^{\max}$$

$$\forall s \in \mathbf{S}^{P} \qquad (27)$$

3.8 Tightening constraints

For a given unit, the duration of all tasks processed in this unit cannot exceed the maximum available time.

$$\sum_{i\in\mathbf{I}_j}\sum_{n} L_{i,j,n} \leq H - \tau_j^{\min}$$

$$\forall j \in \mathbf{J}^p \qquad (28)$$

Where $\tau_j^{\min} = \begin{cases} 0 & \tau_j = 0 \cap \min\limits_{i,i'\in\mathbf{I}_j} \tau_{i',i,j} = 0 \\ \tau_j & \tau_j > 0 \\ \min\limits_{i,i'\in\mathbf{I}_j} \tau_{i',i,j} & \min\limits_{i,i'\in\mathbf{I}_j} \tau_{i',i,j} > 0 \end{cases}$

3.9 Variable bounds

All timing variables must not exceed the scheduling horizon. Furthermore, for states with limited storage capacity the stored amount must not exceed the maximum storage capacity.

$$T_{j,n}^{\mathrm{s}} \leq H \qquad \forall j, n \qquad (29)$$

$$T_{j,n}^{\mathrm{f}} \leq H \qquad \forall j, n \qquad (30)$$

$$T_{s,j,n}^{\mathrm{s}} \leq H \qquad \forall j, n \qquad (31)$$

$$T_{s,j,n}^{\mathrm{f}} \leq H \qquad \forall j, n \qquad (32)$$

$$ST_{s,n} \leq ST_s^{\max} \qquad \forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}), n \qquad (33)$$

3.10 Additional constraints

Several additional constraints are introduced to improve the performance of the proposed model. Constraints (34)-(37) relate $w_{i,j,n,n'}$ with $zI_{j,j',n}$. More specifically, if a unit j' process a consumption task i', and there is an indirect material transfer between units j and j' then unit j must process the related production task i according to (34). Similarly, if a unit j processes a production task i, and there is an indirect material transfer between units j and j' then unit j' must process the related consumption task i' according to (35).

$$w_{i,j,n,n} \geq w_{i',j',n,n} + zI_{j,j',n} - 1$$

$$\forall s \in \mathbf{S}^{IN}, (j, j') \in (\mathbf{J}^p \cap \mathbf{J}_s), j \neq j', i \in (\mathbf{I}_j \cap \mathbf{I}_s^P), i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C), n \qquad (34)$$

$$w_{i',j',n,n} \geq w_{i,j,n,n} + zI_{j,j',n} - 1$$

$$\forall s \in \mathbf{S}^{IN}, (j, j') \in (\mathbf{J}^p \cap \mathbf{J}_s), j \neq j', i \in (\mathbf{I}_j \cap \mathbf{I}_s^P), i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C), n \qquad (35)$$

Similarly, we relate $w_{i,j,n,n'}$ with $zD_{j,j',n}$ for states with FIS policy.

$$w_{i,j,n,n} \geq w_{i',j',n,n} + zD_{j,j',n} - 1$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}), (j, j') \in (\mathbf{J}^p \cap \mathbf{J}_s), j \neq j', i \in (\mathbf{I}_j \cap \mathbf{I}_s^P), i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C), n \qquad (36)$$

$$w_{i',j',n,n} \geq w_{i,j,n,n} + zD_{j,j',n} - 1$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}),\ (j, j') \in (\mathbf{J}^{p} \cap \mathbf{J}_{s}),\ j \neq j',\ i \in (\mathbf{I}_{j} \cap \mathbf{I}_{s}^{P}),\ i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_{s}^{C}),\ n \quad (37)$$

3.11 Minimum run time and amount

A task i must be processed for some minimum duration ($RL_{i,j}^{min}$) in a unit j and/or must process a minimum amount ($Rb_{i,j}^{min}$) once it takes place in some cases. To enforce such minimum run time and amount, we impose the following two constraints.

$$\sum_{i \in \mathbf{I}_j} \sum_{n \leq n'' \leq n'} L_{i,j,n''} \geq \sum_{i \in \mathbf{I}_j} RL_{i,j}^{\min} \cdot w_{i,j,n,n'}$$

$$\forall j \in \mathbf{J}^{p},\ n \leq n' \leq n+\Delta n,\ \Delta n > 0,\ \max_{i \in \mathbf{I}_j}\left(RL_{i,j}^{min}\right) > 0 \quad (38)$$

$$\sum_{i \in \mathbf{I}_j} \sum_{n \leq n'' \leq n'} b_{i,j,n''} \geq \sum_{i \in \mathbf{I}_j} Rb_{i,j}^{\min} \cdot w_{i,j,n,n'}$$

$$\forall j \in \mathbf{J}^{p},\ n \leq n' \leq n+\Delta n,\ \Delta n > 0,\ \max_{i \in \mathbf{I}_j}\left(Rb_{i,j}^{min}\right) > 0 \quad (39)$$

3.12 Changeover time

The changeover time can either be sequence-independent or sequence-dependent. In the latter case, the changeover time depends on the sequence of tasks processed in a unit. We define a parameter $\tau_{i',i,j}$ to denote the sequence-dependent changeover time. In cases of sequence-dependent changeover time, the start time of a unit j during event point n should be after its finish time at event point n' ($n' < n$) plus the sequence-dependent time from task i to task i', if it processes tasks i and i' at event points n and n' respectively. Note that if another task i'' is processed between tasks i and i' (i.e. at event point n'' where $n' < n'' < n$) then we should relax this constraint.

$$T_{j,n}^{\mathrm{s}} \geq T_{j,n'}^{\mathrm{f}} + \tau_{i',i,j} \cdot \sum_{n \leq n'' \leq n+\Delta n} w_{i,j,n,n''} - H\left(1 - \sum_{n'-\Delta n \leq n'' \leq n'} w_{i',j,n'',n'}\right) -$$

$$-H\left(\sum_{i''} \sum_{n' < n''} \sum_{n'' \leq n''' \leq n''+\Delta n} w_{i'',j,n'',n'''}\right)$$

$$\forall j \in \mathbf{J}^{p},\ (i, i') \in \mathbf{I}_{j},\ i \neq i',\ n' < n,\ \tau_{i',i,n} > 0 \quad (40)$$

In the case of sequence-independent changeover time, the sequence of the tasks processed in the unit does not affect the changeover time. We define a parameter τ_j to denote the sequence-independent changeover time, which only depends on units. In such case, the start time of a unit j processing a task i during event point n should be after its finish time at event point n' ($n' < n$) plus its sequence-independent time. Note that the changeover time should be enforced, only if a different task i' is processed at event point n'. Similar to sequence-dependent changeover time, if another task i'' is processed between tasks i and i' then this constraint should be relaxed.

$$T^{\mathrm{s}}_{j,n} \geq T^{\mathrm{f}}_{j,n'} + \tau_j \cdot \sum_{n \leq n'' \leq n+\Delta n} w_{i,j,n,n''} - H\left(1 - \sum_{n'-\Delta n \leq n'' \leq n'} w_{i',j,n'',n'}\right) -$$
$$-H\left(\sum_{i''} \sum_{n'<n''} \sum_{n'' \leq n''' \leq n''+\Delta n} w_{i'',j,n'',n'''}\right)$$
$$\forall j \in \mathbf{J}^p, (i, i') \in \mathbf{I}_j, i \neq i', n' < n, \tau_j > 0 \qquad (41)$$

3.13 Objective function

In this problem, we consider the maximization of profit as objective.

$$z = \sum_{s \in \mathbf{S}^P} (p_s \cdot ST0_s) + \sum_{s \in \mathbf{S}^P} p_s \cdot \left(\sum_{j \in (\mathbf{J}^p \cap \mathbf{J}_s)} \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}^P_s)} \sum_n \rho_{i,s} \cdot b_{i,j,n} \right)$$
$$(42)$$

If minimization of makespan is used as objective, then (43) is introduced.

$$MS \geq T^{\mathrm{f}}_{j,n} \qquad \forall j, n = N \qquad (43)$$

Additionally, the time that state s produced by unit j is processed at event point n should not exceed makespan.

$$T^{\mathrm{f}}_{s,j,n} \leq MS \qquad \forall s \in \mathbf{S}^{IN}, j \in (\mathbf{J}^p \cap \mathbf{J}_s), n = N \quad (44)$$

The length of a task i processed in a unit j at event point n must not exceed the maximum available time.

$$\sum_{i \in \mathbf{I}_j} \sum_n L_{i,j,n} \leq MS - \tau^{\min}_j$$
$$\forall j \in \mathbf{J}^p \qquad (45)$$

Finally, (46) and (47) denote all the continuous and binary variables of the model respectively

$$b_{i,j,n}, bTi_{i,j,i',j',n}, bTd_{i,j,i',j',n}, MS, ST_{s,n}, T^{\mathrm{s}}_{s,jn}, T^{\mathrm{f}}_{s,j,n}, T^{\mathrm{s}}_{jn}, T^{\mathrm{f}}_{jn} \geq 0 \qquad (46)$$

$$w_{i,j,n,n'}, zD_{j,j',n}, zI_{j,j',n} \in \{0, 1\} \qquad (47)$$

We complete the mathematical model **M**, which consists of constraints (1)-(42) and (46)-(47) for maximization of productivity, and (1)-(41) and (43)-(47) for minimization of makespan.

3.14 Extensions

3.14.1 Flexible or swing storage

All the above constraints of model **M** consider dedicated storage. In other words, the storage can hold only one state at any time and during the entire scheduling horizon. In practice, a storage unit may be used to store multiple materials in the scheduling horizon but can hold at most one material at any time, which is called a flexible or swing storage tank. We define two sets $\mathbf{I}^{st}$ and $\mathbf{J}^{st}$ to model flexible or swing storage tasks and tanks, which are also included into *I* and *J* respectively. A binary variable $u_{i,j,n}$ is introduced to denote if a storage task i ($i \in \mathbf{I}^{st}$)is active in a storage tank j ($j \in \mathbf{J}^{st}$) at the end of an event point *n*. At a time, only one storage task can be active in a flexible or swing storage unit.

$$\sum_{i \in (\mathbf{I}^{st} \cap \mathbf{I}_j)} u_{i,j,n} = 1 \qquad \forall j \in \mathbf{J}^{st}, n \tag{48}$$

To monitor the transition from one state s to another state s' in a flexible storage unit j ($j \in \mathbf{J}^{st}$), we define a 0-1 continuous variable $ue_{j,n}$ to denote such transition at the end of event point n.

$$ue_{j,n} \geq u_{i,j,n} - u_{i,j,n+1} \qquad \forall j \in \mathbf{J}^{st}, i \in (\mathbf{I}^{st} \cap \mathbf{I}_j), n < N \tag{49}$$

$$ue_{j,n} \geq u_{i,j,n+1} - u_{i,j,n} \qquad \forall j \in \mathbf{J}^{st},\ i \in (\mathbf{I}^{st} \cap \mathbf{I}_j), n < N \tag{50}$$

We also define a continuous variable $bs_{i,j,n}$ to denote the amount of materials stored in storage unit j ($j \in \mathbf{J}^{st}$) by a task i at an event point *n*. The total amount of materials stored should not exceed the maximum capacity of the storage unit.

$$bs_{i,j,n} \leq V_j^{\max} \cdot u_{i,j,n} \qquad \forall j \in \mathbf{J}^{st},\ i \in (\mathbf{I}^{st} \cap \mathbf{I}_j), n \tag{51}$$

If there is a state transition in a storage tank at event point *n*, then the amount of materials stored at this event point should be zero.

$$\sum_{i \in (\mathbf{I}^{st} \cap \mathbf{I}_j)} bs_{i,j,n} \leq V_j^{\max} \cdot (1 - ue_{i,j,n}) \qquad \forall j \in \mathbf{J}^{st}, n < N \tag{52}$$

Material balance in a storage unit can be ensured by modifying material balance constraints (6a) and (6b),

$$0 = ST0_s + \sum_{j \in \mathbf{J}^p} \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^p)} \rho_{i,s} \cdot b_{i,j,n} + \sum_{j \in \mathbf{J}^p} \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^c)} \rho_{i,s} \cdot b_{i,j,n} - \sum_{j \in \mathbf{J}^{st}} \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s)} bs_{j,n}$$

$$\forall s \in \mathbf{S}^{FFIS},\ n = 1 \qquad \text{(6a-FFIS)}$$

$$0 = \sum_{j \in \mathbf{J}^{st}} \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s)} bs_{j,n-1} + \sum_{j \in \mathbf{J}^{p}} \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P)} \rho_{i,s} \cdot b_{i,j,n} + \sum_{j \in \mathbf{J}^{p}} \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^C)} \rho_{i,s} \cdot b_{i_p,j,n} - $$

$$- \sum_{j \in \mathbf{J}^{st}} \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s)} bs_{j,n}$$

$$\forall s \in \mathbf{S}^{FFIS},\ n > 1 \qquad \text{(6b-FFIS)}$$

where $\mathbf{S}^{FFIS}$ denotes state with flexible finite intermediate storage policy. Additionally, constraints (9) and (13) are slightly modified to consider flexible storage instead of dedicated storage.

$$\sum_{j' \in (\mathbf{J}_s \cap \mathbf{J}^P)} \sum_{i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C)} \left(-\rho_{i',s} \cdot b_{i',j',n}\right) \le \sum_{j \in \mathbf{J}^{st}} \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s)} bs0_{i,j} +$$

$$+ \sum_{j \in (\mathbf{J}_s \cap \mathbf{J}^P)} \sum_{j' \in (\mathbf{J}_s \cap \mathbf{J}^P)} \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P)} \sum_{i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C)} bTi_{i,j,i',j',n}$$

$$\forall s \in \mathbf{S}^{FFIS},\ n = 1 \qquad \text{(9a-FFIS)}$$

$$\sum_{j' \in (\mathbf{J}_s \cap \mathbf{J}^P)} \sum_{i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C)} \left(-\rho_{i',s} \cdot b_{i',j',n}\right) \le \sum_{j \in \mathbf{J}^{st}} \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s)} bs_{i,j,n-1} +$$

$$+ \sum_{j \in (\mathbf{J}_s \cap \mathbf{J}^P)} \sum_{j' \in (\mathbf{J}_s \cap \mathbf{J}^P)} \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P)} \sum_{i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C)} bTi_{i,j,i',j',n}$$

$$\forall s \in \mathbf{S}^{FFIS},\ n > 1 \qquad \text{(9b-FFIS)}$$

$$\sum_{j \in (\mathbf{J}_s \cap \mathbf{J}^P)} \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P)} \left(\rho_{i,s} \cdot b_{i,j,n}\right) + \sum_{j \in \mathbf{J}^{st}} \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s)} bs0_{i,j} \le \sum_{j \in \mathbf{J}^{st}} \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s)} \left(V_j^{max} \cdot u_{i,j,n}\right)$$

$$+ \sum_{j \in (\mathbf{J}_s \cap \mathbf{J}^P)} \sum_{j' \in (\mathbf{J}_s \cap \mathbf{J}^P)} \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P)} \sum_{i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C)} bTd_{i,j,i',j',n}$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FFIS}),\ n = 1 \qquad \text{(13a-FFIS)}$$

$$\sum_{j \in (\mathbf{J}_s \cap \mathbf{J}^P)} \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P)} \left(\rho_{i,s} \cdot b_{i,j,n}\right) + \sum_{j \in \mathbf{J}^{st}} \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s)} bs_{i,j,n-1} \le \sum_{j \in \mathbf{J}^{st}} \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s)} \left(V_j^{max} \cdot u_{i,j,n}\right)$$

$$+ \sum_{j \in (\mathbf{J}_s \cap \mathbf{J}^P)} \sum_{j' \in (\mathbf{J}_s \cap \mathbf{J}^P)} \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P)} \sum_{i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C)} bTd_{i,j,i',j',n}$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FFIS}),\ n > 1 \qquad \text{(13b-FFIS)}$$

Finally, (53) and (54) denote the continuous and binary variables that additionally defined.

$$bs_{i,j,n},\ ue_{j,n} \ge 0 \qquad (53)$$

$$u_{i,j,n} \in \{0, 1\} \qquad (54)$$

We complete the mathematical model **M** with flexible intermediate storage, which consists of constraints (1-5), (6a-FFIS), (6b-FFIS), (7-8), (9-FFIS), (10-12), (13-FFIS), (14)-(42), (46-47), (48-52) and (53)-(54) for maximization of productivity, and (1-5), (6a-FFIS), (6b-FFIS), (7-8), (9-FFIS), (10-12), (13-FFIS), (14-41), (43-47), (48-52) and (53)-(54) for minimization of makespan.

3.14.2 Without storage bypassing

Model **M** considers storage bypassing. However, it can be also extended to address the case where storage bypassing is not allowed. In this case, the material produced should first enter a storage tank and then consumed by downstream processing units. There are two scenarios when storage bypass is not allowed. In the first scenario, a storage tank can receive and deliver materials simultaneously. In other words, the producing amount of state s is transferred to the storage tank first and then immediately consumed by the downstream units. This scenario is similar to the case where the storage bypass is allowed. In the second scenario, a storage tank cannot receive and deliver materials at the same time. In other words, storage tasks should store materials after production. Then, it can be consumed by the downstream processing units. In this scenario, there is no indirect and direct material transfer between units. As a result, the variables $zI_{j,j',n}$, $bTi_{i,j,i',j',n}$, $zD_{j,j',n}$ and $bTd_{i,j,i',j',n}$ and their related constraints (7-14) are omitted from the model.

To sequence related production and consumption tasks in different processing units, we define a binary variable $zz_{j,j',n}$ to denote if there is a material transfer from unit j to unit j' during event point n. Since a storage is not allowed to receive and deliver materials at the same time, a storage tank should transfer materials either from a production unit or to a consumption unit during event point n.

$$zz_{j,j'',n} + zz_{j'',j',n} \le 1$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}),\ \boldsymbol{j''} \in (\mathbf{J}^{st} \cap \mathbf{J}_s),\ (j,j') \in (\mathbf{J}^{p} \cap \mathbf{J}_s),\ \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P)} \rho_{i,s} > 0,$$

$$\sum_{i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C)} \rho_{i',s} < 0,\ n \tag{55}$$

A material transfer between a production unit j and a storage tank j'' at event point n, can take place if the production unit finishes processing task i at event point n.

$$\sum_{j'' \in (\mathbf{J}^{st} \cap \mathbf{J}_s)} zz_{j,j'',n} \geq \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P)} \left(\sum_{n' \leq n} \sum_{n' \leq n'' \leq n' + \Delta n} w_{i,j,n',n''} - \sum_{n' < n} \sum_{n' - \Delta n \leq n'' \leq n'} w_{i,j,n'',n'} \right)$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}), j \in (\mathbf{J}^p \cap \mathbf{J}_s), \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P)} \rho_{i,s} > 0, n \qquad (56a)$$

Similarly, a material transfer between a storage tank j'' and a production unit j at event point n, can take place if the consumption unit finishes processing task i' at event point n.

$$\sum_{j'' \in (\mathbf{J}^{st} \cap \mathbf{J}_s)} zz_{j'',j',n} \geq \sum_{i' \in (\mathbf{I}_j \cap \mathbf{I}_s^C)} \left(\sum_{n' \leq n} \sum_{n' \leq n'' \leq n' + \Delta n} w_{i',j',n',n''} - \sum_{n' < n} \sum_{n' - \Delta n \leq n'' \leq n'} w_{i',j',n'',n'} \right)$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}), j' \in (\mathbf{J}^p \cap \mathbf{J}_s), \sum_{i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C)} \rho_{i',s} < 0, n \qquad (56b)$$

We also define $bz_{j,j',n}$ to denote the amount of material transferred from unit j to unit j' during event point n. In this case, the total amount of materials produced (or consumed) should be transferred to (from) a storage tank.

$$\sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P)} b_{i,j,n} = \sum_{j'' \in (\mathbf{J}^{st} \cap \mathbf{J}_s)} bz_{j,j'',n}$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}), j \in (\mathbf{J}^p \cap \mathbf{J}_s), \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P)} \rho_{i,s} > 0, n \qquad (57a)$$

$$\sum_{i' \in (\mathbf{I}_j \cap \mathbf{I}_s^C)} b_{i',j',n} = \sum_{j'' \in (\mathbf{J}^{st} \cap \mathbf{J}_s)} bz_{j'',j',n}$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}), j' \in (\mathbf{J}^p \cap \mathbf{J}_s), \sum_{i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C)} \rho_{i',s} < 0, n \qquad (57b)$$

The amount of material transferred between two units at event point n must be zero if there is not a material transfer between those units at this event point.

$$bz_{j,j'',n} \leq M \cdot zz_{j,j'',n}$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}), \boldsymbol{j''} \in (\mathbf{J}^{st} \cap \mathbf{J}_s), j \in (\mathbf{J}^p \cap \mathbf{J}_s), \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P)} \rho_{i,s} > 0, n \qquad (58a)$$

$$bz_{j'',j',n} \leq M \cdot zz_{j'',j',n}$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}), \boldsymbol{j''} \in (\mathbf{J}^{st} \cap \mathbf{J}_s), j' \in (\mathbf{J}^p \cap \mathbf{J}_s), \sum_{i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C)} \rho_{i',s} < 0, n \qquad (58b)$$

We also introduce a number of additional constraints to improve the performance of the model, similar to the constraints included for indirect and direct material transfer.

$$\sum_{n'\le n}\sum_{n'\le n''\le n'+\Delta n} w_{i,j,n',n''} - \sum_{n'<n}\sum_{n'-\Delta n\le n''\le n'} w_{i,j,n'',n'} \ge u_{i'',j'',n} + zz_{j,j'',n} - 1$$

$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FFIS}), j \in (\mathbf{J}^{p} \cap \mathbf{J}_s), i \in (\mathbf{I}_j \cap \mathbf{I}_s^{P}), \boldsymbol{j}'' \in (\mathbf{J}^{st} \cap \mathbf{J}_s), i'' \in (\mathbf{I}_{j''} \cap \mathbf{I}_s), n$ (59a)

$$u_{i'',j'',n} \ge \sum_{n'\le n}\sum_{n'\le n''\le n'+\Delta n} w_{i,j,n',n''} - \sum_{n'<n}\sum_{n'-\Delta n\le n''\le n'} w_{i,j,n'',n'} + zz_{j,j'',n} - 1$$

$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FFIS}), j \in (\mathbf{J}^{p} \cap \mathbf{J}_s), i \in (\mathbf{I}_j \cap \mathbf{I}_s^{P}), \boldsymbol{j}'' \in (\mathbf{J}^{st} \cap \mathbf{J}_s), i'' \in (\mathbf{I}_{j''} \cap \mathbf{I}_s), n$ (59b)

$$\sum_{n'\le n}\sum_{n'\le n''\le n'+\Delta n} w_{i',j',n',n''} - \sum_{n'<n}\sum_{n'-\Delta n\le n''\le n'} w_{i',j',n'',n'} \ge u_{i'',j'',n} + zz_{j,j'',n} - 1$$

$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FFIS}), \boldsymbol{j}'' \in (\mathbf{J}^{st} \cap \mathbf{J}_s), i'' \in (\mathbf{I}_{j''} \cap \mathbf{I}_s), j' \in (\mathbf{J}^{p} \cap \mathbf{J}_s), i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^{C}), n$(60a)

$$u_{i'',j'',n} \ge \sum_{n'\le n}\sum_{n'\le n''\le n'+\Delta n} w_{i',j',n',n''} - \sum_{n'<n}\sum_{n'-\Delta n\le n''\le n'} w_{i',j',n'',n'} + zz_{j,j'',n} - 1$$

$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FFIS}), \boldsymbol{j}'' \in (\mathbf{J}^{st} \cap \mathbf{J}_s), i'' \in (\mathbf{I}_{j''} \cap \mathbf{I}_s), j' \in (\mathbf{J}^{p} \cap \mathbf{J}_s), i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^{C}), n$(60b)

To sequence related production and consumption tasks in different processing units, we also need to define $T_{j,n}^{\mathrm{s}}$ and $T_{j,n}^{\mathrm{f}}$ to denote the start and end times of a storage tank *j* at event point *n*. In this case the finish time of a storage tank *j* should always be after the start time of the unit at the same event point.

$T_{j,n}^{\mathrm{f}} \ge T_{j,n}^{\mathrm{s}}$ $\quad \forall\, j \in (\mathbf{J}^{st} \cap \mathbf{J}_s), n$ (61)

Similarly, the start time of a storage tank *j* at event point (*n* + 1) should be after the start time of the unit at the previous event point *n*.

$T_{j,n+1}^{\mathrm{s}} \ge T_{j,n}^{\mathrm{f}}$ $\quad \forall\, j \in (\mathbf{J}^{st} \cap \mathbf{J}_s), n$ (62)

Constraints (63)-(66) are introduced to ensure that the start and finish time of storage tanks are before and after unit *j* processing a producing task *i* if there is material transfer between those units.

$$T_{j'',n}^{\mathrm{s}} \le T_{j,n}^{\mathrm{s}} + M\left(1 - zz_{j,j'',n}\right)$$

$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FFIS}), j \in (\mathbf{J}^{p} \cap \mathbf{J}_s), j' \in (\mathbf{J}^{st} \cap \mathbf{J}_s), \sum_{i\in(\mathbf{I}_j \cap \mathbf{I}_s^{P})} \rho_{i,s} > 0, n$ (63)

$$T_{j'',n}^{\mathrm{f}} \ge T_{j,n}^{\mathrm{f}} + M\left(1 - zz_{j,j'',n}\right)$$

$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FFIS}), j \in (\mathbf{J}^{p} \cap \mathbf{J}_s), , j' \in (\mathbf{J}^{st} \cap \mathbf{J}_s), \sum_{i\in(\mathbf{I}_j \cap \mathbf{I}_s^{P})} \rho_{i,s} > 0, n$ (64)

$$T_{j'',n}^{\mathrm{s}} \le T_{j',n}^{\mathrm{s}} + M\left(1 - zz_{j'',j',n}\right)$$

$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FFIS}), j \in (\mathbf{J}^{p} \cap \mathbf{J}_s), , j' \in (\mathbf{J}^{st} \cap \mathbf{J}_s), \sum_{i\in(\mathbf{I}_j \cap \mathbf{I}_s^{P})} \rho_{i,s} > 0, n$ (65)

$$T^{\mathrm{f}}_{j'',n} \geq T^{\mathrm{f}}_{j,n} + M\left(1 - zz_{j'',j',n}\right)$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FFIS}), j \in (\mathbf{J}^{p} \cap \mathbf{J}_s), , j' \in (\mathbf{J}^{st} \cap \mathbf{J}_s), \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}^P_s)} \rho_{i,s} > 0 , n \qquad (66)$$

Finally, (67) and (68) denote the continuous and binary variables that additionally defined.

$$bz_{i,j,n} \geq 0 \qquad (67)$$

$$zz_{j,j',n} \in \{0, 1\} \qquad (68)$$

The mathematical model **M** with storage bypass transfer not allowed and with the case that a storage is not allowed to receive and deliver materials at the same time, which consists of constraints (1-6), (15)-(42), (46)-(47) and (55)-(68) for maximization of productivity, and (1)-(6), (15)-(41),(43)-(47) and (55)-(68) for minimization of makespan.

3.14.3 No intermediate storage

If there is a state s with no intermediate storage policy, then the model **M** can be slightly modified to extend for this case. As $ST_{s,n} = 0$ in NIS policy, the mass balance constraints (6a-b) can be simplified through removal of $ST_{s,n}$ as follows,

$$0 = \sum_{j \in \mathbf{J}^P} \sum_{i \in \left(\mathbf{I}_j \cap \mathbf{I}^P_s\right)} \rho_{i,s} \cdot b_{i,j,n} + \sum_{j \in \mathbf{J}^P} \sum_{i \in \left(\mathbf{I}_j \cap \mathbf{I}^C_s\right)} \rho_{i,s} \cdot b_{i,j,n}$$

$$\forall s \in \mathbf{S}^{NIS}, n \qquad (6\text{-NIS})$$

Since there is no available storage for state s, then the total amount of materials produced must be directly transferred to downstream processing units. Therefore, there is no indirect material transfer for states with NIS policy. In this case, constraints (7-10) can be removed, while constraints (11-12) can be reformulated. More specifically, the amount of materials produced (consumed) from a task i must be equal to the total amount of materials directly transferred to (from) tasks consuming (producing) the same state.

$$\rho_{i,s} \cdot b_{i,j,n} = \sum_{j' \in (\mathbf{J}_s \cap \mathbf{J}^P)} \sum_{i' \in \left(\mathbf{I}_{j'} \cap \mathbf{I}^C_s\right)} bTd_{i,j,i',j',n}$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{NIS}), j \in (\mathbf{J}^p \cap \mathbf{J}_s), i \in (\mathbf{I}_j \cap \mathbf{I}^P_s), n \qquad (11\text{-NIS})$$

$$-\rho_{i',s} \cdot b_{i',j',n} = \sum_{j \in (\mathbf{J}_s \cap \mathbf{J}^P)} \sum_{i \in \left(\mathbf{I}_j \cap \mathbf{I}^P_s\right)} bTd_{i,j,i',j',n}$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{NIS}), j' \in (\mathbf{J}^p \cap \mathbf{J}_s), i' \in (\mathbf{I}_{j'} \cap \mathbf{I}^C_s), n \qquad (12\text{-NIS})$$

As $ST^{max}_s = 0$, constraint (13a-13b) can be simplified as follows,

$$\sum_{j\in(\mathbf{J}_s\cap\mathbf{J}^P)}\sum_{i\in(\mathbf{I}_j\cap\mathbf{I}_s^P)}\left(\rho_{i,s}\cdot b_{i,j,n}\right)=\sum_{j\in(\mathbf{J}_s\cap\mathbf{J}^P)}\sum_{j'\in(\mathbf{J}_s\cap\mathbf{J}^P)}\sum_{i\in(\mathbf{I}_j\cap\mathbf{I}_s^P)}\sum_{i'\in(\mathbf{I}_{j'}\cap\mathbf{I}_s^C)} bTd_{i,j,i',j',n}$$

$$\forall s\in(\mathbf{S}^{IN}\cap\mathbf{S}^{NIS}),\ n \qquad (13\text{-NIS})$$

Constraint (13-NIS) is redundant as it is ensured by constraints (11-NIS). In addition to the previous modifications, constraint (14) is changed to the following.

$$\sum_{i\in(\mathbf{I}_j\cap\mathbf{I}_s^P)}\sum_{i'\in(\mathbf{I}_{j'}\cap\mathbf{I}_s^C)} bTd_{i,j,i',j',n}\le\min\left\{\max_{i\in(\mathbf{I}_j\cap\mathbf{I}_s^P)}\left(R_{i,j}^{\max}\cdot H\right),\max_{i'\in(\mathbf{I}_{j'}\cap\mathbf{I}_s^C)}\left(R_{i',j'}^{\max}\cdot H\right),ST_s^{\max}\right\}\cdot zD_{j,j',n}$$

$$\forall s\in(\mathbf{S}^{IN}\cap\mathbf{S}^{NIS}),\ j\neq j',\ (j,j')\in(\mathbf{J}^p\cap\mathbf{J}_s),\ n \qquad (14\text{-NIS})$$

As there is no intermediate storage, variables $T_{s,j,n}^{s}$ and $T_{s,j,n}^{\mathrm{f}}$ with related constraints are no longer used. As a result, constraints (17-21) and (26) are removed. Constraints (22-25) are modified to the following constraints for the case of NIS.

$$T_{j',n}^{\mathrm{s}}\ge T_{j,n}^{\mathrm{s}}-M\left(1-zD_{j,j',n}\right)$$

$$\forall s\in(\mathbf{S}^{IN}\cap\mathbf{S}^{NIS}),\ (j,j')\in(\mathbf{J}^p\cap\mathbf{J}_s),\ j\neq j',\ \sum_{i\in(\mathbf{I}_j\cap\mathbf{I}_s^P)}\rho_{i,s}>0,\ \sum_{i'\in(\mathbf{I}_{j'}\cap\mathbf{I}_s^C)}\rho_{i',s}<0,\ n \qquad (22\text{-NIS})$$

$$T_{j',n}^{\mathrm{f}}\ge T_{j,n}^{\mathrm{f}}-M\left(1-zD_{j,j',n}\right)$$

$$\forall s\in(\mathbf{S}^{IN}\cap\mathbf{S}^{NIS}),\ (j,j')\in(\mathbf{J}^p\cap\mathbf{J}_s),\ j\neq j',\ \sum_{i\in(\mathbf{I}_j\cap\mathbf{I}_s^P)}\rho_{i,s}>0,\ \sum_{i'\in(\mathbf{I}_{j'}\cap\mathbf{I}_s^C)}\rho_{i',s}<0,\ n \qquad (23\text{-NIS})$$

$$T_{j',n}^{\mathrm{s}}\le T_{j,n}^{\mathrm{s}}+M\left(1-zD_{j,j',n}\right)$$

$$\forall s\in(\mathbf{S}^{IN}\cap\mathbf{S}^{NIS}),\ (j,j')\in(\mathbf{J}^p\cap\mathbf{J}_s),\ j\neq j',\ \sum_{i\in(\mathbf{I}_j\cap\mathbf{I}_s^P)}\rho_{i,s}>0,\ \sum_{i'\in(\mathbf{I}_{j'}\cap\mathbf{I}_s^C)}\rho_{i',s}<0,\ n \qquad (24\text{-NIS})$$

$$T_{j',n}^{\mathrm{f}}\le T_{j,n}^{\mathrm{f}}+M\left(1-zD_{j,j',n}\right)$$

$$\forall s\in(\mathbf{S}^{IN}\cap\mathbf{S}^{NIS}),\ (j,j')\in(\mathbf{J}^p\cap\mathbf{J}_s),\ j\neq j',\ \sum_{i\in(\mathbf{I}_j\cap\mathbf{I}_s^P)}\rho_{i,s}>0,\ \sum_{i'\in(\mathbf{I}_{j'}\cap\mathbf{I}_s^C)}\rho_{i',s}<0,\ n \qquad (25\text{-NIS})$$

The mathematical model **M** with NIS, which consists of constraints (1-5), (6-NIS), (11-NIS)-(14-NIS), (15-16), (22-NIS)-(25-NIS), (27-30), (36-42) and (46-47) for maximization of productivity, and (1-5), (6-NIS), (11-NIS)-(14-NIS), (15-16), (22-NIS)-(25-NIS), (27-30), (36-41), (43) and (45-47) for minimization of makespan.

3.14.4 Planned maintenance

Two parameters T_j^{ms} and T_j^{mf}, which denote the start and the finish time of maintenance for processing unit *j* are introduced. If a processing unit is under maintenance, then no task can start or end during this period. Maintenance can take place in three different periods; at the beginning of the scheduling horizon, at the end of the scheduling horizon and in the middle the scheduling horizon. In the first case, the start and finish times of all tasks processed in the unit with planned maintenance should start after the finish time of the maintenance.

$$T_{j,n}^{s} \geq T_{j}^{\mathrm{mf}} \qquad \forall\ j \in \mathbf{J}_1^{\mathbf{m}}, n \qquad (69)$$

$$T_{j,n}^{\mathrm{f}} \geq T_{j}^{\mathrm{mf}} \qquad \forall\ j \in \mathbf{J}_1^{\mathbf{m}}, n \qquad (70)$$

where $\mathbf{J}_1^{\mathbf{m}}$ denotes the units with planned maintenance at the beginning of the scheduling horizon.

In the second case, the start and the end time of all tasks processed in the unit must be before the start time of planned maintenance.

$$T_{j,n}^{s} \leq T_{j}^{\mathrm{ms}} \qquad \forall\ j \in \mathbf{J}_2^{\mathbf{m}}, n \qquad (71)$$

$$T_{j,n}^{\mathrm{f}} \leq T_{j}^{\mathrm{ms}} \qquad \forall\ j \in \mathbf{J}_2^{\mathbf{m}}, n \qquad (72)$$

where $\mathbf{J}_2^{\mathbf{m}}$ denotes the units with planned maintenance at the end of the scheduling horizon.

Finally, if maintenance takes place in the middle of the scheduling horizon, then the problem is divided into two parts; in the part where the tasks are processed before planned maintenance and in the part where the tasks are processed after planned maintenance. In this case, it is necessary to define two different sets of event points one for the first part (i.e. N_1) and one at the second part (i.e N_2). In both cases the start and the finish time of all tasks processed in the unit under maintenance should not be within the maintenance period.

$$T_{j,n}^{s} \leq T_{j}^{\mathrm{ms}} \qquad \forall\ j \in \mathbf{J}_3^{\mathbf{m}}, n \in \mathbf{N}_1 \qquad (73)$$

$$T_{j,n}^{\mathrm{f}} \leq T_{j}^{\mathrm{ms}} \qquad \forall\ j \in \mathbf{J}_3^{\mathbf{m}}, n \in \mathbf{N}_1 \qquad (74)$$

$$T_{j,n}^{s} \geq T_{j}^{\mathrm{mf}} \qquad \forall\ j \in \mathbf{J}_3^{\mathbf{m}}, n \in \mathbf{N}_2 \qquad (75)$$

$$T_{j,n}^{\mathrm{f}} \geq T_{j}^{\mathrm{mf}} \qquad \forall\ j \in \mathbf{J}_3^{\mathbf{m}}, n \in \mathbf{N}_2 \qquad (76)$$

where $\mathbf{J}_3^{\mathbf{m}}$ denotes the units with planned maintenance at the middle of the scheduling horizon.

4. Computational studies

We solve three well-established examples from the literature (Shaik and Floudas 2007; Li *et al.* 2010; Omar and Shaik 2019) to illustrate the capabilities of the proposed model. Figures 2-4 depict the STN representations. Tables 1-11 contain the data for all Examples.

In Examples 1a, 2a, and 3a-3d, all units can process tasks without any planning maintenance during the whole scheduling horizon, whilst planning maintenance takes place in Examples 1b-1d, 2b-2d and 3e-3g during the scheduling horizon. The maintenance periods for all units are given in Table 12. To avoid generating solutions where a task *i* is processed in a unit *j* from event point *n* to *n′* $(w_{i,j,n,n'} = 1)$ but the duration of the process is zero, we impose a minimum duration ($RL_{i,j}^{min}$) of 0.1 h for all tasks. We also use the model of Omar and Shaik (2019) (denoted as OS) to solve all these problems for comparison. All examples are solved to zero optimality gap using CPLEX 12/GAMS 24.8.5. on a desktop computer with Intel® Core™ i7-4702HQ 2.2 GHz and 8 GB RAM running Windows 10. The maximum computational time is 1 hour.

Table 1 Data of processing tasks for Examples 1a-1d

Task	Unit	R_i^{max} (ton/h)	Task	Unit	R_i^{max} (ton/h)
I1	J1	20	I8	J7	10
I2	J1	20	I9	J5	10
I3	J2	20	I10	J7	4
I4	J3	20	I11	J5	6
I5	J4	20	I12	J6	6
I6	J5	6	I13	J7	5
I7	J6	5.5	-	-	-

Table 2 Data of states for Examples 1a-1d

State	$ST0_s$	ST_s^{max}	D_i^{min}	D_i^{max}	p_s	State	$ST0_s$	ST_s^{max}	D_i^{min}	D_i^{max}	p_s
Example 1a, c											
S1	∞	∞	-	-	1	S8	0	200	-	-	1
S2	∞	∞	-	-	1	S9	0	200	-	-	1
S3	∞	∞	-	-	1	S10	0	∞	-	-	1
S4	∞	∞	-	-	1	S11	0	∞	-	-	1
S5	0	60	-	-	1	S12	0	∞	-	-	1
S6	0	200	-	-	1	S13	0	∞	-	-	1
S7	0	200	-	-	1	S14	0	∞	-	-	1
Example 1b											
S1	∞	∞	-	-	1	S8	0	200	-	-	1
S2	∞	∞	-	-	1	S9	0	200	-	-	1
S3	∞	∞	-	-	1	S10	0	∞	-	-	1
S4	∞	∞	-	-	1	S11	0	∞	220	270	1
S5	0	60	-	-	1	S12	0	∞	251	300	1
S6	0	200	-	-	1	S13	0	∞	116	140	1
S7	0	200	-	-	1	S14	0	∞	15	25	1

Table 3 Changeover times for Examples 1a-1d

Changeover tasks	Unit	$t^{cl}_{i,i'}$ (h)	τ^{min}_{j} (h)
I2 → I1	J1	4	0
I6 → I9	J5	3	0
I9 → I6	J5	6	0
I9 → I11	J5	6	0
I6 → I11	J5	6	0
I11 → I9	J5	3	0
I7 → I12	J6	6	0
I8 → I10	J7	6	0
I8 → I13	J7	6	0
I10 → I13	J7	6	0
I13 → I8	J7	2	0
I13 → I10	J7	2	0

Table 4 Data of processing tasks for Examples 2a-2d

Task	Unit	$R^{max}_{i,j}$ (ton/h)
I1	J1	10
I2	J1	10
I3	J2	4
I4	J3	4
I5	J2	4
I6	J3	4
I7	J4	1

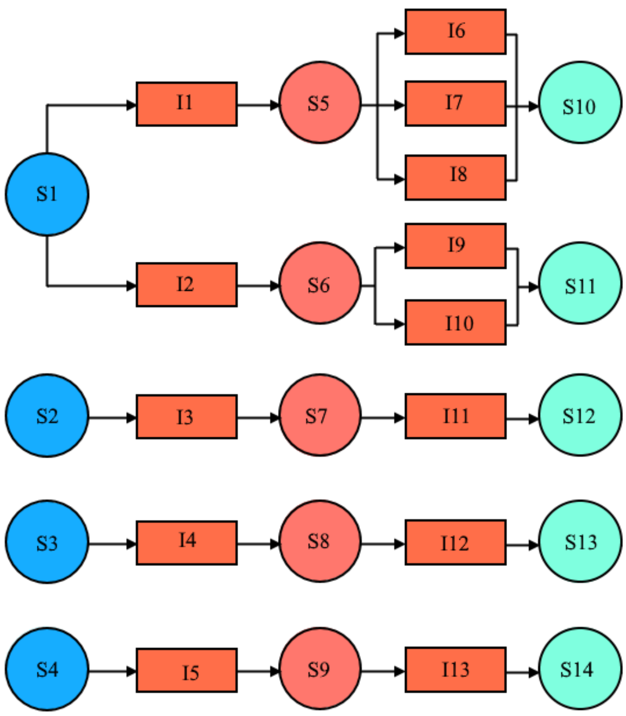

Figure 2 STN representation of Examples 1a-1d

Table 5 Data for storage tanks for Examples 2a-2d

Tank	Maximum storage	S_u
U1	40	S2, S3
U2	40	S2

Table 6 Changeover times for Examples 2a-2d

Changeover tasks	Unit	$t^{cl}_{i,i'}$ (h)	τ^{min}_j (h)
I2 → I1	J5	5	0
I5 → I3	J6	5	0
I6 → I4	J4	5	0

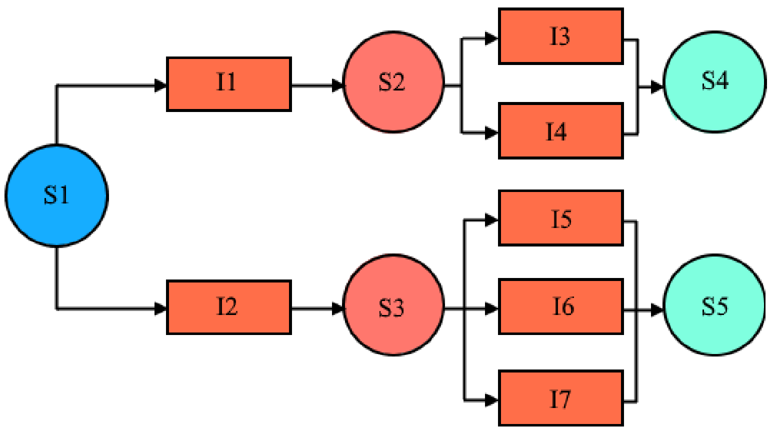

Figure 3 STN representation of Examples 2a-2d

Table 7 Data for states for examples 2a-2d

State	$ST0_s$	ST^{max}_s	D^{min}_i	D^{max}_i	p_s
Example 2a, c					
S1	∞	∞	-	-	1
S2	∞	40	-	-	1
S3	∞	80	-	-	1
S4	∞	∞	-	∞	1
S5	0	∞	-	∞	1
S6	0	∞	-	∞	1
S7	0	∞	-	∞	1
S8	0	∞	-	∞	1
Example 2b					
S1	∞	∞	-	-	1
S2	∞	40	-	-	1
S3	∞	80	-	-	1
S4	∞	∞	100	∞	1
S5	0	∞	100	∞	1
S6	0	∞	20	∞	1
S7	0	∞	20	∞	1
S8	0	∞	10	∞	1

Table 8 Data of processing tasks for Example 3

Task	Unit	$R_{i,j}^{max}$ (ton/h)	Task	Unit	$R_{i,j}^{max}$ (ton/h)	Task	Unit	$R_{i,j}^{max}$(ton/h)
I1	J1	17.00	I10	J3	12.24	I19	J7	2.2410
I2	J1	17.00	I11	J2	12.24	I20	J5	5.8333
I3	J2	17.00	I12	J3	12.24	I21	J6	2.7083
I4	J3	17.00	I13	J4	5.5714	I22	J8	5.3571
I5	J2	17.00	I14	J5	5.5333	I23	J8	5.3571
I6	J3	17.00	I15	J6	2.7083	I24	J7	3.3333
I7	J2	12.24	I16	J5	5.8333	I25	J7	2.2410
I8	J3	12.24	I17	J6	2.7083	I26	J6	2.7083
I9	J2	12.24	I18	J4	5.5714	I27	J7	3.3333

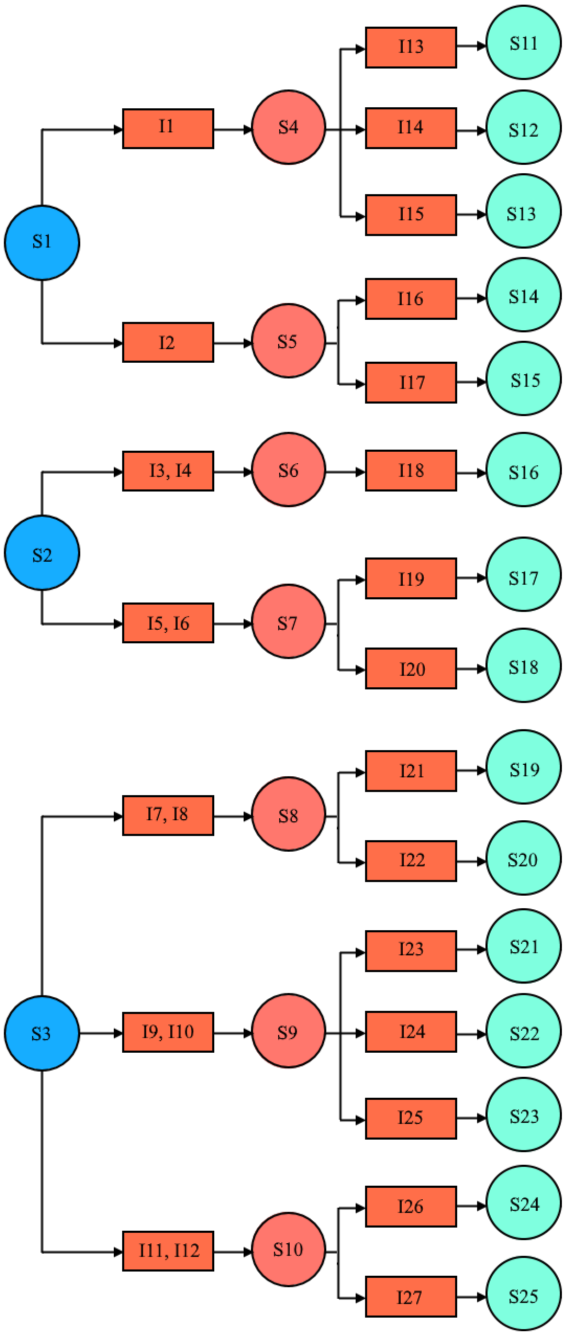

Figure 4 STN representation of Example 3

Table 9 Data of states for Example 3

State	$ST0_s$	ST_s^{max}	D_i^{min}	D_i^{max}	p_s	State	$ST0_s$	ST_s^{max}	D_i^{min}	D_i^{max}	p_s
Example 3a											
S1	∞	∞	-	-	1	S14	0	∞	15	25	1
S2	∞	∞	-	-	1	S15	0	∞	7	20	1
S3	∞	∞	-	-	1	S16	0	∞	47	60	1
S4	0	180	-	-	1	S17	0	∞	8.5	10	1
S5	0	180	-	-	1	S18	0	∞	144	200	1
S6	0	180	-	-	1	S19	0	∞	42.5	60	1
S7	0	180	-	-	1	S20	0	∞	114.5	150	1
S8	0	180	-	-	1	S21	0	∞	53	80	1
S9	0	180	-	-	1	S22	0	∞	2.5	5	1
S10	0	180	-	-	1	S23	0	∞	16.5	25	1
S11	0	∞	220	270	1	S24	0	∞	13.5	18	1
S12	0	∞	251	300	1	S25	0	∞	17.5	25	1
S13	0	∞	116	140	1						
Example 3b											
S1	∞	∞	-	-	1	S14	0	∞	15	∞	1
S2	∞	∞	-	-	1	S15	0	∞	7	∞	1
S3	∞	∞	-	-	1	S16	0	∞	47	∞	1
S4	0	180	-	-	1	S17	0	∞	8.5	∞	1
S5	0	180	-	-	1	S18	0	∞	144	∞	1
S6	0	180	-	-	1	S19	0	∞	42.5	∞	1
S7	0	180	-	-	1	S20	0	∞	114.5	∞	1
S8	0	180	-	-	1	S21	0	∞	53	∞	1
S9	0	180	-	-	1	S22	0	∞	2.5	∞	1
S10	0	180	-	-	1	S23	0	∞	16.5	∞	1
S11	0	∞	220	∞	1	S24	0	∞	13.5	∞	1
S12	0	∞	251	∞	1	S25	0	∞	17.5	∞	1
S13	0	∞	116	∞	1						
Example 3c											
S1	∞	∞	-	-	1	S14	0	∞	15	25	1
S2	∞	∞	-	-	1	S15	0	∞	7	20	1
S3	∞	∞	-	-	1	S16	0	∞	47	60	1
S4	0	60	-	-	1	S17	0	∞	8.5	10	1
S5	0	60	-	-	1	S18	0	∞	144	200	1
S6	0	60	-	-	1	S19	0	∞	42.5	60	1
S7	0	60	-	-	1	S20	0	∞	114.5	150	1
S8	0	60	-	-	1	S21	0	∞	53	80	1
S9	0	60	-	-	1	S22	0	∞	2.5	5	1
S10	0	60	-	-	1	S23	0	∞	16.5	25	1
S11	0	∞	220	270	1	S24	0	∞	13.5	18	1
S12	0	∞	251	300	1	S25	0	∞	17.5	25	1
S13	0	∞	116	140	1						

Example 3d											
S1	∞	∞	-	-	1	S14	0	∞	15	∞	1
S2	∞	∞	-	-	1	S15	0	∞	7	∞	1
S3	∞	∞	-	-	1	S16	0	∞	47	∞	1
S4	0	60	-	-	1	S17	0	∞	8.5	∞	1
S5	0	60	-	-	1	S18	0	∞	144	∞	1
S6	0	60	-	-	1	S19	0	∞	42.5	∞	1
S7	0	60	-	-	1	S20	0	∞	114.5	∞	1
S8	0	60	-	-	1	S21	0	∞	53	∞	1
S9	0	60	-	-	1	S22	0	∞	2.5	∞	1
S10	0	60	-	-	1	S23	0	∞	16.5	∞	1
S11	0	∞	220	∞	1	S24	0	∞	13.5	∞	1
S12	0	∞	251	∞	1	S25	0	∞	17.5	∞	1
S13	0	∞	116	∞	1						

Table 10 Data for storage tanks for Example 3

Tank	Maximum storage	$\boldsymbol{S}_u$	
U1	60	Example 3a, 3c	S4-S10
		Example 3b, 3d	S4, S7, S9
U2	60	Example 3a, 3c	S4-S10
		Example 3b, 3d	S5, S8
U3	60	Example 3a, 3c	S4-S10
		Example 3b, 3d	S6, S10

Table 11 Changeover times for Example 3

Changeover tasks	Unit	$t^{cl}_{i,i'}$ (h)	τ_j^{min} (h)
(I14, I16) → I20	J5	1	1
(I15, I17) → (I21, I26)	J6	4	4
I13 → I18	J4	1	1
(I24, I27) → (I25, I19)	J7	2	2

Table 12 Maintenance periods for Examples 1-3

Example	Unit	Maintenance Period
1b	J5	9 h – 12 h
1c	J5	9h – 12 h
1d	J1	3 h – 6 h
	J5	9h – 12 h
2b	J2	25 h – 30 h
2c	J2	30 h – 35 h
2d	J1	15 h – 20 h
	J2	30 h – 35 h
3e	J5	110 h – 120 h
3f	J5	110 h – 120 h
3g	J1	60 h – 70 h
	J5	110 h – 120 h

The computational results with UIS for Examples 1-3 from both model **M** and the model of Omar and Shaik (2019) are presented in Tables 13-14. While Table 13 depicts the results without planned maintenance, Table 14 demonstrates the results with planned maintenance. From those results, it seems that both mathematical models can generate the optimal solution by using the same number of event points. However, the proposed model **M** leads to slightly fewer binary variables. For instance, model **M** requires 38 binary variables to generate the optimal solution of 399 $ for Example 1a, which is 13.2% less than the model of Omar and Shaik (2019) which requires 43. Despite that, model **M** requires more continuous variables and constraints than the model of Omar and Shaik (2019). For instance, model **M** requires 102 continuous variables and 162 constraints for Example 2a, while the model of Omar and Shaik (2019) requires 42.1% and 24.1% less continuous variables and constraints respectively (59 and 123 respectively). Since the number of binary variables affects the efficiency of the models, the model **M** can generate solutions in significantly less computational time for some examples. For instance, the model of Omar and Shaik (2019) require 1315.2 s for Example 3f, which is two orders of magnitude more than model **M** (67.7 s).

Table 13 Computational results for Examples 1 – 3 with no planned maintenance (UIS policy)

Example	Model	Event Points	Bin. Var.	Cont. Var.	Con str.	RMILP ($)	MILP ($)	Profit ($)	CPU Time (s)
1a	**OS**	2	42	113	223	4985.83	3975.00	399.00	0.08
(H = 12 h)	**M**	2	38	183	291	499.00	399.00	399.00	0.05
2a	**OS**	2	24	59	123	2696.02	2488.00	250.00	0.09
(H = 30 h)	**M**	2	20	102	162	270.00	250.00	250.00	0.05
3a	**OS**	4	208	465	1243	13876.71	13856.00	1388.00	3.2
(H = 120 h)	**M**	4	160	700	1311	1388.00	1388.00	1388.00	0.1
3b	**OS**	4	208	465	1243	27235.68	27097.18	2712.82	501.5
(H = 120 h)	**M**	4	160	700	1343	2724.22	2712.82	2712.82	95.8

OS: Omar and Shaik 2019.

Table 14 Computational results for Examples 1 – 3 with planned maintenance (UIS policy)

Example	Model	Event Points	Bin. Var.	Cont. var.	Constr.	RMILP	MILP	Profit ($)	CPU time (s)
1b	**OS**	3	63	69	355	5384.84	4662.27	467.72	0.4
(H = 12 h)	**M**	3	57	271	465	519.27	467.72	467.72	0.3
1c	**OS**	3	63	169	360	4905.45	3133.00	315.00	0.3
(H = 12 h)	**M**	3	57	271	470	439.00	315.00	315.00	0.2
1d	**OS**	2/1	63	169	360	4905.45	3130.00	315.00	0.2
(H = 12 h)	**M**	2/1	57	271	470	427.10	315.00	315.00	0.3
2b	**OS**	1	12	30	55	2695.20	2493.00	250.00	0.06
(H = 30 h)	**M**	1	10	48	68	250.00	250.00	250.00	0.09
2c	**OS**	2	24	59	123	3146.10	2738.00	275.00	0.1
(H = 35 h)	**M**	2	20	102	162	295.00	275.00	275.00	0.09
2d	**OS**	1/2	36	88	192	3146.10	2483.00	250.00	0.2
(H = 20 h)	**M**	1/2	30	151	254	295.00	250.00	250.00	0.05
3e	**OS**	1	54	119	211	27356.0	26768.8	2678.08	0.08
(H = 120 h)	**M**	1	40	168	230	2678.08	2678.08	2678.08	0.08
3f	**OS**	4	208	465	1243	27235.7	26513.8	2654.48	1315.2
(H = 120 h)	**M**	4	160	700	1311	2665.89	2654.48	2654.48	67.7
3g	**OS**	1/3	208	465	1243	27235.7	26513.9	2654.48	296.6
(H = 120 h)	**M**	1/3	160	700	1311	2665.89	2654.48	2654.48	30.5

OS: Omar and Shaik 2019.

Tables 15 and 16 depicts the computational results for Examples 1-3 with FIS. Note that Examples 2a-2d, 3a-3g are examples with flexible storage. Since the model of Omar and Shaik (2019) do not consider flexible storage, we only solve those examples by using model **M**. For examples with dedicated storage (examples 1a-1d), it seems that model **M** can generate the optimal solution by using fewer binary variables and more continuous variables and constraints, similar to examples with UIS policy. Additionally, it seems that the model of Omar and Shaik (2019) requires more event points in some cases, which further increases the number of binary variables needed to generate the optimal solution. For instance, the model of Omar and Shaik (2019) requires 4 event points to provide the optimal solution for Example 1c, while model **M** requires 3 event points. As a result, the model of Omar and Shaik (2019) requires 35.3% more binary variables than model **M** (116 vs 75 binary variables). Overall, we can conclude that **M** is more generic and efficient than the model of Omar and Shaik (2019).

Table 15 Computational results for Examples 1 – 3 with no planned maintenance (FIS policy)

Example	Model	Event Points	Bin. Var.	Conti. Var.	Constr.	RMILP	MILP	Profit ($)	CPU time (s)
1a	**OS**	4	116	257	833	4985.83	3768.76	379.28	39.9
(H = 12 h)	**M**	4	100	391	1111	499.00	379.28	379.28	28.8
2a (H = 30 h)	**M**	2	26	128	202	270.00	250.00	250.00	0.2
3a (H = 120 h)	**M**	4	216	882	948	1388.00	1388.00	1388.00	0.2
3b (H = 120 h)	**M**	4	216	882	1582	2724.22	2712.82	2712.82	132.6
3c (H = 120 h)	**M**	4	180	856	1474	1388.00	1388.00	1388.00	0.2
3d (H = 120 h)	**M**	4	180	856	1474	2724.22	2712.82	2712.82	58.6

Figures 5 and 6 depict the schedule of example 1c from models **M** and **OS**, respectively. By carefully examining those generated schedules, it can be explained why **OS** requires more event points than **M** in some cases. More specifically, from Figure 5, it seems that task I1 is processed in unit J1 at event point N3. Materials that were produced by J1 at event point N3 are transferred in units J3 and J4 which are processing consuming tasks I7 and I8 respectively. The start time of task I8 at event point N3 is after the start time of producing task I1. Since the model **OS** enforces the start time of related production and consumption tasks to be equal, this schedule is infeasible. In this case, one additional event point is required to avoid generating a suboptimal solution. From Figure 6, task I1 is processed in unit J1 in both event point N3 and N4. Materials that were produced by J1 at event point N3 are transferred in unit J3 which is processing consuming task I7. For the next event point materials are transferred from unit J1 to units J6 and J7. In this schedule, the start time of all related production and consumption tasks is the same.

Table 16 Computational results for Examples 1 – 3 with no planned maintenance (FIS policy)

Example	Model	Event Points	Bin. Var.	Cont. var.	Constr.	RMILP	MILP	Profit ($)	CPU time (s)
1b	**OS**	3	87	193	593	5384.64	4656.27	467.72	0.5
(H = 12 h)	**M**	3	75	295	795	519.27	467.72	467.72	0.3
1c	**OS**	4	116	257	833	4905.85	3133.00	315.00	5.1
(H = 12 h)	**M**	3	75	295	800	439.00	315.00	315.00	0.3
1d	**OS**	2/2	116	257	833	4905.85	3128.00	315.00	3.5
(H = 12 h)	**M**	2/1	75	295	800	427.10	315.00	315.00	0.3
2b (H = 30 h)	**M**	1	23	61	89	250.00	250.00	250.00	0.1
2c (H = 35 h)	**M**	2	26	128	202	295.00	275.00	275.00	0.03
2d (H = 20 h)	**M**	1/2	39	190	313	295.00	250.00	250.00	0.05
3e (H = 120 h)	**M**	1	54	232	299	2678.08	2678.08	2678.08	0.08
3f (H = 120 h)	**M**	4	216	948	1582	2665.89	2654.48	2654.48	55.7
3g (H = 120 h)	**M**	1/3	216	948	1582	2665.89	2654.48	2654.48	11.4

5. Conclusions

In this work, the proposed approach presented in our previous work (Rakovitis *et al.* 2019) was implemented to solve the scheduling of continuous processes problem. In this model, we implement the concept of indirect and direct material transfer, to conditionally sequence and synchronize related production and consumption tasks. We also consider different operating rules, including storage bypass allowed or not allowed and flexible intermediate storage policy. In the latter case, a storage tank can or cannot receive and deliver materials simultaneously. We also extend our model to consider the case where processing units undergo planned maintenance during the scheduling horizon. From the generating results, it seems that the proposed model leads to smaller model sizes with less number of event points and binary variables required in comparison to Omar and Shaik (2019) model. Additionally, the model presented in this work is more efficient, since it can generate optimal schedules in up to two magnitudes less computational time in comparison to the model of Omar and Shaik (2019).

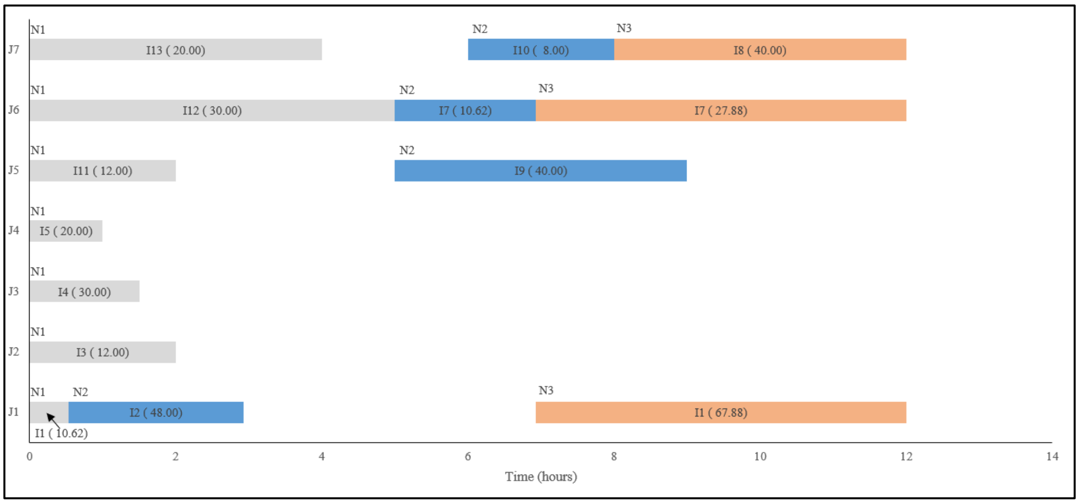

Figure 5 Optimal schedule for example 1c using model **M**

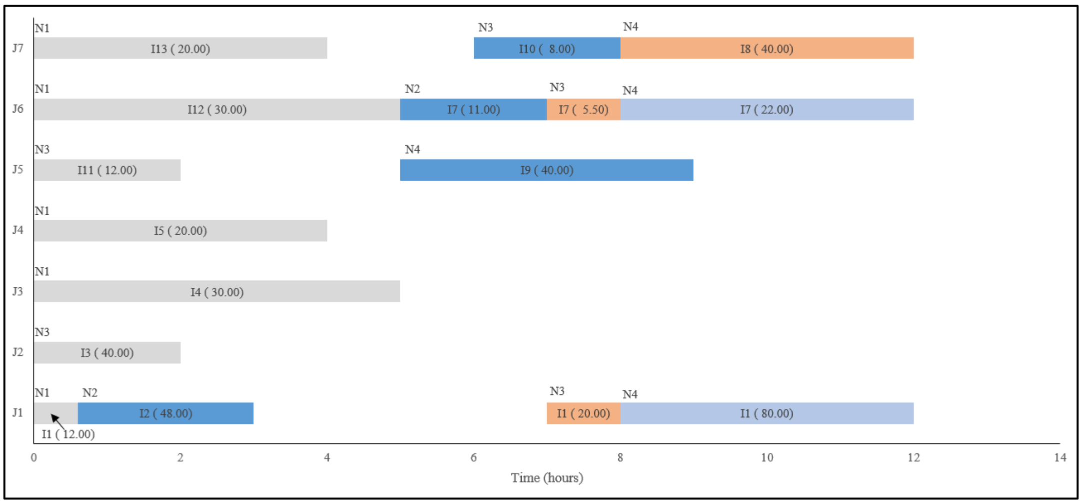

Figure 6 Optimal schedule for example 1c using model **OS**

References

Adiri I., Bruno J., Frostig E., Kan R. A.H.G. Single machine flow-time scheduling with a single breakdown, *Acta Infomatica*, 1989, 26, 679-696

Castro P. M., Barbosa-Póvoa A. P., Matos H. A., Novais A. Q. Simple Continuous-Time Formulation for Short-Term Scheduling of Batch and Continuous Processes, *Industrial Engineering and Chemistry Research*, 2004;43(1);105-118

Dedopoulos I. T., Shah N. 1995. Optimal short-term scheduling of maintenance and production for multipurpose plants, Industrial & Engineering Chemistry Research, 34(1), 192-201

Floudas, C. A., Lin, X., 2004. Continuous-time versus discrete-time approaches for scheduling of chemical processes: a review. Computers and Chemical engineering. 28(11). 2109-2129. https://doi.org/10.1016/j.compchemeng.2004.05.002.

Hadidi L. A., Al-Turki U. M., Rahim M. A. Joint job scheduling and preventive maintenance on a single machine., Interna Journal operation research 2012, 13(2), 174-184

Harjunkoski, I., Maravelias, C. T., Bongers, P., Castro, P. M., Engell, S., Grossmann, I. E., Hooker, J., Méndez, C., Sand, G., Wassick, J., 2014. Scope for industrial application of production scheduling models and solution methods. Computers and Chemical Engineering. 62(5). 161-193. https://doi.org/10.1016/j.compchemeng.2013.12.001.

Hazaras M. J., Swartsz C. L. E., Marlin T. E. 2012, Flexible maintenance within a continuous-time state-task network framework, Computers and Chemical engineering, 46, 167-177

Jain V. , Grossmann I. E. 1998. Cyclic scheduling of continuous parallel-process units with decaying performance, AlChE journal, 44(7), 1623-1636

Karimi I. A., Mcdonald C. M. (1997). Planning and Scheduling of parallel semicontinuous processes 2. Short-term scheduling. *Industrial and Engineering Chemistry Research*, 36(7), pp.2701-2714

Kopanos G. M., Pulgjaner L., Maravelias C. T., Production planning and scheduling of paraller continuous processes with product families, *Industrial Engineering and Chemistry Research*, 2011;50(3);1369-1378

Lee C., Lin C. 2001. Single-machine scheduling with maintenance and repair rate modifying activities, European journal of operational research, 135(3), 493-513

Lee K., Park H. I., Lee I. A Novel Nonuniform Discrete Time Formulation for Short-Term Scheduling of Batch and Continuous Processes, *Industrial and Engineering Chemistry research,* 2001;40(22);4902-4911

Li J., Xiao X., Tang Q., Floudas C. A. Production Scheduling of a Large-Scale Steelmaking Continuous Casting Process via Unit-Specific Event-Based Continuous-Time Models: Short-Term and Medium-Term Scheduling, *Industrial & Engineering Chemistry research*, 2012;51(21);7300-7319

Lu Z., Cui W., Han X. 2015. Integrated production and preventive maintenance scheduling for a single machine with failure uncertainty. Computers and industrial engineering, 80, 236-244

Méndez, C. A., Cerdá, J., Grossmann, I. E., Harjukoski, I., Fahl, M., 2006. State-of-the-art review of optimization methods for short-term scheduling of batch processes. Computers and Chemical Engineering. 30(6-7). 913-946. https://doi.org/10.1016/j.compchemeng.2006.02.008.

Mockus L., Reklaitis G. V. Reklaitis, Continuous time representation approach to batch and continuous process scheduling. 1. MINLP formulation, *Industrial and Engineering Chemistry research*, 1999;38(1);197-203

Rakovitis N., Zhang N., Li J., Zhang L., A new approach for scheduling of multipurpose batch processes with unlimited intermediate storage policy, Frontiers of chemical science and engineering, 2019, 13, 784-802 https://doi.org/10.1007/s11705-019-1858-4

Rakovitis N., Zhang N., Li J., Generic mathematical formulations for scheduling of multipurpose batch plants, AIChE Journal, 2020, under review.

Schilling G., Pantelides C. C. A simple continuous-time process scheduling formulation and a novel solution algorithm, *Computers and Chemical Engineering*, 1996;20(2);S1221-S1226

Shaik M. A., Floudas C. A., Kallrath J., Pitz H. Production scheduling of a large-scale industrial continuous plant: Short-term and medium-term scheduling, *Computers and Chemical Engineering* ,2009;33(3);670-686

Su L. Tsai H. 2010 Flexible preventive maintenance planning for two parallel machines problem to minimize makespan, Journal of Quality in maintenance engineering, 16(3) 288-302

Tang Q H, Li J, Floudas C A, Deng M X, Yan Y B, Xi Z H, Chen P H, Kong J Y. Optimization framework for process scheduling of operation-dependent automobile assembly lines. Optimization Letters, 2012, 6(4): 797-824

Xu D., Cheng Z., Yin Y., Li, H. 2009. Makespan minimization for two parallel machines scheduling with a periodic availability constraint, Computers and Operations research, 36(6), 1809-1812

Yand D., Hung C., Hsu C., Chern M., 2002 Minimizing the makespan in a single machine scheduling problem with a flexible maintenance, Journal of the Chinese institute of industrial engineers, 19(1), 63-66

Yang S., Ma Y., Xu D., Yang J. 2011, Minimizing total completion time on a single machine with a flexible maintenance activity, Computers and operations research, 38(4), 755-770

Yu A. J., Seif J. 2016. Minimizing tardiness and maintenance costs in flow shop scheduling by a lower-boud-based GA. Computers and Industrial engineering, 97, 26-40

Zammori F., Braglia M., Castellano D. 2014. Harmony search algorithm for single-machine scheduling problem with planned maintenance, Computers and industrial engineering, 76, 333-346

Zhang Q., Sundaramoorthy A., Grossmann I. E., Pinto J. M., 2016 A discrete-time scheduling model for continuous power-intensive process networks with various power contracts, Computers and chemical engineering, 84, 382-393

Nomenclature

Indices

i, i', i'' : tasks

j, j' : processing units or storage tanks

n, n', n'' : event points

s, s' : states

Sets

I: tasks

$\mathbf{I}_j$: tasks that can be processed in unit j

$\mathbf{I}_s^p$: production tasks that process state s

$\mathbf{I}_s^c$: consumption tasks that process state s

J: processing units or storage tanks

$\mathbf{J}^p$: processing units

$\mathbf{J}_s$: units that produce/consume state s

$\mathbf{J}^{st}$: storage tanks

$\mathbf{J}_1^m$: processing units with planned maintenance at the end of time horizon

$\mathbf{J}_2^m$: processing units with planned maintenance at the beginning of time horizon

$\mathbf{J}_3^m$: processing units with planned maintenance at the middle of time horizon

N: total number of event points

$\mathbf{N}_1$: number of event points before the maintenance period

$\mathbf{N}_2$: number of event points after the maintenance period

S: states

$\mathbf{S}^{FIS}$: intermediate states with finite storage capacity

$\mathbf{S}^{FFIS}$: intermediate states with flexible finite intermediate storage policy

$\mathbf{S}^{NIS}$: intermediate states with no storage capacity

$\mathbf{S}^R$: raw material states

$\mathbf{S}^{IN}$: intermediate states

$\mathbf{S}^{P}$: final product states

$\mathbf{S}_j$: states that can be stored in storage unit j

Parameters

H: scheduling horizon (h)

P_s: price of state s (\$/ton)

D_s^{min}: minimum demand for state s (ton)

D_s^{max}: maximum demand for state s (ton)

L_i^{min}: minimum duration of task i (h)

L_i^{max}: maximum duration of task i (h)

M: big-M value

$R_{i,j}^{min}$: minimum processing rate of task i in unit j (ton/h)

$R_{i,j}^{max}$: maximum processing rate of task i in unit j (ton/h)

$R_{i,j}$: processing rate of task i in unit j for the case of fixed processing rate (ton/h)

$Rb_{i,j}^{min}$: minimum processing amount of task i in unit j

$RL_{i,j}^{min}$: minimum processing duration of task i in unit j

Δn: Maximum number of event points that a task i is allowed to span over

τ_j^{min}: minimum total clean-up time required in unit j (h)

τ_j: sequence independent clean-up time in unit j (h)

$\tau_{i,i',j}$: sequence dependent clean-up time between unit i' and i in unit j (h)

ST_s^0: initial amount of intermediate state $s \in \mathbf{S}^{in}$ in dedicated storage (ton)

ST_s^{max}: maximum storage capacity of state s (ton)

V_s^{max}: maximum storage capacity of processing unit j (ton)

$\rho_{i,s}$: proportion of state s produced or consumed by task i

Binary variables

$w_{i,j,n,n'}$: binary variable for assignment of task *i* in unit *j* at the beginning of event *n*

$u_{i,j,n}$: binary variable to denote whether a task *i* is active in a storage tank *j* at event point *n*

$zD_{j,j',n}$: binary variable to denote whether direct material transfer takes place between units *j* and *j′* at event point *n*

$zI_{j,j',n}$: binary variable to denote whether indirect material transfer takes place between units *j* and *j′* at event point *n*

$zz_{j,j',n}$: binary variable to denote whether material transfer takes place between units *j* and *j′* at event point *n*

Positive variables

$ST_{s,n}$: storage inventory of state s in dedicated storage at the end of event *n* (ton)

$b_{i,j,n}$: amount of material processed by task *i* in unit *j* at event *n* (ton)

$bs_{i,j,n}$: amount of materials stored in storage tank *j* at event point *n* (ton)

$bTi_{i,j,i',j',n}$: amount of materials, which produced by task *i* processed in unit *j*, were indirectly transferred to unit *j′* which consumes task *i′* at event point *n*

$bTd_{i,j,i',j',n}$: amount of materials, which produced by task *i* processed in unit *j*, were indirectly transferred to unit *j′* which consumes task *i′* at event point *n*

$bz_{j,j',n}$: amount of materials transferred from unit j to unit j' during event point n.

$L_{i,j,n}$: Processing duration of task *i* in unit *j* at event point *n*

MS: makespan

$T^{\mathrm{s}}_{j,n}$: start time of unit *j* at event *n* (h)

$T^{\mathrm{f}}_{j,n}$: finish time of unit *j* at event *n* (h)

$T^{\mathrm{s}}_{s,j,n}$: time that state s produced by unit *j* starts being available at event *n* (h)

$T^{\mathrm{f}}_{s,j,n}$: time that state s produced by unit *j* finishes being available at event *n* (h)

$T^{ms}_{i,n}$: time at which maintenance start for task *i* at event *n* (h)

$T_{i,n}^{mf}$: time at which maintenance finishes for task *i* at event *n* (h)

$ue_{j,n}$: 0-1 continuous variable to denote whether a transition from one state s to another state s' in storage tank *j* takes place at event point *n*

Chapter 6: Scheduling of multitasking multipurpose batch processes

6.1 Introduction

In Chapter 4, an efficient framework for scheduling of multipurpose batch processes was developed. This approach, even though it can significantly reduce the model size of the problem, it still cannot directly solve all multipurpose batch process scheduling problems. The main reason is that in the presented approach, a processing unit can only process one task at a time. Such an assumption, even though it holds in some cases, there are some types of process industry that contains units that can process multiple tasks simultaneously. For instance, scientific service facilities examine several samples by different customers for their chemical and physical properties. These samples, even though they belong to another task (i.e. different samples from different customers), they can be examined in the same processing unit, which contains multiple slots for sample examination.

Despite the great interest for scheduling of single-tasking multipurpose batch processes, scheduling of multitasking batch processes has not gathered the same attention. Only recently several mathematical models considered this scheduling problem. Such models use uniform (Patil et al. 2015), non-uniform (Lagzi et al. 2017b) discrete-time representation and the slot-based representation (Lagzi et al. 2017a) to develop the mathematical formulation. On the other hand, there is no model based on the unit-specific event-based time representation for this problem. Since unit-specific event-based time representation leads to the least possible number of event points, using such a formulation can potentially lead to smaller model sizes and increase efficiency.

In this chapter, two efficient mathematical models for scheduling of multitasking, multipurpose batch processes were developed based on unit-specific event-based approach. While the first model uses timing variables based on tasks, the second model uses timing variables based on units, similar to the proposed efficient framework of Chapter 4. The capabilities of both formulations are examined by solving several problems and comparing the solution quality and computational efficiency with those formulations for multitasking multipurpose batch processes.

Blank Page

6.2 Research contribution 4

Rakovitis, N., Zhang, N., Li, J. A novel unit-specific event-based formulation for short-term scheduling of multitasking processes in scientific service facilities, Computers and Chemical Engineering, 133(2)**,** (2020) doi: doi.org/10.1016/j.compchemeng.2019.106626.

Blank Page

A Novel Unit-Specific Event-Based Formulation for Short-Term Scheduling of Multitasking Processes in Scientific Service Facilities

Nikolaos Rakovitis, Nan Zhang, Jie Li[3]

Centre for Process Integration, School of Chemical Engineering and Analytical Science, The University of Manchester, Manchester, M13 9PL, United Kingdom

Abstract

Scientific service facilities examine a number of samples from different customers for several physical and chemical properties using processing units with large capacities. A processing unit can process a great number of samples simultaneously. The process in such scientific service facility can be treated as a multi-tasking multipurpose batch process. Despite the great interest in developing models for scheduling of process industry during the past three decades, scheduling of multi-tasking multipurpose batch processes in a scientific service facility has not been considered adequately. In this work, we develop three novel mathematical models using the well-established unit-specific event-based modelling approach. The computational results demonstrate that the proposed mathematical models are able to reduce the number of event points required, which leads to a significant reduction in the model size and computational time. One of the proposed models in which the timing variables are defined based on processing units is the most efficient in most cases especially when minimization of makespan is used as the objective, where at least one order of magnitude less computational time than all other models is required to generate the optimum solution compared to other existing models.

Keywords: Scheduling, multi-tasking, batch process, mixed-integer linear programming, unit-specific event-based approach

1 Introduction

Process industries always seek ways to maximize their productivity, minimize their operating cost, and achieve efficient inventory management to survive in a highly competitive market. Scheduling is one of the important managerial tools for such industries to better utilize materials and machines and as a result to increase their profit.

[3] To whom correspondence should be addressed. jie.li-2@manchester.ac.uk. Tel: +44 (0) 161 306 8622

However, most of the existing industries use heuristic-based or spreadsheet-based methods which are only limited to generate a feasible solution for simple processes. Therefore, both academic and industrial research focuses on methods that are able to generate optimal schedules in reasonable computational time. Mathematical programming especially mixed-integer programming approach has gained much attention since it can often generate optimal schedules not only for simple processes but also in complicated ones.

Batch processes are widely used in process industries such as chemicals, pharmaceuticals, food industry, scientific service facilities and iron and steel industry because of their flexibility to produce high valued products, especially if small production of each product is required. Furthermore, they are ideal in cases of seasonal orders by different customers. The batch process is usually classified into single or multi-stage multiproduct batch process and multipurpose batch process (Kopanos and Puigjaner, 2019). In these processes, usually at most one task is allowed to be processed in a processing unit at a time. Scheduling of these processes has received considerable attention in the past three decades (Floudas and Lin, 2004; Méndez *et al.*, 2006; Li *et al.*, 2010; Maravelias, 2012; Harjunkoski *et al.*, 2014). Discrete- and continuous-time modelling approaches have been proposed to develop a great number of mathematical models based on state-task network (Kondili *et al.*, 1993) and resource-task network (Pantelides, 1994). The discrete-time modelling approach divides the scheduling horizon into time intervals of known length where the start and end time of an activity must be exactly at the time interval points. As a result, a great number of time intervals are often required, which significantly increases the model size. The continuous-time modelling approach can be further divided into process-slot (Sundaramoorthy and Karimi, 2005), global event-based (Maravelias and Grossmann, 2003), unit-specific event-based (Ierapetritou and Floudas, 1998; Shaik and Floudas, 2009; Li and Floudas, 2010; Tang et al., 2012; Li et al., 2016), unit-slot (Sursarla et al., 2010; Li and Karimi, 2011) and sequence-based (Méndez and Cerdá, 2000; Hui *et al*., 2000; Méndez and Cerdá, 2003) modelling approaches. The continuous-time modelling approach divides the scheduling horizon into time intervals of unknown length, leading to less time points, batches, slots or event points required compared to the discrete-time modelling approach. The advantages of the unit-specific event-based modelling approach have been well established in the literature (Shaik et al, 2006; Shaik and Floudas, 2009; Li and Floudas, 2010), often requiring less number of event points. The details about these modelling approaches and mathematical models can be found in Floudas and Lin (2004), Méndez *et al.* (2006) and Harjunkoski *et al*. (2014).

In a scientific service facility, a number of samples from different customers are examined for a number of chemical or/and physical properties. In order to examine such properties, a scientific service facility uses a number of machines with each containing a number of slots. Since these machines contain many slots, it is possible to have samples from different customers that are processed at a time simultaneously in a machine. In other words, multiple tasks can be processed in a machine at a time in such scientific service facilities, which is different from the discussed single-tasking batch processes with at most one task being processed in a unit at a time. Each customer requires a different number of physical and chemical properties to be examined. Therefore, each sample group is examined in different processing units. In other words, different samples can follow different processing paths. The processes in scientific service facilities are considered as multi-tasking multipurpose batch process (Lagzi *et al.*, 2017a). A typical scientific service facility receives around 3000-5000 samples from 40-60 different customers every day (Lagzi *et al.*, 2017a).

Most mathematical models that have been developed for the batch process with at most one task being processed in a processing unit at a time cannot be directly applied to the multi-tasking multipurpose batch process in scientific service facilities. Few efforts have focused on optimal scheduling of such multi-tasking batch process in scientific service facilities. Patil *et al.* (2015) developed a discrete-time model for scheduling of multi-tasking batch processes in scientific service facilities. Lagzi *et al.* (2017a) used process-slot continuous-time modelling approach for the same scheduling problem. Lagzi *et al.* (2017b) developed a discrete-time formulation using non-uniform time grid based on the work of Velez and Maravelias (2013) and compared the performance with that of the discrete-time (Patil et al., 2015) and process-slot continuous-time (Lagzi et al., 2017a) formulations. By solving a number of examples, it was concluded that the discrete-time formulations using uniform and non-uniform time grids requires less computational time than process slot-based alternative, especially for large-scale problems. However, those two discrete-time formulations are possible to lead to suboptimum solutions in some cases, especially if a coarse discretization is used since a unit can only start examining a property exactly at time interval points. The non-uniform discrete-time model of Lagzi *et al.* (2017b) was extended to consider allocation of personnel to active machines (Santos *et al.*, 2018) and two conflicting objectives (Lee *et al*., 2019).

In this work, we use the unit-specific event-based modelling approach whose advantages have been well established in the literature (Shaik and Floudas, 2009; Li and Floudas, 2010) to develop efficient models for scheduling of multi-tasking batch processes in a scientific service facility. In this unit-specific event-based modelling approach, timing variables could be defined either based on tasks similar to the definition of Shaik and Floudas (2009) or based on units (Ierapetritou and Floudas 1998). In order to examine the capabilities of both timing variable representations, we develop three different unit-specific event-based mathematical models. While in the first two models we define a number of timing variables based on tasks in the process, in the third model we introduce a number of timing variables based on processing units in the process. The main difference between the first two models are the tightening constraints. The first two models could be considered as an extension of the model of Shaik and Floudas (2009) for allowing multiple tasks to take place in a unit simultaneously. The third model is completely different from all existing models. A number of examples are solved to illustrate the capability of the proposed three formulations and compared with the existing mathematical models in the literature. The computational results demonstrate that the proposed mathematical models are able to reduce the number of event points required, which leads to a much smaller model size compared to the existing models in the literature (Patil *et al.* 2015; Lagzi *et al*. 2017a; Lagzi *et al*. 2017b). The third model with the timing variables defined based on processing units is the most superior since it generates the optimum solution in significantly less computational time, especially when minimization of makespan is used as the objective, where at least one order of magnitude less computational time than all other models is required to generate the optimum solution compared to other existing models.

2 Problem Statement

Figure 1 illustrates a general multi-tasking batch process in a scientific service facility. The scientific service facility receives O (o = 1, 2, 3, ..., O) orders/sample groups from different customers that are required to be examined for a total of P (p = 1, 2, 3, …, P) properties using totally J (j = 1, 2, 3, …, J) machines (or processing units). Each order/sample group contains a number of samples. We assume that all samples in an order/sample group are examined for the same number of properties without loss of generality. This is because if an order/sample group contains samples that are examined for different properties, this order/sample group will be divided into different orders/sample groups. Each order/sample group has to be examined for a number of properties based on the customer request. The property examination sequence for an order/sample group is known a priori. However, each order/sample group could have different property examination sequence and thus follow a different processing path. A machine (or unit) can only examine one property. Each machine (or unit) is allowed to examine a number of samples from different orders/sample groups at the same time depending on its capacity. The examination time of a machine (or unit) j is known and denoted as α_j It only depends on the property that is required to be examined, not the batch size. If a machine (or unit) starts examining some samples, then a new sample can be processed only after the completion of all current samples. In other words, a machine (or unit) cannot be interrupted during the examination. The examination of an order/sample group for a property in a machine (or unit) is considered as a *task*. The examination of different properties for the same or different orders/sample groups is treated as different tasks. The examination of different orders/sample groups on the same machine is also treated as different tasks. There are in total I (i = 1, 2, …, I) tasks and each machine can process $\mathbf{I}_j$ tasks.

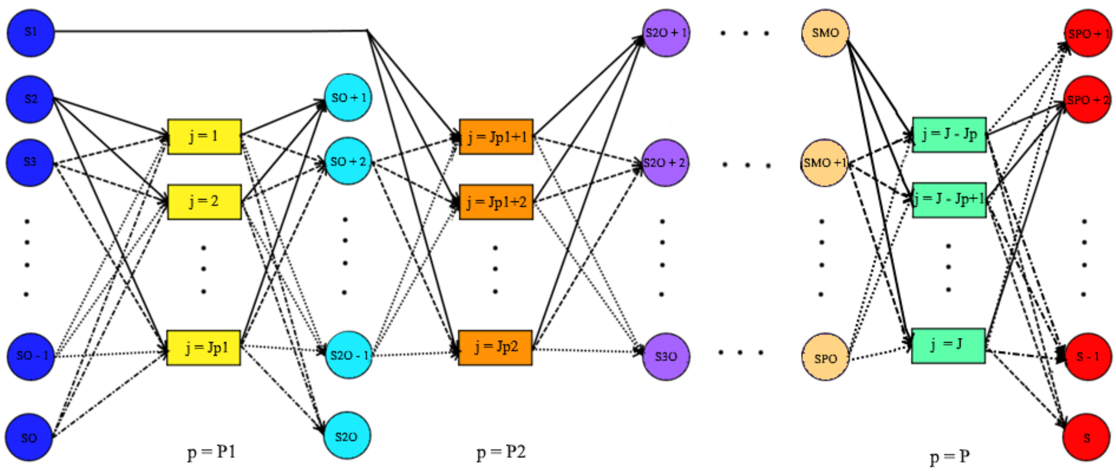

Figure 1 A general multi-tasking batch process in a scientific service facility

An order/sample group has three statuses depending on if its properties are examined. While an order/sample group that is received without any properties examined is called “raw material”, an order/sample group with some properties examined is called “intermediate state”. An order/sample group that has been completely examined is called “final product”. There are total S (s = 1, 2, 3, …, S) states. In Figure 1, states “S1, S2, ..., SO–1, SO” denote “raw material states”, “SO+1, SO+2, …, S2O–1, S2, S2O+1,…, SPO” are “intermediate states” and “SPO+1, SPO+2, …, S-1, S” are “final products”. The “raw material” is included in a set $\mathbf{S}^{R}$, the “intermediate state” is denoted as $\mathbf{S}^{IN}$, and the “final product” is denoted as $\mathbf{S}^{F}$. Each task can “consume” or “produce” at most a state. Tasks that produce a state s are denoted as $\mathbf{I}_s^P$ and tasks that consume a state s are denoted as $\mathbf{I}_s^C$. The portion of a state s that is used for task i is denoted as $\rho_{i,s}$. If a task i consumes a state s, then $\rho_{i,s} = -1$. If a task i produces a state s, then $\rho_{i,s} = 1$.

Each “intermediate state” has its own dedicated storage. There are several intermediate storage policies for each intermediate state including unlimited intermediate storage policy (UIS), finite intermediate storage policy (FIS) and no intermediate storage policy (NIS). There are also several possible wait policies for an intermediate state in a processing unit after processing including unlimited wait policy (UW), limited wait policy (LW) and zero wait policy (ZW). In a scientific service facility, a sample is allowed to stay in the processing time without any restriction after examination. Thus, UW policy is applied. We also assume unlimited intermediate storage policy for samples. With all of these, the scheduling problem can be stated as follows,

Given:

a) O orders/sample groups, the number of samples in each order/sample group, properties and their examination sequence for each order/sample group;
b) J machines (or processing units), minimum and maximum capacities, suitable properties and tasks, processing times;
c) The scheduling horizon H.

Determine:

a) Optimal processing schedule involving task allocations, start and end timings, sequences and batch sizes;
b) Inventory profiles.

Operating rules:

a) More than one tasks are allowed to be processed in a processing unit simultaneously;
b) Each machine (or unit) can examine only one property;
c) Batch splitting and mixing is allowed for each order/sample group.

Assuming:

a) All parameters are deterministic;
b) The processing time of a machine (or unit) j is fixed (denoted as α_j). It only

depends on the property that is required to be examined, not the batch size;

c) Unlimited feed materials are available;
d) Unlimited storage policy for all states;
e) Unlimited resources where required are available;
f) Unlimited wait policy for intermediate states.

The objective of the given problem is to maximize the number of samples examined, during a specified scheduling horizon (maximization of productivity) or to minimize the time required to examine all properties of a specific number of samples (minimization of makespan).

3 Mathematical Formulation

As discussed before, the capabilities of the unit-specific event-based modelling approach have been well established in the literature (Shaik and Floudas, 2009; Li and Floudas, 2010), which is used to develop three mathematical models for the given problem due to different ways for the definition of timing variables. Next, we present these three models in detail.

3.1 Models M1a and M1b

In the models **M1a and M1b**, the timing variables are defined based on tasks in the process, which is similar to the definition of most existing unit-specific event-based models (Shaik and Floudas, 2009; Li and Floudas, 2010). We also follow the approach of Shaik and Floudas (2009) that uses a parameter Δn to regulate the maximum allowable number of event points that a task is allowed to span over. It should be noted that the examination of the same order/sample group in different machines (or units) for the same properties is treated as *different tasks* in order to define timing variables based on tasks.

Allocation constraints

We define three-index binary variables $w_{i,n,n'}$ to denote if a task i is active from event point n to event point n' ($n \leq n'$), which is similar to those in the model of Shaik and Floudas (2009). If a task is allowed to span over multiple event points, it should start or end at only one event point. Constraint (1) guarantees that a task i have no more than one start or end in different event points.

$$\sum_{n-\Delta n \le n' \le n} \sum_{n \le n'' \le n+\Delta n} w_{i,n',n''} \le 1$$

$$\forall j,\ i \in \mathbf{I}_j,\ n,\ \Delta n > 0 \qquad (1)$$

Note that constraint (1) is valid for every task that can be processed in a unit j. Therefore, it allows multiple tasks to take place in the same unit simultaneously. This constraint (1) is different from those of Shaik and Floudas (2009).

We define new 0-1 continuous variables $y_{j,n,n'}$ to denote if a unit j is active from event point n to n'. Since multiple tasks are allowed to take place in the same unit simultaneously, constraints (2) and (3) are introduced to establish the relationship between $w_{i,n,n'}$ and $y_{j,n,n'}$. More specifically, constraint (2) states that when a task i is active from event points n to n' (i.e., $w_{i,n,n'} = 1$), a unit j that is able to process that task i must be also active ($y_{j,n,n'} = 1$). Furthermore, if none of the tasks that can be processed in a unit j is active, this unit j is forced to be inactive ($y_{j,n,n'} = 0$) as indicated in constraint (3).

$$y_{i,n,n'} \ge w_{i,n,n'} \qquad \forall j,\ i \in \mathbf{I}_j,\ n,\ n \le n' \le n+\Delta n \qquad (2)$$

$$y_{i,n,n'} \le \sum_{i \in \mathbf{I}_j} w_{i,n,n'}$$

$$\forall j,\ n,\ n \le n' \le n+\Delta n \qquad (3)$$

It should be noted that constraints (2) and (3) enforce $y_{j,n,n'}$ can take value 0 or 1 and therefore they are defined as 0-1 continuous variables.

Capacity constraints

As previously discussed, multiple sample groups can be examined in a processing unit simultaneously. We define $b_{i,n,n'}$ to denote the batch size of a sample group processed by a task i. The summation of all samples that are examined in the same unit j should be within its minimum unit capacity (B_j^{min}) and maximum unit capacity (B_j^{max}). Therefore, constraint (4) is introduced to avoid a capacity violation.

$$B_j^{\min} \cdot y_{j,n,n'} \le \sum_{i \in \mathbf{I}_j} b_{i,n,n'} \le B_j^{\max} \cdot y_{j,n,n'}$$

$$\forall j,\ n,\ n \le n' \le n+\Delta n \qquad (4)$$

Material balance constraints

The amount of a material state s stored at the beginning of an event point n should be equal to its storage amount at the beginning of the previous event point ($n - 1$) plus the amount of the state s produced at event point $(n-1)$ (i.e., $\rho_{i,s} > 0$), minus the amount of the state consumed at event point n (i.e., $\rho_{i,s} < 0$).

$$ST_{s,n} = ST_{s,n-1} + \sum_{i \in \mathbf{I}_S^P} \rho_{i,s} \sum_{n-1-\Delta n \le n' \le n-1} b_{i,n'n-1} + \sum_{i \in \mathbf{I}_S^C} \rho_{i,s} \sum_{n \le n' \le n+\Delta n} b_{i,n,n'} \qquad \forall s,\ n > 1 \qquad (5)$$

Notice that constraint (5) does not include the amount of a material state s stored at the beginning of the first event point. The amount of a material state s stored at the beginning of the first event point should be equal to the initial amount of state s minus the amount of the state consumed by tasks that start to process state s at the beginning of the first event point.

$$ST_{s,n} = ST0_s + \sum_{i \in \mathbf{I}_S^C} \rho_{i,s} \sum_{n \le n' \le n+\Delta n} b_{i,n,n'} \qquad \forall s,\ n = 1 \qquad (6)$$

The material balance constraints are similar to those of Shaik and Floudas (2009).

Duration constraints

The processing duration of task i is computed using constraint (7) if it is not allowed to span over multiple event points ($\Delta n = 0$).

$$T_{i,n}^{\mathrm{f}} \ge T_{i,n}^{\mathrm{s}} + \alpha_i \cdot w_{i,n,n} \qquad \forall i,\ n,\ \Delta n = 0 \qquad (7)$$

If a task i is allowed to span over multiple event points (i.e, $\Delta n > 0$), then constraints (8) and (9) are applied.

$$T_{i,n}^{\mathrm{f}} \ge T_{i,n}^{\mathrm{s}} \qquad \forall i,\ n,\ \Delta n > 0 \qquad (8)$$

$$T_{i,n'}^{\mathrm{f}} \ge T_{i,n}^{\mathrm{s}} + \alpha_i \cdot w_{i,n,n'} \qquad \forall i,\ n,\ n \le n' \le n+\Delta n,\ \Delta n > 0 \qquad (9)$$

Same task in the same unit

A task i at event point $(n + 1)$ must always start after it completes at the previous event point n as specified by constraint (10).

$$T_{i,n+1}^{\mathrm{s}} \ge T_{i,n}^{\mathrm{f}} \qquad \forall i,\ n < N \qquad (10)$$

If a task is allowed to span over multiple event points, then the start time of task i at event point $(n + 1)$ should be equal to the finish time of the same task at the previous event point n if task i is active at event point n but it continues being active at the next event point, as indicated in constraint (11).

$$T_{i,n+1}^{s} \leq T_{i,n}^{\mathrm{f}} + M \cdot \left[1 - \left(\sum_{n' \leq n} \sum_{n' \leq n'' \leq n' + \Delta n} w_{i,n',n''} - \sum_{n''} \sum_{n' < n, n'' \leq n' \leq n'' + \Delta n} w_{i,n'',n'}\right)\right] +$$
$$+ M \cdot \sum_{n - \Delta n \leq n' \leq n} w_{i,n',n}$$
$$\forall i, n, \Delta n > 0 \qquad (11)$$

Different tasks in the same unit

A task i at an event point (n + 1) must always start after any other task i' that can be processed at the same unit as this task completes at event point n.

$$T_{i,n+1}^{s} \geq T_{i',n}^{\mathrm{f}} \qquad \forall j, i \in \mathbf{I}_j, i' \in \mathbf{I}_j, i \neq i', n < N \qquad (12)$$

Different tasks in different units

Constraint (13) is introduced to define the sequence between tasks in different units that produce and consume the same state s. A consumption task i at event point (n + 1) must start after a production task i' related to the same state s completes at event point n, if the producing task finishes processing materials at event point n.

$$T_{i,n+1}^{s} \geq T_{i',n}^{\mathrm{f}} - M \cdot \left(1 - \sum_{n - \Delta n \leq n' \leq n} w_{i',n',n}\right)$$
$$\forall s \in \mathbf{S}^{IN}, j \neq j', i \in \left(\mathbf{I}_j \cap \mathbf{I}_s^{C}\right), i' \in \left(\mathbf{I}_{j'} \cap \mathbf{I}_s^{P}\right), i \neq i', n < N \qquad (13)$$

Tightening constraints

Shaik and Floudas (2009) introduced a number of tightening constraints in order to tight the relaxation of their MILP formulation. However, these constraints are proposed with the assumption that at most one task is allowed to be processed in a unit at a time. Therefore, these constraints cannot be used in this multi-tasking scheduling problem. In this work, we present two different tightening constraints. In the computational results, we will compare the performance of these two different tightening constraints. The first one is the modification of the tightening constraints from Shaik and Floudas (2009) as indicated in constraint (14).

$$\sum_{n} \sum_{n \leq n' \leq n + \Delta n} \max_{i \in \mathbf{I}_j}(\alpha_i) \cdot y_{i,n,n'} \leq H$$
$$\forall j \qquad (14)$$

In order to develop the second tightening constraints, we introduce new variables

$T_{j,n}^{\mathrm{s}}$ and $T_{j,n}^{\mathrm{f}}$ to denote the start and end time of a unit j at event point n. According to constraint (15), the finish time of a unit j at an event point n must be after the start time of this unit at the same event point plus the maximum processing time of the tasks that are available to be processed in the unit. Furthermore, according to constraint (16), the start time of unit j at event point $(n + 1)$ must always be after the finish time at the previous event point n.

$$T_{j,n}^{\mathrm{f}} \geq T_{j,n}^{\mathrm{s}} + \max_{i \in \mathbf{I}_j}(\alpha_i) \cdot y_{i,n,n'} \qquad \forall\, j, n, n \leq n' \leq n{+}\Delta n \qquad (15)$$

$$T_{j,n+1}^{\mathrm{s}} \geq T_{j,n}^{\mathrm{f}} \qquad \forall\, j, n < N \qquad (16)$$

Variable bounds

All timing variables should take values less than the scheduling horizon, as indicated in constraints (17)-(20).

$$T_{i,n}^{\mathrm{s}} \leq H \qquad \forall\, i, n \qquad (17)$$

$$T_{i,n}^{\mathrm{f}} \leq H \qquad \forall\, i, n \qquad (18)$$

$$T_{j,n}^{\mathrm{s}} \leq H \qquad \forall\, j, n \qquad (19)$$

$$T_{j,n}^{\mathrm{f}} \leq H \qquad \forall\, j, n \qquad (20)$$

The number of samples examined must always be less than the maximum capacity. Therefore, constraint (21) defines the upper limit for these variables.

$$b_{i,n,n'} \leq B_i^{max} \qquad \forall\, i, n, n \leq n' \leq n{+}\Delta n \qquad (21)$$

Objective function

Two objective functions are considered: maximization of productivity and minimization of makespan.

Maximization of productivity

Usually, it is better for a scientific service facility to complete the examination of all samples received during the specified scheduling horizon. However, this may not be achieved. Therefore, it is necessary to maximize the total number of samples that can be examined within the scheduling horizon, as indicated in constraint (22).

$$z = \sum_s p_s \sum_{i \in \mathbf{I}_S^P} \sum_n \sum_{n \leq n' \leq n+\Delta n} \rho_{i,s} \cdot b_{i,n,n'} \qquad (22)$$

where p_s is a weighted value, which takes the value of 1 for “intermediate products” and

5 for “final products”.

Minimization of makespan

Another objective is to minimize the time required to complete the examination of all samples received from customers. We define a variable *MS* to denote that the minimum time needed to examine all properties for all samples, which should exceed the finish time of all tasks at the last event point in the process.

$$T_{i,n}^{\mathrm{f}} \le MS \qquad \forall i,\, n = N \tag{23a}$$

$$\sum_{n} \sum_{n \le n' \le n+\Delta n} \max_{i \in \mathbf{I}_j}(\alpha_i) \cdot y_{i,n,n'} \le MS \qquad \forall j \tag{23b}$$

$$T_{j,n}^{\mathrm{f}} \le MS \qquad \forall i,\, n = N \tag{23c}$$

To achieve minimization of makespan, one additional constraint should be considered to ensure that all samples are examined.

$$ST_{s,n} + \sum_{i \in \mathbf{I}_S^P} \rho_{i,s} \sum_{n-\Delta n \le n' \le n} b_{i,n',n} \ge D_s \qquad \forall\, s \in \mathbf{S}^P,\, n = N \tag{24}$$

where D_s is the total amount of samples that have to be examined. Note that at the last event point all samples have to be examined for all properties. Therefore, we only consider “production states” in (24). Furthermore, the total number of samples received from customers should be equal to the number of samples required to be examined at the last event point.

We complete the model **M1a** which comprises eqs. 1-14, 17-18 and 21-22 for maximization of productivity and eqs. 1-13, 21, 23a, 23b and 24 for minimization of makespan. This model **M1a** uses the first tightening constraints (i.e., constraints 14 and 23b). Another model **M1b** using the second tightening constraints (i.e., constraints 15-16) are completed which comprises eqs. 1-13 and 15-22 for maximization of productivity and eqs. 1-13, 15-16, 21, 23a, 23c and 24 for minimization of makespan. It should be noted that although different tasks in the same unit may not start at the same time from the schedule generated using the models **M1a** and **M1b**, it is easy to revise the schedule to make sure that different tasks in the same unit start at the same time without any effect on the objective function.

3.2 Model M2

In this model, the timing variables are defined based on units. Since multiple tasks are allowed to take place at the same units simultaneously, we want to know their active status of each unit at a time. Thus, we define binary variables $w_{j,n,n'}$ to denote if a unit j is active from an event point n to another event point n' ($n' > n$).

Allocation constraints

Although a unit j is allowed to be active over multiple event points, it can start or end only at one event point, as indicated in constraint (25).

$$\sum_{n-\Delta n \le n' \le n} \sum_{n \le n'' \le n+\Delta n} w_{j,n',n''} \le 1$$

$$\forall j,\ n,\ \Delta n > 0 \tag{25}$$

Note that constraint (25) is valid for each unit j without involving any task. Thus, it does not restrict the number of tasks that are allowed to be processed in a unit j.

Capacity constraints

We define continuous variables $b_{i,j,n,n'}$ to denote the batch size that is processed by a task i in a unit j from event point n to event point n'. Recall that multiple tasks are allowed to be processed in a unit j simultaneously. The total batch size processed in a unit j should be within the minimum (B_j^{min}) and maximum (B_j^{max}) capacities of this unit j at a time, as indicated by constraint (26).

$$B_j^{\min} \cdot w_{j,n,n'} \le \sum_{i \in \mathbf{I}_j} b_{i,j,n,n'} \le B_j^{\max} \cdot w_{j,n,n'}$$

$$\forall j,\ n,\ n \le n' \le n+\Delta n \tag{26}$$

Material balance constraints

The amount of a material state s stored at the beginning of event point n should be equal to the amount of the state s stored at the beginning of event point (n – 1), plus the amount of this state s produced by tasks at the end of event point (n – 1) (i.e., $\rho_{i,s} > 0$), minus the amount of state s consumed by tasks at the beginning of event point n (i.e., $\rho_{i,s} < 0$).

$$ST_{s,n} = ST_{s,n-1} + \sum_j \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_S^P)} \rho_{i,s} \sum_{n-1-\Delta n \le n' \le n-1} b_{i,j,n',n-1} +$$

$$+ \sum_j \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_S^C)} \rho_{i,s} \sum_{n \le n' \le n-\Delta n} b_{i,j,n,n'}$$

$$\forall s,\ n > 1 \tag{27}$$

The amount of a material state s stored at the beginning of the first event point should be equal to the initial amount of state *s* minus the amount of the state consumed by tasks that start to process state *s* at the beginning of the first event point.

$$ST_{s,n} = ST0_s + \sum_j \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_S^C)} \rho_{i,s} \sum_{n \le n' \le n + \Delta n} b_{i,j,n,n'}$$

$$\forall s,\ n = 1 \qquad (28)$$

Duration constraints

The processing duration of a unit *j* is defined by constraint (29). The constraint assumes constant processing time (α_j) for all tasks, which is unit dependent only. This is true for property examination in a scientific service facility. Constraint (29) indicates that the end time of a unit *j* at event point n' must be greater than the start time of this unit *j* at event point *n* plus the constant processing time if unit *j* is active from event point *n* to event point n'.

$$T^{\mathrm{f}}_{j,n'} \ge T^{\mathrm{s}}_{j,n} + \alpha_j \cdot w_{j,n,n'} \qquad \forall j,\ n,\ n \le n' \le n + \Delta n \qquad (29)$$

Sequencing constraints

To sequence different tasks in different units, we define continuous variable $T_{s,n}$ to denote the time that state *s* is available at event point *n*. The end time of tasks that produce a state *s* from event point n' to event point *n* must be before the time that state *s* is available at event point *n* as denoted by constraint (30). We assume that "Raw material states" are available at the beginning of the scheduling horizon, while "final product states" are not "consumed" by any task. Therefore, they are not considered in constraint (30).

$$T_{s,n} \ge T^{\mathrm{f}}_{j,n} - M\left(1 - \sum_{n - \Delta n \le n' \le n} w_{j,n',n}\right)$$

$$\forall\ s \in \mathbf{S}^{IN},\ j,\ n,\ \sum_{i \in \mathbf{I}_j} \rho_{i,s} > 0 \qquad (30)$$

Furthermore, the start time of tasks that consume state *s* from event point (*n* + 1) to event point n', must be after the time that state *s* is available to be consumed at event point *n* as specified by constraint (31).

$$T_{s,n} \le T^{\mathrm{s}}_{j,n+1} + M\left(1 - \sum_{n+1 \le n' \le n+1+\Delta n} w_{j,n+1,n'}\right)$$

$$\forall\ s \in \mathbf{S}^{IN},\ j,\ n < N,\ \sum_{i \in \mathbf{I}_j} \rho_{i,s} < 0 \qquad (31)$$

Similar to constraint (30), constraint (31) is also only valid for "intermediate material states".

A unit j at event point (n + 1) must always start after any other process in unit j ends at event point n. This is because the property examination for a new order/sample groups should wait until the current examination completes.

$$T^{\mathrm{s}}_{j,n+1} \geq T^{\mathrm{f}}_{j,n} \qquad \forall\, j,\, n < N \qquad (32)$$

The start time that state s is available to be consumed at event point n must always be before the time that is available to be consumed at the next event point (n + 1), as denoted by constraint (33).

$$T_{s,n} \leq T_{s,n+1} \qquad \forall\, s \in \mathbf{S}^{IN},\, n < N \qquad (33)$$

Variable bounds

$$T^{\mathrm{s}}_{j,n} \leq H \qquad \forall\, j,\, n \qquad (34)$$

$$T^{\mathrm{f}}_{j,n} \leq H \qquad \forall\, j,\, n \qquad (35)$$

$$T_{s,n} \leq H \qquad \forall\, s,\, n \qquad (36)$$

$$b_{i,j,n,n'} \leq B^{max}_{i} \qquad \forall\, j,\, i \in \mathbf{I}_j,\, n \leq n' \leq n + \Delta n \qquad (37)$$

Objective function

Similar to the models **M1a** and **M1b**, two objective functions are considered for the model **M2** including maximization of profit and minimization of makespan.

Maximization of productivity

Given a specific scheduling horizon, objective function (38) is used to maximize the total number of samples that can be examined at the end of the scheduling horizon.

$$z = \sum_{s} p_s \sum_{j} \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}^{P}_{s})} \sum_{n} \sum_{n \leq n' \leq n + \Delta n} \rho_{i,s} \cdot b_{i,j,n,n'} \qquad (38)$$

Minimization of makespan

For minimization of makespan, the objective function (39a) is introduced.

$$T^{\mathrm{f}}_{j,n} \leq MS \qquad \forall\, j,\, n = N \qquad (39a)$$

$$T_{s,n} \leq MS \qquad \forall\, j,\, n = N \qquad (39b)$$

Similar to models **M1a** and **M1b**, one more constraint should be considered to ensure that all properties are examined in all samples in the case of minimization of makespan.

$$ST_{s,n} + \sum_{j} \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_S^P)} \rho_{i,s} \sum_{n-\Delta n \le n' \le n} b_{i,j,n',n} \ge D_s$$

$$\forall\ s \in \mathbf{S}^P,\ n = N \qquad (40)$$

We complete the model **M2** which comprises eqs. 25-38 for maximization of productivity and eqs. 25-33, 37, 39-40 for minimization of makespan.

4 Computational studies

We solve 32 examples to illustrate the capabilities of the proposed models. Example 1 is the illustrative example from Lagzi *et al.* (2017a) in which two groups of samples are examined for four properties in the facility having six machines. The necessary data are given in Tables 1-2. Examples 2-20 are generated randomly following discrete uniform distribution. Examples 2-6 have ten groups of samples with each containing from 50 to 80 samples. These sample groups have to be examined for 1 to 4 properties. Examples 7-8 have five groups of samples with 1-4 properties to be examined. One sample has to be examined for more than once for the same property since a property needs to be examined in two or more different conditions such as varying temperature or pressure. This could often happen in a scientific service facility, as illustrated in Lagzi *et al.* (2017a). Examples 9-19 involve 2-16 groups of samples with each containing from 50 to 80 samples having 3-8 properties to be examined. The necessary data for Examples 2-20 can be found in the **Supplementary Material**. A scheduling horizon of 480 min (i.e., 8 hours) is considered for Examples 2-19. Example 20 has 100 groups of samples with each containing 200 to 300 samples having a total 25049 samples to be examined. There are 25 properties that are required to be examined with each group having 8-9 properties. The scheduling horizon is 40 hours.

Table 1 Sample group data for Example 1

Sample group	Processing pathway	Number of samples
1	$P_1 - P_3 - P_4$	120
2	$P_1 - P_2 - P_3 - P_4$	100

Table 2 Data on processing units and properties for Example 1

Property	Unit	Unit capacity (samples) (Min – Max)	α_i (min)
1	1	0-140	50
2	2	0-70	30
	3	0-70	30
3	4	0-50	60
	5	0-50	60
4	6	0-120	195

Table 3 Possible processing paths for Examples 20-32

No. of path	Processing path
1	$P_1 - P_2 - P_3 - P_4 - P_{24} - P_6 - P_{13} - P_{16} - P_{11}$
2	$P_1 - P_2 - P_3 - P_4 - P_{24} - P_6 - P_5 - P_{11}$
3	$P_1 - P_2 - P_3 - P_4 - P_{24} - P_7 - P_{13} - P_{20} - P_{12}$
4	$P_1 - P_2 - P_3 - P_4 - P_{24} - P_6 - P_{18} - P_{11}$
5	$P_1 - P_2 - P_3 - P_4 - P_{24} - P_6 - P_{13} - P_{20} - P_{12}$
6	$P_1 - P_2 - P_3 - P_4 - P_{25} - P_6 - P_{13} - P_{17} - P_{10}$
7	$P_1 - P_2 - P_3 - P_4 - P_8 - P_{13} - P_{20} - P_{12}$
8	$P_1 - P_2 - P_3 - P_4 - P_{24} - P_{23} - P_{19} - P_{11}$
9	$P_1 - P_2 - P_3 - P_4 - P_{24} - P_{23} - P_{14} - P_{21} - P_{12}$
10	$P_1 - P_2 - P_3 - P_4 - P_{24} - P_{23} - P_{15} - P_9 - P_{22}$
11	$P_1 - P_2 - P_3 - P_4 - P_{24} - P_{23} - P_5 - P_{11}$

Examples 21-32 have the same number of processing units and properties to those of Lagzi *et al.* (2017b). The processing time, and maximum capacity of each processing unit in Examples 21-32 are also exactly same as those of Lagzi et al. (2017b). Other data are generated randomly following the discrete uniform distribution since they are not provided in Lagzi et al. (2017b). While 5 sample groups are considered for Examples 21-25, 10 sample groups are involved in Examples 26-30. Each sample group in Examples 21-30 contains from 50 to 80 samples. Example 31 considers 100 sample groups with each group containing 200-300 samples. A total of 25245 samples have to be examined. Example 32 contains 100 sample groups with 250-350 samples for each group. For Example 32, a total of 30067 samples have to be examined. Two scheduling horizons including H = 480 min and H = 1440 min are investigated for Examples 21-30, whilst a scheduling horizon of 40 hours (H = 2400 min) was examined for Examples 31-32. All sample groups are able to be processed using 11 predefined processing paths with each having 1-10 machines as shown in Table 3 and Figure 2. It should be noted that Examples 20-32 represent large-scale actual scientific service facilities (Lagzi *et al.*, 2017b). The necessary data for Examples 21-32 can be found in the **Supplementary Material**. All examples vary with the number of sample groups, properties, machines, and scheduling horizon. All examples are solved in CPLEX 12/GAMS 24.6.1 on a desktop computer with Intel® Core™ i5-2500 3.3 GHz and 8 GB RAM running Windows 7. We set the maximum CPU time for all examples as 1 hour.

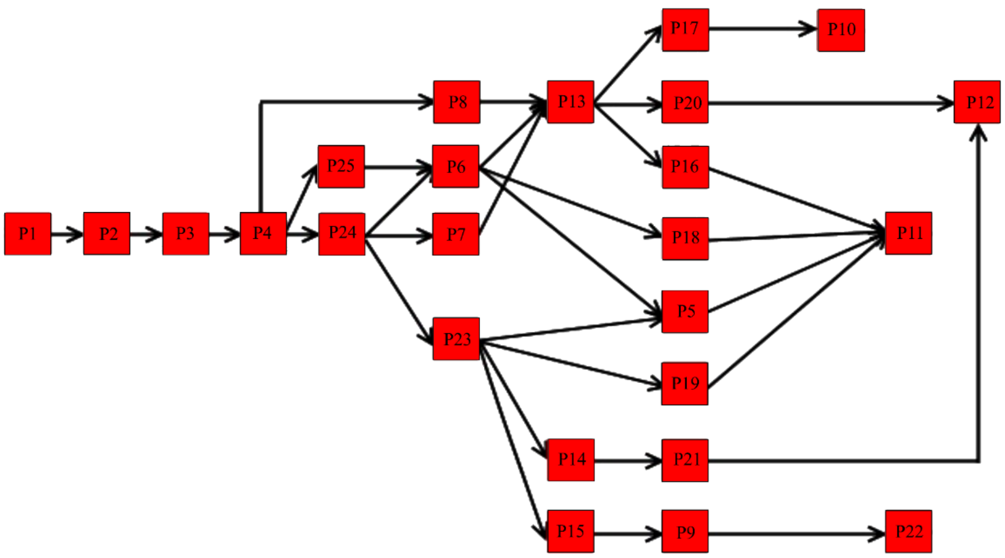

Figure 2 Possible processing routes for Examples 20-32

We also compare the performance of the proposed three models **M1a, M1b** and **M2** with those discrete-time models of Patil *et al.* (2015) and Lagzi *et al*. (2017b) and the process-slot continuous-time model of Lagzi *et al.* (2017a). While the discrete-time model of Patil *et al*. (2015) uses a uniform time grid for all units in which the scheduling horizon is divided into time intervals of equal length, the model of Lagzi *et al*. (2017b) uses non-uniform time grid in which the scheduling horizon is divided into time intervals of varying length. The length of each time interval in Patil *et al*. (2015) is equal to the greatest common factor of all tasks since all tasks have to start and end exactly at the time points. In the model of Lagzi *et al*. (2017b) the maximum time interval length was set to 60 minutes. For units with processing time less than 60 minutes the length of each time interval was equal to the processing time. On the other hand, for units with larger processing times, the time interval is set to the maximum length (i.e., 60 minutes). It should be mentioned that the models from Patil *et al.* (2015) and Lagzi *et al.* (2017a-b) did not consider makespan minimization. These models are extended for minimization of makespan in this paper which are presented in **Appendices A and B**.

Example 1

This example involves two groups of samples (group 1 and group 2). There are in total 4 properties (P1-P4) using in 6 units (J1-J6). The property examination sequences for two groups of samples are P1-P3-P4 for group 1 and P1-P2-P3-P4 for group 2. Property 1 is examined in unit J1. Property 2 is examined in units J2 and J3. Property 3 is examined in units J4 and J5. Property 4 is examined in unit J6. The examination of each property from a sample group is denoted as a *task*. For instance, we use task I1 to denote the examination of P1 for the sample group 1 in unit J1 and use task I2 to denote the examination of P1 for the sample group 2 in unit J1. There are in total 10 tasks (I1-I10). We use state S1 to denote the initial status of the sample group 1. We use states S1-S9 to denote the status of the two sample groups. The state-task network for this example is illustrated in Figure 3. The computational results are given in Table 4. From Table 4, it can be seen that all six models solve Example 1 in very small CPU time (< 1 s) for both objective functions. However, the proposed models **M1a, M1b** and **M2** lead to smaller model size than the existing models and hence they can be potentially superior, especially in the case of minimization of makespan, where they also lead to a tighter MILP relaxation. The optimal solutions are generated with $\Delta n = 0$ from the proposed models **M1a, M1b** and **M2.** The optimal schedule generated using the mathematical model **M2** with maximization of productivity is depicted in Figure 4. From the optimal schedule (Figure 4) it can be observed that two tasks are processed in the same unit simultaneously. For instance, unit J1 examines the property P1 of 70 samples from the sample group 1 and 70 samples from the sample group 2 simultaneously during 0 to 50 minutes.

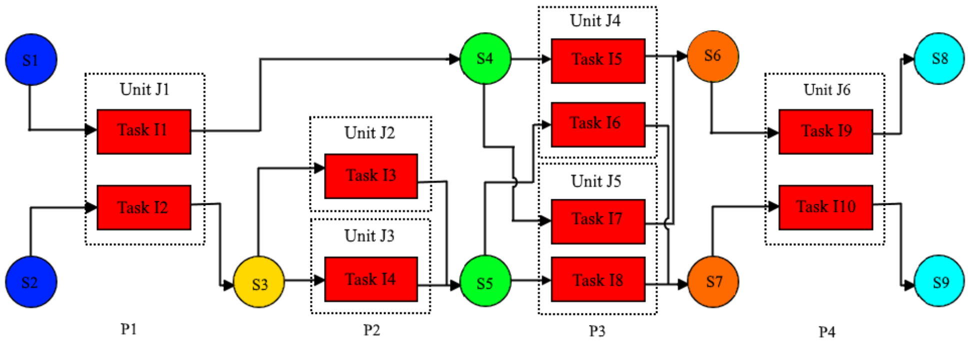

Figure 3 State-task network representation for Example 1

Another remarkable finding is that the discrete-time model of Patil *et al.* (2015) leads to a much tighter MILP relaxation than all other models when the objective is to maximize productivity. More specifically, it is interesting that the solution from the relaxed MILP is exactly identical to the optimal solution. Even though the discrete-time model of Patil *et al.* (2015) leads to large model size, it requires similar computational time than the rest. On the contrary, for minimization of makespan the discrete-time formulation of Patil *et al*. (2015) leads to a much worse relaxation than other models. By comparing the uniform discrete-time model of Patil *et al.* (2015) with that non-uniform discrete-time model of Lagzi *et al.* (2017b), it can be concluded that the use of a non-uniform discretization can lead to smaller model size. More specifically, if maximization of productivity is used as objective, the model of Lagzi *et al.* (2017b) requires approximately half discrete variables than the model of Patil *et al.* (2015) (715 vs 1394). Therefore, Lagzi *et al.* (2017b) formulation can potentially be more efficient in terms of computational time than the discrete-time model of Patil *et al.* (2015). However, since a coarser discretization is used, it can lead to suboptimum solutions. For instance, if minimization of makespan is used as objective, the model of Lagzi *et al.* (2017b) leads to 9.9% worse solution than other models (555 min vs 500 min).

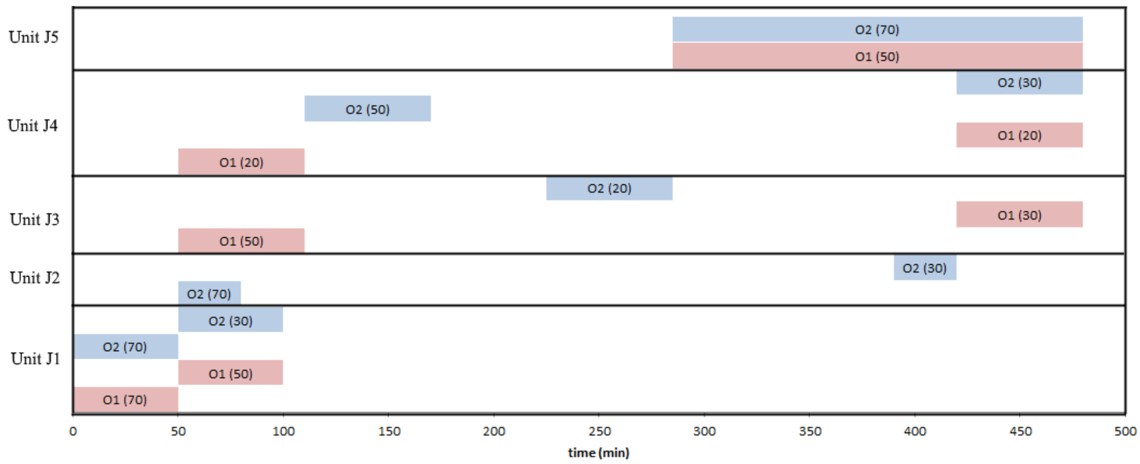

Figure 4 Optimal schedule for Example 1 using the model M2 with maximization of productivity

From Table 4 it can also be seen that all continuous-time formulations (i.e., Lagzi et al. 2017a, **M1a**, **M1b** and **M2**) require the same number of event points or slots for both objective functions. Although the models **M1a** and **M1b** lead to the same MILP relaxation, the model **M1b** has more continuous variables and constraints than **M1a** due to the introduction of additional variables $T^{s}_{j,n}$ and $T^{f}_{j,n}$ with additional related constraints (e.g., constraints 15 and 16). As a result, the model **M1a** leads to slightly smaller CPU time than the model **M1b**.

Table 4 Computational results for Example 1

Objective	Model	Event points/slots/time intervals	CPU time (s)	RMILP	MILP	Disc. Var.	Cont. Var.	Constr.
Max	Patil et al. (2015)	96	0.312	1140	1140	1394	1023	1835
productivity	Lagzi et al. (2017b)	81	0.125	1140	1140	715	721	1193
(H = 480 min)	Lagzi et al. (2017a)	4	0.187	1440	1140	320	360	1426
	M1a	4	0.078	1440	1140	80	141	301
	M1b	4	0.093	1440	1140	80	189	337
	M2	4	0.109	1440	1140	64	100	225
Min	Patil et al. (2015)	150	0.968	160.8	500	2258	1351	4126
makespan	Lagzi et al. (2017b)	43	0.218	68.42	555	419	379	833
	Lagzi et al. (2017a)	5	0.344	10.23	500	384	408	1694
	M1a	5	0.109	195.0	500	100	176	393
	M1b	5	0.141	195.0	500	100	236	441
	M2	5	0.093	357.5	500	80	126	300

Note Δn = 0 for this example

Other examples

The computational results for Examples 2-32 with the objective of maximization of productivity are given in Tables 5-8, whilst the results for Examples 2-19, 21-23 and 27 with the objective of minimization of makespan are given in Tables 9-11. The column "event points" in Tables 4-11 presents the number of event points required for **M1a**, **M1b**, and **M2**, the number of slots required for the model of Lagzi et al. (2017a) and the number of time intervals required for the models of Patil et al. (2015) and Lagzi et al. (2017b).

Maximization of productivity

Table 5 presents the computational results for Examples 2-8. From Table 5, it can be concluded that the model **M2** is the most superior since it requires less computational time than the models of Patil *et al.* (2015), Lagzi *et al.* (2017a), **M1a** and **M1b**. The main reason is that the model **M2** leads to a much smaller model size especially less number of discrete variables required. For example, it can generate the optimum solution for

Example 2 by using 89.1% less constraints than the model of Lagzi *et al.* (2017a) (i.e., 703 vs. 6424), 85.7% less constraints than the model of Patil *et al.* (2015) (i.e., 703 vs. 4907), 73.8% less constraints than the model **M1a** (i.e., 703 vs. 2682) and 74.2% less constraints than the model **M1b** (i.e., 703 vs. 2730). Furthermore, the model **M2** requires 82.1% less discrete variables than the model of Lagzi *et al.* (2017a) (i.e., 260 vs. 1456), 94.1% less discrete variables than the model of Patil et al. (2015) (i.e., 260 vs. 4412) and 43.5% less discrete variables than the models **M1a** and **M1b** (260 vs. 460). Even though the model **M2** also leads to a smaller model size than the model of Lagzi *et al.* (2017b), (51.9% less discrete variables, 62.8% less continuous variables and 12.8% less constraints), it requires more computational time. This is mainly because the model of Lagzi *et al.* (2017b) leads to a much tighter MILP relaxation. However, in most cases, the model of Lagzi *et al.* (2017b) is only limited to generate a suboptimum solution in contrast to the proposed model **M2** which generates the optimum solution for all these examples.

The uniform discrete-time formulation of Patil *et al.* (2015) performs better than the models **M1a**, **M1b** and the process-slot continuous-time model of Lagzi *et al.* (2017a). This is mainly due to the fact that the solution from the relaxed MILP of Patil et al. (2015) is exactly the same as the optimum solution for all these examples. However, the exceptionally high model size makes the model inferior to the model **M2**. By comparing the models **M1a, M1b** and **M2** and the process-slot model of Lagzi *et al.* (2017a), it can be concluded that proposed models **M1a**, **M1b** and **M2** generate optimum solutions in much less computational time, due to the fact that they are much tighter and hence lead to smaller model size. More importantly, the process-slot model of Lagzi *et al.* (2017a) requires more slots than the models **M1a**, **M1b** and **M2** in some examples. This is because all tasks in the process have to start or end at the same slot points. It can also be observed that the model **M2** requires more event points than models **M1a** and **M1b** for Example 5. The main possible reason is due to the constraints (30)-(31), which impose all states that can be processed in a unit j to be available after the unit finishes tasks or before the unit begins to process tasks once the unit is active regardless which task is processed in a unit j. Consequently, more event points than models **M1a** and **M1b** are required to generate the optimal solution for this Example 5. It is interesting that even though the model **M2** requires one more event point than these models, it still leads to a much smaller model size. As a result, it generates the optimum solution in significantly less amount of CPU time than **M1a** (0.328 s vs. 20.64 s) and **M1b** (0.328 s vs. 19.33 s).

Table 5 Computational results for Examples 2-8 with maximization of productivity

Example	Model	Event points	CPU time (s)	RMILP (cu)	MILP (cu)	Discr. Var.	Cont. Var.	Constr.
2	Patil et al. (2015)	96	0.312	3100	3100	4412	3937	4907
	Lagzi et al. (2017b)	18	0.078	2500	2500	541	698	806
	Lagzi et al. (2017a)	6	1755	4065	3100	1456	2044	6424
	M1a	5	4.54	3377	3100	460	696	2682
	M1b	5	4.21	3377	3100	460	756	2730
	M2	5	0.125	3377	3100	260	290	703
3	Patil et al. (2015)	96	0.826	3147	3147	5150	4417	5387
	Lagzi et al. (2017b)	18	0.046	2547	2547	602	698	806
	Lagzi et al. (2017a)	8	3600[a]	4822	3047	2196	2709	9402
	M1a	7	95.4	3424	3147	770	1135	5030
	M1b	7	87.9	3424	3147	770	1219	5102
	M2	7	0.218	3424	3147	427	623	1211
4	Patil et al. (2015)	96	0.998	3481	3481	4914	4321	5291
	Lagzi et al. (2017b)	18	0.062	2881	2881	569	766	874
	Lagzi et al. (2017a)	9	3600[b]	4881	3481	2320	2980	10045
	M1a	8	324.9	3758	3481	832	1241	5337
	M1b	8	299.1	3758	3481	832	1337	5421
	M2	8	0.312	3758	3481	464	702	1367
5	Patil et al. (2015)	96	0.531	3219	3219	4301	3841	4811
	Lagzi et al. (2017b)	18	0.062	2619	2619	501	681	789
	Lagzi et al. (2017a)	7	3600[c]	4679	3219	1632	2328	7245
	M1a	6	20.6	3496	3219	540	817	3074
	M1b	6	19.3	3496	3219	540	889	3134
	M2	7	0.328	3496	3219	357	545	1013
6	Patil et al. (2015)	96	1.98	2971	2971	4691	4225	5195
	Lagzi et al. (2017b)	18	0.140	2621	2621	520	749	857
	Lagzi et al. (2017a)	8	3600[d]	5156	2971	2016	2664	8782
	M1a	7	3600[e]	3498	2971	693	1058	4341
	M1b	7	3600[f]	3498	2971	693	1142	4413
	M2	7	0.546	3498	2971	385	597	1133
7	Patil et al. (2015)	96	0.296	1779	1779	3191	2401	3371
	Lagzi et al. (2017b)	18	0.094	1779	1779	377	426	534
	Lagzi et al. (2017a)	5	468.5	2149	1779	900	1008	3796
	M1a	5	3.12	2149	1779	310	466	1477
	M1b	5	2.91	2149	1779	310	526	1525
	M2	5	0.140	2149	1779	185	266	498
8	Patil et al. (2015)	96	0.265	2110	2110	2901	2401	3371
	Lagzi et al. (2017b)	18	0.031	1510	1510	340	426	534
	Lagzi et al. (2017a)	7	199.7	2470	2110	1104	1320	4754
	M1a	6	0.936	2210	2110	336	595	2015
	M1b	6	0.952	2210	2110	336	523	1613
	M2	6	0.250	2210	2110	204	323	588

[a]Relative gap 36.8%. [b]Relative gap 28.7%. [c]Relative gap 30.1%. [d]Relative gap 42.4%. [e]Relative gap 7.27%. [f]Relative gap 7.27%. Note $\Delta n = 0$ for all examples.

Table 6 Computational results for Examples 9-20 with maximization of productivity

Example	Model	Event points	CPU time (s)	RMILP (cu)	MILP (cu)	Disc. Var.	Cont. Var.	Constr.
9	Patil et al. (2015)	480	1.46	2339	2339	10370	8641	20185
	Lagzi et al. (2017b)	30	0.047	1986	1986	353	523	1214
	Lagzi et al. (2017a)	5	0.078	2339	2339	576	774	2562
	M1a	5	0.094	2339	2339	180	301	731
	M1b	5	0.063	2339	2339	180	361	779
	M2	5	0.125	2339	2339	120	203	334
10	Patil et al. (2015)	480	0.749	1175	1175	8659	8161	12971
	Lagzi et al. (2017b)	9	0.063	775	775	160	137	217
	Lagzi et al. (2017a)	5	65.95	1783	1175	552	768	2480
	M1a	3	0.063	1175	1175	102	172	381
	M1b	3	0.016	1175	1175	102	208	405
	M2	3	0.110	1175	1175	69	112	166
11	Patil et al. (2015)	480	1.22	958	958	12847	10588	15371
	Lagzi et al. (2017b)	10	0.015	858	858	243	199	281
	Lagzi et al. (2017a)	4	897.7	1308	958	660	690	2740
	M1a	4	167.3	1284	958	216	329	944
	M1b	4	157.9	1284	958	216	377	980
	M2	4	0.094	1284	958	132	188	333
12	Patil et al. (2015)	480	0.733	868	868	3781	3841	6727
	Lagzi et al. (2017b)	21	0.093	773	773	102	161	235
	Lagzi et al. (2017a)	4	0.109	868	868	180	190	785
	M1a	4	0.062	868	868	48	93	180
	M1b	4	0.031	868	868	48	117	198
	M2	4	0.109	868	868	36	75	108
13	Patil et al. (2015)	480	0.561	797	797	3489	3361	6247
	Lagzi et al. (2017b)	25	0.125	557	557	109	169	249
	Lagzi et al. (2017a)	4	0.047	797	797	160	185	717
	M1a	4	0.078	797	797	40	81	152
	M1b	4	0.031	797	797	40	105	170
	M2	4	0.063	797	797	32	68	96

14	Patil et al. (2015)	480	0.577	363	363	4828	3841	7208
	Lagzi et al. (2017b)	10	0.062	283	283	86	73	131
	Lagzi et al. (2017a)	7	1981	603	363	384	384	1667
	M1a	7	6.66	603	363	112	197	429
	M1b	7	6.26	603	363	112	253	477
	M2	7	0.328	603	363	84	149	251
15	Patil et al. (2015)	480	2.89	1279	1254	6264	6721	10569
	Lagzi et al. (2017b)	25	0.046	666	666	212	337	455
	Lagzi et al. (2017a)	8	162.0	1404	1254	540	585	2341
	M1a	7	5.01	1404	1254	154	281	624
	M1b	7	4.60	1404	1254	154	337	672
	M2	7	0.234	1404	1254	105	221	348
16	Patil et al. (2015)	480	1.17	1340	1340	7842	6241	10570
	Lagzi et al. (2017b)	45	0.047	1040	1040	501	573	817
	Lagzi et al. (2017a)	5	0.764	1340	1340	432	462	1858
	M1a	4	0.093	1340	1340	104	177	395
	M1b	4	0.031	1340	1340	104	217	425
	M2	5	0.218	1340	1340	90	156	254
17	Patil et al. (2015)	480	6.40	1275	1275	9260	10081	14891
	Lagzi et al. (2017b)	27	0.109	1168	1168	303	547	699
	Lagzi et al. (2017a)	18	3600[a]	2370	1275	1672	1881	7257
	M1a	18	3600[b]	1687	1275	612	1081	2706
	M1b	18	3600[c]	1687	1275	612	1261	2876
	M2	18	103.0	1687	1275	396	848	1426
18	Patil et al. (2015)	480	217.6	2330	2330	32445	38401	45135
	Lagzi et al. (2017b)	27	0.063	1132	1132	1067	2081	2293
	Lagzi et al. (2017a)	20	3600[d]	3259	1194	6300	7350	26491
	M1a	17	3600[e]	2744	2294	2312	3792	16722
	M1b	17	3600[f]	2744	2300	2312	4030	16946
	M2	17	3600[g]	2744	2330	1275	2687	4715
19	Patil et al. (2015)	480	478.1	3752	3692	48448	60001	67697
	Lagzi et al. (2017b)	27	0.266	2340	2340	1253	3251	3447
	Lagzi et al. (2017a)	16	3600[h]	6856	2604	7956	8959	32950
	M1a	22	3600[i]	4575	3357	4796	7723	43695
	M1b	22	3600[j]	4575	3376	4796	8075	44031
	M2	21	3600[k]	4563	3692	2457	5142	9153

20	Patil et al. (2015)	-	-	-	-	-	-	-
	Lagzi et al. (2017b)	41	57.5	53100	53100	71574	109575	42361
	Lagzi et al. (2017a)	-	-	-	-	-	-	-
	M1a	-	-	-	-	-	-	-
	M1b	-	-	-	-	-	-	-
	M2	15	3600[l]	196745	69664	44865	73536	186189

[a]Relative gap 46.2%. [b]Relative gap 24.4%. [c]Relative gap 42.4%. [d]Relative gap 15.4%. [e]Relative gap 16.4%. [f]Relative gap 16.2%. [g]Relative gap 4.03%. [h]Relative gap 56.1%. [i]Relative gap 26.7%. [j]Relative gap 26.2%. [k]Relative gap 18.6%. [l] Relative gap 63.7%. Note $\Delta n = 0$ for all examples.

By examining the models **M1a** and **M1b**, both of them lead to the same MILP relaxation for all these examples. The model **M1a** leads to smaller number of continuous variables and constraints but the same number of discrete variables compared to the model **M1b**. For instance, for Example 4 the model **M1a** require 1086 continuous variables and 4667 constraints to generate the optimal solution, while the model **M1b** require 1170 and 4739 respectively. However, the model **M1a** requires slightly more computational time than the model **M1b** in some cases. For both models **M1a** and **M1b** the computational time required is within the same order of magnitude. For instance, the model **M1a** needs 84.69 s to generate the optimum solution for Example 4, whereas the model **M1b** requires 80.11 s.

Table 6 lists the computational results for Examples 9-20. As it is demonstrated, the model **M2** is the most superior among all six formulations for most examples (Examples 9-17). The optimal schedule for Example 17 is illustrated in Figure 5. From Figure 5, it can be again confirmed that samples from different customers can be processed simultaneously in the same unit. However, the model **M2** generate the optimum solution, but it fails to converge within 1 hour for more complex examples (e.g., Examples 18-19), whereas the uniform discrete-time formulation of Patil *et al.* (2015) generates the optimum solution for Examples 18-19 within 1 hour. Consequently, it seems that the tighter relaxation of the discrete-time model of Patil *et al*. (2015) makes it more efficient for more complex problems when productivity is maximized. The process-slot model of Lagzi *et al.* (2017a) and the models **M1a** and **M1b** require more computational time to generate the optimum solution than the discrete-time model of Patil *et al.* (2015) for most examples even though they lead to smaller model size. Among all models, the one of Lagzi *et al*. (2017a) seems to have the worse MILP relaxation and hence it performs worse than the other models for most examples. The model of Lagzi *et al.* (2017b) requires significantly less computational time than all other models for all examples especially for large and complex examples (Examples 18-20) with less than one second required to generate a solution for Examples 18-19 and less than one minute for Example 20. However, the solution quality is much worse compared to other models. Similar observations can be made for the models **M1a** and **M1b** as those from previous Examples 2-8. For instance, the proposed model **M2** is able to generate the solution of 69664 cu for Example 20, while the model of Lagzi *et al.* (2017b) generates a significantly worse solution of 53100 cu.

Table 7 Computational results of Examples 21-25 with maximization of productivity

Example	Model	Event points	CPU time (s)	RMILP (cu)	MILP (cu)	Discr. Var.	Cont. Var.	Constr.
21a	Patil et al. (2015)	480	7.27	666	666	71220	64375	65850
(H=	Lagzi et al. (2017b)	49	0.078	666	666	2923	2305	3369
480min)	Lagzi et al. (2017a)	3	0.202	846	666	3744	4920	16330
	M1a	3	0.047	846	666	966	1330	4325
	M1b	3	0.046	846	666	966	1768	4617
	M2	3	0.094	846	666	702	669	2284
21b	Patil et al. (2015)	1440	13.4	666	666	264570	141745	197370
(H=	Lagzi et al. (2017b)	145	0.109	666	666	9652	6913	10105
1440min)	Lagzi et al. (2017a)	3	1.65	846	666	3744	4920	16330
	M1a	3	0.078	846	666	966	1330	4325
	M1b	3	0.062	846	666	966	1768	4617
	M2	3	0.110	846	666	702	669	2284
22a	Patil et al. (2015)	480	1.54	682	682	63822	56862	58635
(H=	Lagzi et al. (2017b)	49	0.062	682	682	2879	2305	3341
480min)	Lagzi et al. (2017a)	3	0.218	862	682	3248	3964	13888
	M1a	3	0.140	862	682	876	1192	3862
	M1b	3	0.078	862	682	876	2337	5806
	M2	3	0.032	862	682	609	573	2039
22b	Patil et al. (2015)	1440	9.24	682	682	229373	132271	175755
(H=	Lagzi et al. (2017b)	145	0.109	682	682	9331	6913	10021
1440min)	Lagzi et al. (2017a)	3	1.84	862	682	3248	3964	13888
	M1a	3	0.078	862	682	876	1192	3862
	M1b	3	0.063	862	682	876	1534	4090
	M2	3	0.094	862	682	609	573	2039
23a	Patil et al. (2015)	480	1.98	662	662	59587	50034	55268
(H=	Lagzi et al. (2017b)	49	0.032	662	662	2572	2305	13966
480min)	Lagzi et al. (2017a)	3	0.187	842	662	2880	3508	12310
	M1a	3	0.140	842	662	780	1073	3365
	M1b	3	0.047	842	662	780	1375	3565
	M2	3	0.062	842	662	540	531	1828

23b	Patil et al. (2015)	1440	10.1	662	662	212718	115783	165668
(H=	Lagzi et al. (2017b)	145	0.188	662	662	8357	6913	9853
1440min)	Lagzi et al. (2017a)	3	2.68	842	662	2880	3508	12310
	M1a	3	0.078	842	662	780	1073	3365
	M1b	3	0.062	842	662	780	1375	3565
	M2	3	0.047	842	662	540	531	1828
24a	Patil et al. (2015)	480	1.95	650	650	28860	93267	60559
(H=	Lagzi et al. (2017b)	49	0.062	650	650	2882	2305	3373
480min)	Lagzi et al. (2017a)	3	0.187	830	650	3296	4132	14194
	M1a	3	0.109	830	650	876	1201	3977
	M1b	3	0.031	830	650	876	1561	4217
	M2	3	0.047	830	650	618	591	2054
24b	Patil et al. (2015)	1440	9.89	650	650	230662	135305	181519
(H=	Lagzi et al. (2017b)	145	0.172	650	650	9355	6913	10117
1440min)	Lagzi et al. (2017a)	3	2.64	830	650	3296	4132	14194
	M1a	3	0.062	830	650	876	1201	3977
	M1b	3	0.063	830	650	876	1561	4217
	M2	3	0.031	830	650	618	591	2054
25a	Patil et al. (2015)	480	1.81	688	688	62470	59656	57191
(H=	Lagzi et al. (2017b)	49	0.078	688	688	2837	2353	15288
480min)	Lagzi et al. (2017a)	3	0.125	868	688	3280	3816	13852
	M1a	3	0.047	868	688	906	1216	4021
	M1b	3	0.046	868	688	906	1540	4237
	M2	3	0.062	868	688	615	560	2083
25b	Patil et al. (2015)	1440	35.5	688	688	228641	137325	171431
(H=	Lagzi et al. (2017b)	145	0.062	688	688	9291	6892	22318
1440min)	Lagzi et al. (2017a)	3	15.5	868	688	3280	3816	13852
	M1a	3	0.078	868	688	906	1216	4021
	M1b	3	0.140	868	688	906	1540	4237
	M2	3	0.078	868	688	615	560	2083

Note $\Delta n = 0$ for all examples.

Table 8 Computational results for Examples 26-32 with maximization of productivity

Example	Model	Event points	CPU time (s)	RMILP (cu)	MILP (cu)	Disc. Var.	Cont. Var.	Constr.
26a	Patil et al. (2015)	480	6.08	1280	1280	95756	118676	87448
(H=	Lagzi et al. (2017b)	49	0.14	1280	1280	4145	4561	5701
480min)	Lagzi et al. (2017a)	5	60.1	2288	1280	8424	12042	36275
	M1a	5	0.344	2173	1280	2830	3646	16649
	M1b	5	0.296	2173	1280	2830	4326	17193
	M2	5	0.156	2173	1280	1755	1496	7126
26b	Patil et al. (2015)	1440	18.1	1280	1280	360876	281716	262168
(H=	Lagzi et al. (2017b)	145	0.171	1280	1280	7669	13681	16198
1440min)	Lagzi et al. (2017a)	5	3600[a]	2288	1280	8424	12042	36275
	M1a	5	10.5	2288	1280	2830	3646	16649
	M1b	5	9.24	2288	1280	2830	4326	17193
	M2	5	0.188	2288	1280	1755	1496	7126

27a	Patil et al. (2015)	480	8.83	1284	1284	111175	111435	90816
(H=	Lagzi et al. (2017b)	49	0.141	1284	1284	5012	4513	5637
480min)	Lagzi et al. (2017a)	5	96.6	2244	1284	8856	13392	38669
	M1a	5	0.483	2140	1284	2920	3776	18010
	M1b	5	0.421	2140	1284	2920	4546	18626
	M2	5	0.109	2140	1284	1845	1577	7388
27b	Patil et al. (2015)	1440	18.7	1284	1284	421995	245095	272256
(H=	Lagzi et al. (2017b)	145	0.172	1284	1284	16630	13537	16909
1440min)	Lagzi et al. (2017a)	5	3600[b]	2244	1284	8856	13392	38669
	M1a	5	4.76	2244	1284	2920	3776	18010
	M1b	5	4.38	2244	1284	2920	4546	18626
	M2	5	0.188	2244	1284	1845	1577	7388
28a	Patil et al. (2015)	480	2.53	1260	1260	103942	124920	86486
(H=	Lagzi et al. (2017b)	49	0.140	1260	1260	4868	4561	5605
480min)	Lagzi et al. (2017a)	5	158.3	2220	1260	9144	12084	38633
	M1a	5	0.421	2116	1260	3140	3951	19027
	M1b	5	0.358	2116	1260	3140	4621	19563
	M2	5	0.110	2116	1260	1905	1486	7768
28b	Patil et al. (2015)	1440	12.2	1260	1260	404323	281299	259286
(H=	Lagzi et al. (2017b)	145	0.234	1260	1260	16336	13681	16813
1440min)	Lagzi et al. (2017a)	5	3600[c]	2220	1260	9144	12084	38633
	M1a	5	15.4	2220	1260	3140	3951	19027
	M1b	5	14.6	2220	1260	3140	4621	19563
	M2	5	0.156	2220	1260	1905	1486	7768
29a	Patil et al. (2015)	480	7.97	1188	1188	97698	118659	86968
(H=	Lagzi et al. (2017b)	49	0.078	1188	1188	4413	4513	5653
480min)	Lagzi et al. (2017a)	5	670.0	2196	1188	8544	12072	36685
	M1a	5	0.374	2081	1188	2880	3691	16875
	M1b	5	0.265	2081	1188	2880	4371	17419
	M2	5	0.047	2081	1188	1780	1487	7223
29b	Patil et al. (2015)	1440	18.0	1188	1188	376692	271665	260728
(H=	Lagzi et al. (2017b)	145	0.172	1188	1188	14798	13537	16959
1440min)	Lagzi et al. (2017a)	5	3600[d]	2196	1188	8544	12072	36685
	M1a	5	6.33	2196	1188	2880	3691	16875
	M1b	5	5.88	2196	1188	2880	4371	17419
	M2	5	0.156	2196	1188	1780	1487	7223
30a	Patil et al. (2015)	480	4.76	1348	1348	110609	114886	94182
(H=	Lagzi et al. (2017b)	49	0.078	1348	1348	5082	4561	5813
480min)	Lagzi et al. (2017a)	5	737.0	2308	1348	8976	13836	39385
	M1a	5	0.452	2204	1348	2940	3816	17978
	M1b	5	0.359	2204	1348	2940	4616	18618
	M2	5	0.078	2204	1348	1870	1616	7465
30b	Patil et al. (2015)	1440	20.2	1348	1348	411010	264725	282342
(H=	Lagzi et al. (2017b)	145	0.250	1348	1348	16751	13681	17437
1440min)	Lagzi et al. (2017a)	5	3600[e]	2308	1348	8976	13836	39385
	M1a	5	5.48	2308	1348	2940	3816	17978
	M1b	5	5.15	2308	1348	2940	4616	18618
	M2	5	0.078	2308	1348	1870	1616	7465

31	Patil et al. (2015)	-	-	-	-	-	-	-
(H=	Lagzi et al. (2017b)	241	5.98	34805	35005	228859	228481	235461
2400min)	Lagzi et al. (2017a)	-	-	-	-	-	-	-
	M1a	-	-	-	-	-	-	-
	M1b	-	-	-	-	-	-	-
	M2	40	3600[f]	46516	36965	119360	78029	306653
32	Patil et al. (2015)	-	-	-	-	-	-	-
(H=	Lagzi et al. (2017b)	241	5.98	39827	39827	220892	228481	235461
2400min)	Lagzi et al. (2017a)	-	-	-	-	-	-	-
	M1a	-	-	-	-	-	-	-
	M1b	-	-	-	-	-	-	-
	M2	40	3600[g]	56727	49991	117360	78029	302753

[a]Relative gap 44.1%. [b]Relative gap 42.8%. [c]Relative gap 43.2% [d]Relative gap 45.9%. [e]Relative gap 41.6%. [f]Relative gap 25.4%. [g]Relative gap 8.1%. Note Δn = 0 for all examples.

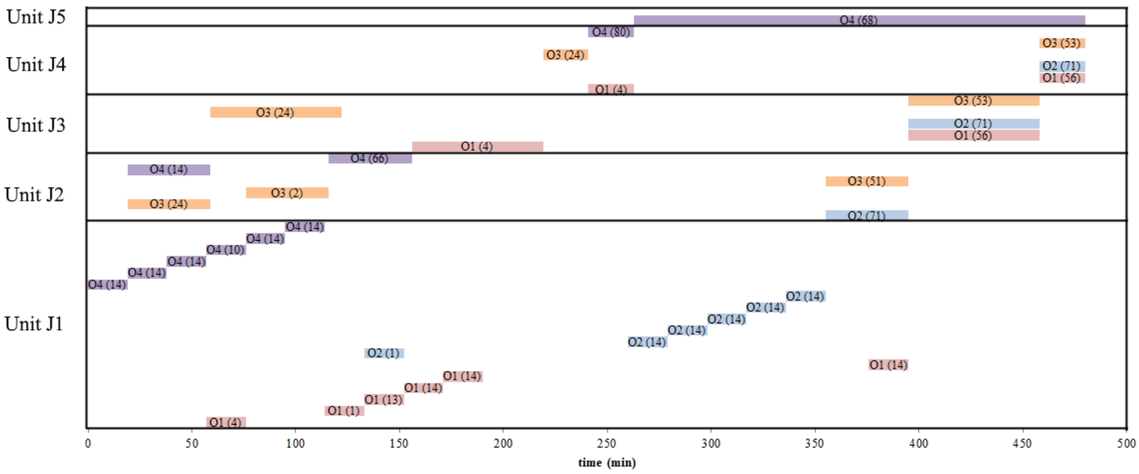

Figure 5 Optimal schedule for Example 17 using the model M2 with maximization of productivity

Tables 7-8 present the computational results for Examples 21-32. From Tables 7-8, it seems that the discrete-time model of Patil *et al.* (2015) as well as the models **M1a**, **M1b** and **M2** are more efficient compared to the process-slot model of Lagzi *et al.* (2017a). For instance, in Example 27b, the formulation of Lagzi *et al.* (2017a) could generate the optimal solution but could not converge after 1 hour, while all other formulations are able to generate the optimal solution in less than 1 minute. The developed model **M2** is more superior compared to the discrete-time model of Patil *et al.* (2015). For example, the model **M2** requires 99.0% less computational time than the discrete-time model of Patil *et al.* (2015) (0.188 s vs. 18.7 s) to generate the optimal solution for Example 27b. The model of Lagzi *et al.* (2017b) is also able to generate the optimum solution for Examples 21-30. Furthermore, the tight relaxation of the Lagzi *et al.* (2017b) model makes it as efficient as the mathematical models **M1a**, **M1b** and **M2** for this set of examples. However, the model of Lagzi *et al.* (2017b) fails to generate the optimal solution for large-scale examples. For instance, for Example 31 the model of Lagzi *et al.* (2017b) generates a solution of 35005 cu, while the proposed model **M2** generates a better solution of 36965 cu. Similarly, for Example 32 the proposed model **M2** is able to generate a significantly better solution of 49991 cu than the model of Lagzi *et al.* (2017b) which generates a solution of 39827 cu. On the other hand, both mathematical models **M1a** and **M1b** are more efficient than the discrete-time model of Patil *et al.* (2015). More specifically, both **M1a** and **M1b** models require at least one order of magnitude less computational time than the model of Patil *et al.* (2015) to generate the optimal solution for all these examples. This is because both models lead to significantly smaller model size. For instance, both models **M1a** and **M1b** require 98.6% less discrete variables (966 vs 71220) to generate the optimal solution for Example 21a.

Minimization of makespan

Table 9 gives the computational results for Examples 2-8. From Table 9, it seems that the process-slot model of Lagzi *et al.* (2017a) is not suitable for the problem of makespan minimization since it is unable to generate the optimum solution within 1 hour for most examples. The model of Lagzi *et al.* (2017b) is also not suitable since it fails to generate the optimum solution for all these examples. The discrete-time formulation of Patil *et al.* (2015) and the model **M2** are able to generate the optimum solution within 1 hour. Between these two models, the model **M2** is more efficient since it generates the optimum solution in at least two orders of magnitude less computational time. This is due to the fact that the model **M2** leads to both much smaller model size and a tighter MILP relaxation. Similarly, the models **M1a** and **M1b** require less computational time than the discrete-time formulation of Patil *et al.* (2015), due to their much smaller model size and tighter MILP relaxation. However, in some cases such as Examples 3 and 6, the proposed models fail to converge after 1 hour, while the discrete-time of Patil *et al.* (2015), requires significantly less computational time to converge (93.5 s and 17.3 s respectively).

Table 9 Computational results for Examples 2-8 with minimization of makespan

Example	Model	Event points	CPU time (s)	RMILP (min)	MILP (min)	Disc. Var.	Cont. Var.	Constr.
2	Patil et al. (2015)	250	43.62	16.34	1005	12420	10251	21810
	Lagzi et al. (2017b)	43	0.327	198.29	1215	1523	1723	2187
	Lagzi et al. (2017a)	7	3600[a]	95.46	1005	1664	2336	7364
	M1a	6	2.70	224.25	1005	552	835	3358
	M1b	6	2.64	224.25	1005	552	907	3418
	M2	6	0.249	843.38	1055	312	474	1283
3	Patil et al. (2015)	250	93.5	23.90	1065	14544	11501	25636
	Lagzi et al. (2017b)	43	0.312	351.82	1275	1696	1933	2396
	Lagzi et al. (2017a)	8	3600[b]	99.50	1065	2196	2709	9411
	M1a	8	3600[c]	125.13	1065	880	1297	5894
	M1b	8	3600[d]	125.13	1065	880	1393	5978
	M2	8	0.187	934.38	1065	488	717	2068
4	Patil et al. (2015)	250	147.5	18.43	1055	13846	11251	24626
	Lagzi et al. (2017b)	43	0.421	229.15	1275	1605	1891	2355
	Lagzi et al. (2017a)	8	3600[e]	79.61	1055	2088	2682	9039
	M1a	7	84.7	143.54	1055	728	1086	4667
	M1b	7	80.1	143.54	1055	728	1170	4739
	M2	8	1.17	845.00	1055	464	702	1995
5	Patil et al. (2015)	250	49.2	18.34	1035	12155	10001	21216
	Lagzi et al. (2017b)	43	0.343	209.43	1215	1415	1682	2145
	Lagzi et al. (2017a)	6	9.35	67.28	1035	1428	2037	6337
	M1a	6	1.01	230.75	1035	540	817	3128
	M1b	6	1.04	230.75	1035	540	889	3188
	M2	6	0.218	864.50	1035	306	463	1247
6	Patil et al. (2015)	250	17.3	69.61	1230	13315	11001	23816
	Lagzi et al. (2017b)	45	0.280	327.38	1455	1552	1938	2421
	Lagzi et al. (2017a)	8	3600[f]	152.92	1230	2016	2664	8791
	M1a	8	3600[g]	144.63	1230	792	1209	5088
	M1b	8	3600[h]	144.63	1230	792	1305	5172
	M2	8	0.280	1018.88	1230	440	687	1923
7	Patil et al. (2015)	600	103.0	5.84	570	20287	15001	58986
	Lagzi et al. (2017b)	23	0.078	209.44	665	521	551	793
	Lagzi et al. (2017a)	6	3600[i]	56.35	570	1050	1176	4444
	M1a	6	197.5	106.38	570	372	559	1847
	M1b	6	194.4	106.38	570	372	631	1907
	M2	6	0.655	360.75	570	222	323	878
8	Patil et al. (2015)	650	69.02	25.68	635	20199	16251	59412
	Lagzi et al. (2017b)	25	0.109	471.00	785	525	601	863
	Lagzi et al. (2017a)	7	784.1	25.92	635	1104	1320	4758
	M1a	6	5.210	134.13	635	336	523	1645
	M1b	6	4.633	134.13	635	336	595	1705
	M2	7	0.686	481.00	635	238	380	986

[a]Relative gap 34.1%%. [b]Relative gap 25.9% [c]Relative gap 22.5%. [d]Relative gap 25.9%. [e]Relative gap 45.6%. [f]Relative gap 32.9%. [g]Relative gap 10.8% [h]Relative gap 11.0%. [i]Relative gap 1.62%. Note $\Delta n = 0$ for all examples.

Table 10 presents the computational results for Examples 9-19. From these examples, it can be concluded that the model **M2** is superior compared to the other five models, even though it requires more number of event points compared to models **M1a** and **M1b** in some cases. It can also be observed that the model **M2** is able to generate solutions for more complex examples, (Examples 18-19). More specifically, the model **M2** can generate the optimum solution for Example 18 within a minute (i.e., 28.5 s). However, the models of Patil *et al.* (2015) and Lagzi *et al.* (2017a) are unable to generate a feasible solution and the models **M1a** and **M1b** fail to generate the optimum solution after 1 hour. The superiority of the model **M2** lays to the fact that it not only leads to smaller model size but also leads to tighter MILP relaxation for makespan minimization problems. The optimal schedule for Example 17 using model **M2** is illustrated in Fig. 4. Similarly, it can be confirmed that the proposed model allows more than one tasks to take place in a processing unit simultaneously (Fig. 4 and Table 12). From Table 10, it seems that both the models **M1a** and **M1b** perform better than the models Patil et al. (2015), Lagzi et al. (2017a) and Lagzi et al. (2017b). The main reason that the models **M1a** and **M1b** are more efficient is that they both lead to smaller model size and tighter MILP relaxation. For instance, in Example 15, the model **M1a** requires one order of magnitude less computational time than the discrete-time formulation of Patil *et al.* (2015) (1.451 s vs 13.74 s), two orders of magnitude less computational time than the process-slot model of Lagzi *et al.* (2017a) (1.451 s vs 119.7 s) and one order of magnitude less computational time than the model of Lagzi et al. (2017b) (1.451 s vs 13.7 s). However, both **M1a** and **M1b** fail to generate optimum solutions in more complex problems (Examples 17-19). By comparing the models **M1a** and **M1b**, both tightening constraints lead to the same MILP relaxation, indicating that they are very similar. Furthermore, the model **M1a** leads to slightly smaller model size. Despite that, for both models the computational time required is within the same order of magnitude.

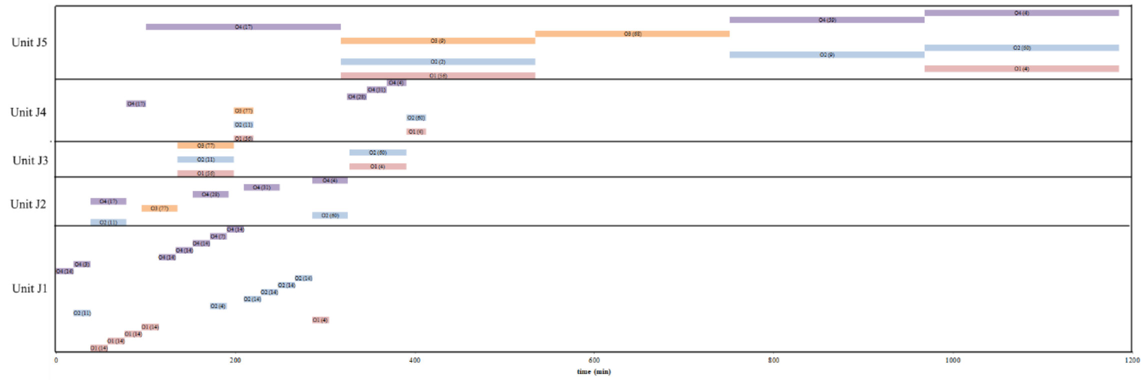

Figure 6 Optimal schedule for Example 17 using model M2 with minimization of makespan

Table 10 Computational results for Examples 9-19 with minimization of makespan

Example	Model	Event points	CPU time (s)	RMILP (min)	MILP (min)	Disc. Var.	Cont. Var.	Constr.
9	Patil et al. (2015)	200	4.57	3.97	176	3650	3601	13338
	Lagzi et al. (2017b)	60	0.203	52.19	212	918	1081	1685
	Lagzi et al. (2017a)	7	3600[a]	6.64	176	768	992	3394
	M1a	5	0.406	38.28	176	180	301	753
	M1b	5	0.421	38.28	176	180	361	801
	M2	6	0.265	132.96	176	144	246	560
10	Patil et al. (2015)	1300	317.2	34.89	1272	27519	22101	75296
	Lagzi et al. (2017b)	35	0.281	227.59	1519	773	596	1144
	Lagzi et al. (2017a)	7	23.2	432.31	1272	736	984	3284
	M1a	7	0.842	443.43	1272	238	400	990
	M1b	7	0.858	443.43	1272	238	484	1062
	M2	7	0.093	1146.24	1272	161	276	621
11	Patil et al. (2015)	1100	1984	5.01	1045	33334	24201	92636
	Lagzi et al. (2017b)	38	0.593	156.09	1342	1133	837	1396
	Lagzi et al. (2017a)	8	3600[b]	239.51	1056	1188	1170	4939
	M1a	8	2.50	225.76	1045	432	657	2071
	M1b	8	2.37	225.76	1045	432	753	2155
	M2	8	0.171	712.64	1045	264	392	1050
12	Patil et al. (2015)	500	1.79	2.49	465	3961	4001	11659
	Lagzi et al. (2017b)	24	0.047	99.70	551	116	185	315
	Lagzi et al. (2017a)	4	0.047	45.48	465	180	190	786
	M1a	4	0.078	123.70	465	48	93	187
	M1b	4	0.094	123.70	465	48	117	205
	M2	4	0.124	225.56	465	36	75	142
13	Patil et al. (2015)	300	0.484	3.91	289	2049	2101	6118
	Lagzi et al. (2017b)	21	0.015	109.99	299	91	142	244
	Lagzi et al. (2017a)	4	0.125	35.40	289	160	185	718
	M1a	4	0.093	68.08	289	40	81	158
	M1b	4	0.094	68.08	289	40	105	176
	M2	4	0.109	116.03	289	32	68	123

14	Patil et al. (2015)	750	69.6	6.26	703	8068	6001	21212
	Lagzi et al. (2017b)	16	0.109	299.86	759	162	121	279
	Lagzi et al. (2017a)	9	300.2	193.48	703	480	620	2190
	M1a	9	5.37	195.94	703	144	253	566
	M1b	9	5.38	195.94	703	144	325	630
	M2	9	0.078	367.81	703	108	193	420
15	Patil et al. (2015)	600	13.7	4.02	555	8064	6401	26642
	Lagzi et al. (2017b)	101	0.374	91.97	625	939	1401	2138
	Lagzi et al. (2017a)	8	119.7	88.09	555	540	585	2343
	M1a	7	1.45	137.39	555	154	281	637
	M1b	7	1.48	137.39	555	154	337	685
	M2	7	0.203	307.50	555	105	221	469
16	Patil et al. (2015)	500	2.12	6.92	466	8202	6501	22707
	Lagzi et al. (2017b)	51	0.062	253.66	520	583	651	1101
	Lagzi et al. (2017a)	6	8.13	0.00	466	504	539	2175
	M1a	4	0.062	159.19	466	104	177	410
	M1b	4	0.234	159.19	466	104	217	440
	M2	5	0.140	327.32	466	90	156	341
17	Patil et al. (2015)	1200	1447	8.18	1185	25100	25201	73824
	Lagzi et al. (2017b)	70	0.094	305.17	1297	876	1450	2071
	Lagzi et al. (2017a)	19	3600[c]	108.11	1185	1760	1980	7644
	M1a	19	3600[d]	255.29	1185	646	1141	2880
	M1b	19	3600[e]	255.29	1185	646	1331	3060
	M2	19	151.4	919.06	1185	418	896	2051
18	Patil et al. (2015)	-	-	-	-	-	-	-
	Lagzi et al. (2017b)	376	4.20	1228.30	1725	12692	30001	33492
	Lagzi et al. (2017a)	-	-	-	-	-	-	-
	M1a	52	3600[f]	167.75	1820	7072	11597	52711
	M1b	52	3600[g]	167.75	1966	7072	12325	53425
	M2	52	28.5	1562.00	1696	3900	8357	21628
19	Patil et al. (2015)	-	-	-	-	-	-	-
	Lagzi et al. (2017b)	365	27.28	3604.89	6095	22983	45502	50370
	Lagzi et al. (2017a)	-	-	-	-	-	-	-
	M1a	-	-	-	-	-	-	-
	M1b	-	-	-	-	-	-	-
	M2	59	3600[h]	3573.47	3722	6903	14642	38785

[a]Relative gap 5.24%. [b]Relative gap 25.9%. [c]Relative gap 60.7%. [d]Relative gap 43.3%. [e]Relative gap 39.4%. [f]Relative gap 89.4%. [g]Relative gap 89.6%. [h]Relative gap 1.98%. Note $\Delta n = 0$ for all examples.

Table 11 presents the computational results for Examples 21-23 and 27, which are highly complex. The computational results for Examples 20, 24-26 and 28-32 are not presented because none of the models could generate optimal solutions or even feasible solutions for these six examples within the predefined CPU time (i.e., 1 hr). From Table 11, it seems that the model **M2** is able to generate the best solution within 1 hour. The discrete-time models of Patil *et al.* (2015) and Lagzi *et al.* (2017b) are unable to generate a feasible solution within 1 hour. Therefore, it can be concluded the model **M2** is superior compared to the other four models. The models **M1a** and **M1b** are only able to generate feasible solutions for Examples 21-23. However, they both fail to converge after 1 hour. Similar observations can be done regarding the model size of **M1a** and **M1b**.

Table 11 Computational results for Examples 21-23 and 27 with minimization of makespan

Example	Model	Event points	CPU time (s)	RMILP (min)	MILP (min)	Disc. Var.	Cont. Var.	Constr.
21	Patil et al. (2015)	-	-	-	-	-	-	-
	Lagzi et al. (2017b)	-	-	-	-	-	-	-
	Lagzi et al. (2017a)	13	3600[a]	14.25	7431	13104	18340	59434
	M1a	13	3600[b]	169.00	5131	4186	5760	22360
	M1b	13	3600[c]	169.00	5131	4186	7658	24112
	M2	13	3600[d]	624.38	5131	3042	3039	13097
22	Patil et al. (2015)	-	-	-	-	-	-	-
	Lagzi et al. (2017b)	-	-	-	-	-	-	-
	Lagzi et al. (2017a)	87	3600[e]	15027.1	39846	71456	94248	320980
	M1a	87	3600[f]	16261.4	20930	25404	34540	138580
	M1b	87	3600[g]	16261.4	20930	25404	44458	148384
	M2	87	3600[h]	620.63	20880	17661	17793	82658
23	Patil et al. (2015)	-	-	-	-	-	-	-
	Lagzi et al. (2017b)	-	-	-	-	-	-	-
	Lagzi et al. (2017a)	14	3600[i]	56.5	6850	10800	14355	48340
	M1a	14	3600[j]	342.00	5131	3640	5013	18767
	M1b	14	3600[k]	342.00	5131	3640	6413	20067
	M2	14	3600[l]	620.63	5131	2520	2632	11472
27	Patil et al. (2015)	-	-	-	-	-	-	-
	Lagzi et al. (2017b)	-	-	-	-	-	-	-
	Lagzi et al. (2017a)	-	-	-	-	-	-	-
	M1a	-	-	-	-	-	-	-
	M1b	-	-	-	-	-	-	-
	M2	18	3600[m]	1203.8	5121	6642	5893	31678

[a]Relative gap 99.7%. [b]Relative gap 91.2%. [c]Relative gap 78.9%. [d]relative gap 59.5%. [e]Relative gap 58.9%. [f]Relative gap 12.9%. [g]Relative gap 5.27%. [h]Relative gap 21.9%. [i]Relative gap 49.4%. [j]Relative gap 22.5%. [k]Relative gap 49.4%. [l]Relative gap 21.8%. [m]Relative gap 57.0%. Note Δn = 0 for all examples.

5 Conclusions

In this paper, we developed three novel MILP mathematical formulations using the well-established unit-specific event-based modelling approach for scheduling of multi-tasking multipurpose batch processes in a scientific service facility. Multiple tasks were allowed to be processed simultaneously in the same units. While the timing variables were defined based on tasks of the process in the first two models (**M1a and M1b**), they were introduced based on processing units of the process in the third model (**M2**). Two different tightening constraints were proposed in the models **M1a** and **M1b** to improve their MILP relaxation. The computational results demonstrate that the model **M2** is the most efficient for most examples since it generates the optimum solution in significantly less amount of computational time than all other models. The proposed tightening constraints for the models **M1a** and **M1b** resulted in the same MILP relaxation for all examples. Although the model **M1a** has less number of constraints and continuous variables than the model **M1b**, it seems that their performance is almost the same. The future work is to employ rolling-horizon decomposition approach to solve all examples especially those large-scale complex problems that the best model **M2** fail to solve. Even though this work is focused on scientific service facilities, it can be also implemented in any multipurpose batch process industry which allows multiple tasks to take place simultaneously in a processing unit.

Table 12 Optimal results for Example 17 with minimization of makespan

Unit	Order/Sample group	Samples	Start time (min)	End time (min)
J1	O1	14	38	57
		14	57	76
		14	76	95
		14	95	114
		4	285	304
	O2	11	19	38
		4	171	190
		14	209	228
		14	228	247
		14	247	266
		14	266	285
	O4	14	0	19
		3	19	38
		14	114	133
		14	133	152
		14	152	171
		7	171	190
		14	190	209

J2	O2	11	38	78
		60	285	325
	O3	77	95	135
	O4	17	38	78
	O4	28	152	192
		31	209	249
		4	285	325
J3	O1	56	135	198
		4	327	390
	O2	11	135	198
		60	327	390
	O3	77	135	198
J4	O1	56	198	220
		4	390	412
	O2	11	198	220
		60	390	412
	O3	77	198	220
	O4	17	78	100
		28	324	346
		31	346	368
		4	368	390
J5	O1	56	317	534
		4	968	1185
	O2	2	317	534
		9	751	968
		60	968	1185
	O3	9	317	534
		68	534	751
	O4	17	100	317
		59	751	968
		4	968	1185

Acknowledgements

Nikolaos Rakovitis would like to acknowledge financial support from the postgraduate award by The University of Manchester.

Nomenclature

Sets

I: tasks

$\mathbf{I}_j$: units that can process task i

$\mathbf{I}_s^C$: tasks that consume state s

$\mathbf{I}_s^P$: tasks that produce state s

J: units

N: event points

P: processes

$\mathbf{P}_J$: units that are able to process process p

S: states

$\mathbf{S}^R$: raw material states

$\mathbf{S}^{IN}$: intermediate states

$\mathbf{S}^P$: product states

Indicies

i: tasks

j: units

s: states

n: event points

Parameters

α_i: processing time of task i

α_j: processing time of unit j

B_i^{max}: maximum amount of materials that can be processed at task i

B_j^{min}: minimum capacity of unit j

B_j^{max}: maximum capacity of unit j

D_s: total amount of samples that have to be examined

H: scheduling horizon

$\rho_{i,s}$: proportion of state s that is consumed/produced from task i

$ST0_s$: initial amount of available state s

B_i^{max}: maximum amount of materials that can be processed at task i

B_j^{min}: minimum capacity of unit j

B_j^{max}: maximum capacity of unit j

D_s: total amount of samples that have to be examined

H: scheduling horizon

$\rho_{i,s}$: proportion of state s that is consumed/produced from task i

$ST0_s$: initial amount of available state s

Δn: maximum number of event points that a task i is allowed to span

M: a large positive number

Binary variables

$w_{i,n,n'}$: 1 if task i is active from event point n to event point n'

$w_{j,n,n'}$: 1 if unit j is active from event point n to event point n'

Integer variables

$b_{i,n,n'}$: amount of materials that are processed in task i from event point n to event point n'

$b_{i,j,n,n'}$: amount of materials that are processed in task i which takes place at unit j from event point n to event point n'

Continuous variables

MS: makespan

$ST_{s,n}$: amount of state s that has to be stored at event point n

$T_{i,n}^{s}$: start time of task i at event point n

$T_{i,n}^{f}$: end time of task i at event point n

$T_{j,n}^{s}$: start time of unit j at event point n

$T_{j,n}^{f}$: end time of unit j at event point n

$y_{j,n,n'}$: 1 if unit j is active from event point n to event point n'

z: total profit

Appendix A Discrete-time mathematical model proposed by Patil et al. (2015)

Sets

I: tasks

$\mathbf{I}_p$: tasks that belong in process p

$\mathbf{I}_s^C$: tasks that consume state s

$\mathbf{I}_s^P$: tasks that produce state s

J: units

P: processes

$\mathbf{P}_J$: units that are able to process process p

S: states

T: time slots

Parameters

B_j^{min}: minimum capacity of unit j

B_j^{max}: maximum capacity of unit j

R_p: number of resources available for property p

$ST0_s$: initial amount of available state s

Tr_p: Duration of examination of property p

$\rho_{i,s}$: proportion of state s that is consumed/produced from task i at time slot t

Binary variables

$y_{j,t}$: binary variable which take the value 1 if unit j is active at time slot t

Integer variables

$b_{i,p,t}$: amount of materials that are processed in task i which belongs to process p at time slot t

Continuous variables

$ST_{s,t}$: amount of state s that has to be stored at time slot t

$$ST_{s,t} = ST_{s,t-1} + \sum_{i \in \mathbf{I}_S^P} \rho_{i,s} \sum_p b_{i,p,t-Tr_p} + \sum_{i \in \mathbf{I}_S^C} \rho_{i,s} \sum_p b_{i,p,t} \qquad \forall s,\ t > 2 \qquad \text{(A1)}$$

$$ST_{s,t} = ST0_s + \sum_{i \in \mathbf{I}_S^C} \rho_{i,s} \sum_p b_{i,p,t} \qquad \forall s,\ t = 2 \qquad \text{(A2)}$$

$$y_{j,t} B_j^{\min} \le \sum_{i \in \mathbf{I}_p} b_{i,p,t} \le y_{j,t} B_j^{\max} \qquad \forall p,\ j \in \mathbf{P}_j,\ t \qquad \text{(A3)}$$

$$\sum_{j \in \mathbf{P}_j} \sum_{t-Tr_p+1 \le t' \le t} y_{j,t'} \le R_p \qquad \forall p,\ t \qquad \text{(A4)}$$

Maximization of productivity

$$z = \sum_s p_s \sum_{i \in \mathbf{I}_S^P} \sum_p \sum_t \rho_{i,s} \cdot b_{i,p,t} \qquad \text{(A5)}$$

Minimization of makespan

$$\big((t-1) + Tr_p\big) \cdot y_{j,t} \le MS \qquad \forall p,\ j \in \mathbf{P}_j,\ t \qquad \text{(A6)}$$

$$ST_{s,t} + b_{i,p,t} \ge D_s \qquad \forall s,\ i,\ p,\ t = T \qquad \text{(A7)}$$

While the model of Patil et al. (2015) for maximization of productivity includes constraints A1-A5, the model of Patil et al. (2015) for makespan minimization consists of A1-A4 and A6-A7.

Appendix B Continuous-time mathematical model proposed by Lagzi et al. (2017a)

Sets

I: tasks

$\mathbf{I}_j$: units that can process task i

$\mathbf{I}_s^C$: tasks that consume state s

$\mathbf{I}_s^P$: tasks that produce state s

J: units

N: event points

P processes

$\mathbf{P}_J$: units that are able to process process p

S: states

Parameters

B_j^{min}: minimum capacity of unit j

B_j^{max}: maximum capacity of unit j

H: scheduling horizon

TA_i: earliest available time that task i is available

TM_j: earliest available time that unit j is available

α_i: initial amount of materials in task i

τ_j: processing time of unit j

Binary variables

$Y_{i,j,n}$: binary variable which take the value one if task i is assigned to unit j to start being processed at event point n

Integer variables

$B_{i,j,n}$: amount of materials from task i that begins processing in unit j at time point n

$BE_{i,j,n}$: amount of materials from task i that completes its processing at unit j at event point n

$BR_{i,j,n}$: amount of materials from task i that continues its processing at unit j at time point n

Continuous variables

SL_n: length of time slot n

T_n: location of event point n

$TR_{i,j,n}$: amount of time remaining to complete processing materials from task i that continue to be processed at unit j at event point n

$w_{i,k,n}$: amount of materials from task i that have visited process p_{k-1}^{i} and are waiting to visit process p_k^i at event point n

$YE_{i,j,n}$: 0-1 continuous variable which take the value one if a subset of materials from task i completed their processing at unit j at event point n

$YR_{i,j,n}$: 0-1 continuous variable which take the value one if a subset of materials from task i continues to be processed at unit j at event point n

$Z_{j,n}$: 0-1 continuous variable which take the value one if unit j starts processing materials at event point n

$$\sum_{n>1} SL_n = H \qquad \text{(B1)}$$

$$T_n - T_{n-1} = SL_n \qquad \forall\, n > 1 \qquad \text{(B2)}$$

$$T_n \geq \left(\max TA_i TM_j \cdot Y_{i,j,n}\right) \qquad \forall j,\, i \in \mathbf{I}_j,\, n \qquad \text{(B3)}$$

$$z_{j,n} \geq Y_{i,j,n} \qquad \forall j,\, i \in \mathbf{I}_j,\, n \qquad \text{(B4)}$$

$$z_{j,n} \leq \sum_{i \in I_j} Y_{i,j,n} \qquad \forall j,\, n \qquad \text{(B5)}$$

$$YR_{i,j,n} = YR_{i,j,n-1} + Y_{i,j,n} - YE_{i,j,n} \qquad \forall j,\, i \in \mathbf{I}_j,\, n > 1 \qquad \text{(B6)}$$

$$z_{j,n} \geq YE_{i,j,n} \qquad \forall j,\, i \in \mathbf{I}_j,\, n \qquad \text{(B7)}$$

$$z_{j,n} \leq 1 - YR_{i,j,n} \qquad \forall j,\, i \in \mathbf{I}_j,\, n \qquad \text{(B8)}$$

$$1 - \sum_{i \in I_j} YR_{i,j,n} \leq z_{j,n} \qquad \forall j,\, n \qquad \text{(B9)}$$

$$Y_{i,j,n} \leq 1 - Y_{I_0,j,n} \qquad \forall j,\, i \in \mathbf{I}_j,\, i \neq I_0,\, n \qquad \text{(B10)}$$

$$Y_{i,j,n} \cdot B_j^{\min} \leq B_{i,j,n} \leq Y_{i,j,n} \cdot B_j^{\max} \qquad \forall j,\, i \in \mathbf{I}_j,\, n \qquad \text{(B11)}$$

$$z_{j,n} \cdot B_j^{\min} \leq \sum_{i \in \mathbf{I}_j} B_{i,j,n} \leq z_{j,n} \cdot B_j^{\max}$$

$\forall j, n$ (B12)

$$YR_{i,j,n} \cdot B_j^{\min} \le BR_{i,j,n} \le YR_{i,j,n} \cdot B_j^{\max}$$ $\forall j, i \in \mathbf{I}_j, n$ (B13)

$$(1 - z_{j,n}) \cdot B_j^{\min} \le \sum_{i \in \mathbf{I}_j} BR_{i,j,n} \le (1 - z_{j,n}) \cdot B_j^{\max}$$

$\forall j, n$ (B14)

$$YE_{i,j,n} \cdot B_j^{\min} \le BE_{i,j,n} \le YE_{i,j,n} \cdot B_j^{\max}$$ $\forall j, i \in \mathbf{I}_j, n$ (B15)

$$z_{j,n} \cdot B_j^{\min} \le \sum_{i \in \mathbf{I}_j} BE_{i,j,n} \le z_{j,n} \cdot B_j^{\max}$$

$\forall j, n$ $\forall j, n$ (B16)

$$\sum_n \sum_{j \in \mathbf{J}_p, p = p_k^i} B_{i,j,n} \le \alpha_i$$

$\forall\ i, p, k = 1$ (B17)

$$BR_{i,j,n} + BE_{i,j,n} = BR_{i,j,n-1} + B_{i,j,n-1}$$ $\forall j, i \in \mathbf{I}_j, n$ (B18)

$$W_{i,k,n} = \alpha_i + \sum_{j \in \mathbf{J}_p, p = p_{k-1}^i} B_{i,j,n} - \sum_{j \in \mathbf{J}_p, p = p_k^i} BE_{i,j,n}$$

$\forall\ i, k, n = 1$ (B19)

$$W_{i,k,n} = W_{i,k,n-1} + \sum_{j \in \mathbf{J}_p, p = p_{k-1}^i} B_{i,j,n} - \sum_{j \in \mathbf{J}_p, p = p_k^i} BE_{i,j,n}$$

$\forall\ i, k, n > 1$ (B20)

$$TR_{i,j,n} \le \tau_j \cdot YR_{i,j,n}$$ $\forall j, i \in \mathbf{I}_j, n$ (B21)

$$TR_{i,j,n+1} \ge TR_{i,j,n} + \tau_j \cdot YR_{i,j,n}$$ $\forall j, i \in \mathbf{I}_j, n < N$ (B22)

Maximization of productivity

$$z = \sum_s p_s \sum_j \sum_{i \in (\boldsymbol{I}_j \cap \boldsymbol{I}_S^P)} \sum_n \rho_{i,s} \cdot BE_{i,j,n}$$

(B23)

Minimization of makespan

$$T_n \le MS$$ $\forall\ n = N$ (B24)

$$\sum_j \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_S^P)} \sum_n \rho_{i,s} \cdot BE_{i,j,n} \ge D_s$$

$\forall\ s \in \mathbf{S}^P$ (B25)

While the model of Lagzi et al. (2017a) for maximization of productivity includes constraints B1-B23, the model of Lagzi et al. (2017a) for makespan minimization consists of B1-B22 and B24-B25.

References

Floudas, C. A., Lin, X., 2004. Continuous-time versus discrete-time approaches for scheduling of chemical processes: a review. Computers and Chemical engineering. 28(11). 2109-2129. https://doi.org/10.1016/j.compchemeng.2004.05.002.

Harjunkoski, I., Maravelias, C. T., Bongers, P., Castro, P. M., Engell, S., Grossmann, I. E., Hooker, J., Méndez, C., Sand, G., Wassick, J., 2014. Scope for industrial application of production scheduling models and solution methods. Computers and Chemical Engineering. 62(5). 161-193. https://doi.org/10.1016/j.compchemeng.2013.12.001.

Hui, C., Gupta, A., van der Meulen, H. A. J. 2000. A novel MILP formulation for short-term scheduling of multi-stage multi-product batch plants with sequence-dependent constraints. Computers & Chemical Engineering, 24(12). 2705 – 2717 https://doi.org/10.1016/S0098-1354(00)00623-2

Ierapetritou, M. G., Floudas, C., 1998. A. Effective continuous-time formulation for short-term scheduling. 1. Multipurpose batch processes. Industrial & Engineering Chemistry. 37(11). 4341-4359. https://doi.org/10.1021/ie970927g.

Kondili, E., Pantelides, C. C., Sargent, R. W. H., 1993. A general algorithm for short-term scheduling of batch operations-I MILP formulation. Computers & Chemical Engineering. 17(2). 211-227. https://doi.org/10.1016/0098-1354(93)80015-F.

Kopanos, G., Puigjaner, L., 2019. Solving Large-Scale Production Scheduling and Planning in the Process industries, Springer, https://doi.org/10.1007/978-3-030-01183-3

Lagzi, S., Fukasawa, R., Ricardez-Sandoval, L., 2017a. A multitasking continuous time formulation for short-term scheduling of operations in multipurpose plants. Computers & Chemical Engineering. 97. 135-146. https://doi.org/10.1016/j.compchemeng.2016.11.012.

Lagzi, S., Lee, D. Y., Fukasawa, R., Ricardez-Sandoval, L., 2017b. A computational study of continuous and discrete time formulations for a class of short-term scheduling problems for multipurpose plants. Industrial and Engineering Chemistry Research. 56(31). 8940-8953. https://doi.org/ 10.1021/acs.iecr.7b01718.

Lee, S. Y., Fukasawa, R., Ricardez-Sandoval, L., 2019. Bi-objective short-term scheduling in a rolling horizon framework: a priori approaches with alternative operational objectives. Computers and Operations research. 111.141-154 https://doi.org/10.1016/j.cor.2019.06.006

Li J., Karimi I. A., 2011. Scheduling gasoline blending operations from recipe determination to shipping using unit slots. Industrial & Engineering Chemistry Research, 50(15). 9156-9174 https://doi.org/10.1021/ie102321b

Li J., Xiao X., Floudas C. A., 2016. Integrated gasoline blending and order delivery operations: part I. short-term scheduling and global optimization for single and multi-period operations. AIChE Journal, 62(6). 2043-2070 https://doi.org/10.1002/aic.15168

Li, J., Floudas, C.A., 2010. Optimal Event Point Determination for Short-Term Scheduling of Multipurpose Batch Plants via Unit-Specific Event-Based Continuous-Time Approaches, Industrial & Engineering Chemistry Reshearch. 49(16). 7446-7469. https://doi.org/10.1021/ie901842k.

Li, J., Sursarla, N., Karimi, I. A., Shaik, M. A., Floudas, C. A., 2010. An Analysis of Some Unit-Specific Event-Based Models for the Short-Term Scheduling of Noncontinuous Processes. Industrial and Engineering Chemistry Research. 49(2). 633-647. https://doi.org/ 10.1021/ie801879n.

Maravelias C. T., Grossmann I. E., 2003. New general continuous-time state-task network formulation for short-term scheduling of multipurpose batch plants. Industrial & Engineering Chemistry Research, 42(13). 3056-3074 https://doi.org/10.1021/ie020923y

Maravelias, C. T., 2012. General framework and modeling approach classification for chemical production scheduling. AIChE journal. 58(6). 1812-1828. https://doi.org/10.1002/aic.13801.

Méndez C. A., Cerdá J. 2000. Optimal scheduling of a resource-constrained multiproduct batch plant supplying intermediates to nearby end-product facilities. Computers & Chemical Engineering, 24(2-7). 369-376 https://doi.org/10.1016/S0098-1354(00)00482-8

Méndez C. A., Cerdá J. 2003. An MILP continuous-time framework for short-term scheduling of multipurpose batch processes under different operation strategies. Optimization and Engineering, 4(1-2). 7-22 https://doi.org/10.1023/A:1021856229236

Méndez, C. A., Cerdá, J., Grossmann, I. E., Harjukoski, I., Fahl, M., 2006. State-of-the-art review of optimization methods for short-term scheduling of batch processes. Computers and Chemical Engineering.30(6-7). 913-946.https://doi.org/10.1016/j.compchemeng.2006.02.008.

Pantelides C. Unified frameworks for optimal process planning and scheduling. Proceedings of the Second Conference on Foundations of Computer Aided Operations, 1994, 253-274

Patil, B. P., Fukasawa, R., Ricardez-Sandoval, L. A., 2015. Scheduling of operations in a large-scale Scientific services facility via multicommodity flow and an optimization-based algorithm. Industrial and Engineering Chemistry Research. 54(5). 1628-1639. https://doi.org/10.1021/ie503660f.

Santos F. Fukasawa, R., Ricardez-Sandoval, L., 2018. An integrated personnel allocation and machine scheduling problem for industrial size multipurpose plants. IFAC-PapersOnLine. 51(18). 156-161 https://doi.org/10.1016/j.ifacol.2018.09.292

Shaik M. A., Janak S. L., Floudas C. A., 2006. Continuous-time models for short-term scheduling of multipurpose batch plants: a comparative study. Industrial & Engineering Chemistry Research, 45(18). 6190-6209 https://doi.org/10.1021/ie0601403

Shaik, M. A., Floudas, C. A., 2008. Unit-specific event-based continuous-time approach for short-term scheduling of batch plants using RTN framework. Computers & Chemical Engineering. 32(1-2). 260-274. https://doi.org/10.1016/j.compchemeng.2007.05.007.

Shaik, M. A., Floudas, C. A., 2009. Novel Unified Modeling Approach for Short-Term Scheduling. Industrial & Engineering Chemistry Research. 48(6). 2947-2964. https://doi.org/10.1021/ie8010726.

Sundaramoorthy, A., Karimi, I. A., 2005. A simpler better slot-based continuous-time formulation for short-term scheduling in multipurpose batch plants. Chemical engineering science. 60(10). 2679-2702. https://doi.org/10.1016/j.ces.2004.12.023.

Sursarla N., Li J., Karimi I., 2010. A novel approach to scheduling multipurpose batch plants using unit-slots. AIChE Journal, 56(7). 1859-1879 https://doi.org/10.1002/aic.12120

Tang Q. H., Li. J., Floudas C. A., Deng M. X., Yan Y. B., Xi Z. H., Chen. P. H., Kong J. Y., 2012. Optimization framework for process scheduling of operation-dependent automobile assembly lines. Optimization Letters, 6(4). 797-824 https://doi.org/10.1007/s11590-011-0303-5

Velez S., Maravelias C. T., 2013. Multiple and nonuniform time grids in discrete-time MIP models for chemical production scheduling. Computers and chemical engineering, 53(11). 70-85 https://doi.org/10.1016/j.compchemeng.2013.01.014

Blank Page

6.3 Rolling horizon decomposition approach for large-scale multi-tasking multipurpose batch process scheduling problems

6.3.1 Introduction

In Chapter 6.2 we presented three unit-specific event-based mathematical models for scheduling of multi-tasking, multipurpose batch processes. While the first two models use timing variables based on tasks, the timing variables of the third model is based on units. The latter model is the most efficient to solve the multi-tasking multipurpose batch problem, since it leads to significantly smaller model sizes, which leads to smaller computational times. For minimization of makespan, the model also leads to tigher relaxation and as a result, it is even able to generate optimum solutions for examples that existing model even fail to generate a feasible solution.

Despite the high efficiency of the unit-specific event-based mathematical model using timing variables based on units, it seems that even this model cannot handle large-scale and complex problems. For instance, for some examples, with minimization of makespan as objective, any of the examined mathematical models were able to generate a feasible solution after one hour. The main reason behind this issue is that significantly high number of orders with many properties have to be examined. This leads to exceptionally large model sizes, which makes it impossible for a short-term model to solve it directly.

As discussed before, a number of rolling horizon decomposition approaches were developed for the job-shop scheduling problem and the multipurpose batch problem (Singer 2001; Lin *et al.* 2002; Janak *et al.* 2004; Shaik *et al.* 2009; Li *et al.* 2012; Yan *et al.* 2013; Mohammadi and Poursabzi 2014). On the other hand, a rolling horizon decomposition approach was not implemented in the multi-tasking problem before. Additionally, there is no decomposition approach that can effectively solve problems with the same due dates for a large number of due dates or problems with no or the same due dates.

To tackle this problem, we enhance the rolling horizon decomposition approach developed for multipurpose batch processes (Lin *et al.* 2002; Janak *et al.* 2004; Shaik *et al.* 2009; Li *et al.* 2012). In this decomposition approach the properties examined are divided into a number of groups using mixed-integer programming. Each group is a subproblem and the allocation and sequencing of samples included in the same group is considered simultaneously. To effectively divide a large-scale problem, the number of groups generated are minimized, while minimizing the difference in the number of orders included in each group. Additionally, the number of properties to be included in the model are controlled, to generate subproblems that the short-term mathematical model is able to generate the optimum solution in small computational time. The results demonstrate that the proposed enhanced rolling horizon decomposition model can successfully decompose and solve problem, which all mathematical models fail even to generate a feasible solution.

6.3.2 Enchased Rolling horizon decomposition approach

As already discussed, there are multiple properties that have to be examined for each order/sample group. The scientific service facility cannot randomly examine those properties. Instead, the property examination sequence is predefined. To effectively divide the scheduling horizon, we introduce a new set K (k = 1, 2, …, K) which denotes the k^{th} property that has to be examined of each order. For instance, k = 3 denotes the third property that has to be examined for a given order.

To enhance the rolling horizon decomposition approach, we first introduce a binary variable Y_g to denote the active groups/subproblems for the given problem. To monitor the properties of each order examined in each subproblem, we introduce a binary variable $Y^{o}_{o,k,g}$. According to constraint (1), we can examine the k^{th} property of a given order during a group/subproblem only if the group/subproblem is active.

$$Y^{o}_{o,k,g} \leq Y_g \qquad \forall o, k \in O_k, g \qquad (1)$$

Additionally, if a group g is active, then it should include at least one property during this subproblem.

$$\sum_{o,k \in O_k} Y^{o}_{o,k,g} \geq Y_g \qquad \forall\, g \qquad (2)$$

During the scheduling horizon all properties of all orders should be examined once.

$$\sum_{g} Y^{o}_{o,k,g} = 1 \qquad \forall o, k \in O_k \qquad (3)$$

The examination of the k^{th} property of a given order o can take place in a group g, only if (k – 1) property of the same order is already examined in a previous group $g' < g$, or it is included in the current group g.

$$Y^{o}_{o,k',g} \leq Y^{o}_{o,k,g} + \sum_{g'<g} Y^{o}_{o,k,g'}$$

$$\forall o, k, k' \in O_k, k' = k + 1 \qquad (4)$$

A group g + 1 cannot be selected if the previous group g is not selected.

$$Y_{g+1} \leq Y_g \qquad \forall\, g \qquad (5)$$

To monitor the number of properties of each order o that a group g contains, we introduce a continuous variable $TNO_{o,g}$.

$$TNO_{o,g} = \sum_{k \in O_k} Y^{o}_{o,k,g}$$

$$\forall\, o, g \qquad (6)$$

Constraint (7a) sequences the number of properties of each order included in each group in decreasing order.

$$TNO_{o,g+1} \leq TNO_{o,g} \qquad \forall\, o, g < G \qquad (7a)$$

Constraint (7a) limits the minimum number of properties that can be included in a group g. For instance, in each group, at least one property from each order should be included. Otherwise, if no properties from a given order o are included at group g, then no properties from this order can be included in the next groups $g' > g$. As a result, the minimum number of orders that can be included in each group (with the exception of the last group) is $|O|$. For very large examples, this however can still lead to subproblems with large model sizes, which requires excessive computational time. To avoid this case, we relax constraint (7a) by introducing a parameter $M^{\max}$.

$$TNO_{o,g+1} \leq TNO_{o,g} + M^{\max} \qquad \forall\, o, g < G \qquad (7b)$$

Additionally, the total number of properties included in a group g is monitored by using the continuous variable $TNOP_g$.

$$TNOP_g = \sum_{o,k \in O_k} Y_{o,k,g}^{o}$$

$$\forall\, g \qquad (8)$$

To avoid generating subproblems with many properties to be examined, that require excessive computational time to generate the optimum solution, we introduce a parameter $L^{\max}$, which denotes the maximum number of properties that can be included in a group g.

$$TNOP_g \leq L^{\max} \qquad \forall\, g \qquad (9)$$

Finally, we use two penalties *PEN1* and *PEN2* in order to minimize the difference in the total number of properties included in each group g.

$$PEN1 \geq TNO_{o,g} \qquad \forall\, o, g \qquad (10)$$

$$PEN2 \leq TNO_{o,g} + |G| \cdot \left(1 - Y_g\right) \qquad \forall\, k, g \qquad (11)$$

The objective of this model is to minimize the number of groups selected. In this way we minimize the number of subproblems that the main problem is divided.

$$obj = w_1 \cdot \sum_{g} Y_g + w_2(PEN1 - PEN2)$$

$$(12)$$

Where w_1 and w_2 are the importance weight parameters.

For each subproblem, the number of event points (EN_g) used are equal to the maximum number of samples that a unit is able to process plus the number of different properties included in the given group g.

$$EN_g = \max_{p}\left(\left\lceil \sum_{o \in P_o}\left(\frac{\sum_{k \in K_p, K_o} samples_o}{\sum_{j \in J_p} B_j^{\max}}\right)\right\rceil\right) + \left|P_g\right| - 1$$

$$\forall g \qquad (13)$$

6.3.3 Computational results

We implement the proposed enhanced rolling horizon decomposition approach to solve

Examples 20-31 from Rakovitis *et al.* (2020) with minimization of makespan as objective. Examples 21-25 contain 5 orders, while examples 26-30 contain 10 orders. On the other hand, Examples 20 and 31 large-scale examples, with 100 orders and 200-300 samples each. Additionally, 25 properties can be examined in the facility in 84 processing units for all Examples. The relevant data for all these examples are presented in Rakovitis *et al.* (2020). Table 1 depicts the parameters L^{max} and M^{max} used to solve this problem. All examples are solved to 1% of optimality gap using CPLEX 12/GAMS 24.6.1. on a desktop computer with Intel® Core™ i5-2500 3.3 GHz and 8 GB RAM running Windows 7. The maximum computational time is one hour for all examples.

Table 1 Additional data for Examples 20-31

Example	L^{max}	M^{max}	Example	L^{max}	M^{max}
20	100	0	26	10	0
21	5	0	27	10	0
22	5	0	28	10	0
23	5	0	29	10	0
24	5	0	30	10	0
25	5	0	31	50	1

Table 2 presents the results generated for all examined examples. The results demonstrate that the proposed approach is able to generate a schedule for all examples, even for those examples that the short-term model of Rakovitis *et al.* (2020) is not able to generate a feasible solution after one hour. For example, by only using the short-term model of Rakovitis *et al.* (2020) we can only generate a solution for Examples 21-23 and 27. On the contrary, the rolling horizon decomposition approach can generate a solution for all Examples 20-31. This approach can even generate solutions for very large and complex examples (Example 31). However, for this example excessive computational time is required even with rolling horizon decomposition approach. Another conclusion is that the proposed approach is able to generate slightly worse solutions than by only using the short-term model. For instance, in Example 21 only using the short-term model, leads to a solution with a makespan of 5131 min (Rakovitis *et al.* 2020), after one hour of computational time. On the other hand, the proposed rolling horizon decomposition approach requires less than one second (0.7 s) to generate an approximately 9% worse solution (5621 min). As a result, the benefits of significantly reducing the computational time overpass the fact that a slightly worse solution is generated.

Table 2 Summary of computational results for large-scale examples

Example	Makespan (min)	Total CPU time (s)	Example	Makespan (min)	Total CPU time (s)
20	36199	63.9	26	35075	44.3
21	5621	0.7	27	7156	1.0
22	22295	1.2	28	18860	10.9
23	5815	0.8	29	33775	12.0
24	19295	1.2	30	23235	2.3
25	20795	1.1	31	642652	15079.1

Table 3 depicts more details for each subproblem solved for example 20. In Example 20, 100 orders have to be examined for 8 to 9 different properties. We set the maximum number of properties (tasks) to be examined (L^{max}) is 100. Therefore, in each subproblem at most 100 properties can be examined. This is the case for the first eight subproblems, while for the last subproblem, only 50 properties are examined. Note that different orders can be examined for the same property within a specific subproblem. However, the examination of a property for different orders is considered as a different task.

Table 3 Computational results for each subproblem for Example 20

Sub-problem	Properties (tasks) Examined	Makespan (min)	CPU time (s)	Integer variables	Continuous variables	Constraints
1	100	2660	0.19	3976	6553	11728
2	100	4910	1.3	3456	5116	12868
3	100	6326	2.0	3408	6819	14724
4	100	10674	0.84	3388	3981	11464
5	100	16969	0.50	4053	3985	12889
6	100	19843	1.1	2328	3421	8564
7	100	22699	6.7	6842	6288	21246
8	100	27424	51.0	6058	7409	20890
9	50	36199	0.27	2752	5896	11486

From Table 3, it seems that the rolling horizon decomposition approach can successfully decompose the problem that the short-term model can easily solve. For instance, the short-term model requires less than one minute to generate the optimal solution for a given subproblem. For most subproblems, less than one second is required to generate the optimal solution (subproblems 1, 4, 5, 9). Additionally, it seems that all examples lead to similar model sizes for all subproblems. As a result, the rolling horizon decomposition approach proposed can efficiently divide the problem in smaller subproblem that the short-term model can efficiently solve.

6.3.4 Conclusions

Even though the proposed short-term model for scheduling of multi-tasking multipurpose batch processes can be very efficient, it seems that in some cases it may fail to generate the optimal solution or even to fail to generate a feasible solution in small computational time. In this work, we enhance the rolling horizon decomposition approach that is able to decompose problems without due dates. From the results generated, it seems that the proposed approach can efficiently decompose the problem, in smaller subproblems. As a result, implementing this approach to a number of multitasking multipurpose batch process scheduling problems, can significantly reduce the computational time required to generate a slightly worse solution. Additionally, the proposed approach can generate a solution for all examples, in contrast to the case that only the short-term model is used.

Blank Page

Chapter 7: Energy-efficient scheduling of flexible job-shops

7.1 Introduction

The flexible job-shop scheduling problem has been well examined in the past decades with multiple metaheuristics and mathematical modelling methodologies proposed to solve this problem. However, the majority of the formulations only take into consideration the economic performance without considering energy consumption. Such approaches, even though they generate schedules where the processing units process all jobs at the earliest possible time, they often lead to significantly high energy demands. Therefore, it is crucial to consider energy consumption in the scheduling problem. Only a few approaches considered energy-efficient scheduling of flexible job-shops. Such methodologies, either fail to generate the optimal solution even for small examples, since they do not consider switching off and on the processing units, or they lead to large model sizes and excessive computational time required to generate a solution.

In this chapter, the proposed framework presented in the previous research contributions is implemented to solve the flexible job-shop scheduling problem by considering energy consumption. Switching off and on of processing units is also considered. Additionally, two mathematical models using sequence-based representation is proposed to compare its performance with the proposed unit-specific event-based framework. Several examples were solved to examine the capabilities of the models as well as the proposed formulations. For large-scale problems, that require excessive computational time, an enhanced rolling horizon decomposition approach, by developing mixed-integer mathematical programming to group operations is proposed. Finally, the capabilities of hybrid algorithms were examined by combining the mathematical programming with the genetic evolutionary programming (GEP) approach. More specifically, GEP is used to generate the allocation and sequence of operations into units and mathematical programming to develop the optimal timings of those operations into units.

Blank Page

7.2 Research contribution 5

Rakovitis, N., Zhang, N., Li, J. Zhang, L. Novel Approaches for Energy-Efficient Scheduling of Flexible Job-Shop Problems, to be submitted to European Journal of Operational Research

Blank Page

Novel Approaches for Energy-Efficient Scheduling of Flexible Job-Shop Problems

Nikolaos Rakovitis[1], Dan Li[1], Matthew Mclaughlan[1], Nan Zhang[1], Jie Li[1,§] and Liping Zhang[2]

[1]Centre for Process Integration, Department of Chemical Engineering and Analytical Science, The University of Manchester, Manchester, M13 9PL, United Kingdom

[2]Department of Industrial Engineering, School of Machinery and Automation, Wuhan University of Science and Technology, Wuhan, Hubei, 430081 P. R China

Abstract

In this work, we develop three mathematical models for scheduling of energy-efficient flexible job shops, based on unit-specific event-based and local sequence-based approach. The computational results demonstrate that the proposed model based on the unit-specific event-based representation is superior to the existing models with the same or better solutions in less computational time. Furthermore, it can generate feasible solutions for some large-scale examples that the previous models fail to solve. To solve larger-scale problems, we enhance the existing rolling-horizon decomposition approach in which a grouping strategy using mixed-integer programming divides the entire horizon into different subproblems. This enhanced rolling-horizon decomposition approach can generate *good* solutions for those large-scale examples that cannot be directly solved using the mathematical models in significantly less computational time. It can also achieve up to 43.1% less energy consumption for most examples in comparison to the existing efficient gene-expression programming-based algorithm. Finally, we combine the approaches of mathematical modelling and GEP. Such approach leads to up to 20% less energy consumption than the solutions generated by only implementing GEP.

Keywords: Scheduling, mixed-integer programming, flexible job-shops, energy-efficient, unit-specific event-based approach, sequence-based approach

[§] To whom correspondence should be addressed. jie.li-2@manchester.ac.uk. Tel: +44 (0) 161 306 8622

1. Introduction

The process industries such as chemical industry, car industry and iron and steel industry usually receive multiple orders from different customers daily. Each facility process several jobs in the available processing units/machines to fulfil the customers' demands,. Each of these jobs contains multiple operations, where several units are processing . The main objective of such a facility is to determine the best sequence of operations on the processing units to eliminate their operational cost and satisfy their customer demands simultaneously. Furthermore, facilities aim to reduce their energy consumption, which also contributes to their expenses, as well as their environmental footprint. Such a scheduling problem is commonly known as job-shop scheduling problem (JSSP) (Bowman 1959). In JSSP, a processing unit can process at most one operation at each time. However, it can process multiple of those operations during the scheduling horizon. With highly increasing customer demands, facilities often install several processing units that can process the same type of operations instead of one processing unit. Scheduling of such facilities is commonly known as flexible job-shop scheduling problem (FJSSP) (Wagner 1959). FJSSP is a more general case of the classical job-shop scheduling problem.

The flexible job-shop scheduling problem has gathered considerable attention during the past decades. The first attempts (Brandimarte 1993; Paulli 1995) were solving the problem by using the two-stage hierarchical method. While Brandimarte (1993) generated schedules based on Tabu search, Paulli (1995) used several dispatching rules to solve the same problem. The first stage defines the assignment of operations into machines, while the second stage then determines the best sequence of those operations in each processing unit. Using such an approach can significantly reduce the computational time. However, it is only limited to generate a feasible solution. Integrated methods were later proposed, to improve solution quality. These integrated methods solve both the assignment and sequence of operations into simultaneously. Different research groups developed several metaheuristic approaches including tabu search (Hurink *et al.* 1994; Mastrolilli *et al.* 2000; Saidi-Mehrabad and Fattahi 2007; Fattahi *et al.* 2007; Liouane *et al.* 2007), genetic algorithm (Chen *et al.* 1999; Pezzella *et al.* 2008; Zhang *et al.* 2011; Al-Hinai and ElMekkawy 2011), artificial immune algorithm (Bagheri *et al.* 2010; Roshainaei *et al.* 2013), imperialist competitive algorithm (Karimi *et al.* 2017), ant colony optimization (Liouane *et al.* 2007), simulated annealing (Fattahi *et al.* 2007), variable neighbourhood search (Yazdani *et al.* 2010) and hybrid methods such as particle swarm optimization and tabu search (Zhang *et al.* 2009), genetic algorithm and local search (Gao *et al.* 2006; Gao *et al.* 2008). Even though these metaheuristics can efficiently generate *good* feasible schedules for the FJSSP, they cannot guaranty the solution optimality. Additionally, they also require excessive computational time to solve industrial-scale problems. Such issue led to the development of mathematical programming approaches for this scheduling problem (Choi and Choi 2002; Gao *et al.* 2006; Fattahi *et al.* 2007; Özgüven *et al.* 2010; Roshainaei *et al.* 2013; Karimi *et al.* 2017). Chaudhy and Khan (2016) and Xie *et al.* (2019) includes more information for different approaches for solving the FJSSP.

Most of the discussed works considered the economic performance of the FJSSP only without incorporating energy consumptions during scheduling. As reported, the existing real-life industries suffer from high energy consumption. The processing units can consume up to 65% of the total energy consumption during the period that they remain idle (Gutowski *et al.* 2005; Devoldere *et al.* 2007; Nguyen *et al.* 2019). A processing unit consumes this amount of energy even if it does not process any operations/tasks to maintain its functionality. Since most of the existing approaches only consider makespan minimization, they generate schedules where one or more processing units can remain idle for long periods during the scheduling horizon, resulting in significantly high energy consumption. Furthermore, most of the existing formulations do not consider switching off-on strategy, which can save energy if a processing unit does not process an operation for long periods. The switching off-on strategy can potentially lead to significant energy savings of at least 13% (Mouzon *et al.* 2007).

Only a few approaches considered the case of developing energy-efficient schedules for the flexible job-shop problem. Zhang et al. (2017) developed an efficient algorithm to create *good* dispatching rules that can generate schedules using gene expression programming (GEP). GEP is an evolutionary algorithm which is used to develop an efficient model. Similarly, to other evolutionary approaches, a population of random chromosomes is used, which are evolved through the mutation and selection procedure. Each chromosome can be converted to a formula, model or, in this case, into an efficient dispatching rule. This approach can generate several dispatching rules by using a set of examples as "training sets". For more information on GEP, the reader can be refer to Zhang et al. (2017). Even though the GEP-based algorithm of Zhang et al. (2017) can generate good schedules, even for large-scale problems, it cannot guarantee the solution optimality. As we demonstrate later, the solution obtained from the GEP-based approach is a bit far from the optimal solution. Zhang et al. (2017) also developed a mixed-integer linear programming (MILP) model, which can solve small-scale problems to optimality. However, the model requires huge computational time or fails to generate schedules for large-scale examples. Wang *et al.* (2018) developed a two-stage optimization method for energy-efficient scheduling of flexible job shops. In the first stage, the assignment of operations into processing units is determined using a modified genetic algorithm while in the second stage a hybrid genetic algorithm-particle swarm optimization approach is used to generate the optimal sequence of operations on each processing unit. Although the proposed meta-heuristic approach can generate feasible schedules for large-scale examples, it often fails to provide the optimal energy-efficient schedule as the assignment and sequencing problems are not solved simultaneously. It also did not consider the machine switching off-on strategy, which could further reduce energy consumption. Meng *et al.* (2019) developed six mathematical formulations using the modelling approach of Wanger (1959) for a such scheduling problem. By comparing those models with the mathematical model of Zhang et al. (2017), they concluded that most of their models are more efficient than that of Zhang et al. (2017) due to smaller model size and less computational time required. However, these models still require excessive computational time or fail to find feasible solutions for large-scale problems. Finally, several works have considered the multi-objective optimisation problem of both minimizing makespan and total energy consumption for both JSSP (May *et al.* 2015) and FJSSP (Dai *et al.* 2013; Lei *et al.* 2016; Mokhtari and Hasani 2017; Zhang *et al.* 2018; Wu and Sun 2018), which will be our future work to extend our approach for the multi-objective optimisation problem.

In this work, we first develop three novel mathematical formulations for the energy-efficient scheduling of flexible job-shop problem using the improved unit-specific event-based (Rakovitis *et al.* 2019) and the local sequence-based (Méndez and Cerdá, 2000) time representation. For the local sequence-model, we examine two different sets of binary variables to define the sequencing between operations. The proposed formulations lead to a tighter MILP relaxation and smaller model size compared to the existing models of Zhang *et al*. (2017) and Meng *et al*. (2019). As a result, they can generate the same or better feasible solutions than the models of Zhang *et al*. (2017) and Meng *et al*. (2019). Furthermore, they can generate solutions for examples that the existing mathematical models of Zhang *et al*. (2017) and Meng *et al*. (2019) fail after a specified computational time (e.g., 1 hour). The model based on unit-specific event-based time representation is the most efficient and robust since it can generate better solutions than all models. Additionally, it can develop solutions for most examples. To solve large-scale and computationally expensive problems, we enhance the rolling horizon decomposition approach (Lin *et al.*, 2002; Janak *et al.*, 2006; Li et al., 2012) in which a grouping strategy using the mixed-integer programming further divides the optimization problem with the same due date into sub-problems. The computational results demonstrate the proposed decomposition approach can generate optimal schedules for small-scale examples, while for large-scale ones, it can generate improved schedules than eGEP with up to 27.6% additional energy savings. It can also improve the solution quality with up to 28.5% energy savings for the examples with more than ten jobs in significantly less computational time in comparison to the short-term model. Finally, we develop a hybrid algorithm through a simple combination of the mathematical programming approach with the GEP-based approach. In the hybrid algorithm, we use the GEP-based algorithm of Zhang *et al*. (2017) to generate the allocation of operations to the processing units and their sequence on these processing units. Then, the two sequence-based models are used to determine the optimal timings of operations on the processing units. The computational results demonstrate that this hybrid approach can generate improved solutions with up to 20% energy savings in comparison to the GEP-based method. By comparing the results between the enhanced rolling horizon decomposition and the hybrid approach, it seems that, even though the hybrid approach can lead to higher energy savings, there are many cases where the enhanced rolling horizon decomposition approach can generate a schedule with less energy consumption.

2. Problem description

Figure 1 illustrates a typical flexible job-shop facility. There are K $(k = 1, 2, 3, \dots, K)$ jobs to be processed with up to L $(l = 1, 2, 3, \dots, L)$ operations in each job and J $(j = 1, 2, 3, \dots, J)$ processing units/machines. Each job k contains $\mathbf{L}_k$ operations. Each operation l can be processed in $\mathbf{J}_l$ units. At a time, at most one operation can be processed in a processing unit. Each operation is processed exactly once during the entire scheduling horizon. The processing sequence of operations in a job i is known a prior. An operation in a job i can only start if all precedent operations in this job have already been processed. The disjunctive graph is a method used to represent job-shop and flexible job-shop facilities (Roy and Sussmann 1964).

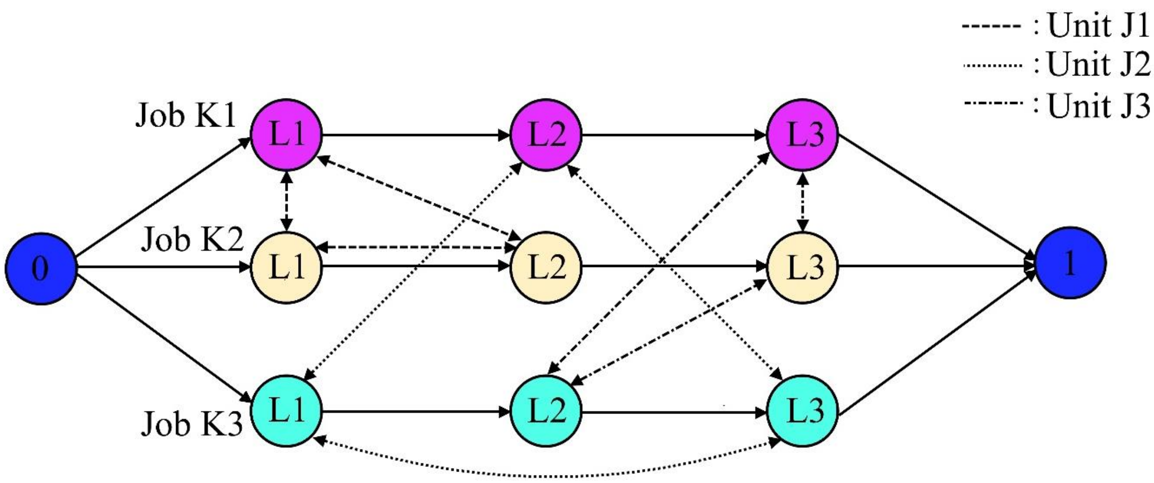

Figure 1 A typical flexible job-shop facility using the disjunctive graph representation

At a time, a processing unit j can either process operations or be idle. When it processes an operation l that belongs to a job k, it should process for some duration denoted as $\alpha_{k,l,j}$. Energy consumed during processing includes direct and indirect energy. While direct energy is the energy consumed by processing units to process operations, indirect energy is consumed within the facility for processing an operation. Indirect energy is not directly related to the processing of an operation. For instance, the facility should be properly lighted so that personnel can operate processing units. The cutting power that a unit j directly requires to process an operation l in a job k is denoted as $PC_{k,l,j}$. After a processing unit finishes an operation, it can remain on until the next processing, or it can be switched off and on right before the next processing. While the former is called standby mode, the latter is called switch off/on mode. During standby, a processing unit requires energy to maintain its functionality. Such energy is called standby energy. Standby energy consumption is related to the time (ST_j) that the unit remains idle. The unit unload power for standby is constant, and it is denoted as PU_j. Energy consumed during switch off and on mode is assumed to be a constant for each machine/processing unit, which is denoted as EO_j. The time that the processing unit remains idle is denoted as ST_j. The standby energy should not be higher than the switch off-on energy between two operations. Figure 2 illustrates the energy consumption profile for a processing unit.

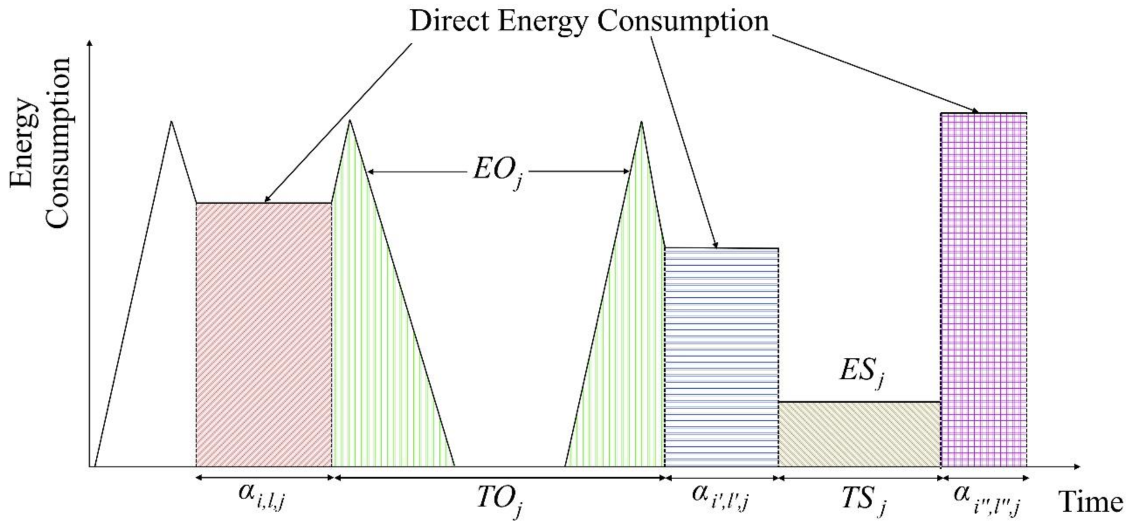

Figure 2 The energy consumption profile for a machine/processing unit

With these, the energy-efficient scheduling of flexible job-shop problem can be stated as follows,

Given:

a) K jobs to be processed, and corresponding $\mathbf{L}_k$ operations;
b) J processing units, suitable operations that can be processed, processing times;
c) Unit cutting power ($PC_{k,l,j}$), indirect energy consumption coefficient (β), unit unload power in standby (PU_j) and the switch off-on energy consumption (EO_j);
d) Scheduling horizon.

Determine:

a) Optimal processing schedule including the allocation of operations to units, their sequences, and timings on each unit;
b) Optimal operating mode for a unit;
c) Optimal energy consumption profile.

Operating rules:

a) At most one operation can be processed in a unit at a time.
b) An operation must be processed exactly once during the scheduling horizon.
c) An operation in a job can start only after all precedent operations in the same job have finished.

Assumptions

a) All parameters are deterministic;
b) Unlimited unit wait policy;
c) Unlimited resources are available;

d) All jobs must be completed in the scheduling horizon.

e) All processing units are switched off at the beginning of the scheduling horizon. They are switched on right before the time that the first operation has to be processed in this unit;

f) All processing units are switched off after the finish processing the last operation.

The objective is to minimize total energy consumption, which consists of direct, indirect, standby and switch off-on energy consumptions.

3. Mathematical formulations

We develop four mathematical formulations using the unit-specific event-based modelling approach and the sequence-based modelling approach. While the first model is based on unit-specific event-based representation, the next two models use the local sequence-based approach.

3.1. Unit-specific event-based formulation (M1)

The improved unit-specific event-based modelling approach (Rakovitis *et al.* 2019) is used to develop the model **M1** since its advantages have been well established in the literature. In this modelling approach, the scheduling horizon is divided based on processing units (Rakovitis *et al.* 2019; Rakovitis *et al.* 2020). The start and end times of the same event point on different units can differ. Furthermore, A parameter Δn is used to denote the maximum number of event points that a task is allowed to span over. The state-task network representation (Kondili *et al.* 1993) is used to represent the process, as illustrated in Figure 3, which is the STN representation of Figure 1. In this representation, an operation is denoted as a *task*, and it is represented with a rectangle. A circle represents an operation that can "produce" or "consume" a state. Then the processing sequence of two operations in a job is established with the state. It is assumed that the first operation or task in each job consumes a "feed" state denoted as $\mathbf{S}^{F}$, while the last task of each job produces a "product" state, denoted as $\mathbf{S}^{P}$. Other operations or tasks in a job "produce" or "consume" intermediate states, denoted as $\mathbf{S}^{IN}$. A parameter $\rho_{i,s}$ is used to indicate whether a state is consumed (i.e., $\rho_{i,s} = -1$) or produced (i.e., $\rho_{i,s} = 1$) by a task i. Set $\mathbf{I}_j$ is defined to denote tasks that can be processed in a processing unit j, while set $\mathbf{J}_i$ denotes units that can process a task i.

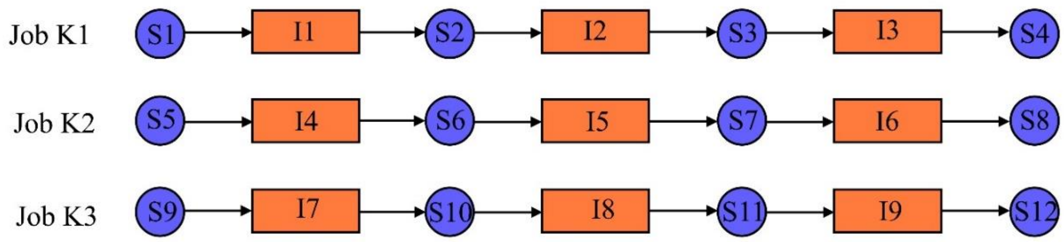

Figure 3 STN representation of a typical flexible job-shop facility

3.1.1. Allocation constraints

We introduce a binary variable $w_{i,j,n,n'}$ to denote if a task i is processed in a unit j from event point n to event point n' as given below,

$$w_{i,j,n,n'} = \begin{cases} 1 & \text{if a task } i \text{ is processed in a unit } j \text{ from event point } n \text{ to event point } n' \\ 0 & \text{otherwise} \end{cases}$$

At a time, a processing unit can process at most one task.

$$\sum_{i \in \mathbf{I}_j} \sum_{n-\Delta n \le n' \le n} \sum_{n \le n'' \le n'+\Delta n} w_{i,j,n',n''} \le 1 \qquad \forall j, n \qquad (1)$$

All tasks must be processed once during the scheduling horizon.

$$\sum_{j \in \mathbf{J}_i} \sum_{n} \sum_{n \le n' \le n+\Delta n} w_{i,j,n,n'} = 1 \qquad \forall i \qquad (2)$$

If two tasks i and i' belonging to the same job are related to the same state (i.e. task i produces a state which is consumed by task i'), then the task i' can only start being processed at event point n if its related production task i ends being processed at the same event point n or a previous event point n'.

$$\sum_{j \in \mathbf{J}_i} \sum_{n' \le n} \sum_{n'-\Delta n \le n'' \le n'} w_{i,j,n'',n'} \ge \sum_{n \le n' \le n+\Delta n} w_{i',j',n,n'}$$

$$\forall s \in \mathbf{S}^{\mathrm{IN}}, i \in \mathbf{I}_S^{\mathrm{P}}, j', i' \in \left(\mathbf{I}_{j'} \cap \mathbf{I}_S^{\mathrm{C}}\right), n \qquad (3)$$

The number of tasks processed in a processing unit j should be within the minimum (N_j^{min}) and maximum (N_j^{max}) limits.

$$\sum_{i \in \mathbf{I}_j} \sum_{n} \sum_{n \le n' \le n+\Delta n} w_{i,j,n,n'} \ge N_j^{\min} \qquad \forall j \qquad (4)$$

$$\sum_{i\in\mathbf{I}_j}\sum_{n}\sum_{n\le n'\le n+\Delta n} w_{i,j,n,n'} \le N_j^{\max} \qquad \forall j \qquad (5)$$

The minimum and maximum number of tasks that can be processed in a unit j can be easily calculated if the available processing units for each task are known. More specifically, the minimum number of tasks processed is equal to the number of tasks that are only available to be processed exclusively in this unit ($\mathbf{I}_j^e$), while the maximum number of tasks processed is equal to all tasks that are suitable to be processed in the unit.

$$N_j^{\min} = \left|\mathbf{I}_j^e\right| \qquad \forall j \qquad (6)$$

$$N_j^{\max} = \left|\mathbf{I}_j\right| \qquad \forall j \qquad (7)$$

3.1.2. Standby energy calculation

We introduce a binary variable $x_{j,n}$ to denote if a unit j remains in the standby mode at event point n below,

$$x_{j,n} = \begin{cases} 1 & \text{if unit } j \text{ remains in the standby mode at the begining of event point } n \\ 0 & \text{otherwise} \end{cases}$$

We also define a positive continuous variable $ES_{j,n}$ to denote the standby energy consumption of a unit j calculated at the beginning of event point n. Constraints (9) and (10) enforce the standby energy consumption of a unit j at event point n to be equal to the idle time multiplying the unit unload power of the unit, if the unit remains in standby mode. Otherwise, constraint (8) enforces the standby energy consumption to be equal to zero.

$$ES_{j,n} \le EO_j \cdot x_{j,n} \qquad \forall j, n \qquad (8)$$

$$ES_{j,n} \le \left(T_{j,n}^{\mathrm{s}} - T_{j,n-1}^{\mathrm{f}}\right) \cdot PU_j \qquad \forall j, n > 1 \qquad (9)$$

$$ES_{j,n} \ge \left(T_{j,n}^{\mathrm{s}} - T_{j,n-1}^{\mathrm{f}}\right) \cdot PU_j - M \cdot PU_j \cdot \left(1 - x_{j,n}\right) \qquad \forall j, n > 1 \qquad (10)$$

3.1.3. Duration constraints

Once a task is processed on a unit j, it must be processed for some duration ($\alpha_{i,j}$). Therefore, the end time of a unit j at event point n must be equal to the start time plus the processing time of the task processed on this unit j. If a unit j does not process any task, then the finish time of this unit j should be equal to the start time at event point n.

$$T^{\mathrm{f}}_{j,n} = T^{\mathrm{s}}_{j,n} + \sum_{i \in \mathbf{I}_j} \sum_{n \le n' \le n+\Delta n} \left(\alpha_{i,j} \cdot w_{i,j,n,n'} \right) \qquad \forall j, n \qquad (11)$$

Note that the end time of a unit j at event point n is enforced to be exactly equal to the start time plus the total processing time to correctly monitor the standby energy consumption.

3.1.4. Sequencing constraints

A processing unit j at an event point $(n+1)$ must always start after the finish time of this unit at the previous event point n.

$$T^{\mathrm{s}}_{j,n+1} \ge T^{\mathrm{f}}_{j,n} \qquad \forall j, n < N \qquad (12)$$

Different tasks in different units

We need to sequence tasks that are related to the same state but processed in different units. More specifically, a consumption task i' must start after the finish time of its related production task i at event point n. We define a continuous variable $T_{s,n}$ to denote the time that a state s is available to be consumed at event point n. In this case, the finish time of a unit j, processing a task which "produces" state s should be before $T_{s,n}$.

$$T_{s,n} \ge T^{\mathrm{f}}_{j,n} - M\left(1 - \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}^P_s)} \sum_{n-\Delta n \le n' \le n} w_{i,j,n',n}\right) \qquad \forall s \in \mathbf{S}^{IN}, j, \sum_{i \in \mathbf{I}_j} \rho_{s,i} > 0, n \qquad (13)$$

Furthermore, the start time of unit j at event point n, processing a task which "consumes" state s should be after the time that state s is available at the same event point n.

$$T_{s,n} \le T^{s}_{j,n} - M\left(1 - \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}^C_s)} \sum_{n \le n' \le n+\Delta n} w_{i,j,n,n'}\right) \qquad \forall s \in \mathbf{S}^{IN}, j, \sum_{j'} \sum_{i' \in (\mathbf{I}_{j'} \cap \mathbf{I}^P_S)} \rho_{s,i'} > 0, n \qquad (14)$$

Finally, the time that a state s is available at event point n must be before the time that the same state is available at the next event point $(n+1)$.

$$T_{s,n} \le T_{s,n+1} \qquad \forall s \in \mathbf{S}^{IN}, n < N \qquad (15)$$

3.1.5. Makespan calculation

It is necessary to calculate makespan to calculate the total energy consumed in a facility.

We define a variable MS to denote makespan. Makespan is the earliest time that all tasks have already been processed.

$$T_{j,n}^{\mathrm{f}} \leq MS \qquad \forall j, n = N \qquad (16)$$

The time that state s is available at event point n cannot exceed makespan.

$$T_{s,n} \leq MS \qquad \forall s \in \mathbf{S}^{IN}, n \qquad (17)$$

3.1.6. Additional constraints

If a job k has a non-zero release time (r_k), then all tasks belonging to this job have a non-zero release time (denoted as r_i). The start time of a unit j processing a task i that belongs to this job should be after the release time.

$$T_{j,n}^{\mathrm{s}} \geq \sum_{i \in \mathbf{I}_j} \sum_{n \leq n' \leq n+\Delta n} r_i \cdot w_{i,j,n,n'}$$

$$\forall j, n, r_i > 0 \qquad (18)$$

Similarly, if a job k has a due date (d_k), then all tasks belonging to this job has a non-zero due date (denoted as d_i). The completion time of a unit j processing a task i that belongs in this job should be before the due date.

$$T_{j,n}^{\mathrm{f}} \leq \sum_{i \in \mathbf{I}_j} \sum_{n-\Delta n \leq n' \leq n} d_i \cdot w_{i,j,n',n} + M\left(1 - \sum_{i \in \mathbf{I}_j} \sum_{n-\Delta n \leq n' \leq n} w_{i,j,n',n}\right)$$

$$\forall j, n, d_i > 0 \qquad (19)$$

Additionally, constraints (20) and (21) are introduced to avoid violation of forbidden sequencing paths and assignments.

$$\sum_n \sum_{n \leq n' \leq n+\Delta n} w_{i,j,n,n'} + \sum_n \sum_{n \leq n' \leq n+\Delta n} w_{i',j',n,n'} \leq 1$$

$$\forall k, (j, j') \in \mathbf{FP}, i \in (\mathbf{I}_j \cap \mathbf{I}_k), i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_k), i \neq i' \qquad (20)$$

$$\sum_n \sum_{n \leq n' \leq n+\Delta n} w_{i,j,n,n'} = 0$$

$$\forall j, i \in \mathbf{FI}_j \qquad (21)$$

Finally, (22) and (23) denote all the continuous and binary variables of model **M** respectively.

$$ES_{j,n}, MS, T_{s,n}, T_{j,n}^{\mathrm{s}}, T_{j,n}^{\mathrm{f}} \geq 0 \qquad (22)$$

$$w_{i,j,n,n'}, x_{j,n} \in \{0, 1\} \qquad (23)$$

3.1.7. Objective function

The objective is to minimize total energy consumption (denoted as *TEC*), which is classified in four different types of energy consumption. In constraint (24), the first term calculates direct energy consumption, which is equal to the processing time multiplying the unit cutting power, the second term calculates the indirect energy consumption, which is proportional to the makespan, while the third term calculates the standby energy consumption and the switch off and on energy consumption.

$$TEC = \sum_{j} \sum_{i \in \mathbf{I}_j} \sum_{n} \sum_{n \le n' \le n + \Delta n} \left(w_{i,j,n,n'} \cdot \alpha_{i,j} \cdot PC_{i,j} \right) + \beta \cdot MS + \\ + \sum_{j} \sum_{n > 1} \left[ES_{j,n} + EO_j \cdot \left(1 - x_{j,n}\right) \right] \tag{24}$$

We complete our mathematical model **M1**, which consists of constraints 1-23 with the objective function in 24. The model **M1** is a MILP formulation.

3.1.8. Extensions

The model **M1** is not difficult to extend for the case with varying processing times. If the variable processing time is assumed to be linearly dependent on the processing batch size ($b_{i,j,n,n'}$), which is denoted as $\alpha_{i,j,n,n'} + \beta_{i,j} \cdot b_{i,j,n,n'}$, then constraint 11 can change by replacing the term $\alpha_{i,j} \cdot w_{i,j,n,n'}$ with $\alpha_{i,j} \cdot w_{i,j,n,n'} + \beta_{i,j} \cdot b_{i,j,n,n'}$. The following constraints can be added for the batch size.

$$T^{\mathrm{f}}_{j,n} = T^{\mathrm{s}}_{j,n} + \sum_{i \in \mathbf{I}_j} \sum_{n \le n' \le n + \Delta n} \left(\alpha_{i,j} \cdot w_{i,j,n,n'} + \beta_{i,j} \cdot b_{i,j,n,n'} \right) \qquad \forall j, n \tag{11a}$$

$$w_{i,j,n,n'} \cdot B^{\min}_{i,j} \le b_{i,j,n,n'} \le w_{i,j,n,n'} \cdot B^{\max}_{i,j} \qquad \forall j, i \in \mathbf{I}_j, n \le n' \le n + \Delta n \tag{11b}$$

As already discussed, it is assumed that all processing units switch off before they process the first operation and after they finish processing the last operation. However, it can be more energy efficient for a unit to remain idle between two scheduling horizons. Therefore, we can easily to omit such assumption by calculating the time that the last operation of unit *j* in scheduling horizon *H* finishes in comparison to the next scheduling horizon (*H* + 1). We use a parameter T_j^0 to denote the time that a unit remaining idle before the next scheduling horizon starts $|(H\ +\ 1)^{start}|$. After the scheduling problem with a scheduling horizon *H* is solved, then we can calculate the value of T_j^0 as follows,

$T_j^0 = T_{j,n}^{\mathrm{f}} - |(H+1)^{start}|$

$$\forall j,\, n,\ \sum_{j}\sum_{i\in \mathbf{I}_j}\sum_{n\le n'\le n+\Delta n} w_{i,j,n,n'} = 1 \wedge \sum_{j}\sum_{i\in \mathbf{I}_j}\sum_{n''>n}\sum_{n''\le n'''\le n''+\Delta n} w_{i,j,n,n'} = 0 \qquad (25)$$

Note that $T_j^0 < 0$, since it is calculated based on the next scheduling horizon (H + 1). In this case the standby energy consumption for the first event point is calculated as follows

$$ES_{j,n} \le \left(T_{j,n}^{\mathrm{s}} - T_j^0\right) \cdot PU_j \qquad \forall j,\, n = 1 \qquad (26)$$

$$ES_{j,n} \ge \left(T_{j,n}^{\mathrm{s}} - T_j^0\right) \cdot PU_j - M \cdot PU_j \cdot \left(1 - x_{j,n}\right) \qquad \forall j,\, n = 1 \qquad (27)$$

Finally, the objective function is modified to consider the standby and switch off-on energy consumption at the first event point.

$$TEC = \sum_{j}\sum_{i\in \mathbf{I}_j}\sum_{n}\sum_{n\le n'\le n+\Delta n}\left(w_{i,j,n,n'} \cdot \alpha_{i,j} \cdot PC_{i,j}\right) + \beta \cdot MS + $$
$$+ \sum_{j}\sum_{n}\left[ES_{j,n} + EO_j \cdot \left(1 - x_{j,n}\right)\right] \qquad (24a)$$

Finally, if there is a changeover time from a task to another in a unit j, then it can be ensured using the following constraints.

$$T_{j,n+1}^{\mathrm{s}} \ge T_{j,n}^{\mathrm{f}} + \tau_j \cdot \sum_{n+1\le n''\le n+1+\Delta n} w_{i,j,n+1,n''} - H\left(1 - \sum_{n-\Delta n\le n''\le n} w_{i',j,n'',n}\right)$$
$$\forall j, i, i' \in \mathbf{I}_j, i \ne i', n < N, \tau_j > 0 \qquad (28a)$$

where a parameter τ_j denotes the sequence-independent changeover time, which only depends on units.

$$T_{j,n}^{\mathrm{s}} \ge T_{j,n'}^{\mathrm{f}} + \tau_{i',i,j} \cdot \sum_{n\le n''\le n+\Delta n} w_{i,j,n,n''} - H\left(1 - \sum_{n'-\Delta n\le n''\le n'} w_{i',j,n'',n'}\right) -$$
$$-H\left(1 - \sum_{i''}\sum_{n'\le n''\le n}\sum_{n''\le n'''\le n''+\Delta n} w_{i'',j,n'',n'''}\right)$$

$$\forall j, i, i' \in \mathbf{I}_j, i \ne i', n' < n, \tau_{i',i,n} > 0 \qquad (28b)$$

where a parameter $\tau_{i',i,j}$ denotes the sequence-dependent changeover time.

3.2. Local sequence-based formulation (M2a)

In this model, we use the local sequence-based modelling approach where the immediate precedence of two operations processed on a unit is examined. Therefore, we do not use any time interval, slot or event point to divide the scheduling horizon. To denote if an operation l belonging to a job k is processed immediately before another operation l' in a job k' on a unit j, we introduce a binary variable $x_{k',l',k,l,j}$ as follows,

$$x_{k,l,k',l',j} = \begin{cases} 1 & \text{if an operation } l \text{ in a job } k \text{ immidiately precedes operation } l' \text{ in job } k' \text{ in a unit } j \\ 0 & \text{otherwise} \end{cases}$$

Note that one operation can have at most one immediate predecessor as we use the local sequence-based modelling approach.

3.2.1. Allocation constraints

We define two 0-1 continuous variables $w_{k,l,j}$ and $XF_{k,l,j}$ as follows,

$$w_{k,l,j} = \begin{cases} 1 & \text{if an operation that belongs to job } k \text{ is processed in a unit } j \\ 0 & \text{otherwise} \end{cases}$$

$$XF_{k,l,j} = \begin{cases} 1 & \text{if an operation } l \text{ in a job } k \text{ is the first operation that is processed in a unit } j \\ 0 & \text{otherwise} \end{cases}$$

If an operation l is processed in a processing unit j, then it should be either be processed first or be immediately preceded by another operation l'.

$$\sum_{k'} \sum_{\substack{l' \in \mathbf{LKJ}_{k',j} \\ (k \neq k' \vee (k = k' \wedge l \neq l'))}} x_{k',l',k,l,j} + XF_{k,l,j} = w_{k,l,j} \qquad \forall k, j, l \in \mathbf{LKJ}_{k,j} \qquad (29)$$

where $\mathbf{LKJ}_{k,j}$ denotes the operations that a unit j is able to process.

If an operation l is not processed in a unit j, then it should not immediately precede any other operation processed in this unit j.

$$\sum_{k'} \sum_{\substack{l' \in \boldsymbol{LKJ}_{k',j} \\ (k \neq k' \vee (k = k' \wedge l \neq l'))}} x_{k,l,k',l',j} \leq w_{k,l,j} \qquad \forall k, j, l \in \mathbf{LKJ}_{k,j} \qquad (30)$$

At most one operation can be processed first in a unit j during the scheduling horizon.

$$\sum_{k} \sum_{l \in \mathbf{LKJ}_{k,j}} XF_{k,l,j} \leq 1 \qquad \forall j \qquad (31)$$

Constraint (32) is introduced to ensure that all operations are processed exactly once in the scheduling horizon.

$$\sum_{j \in \mathbf{J}_{k,l}} w_{k,l,j} = 1$$

$\forall k,\, l \in \mathbf{L}_k$ (32)

The number of operations that can be processed in a processing unit j should be within the minimum (N_j^{min}) and maximum (N_j^{max}) limits, which are similar to model **M1**.

$$\sum_{k} \sum_{l \in \mathbf{LKJ}_{k,l,j}} w_{k,l,j} \geq N_j^{\min}$$

$\forall j$ (33)

$$\sum_{k} \sum_{l \in \mathbf{LKJ}_{k,l,j}} w_{k,l,j} \leq N_j^{\max}$$

$\forall j$ (34)

To monitor standby and off-on mode of a processing unit during the periods that it does not process any operation, we introduce a binary variable $z_{k,l,j}$ and a 0-1 continuous variable $y_{k,l,j}$ defined below,

$$y_{k,l,j} = \begin{cases} 1 & \text{if a unit } j \text{ remains standby after it processes an operation } l \text{ that belongs to job } k \\ 0 & \text{otherwise} \end{cases}$$

$$z_{k,l,j} = \begin{cases} 1 & \text{if a unit } j \text{ is switched off after it processes an operation } l \text{ that belongs to job } k \\ 0 & \text{otherwise} \end{cases}$$

If a unit j is idle, it can be switched off or remain standby only after it completes processing an operation l that belongs to a job k. Both $y_{k,l,j}$ and $w_{k,l,j}$ should be zero if this unit j does not process an operation l of a job k. To ensure this, we impose the following constraints.

$$y_{k,l,j} + z_{k,l,j} = w_{k,l,j} \qquad \forall k, j, l \in \mathbf{LKJ}_{k,j} \qquad (35)$$

3.2.2. Sequencing constraints

Operations in the same job

To model the timing of an operation that is processed in a unit, we define a positive continuous variable $T_{k,l}$, to denote the start time of an operation l in a job k that is processed in a unit. Note that it is not necessary to define this timing variable based on a specific unit j, as an operation l that in a job k can only be processed once during the scheduling horizon. An operation l that belongs to a job k should start after the previous operation l' ($l' = l - 1$) in the same job finishes.

$$T_{k,l} \geq T_{k,l'} + \sum_{j \in \mathbf{J}_{k,l'}} \left(\alpha_{k,l',j} \cdot w_{k,l',j}\right)$$

$$\forall k,\ l' \in \mathbf{L}_k,\ l \in \mathbf{L}_k,\ l' = l - 1 \qquad (36)$$

Operations in different jobs processed in the same unit

We define a continuous variable $ST_{k,l,j}$ to denote the time of a unit j on standby mode after it completes processing an operation l in a job k, respectively. An operation l in a job k in a unit should start after its direct predecessor l' finishes plus the idle time, as indicated in constraints (37)-(38). Note that if the unit j is in switch off-on mode, then constraint (38) should be relaxed due to a longer idle time required compared to that in the standby mode.

$$T_{k',l'} \geq T_{k,l} + \sum_{j \in \mathbf{J}_{k,l}} \left(\alpha_{k,l,j} \cdot w_{k,l,j} + ST_{k,l,j}\right) - H\left(1 - \sum_{j \in \left(\mathbf{J}_{k,l} \cap \mathbf{J}_{k',l'}\right)} x_{k,l,k',l',j}\right)$$

$$\forall k,\ k',\ k \neq k',\ l \in \mathbf{L}_k,\ l' \in \mathbf{L}_{k'} \qquad (37)$$

$$T_{k',l'} \leq T_{k,l} + \sum_{j \in \mathbf{J}_{k,l}} \left(\alpha_{k,l,j} \cdot w_{k,l,j} + ST_{k,l,j}\right) + H\left(1 - \sum_{j \in \left(\mathbf{J}_{k,l} \cap \mathbf{J}_{k',l'}\right)} x_{k,l,k',l',j}\right) +$$

$$+ H \cdot \sum_{j \in \mathbf{J}_{k,l}} z_{k,l,j}$$

$$\forall k,\ k',\ k \neq k',\ l \in \mathbf{L}_k,\ l' \in \mathbf{L}_{k'} \qquad (38)$$

3.2.3. Standby energy calculation

We define a continuous variable $ES_{k,l}$ to denote the standby energy consumption of a unit after operation l finishes being processed. It is equal to the time of this unit j on standby mode after it processes operation l multiplies the unload power rate (PU_j).

$$ES_{k,l} = \sum_{j \in \mathbf{J}_{k,l}} \left(ST_{k,l,j} \cdot PU_j\right)$$

$$\forall k,\ l \in \mathbf{L}_k \qquad (39)$$

In any case, the standby energy consumption should be less than the energy consumed by a unit j in the switch off-on mode (EO_j). Therefore, we set an upper limit for the time of a unit on standby mode.

$$ST_{k,l,j} \leq min\left(H, \frac{EO_j}{PU_j}\right) \cdot y_{k,l,j}$$

$\forall k, j, l \in \mathbf{LKJ}_{k,j}$ (40)

3.2.4. Makespan calculation

As already discussed, makespan is the earliest time that all tasks have been processed.

$$T_{k,l} + \sum_{j \in \mathbf{J}_{k,l}} \left(w_{k,l,j} \cdot \alpha_{k,l,j} + ST_{k,l,j}\right) \leq MS$$

$\forall k, l \in \mathbf{L}_k$ (41)

3.2.5. Tightening constraints

The processing time of all operations processed in a unit *j* plus the idle time should be less than the makespan.

$$\sum_k \sum_{l \in \mathbf{KLJ}_{k,l,j}} \left(w_{k,l,j} \cdot \alpha_{k,l,j} + ST_{k,l,j}\right) \leq MS$$

$\forall j$ (42)

3.2.6 Additional constraints

Similar to model **M1,** the release time and the due dates of each job should be respected.

$$T_{k,l} \geq r_k \qquad \forall k, l \in (\mathbf{L}_k \cap \mathbf{LR}_k) \qquad (43)$$

$$T_{k,l} + \sum_{j \in \mathbf{J}_{k,l}} \left(w_{k,l,j} \cdot \alpha_{k,l,j} + ST_{k,l,j}\right) \leq d_k$$

$\forall k, l \in (\mathbf{L}_k \cap \mathbf{LD}_k)$ (44)

where $\mathbf{LR}_k$ is the jobs with non-zero release time and $\mathbf{LD}_k$ is the jobs that have a due date. Additionally, constraints (43) and (44) are introduced to avoid violation of forbidden sequencing paths and assignments.

$$w_{k,l,j} + w_{k,l',j'} \leq 1 \qquad \forall k, l, l' \in \mathbf{L}_k, (j, j') \in \mathbf{FP} \qquad (45)$$

$$w_{k,l,j} = 0 \qquad \forall k, l \in \mathbf{L}_k, l \in \mathbf{JP}_{k,j} \qquad (46)$$

where **FP** is the set including the forbidden sequencing paths, $\mathbf{JP}_{k,j}$ is the set including the forbidden assignment.

3.2.7. Objective function

The objective is to minimize the total energy consumption, which is similar to model **M1**.

$$z = \sum_j \sum_k \sum_{l \in \mathbf{LKJ}_{k,j}} \left(w_{k,l,j} \cdot \alpha_{k,l,j} \cdot PC_{k,l,j}\right) + \beta \cdot MS + \sum_k \sum_{l \in \mathbf{L}_k} ES_{k,l} +$$
$$+ \sum_j \sum_k \sum_{l \in \mathbf{LKJ}_{k,j}} \left(EO_j \cdot z_{k,l,j}\right)$$

(47)

Bounds on variables

The start time of an operation l in a job k should be always after the minimum time required for all previous operations in the same job to be processed.

$$T_{k,l} \geq \sum_{\substack{l' < l \\ l' \in \mathbf{L}_k}} \left\{ \min_j \left(\alpha_{k,l',j}\right) \right\}$$

$\forall k, l \in \mathbf{L}_k$ (48)

Finally, (47) and (48) denote all the continuous and binary variables of the model respectively.

$ES_{k,l}, ST_{k,l,j}, T_{k,l} >= 0$ (49)

$0 =< w_{k,l,j}, XF_{k,l,j}, y_{k,l,j} <= 1$ (50)

$z_{k,l,j}, x_{k',l',k,l,j} \in \{0, 1\}$

We complete the local sequence-based formulation denoted as **M2a**, which comprises constraints 29-46, 48-50 with the objective function in constraint 47.

3.2.8 Extension

Similar to model **M1**, model **M2a** can also be extended for the case with the varying processing time. If the variable processing time is assumed to be linearly dependent on the processing batch size ($B_{k,l,j}$), which is denoted as $\alpha_{k,l,j} + \beta_{k,l,j} \cdot B_{k,l,j}$, then constraints 34-36, 42 and 46 can change by replacing the term $\alpha_{k,l,j} \cdot w_{k,l,j}$ with $\alpha_{k,l,j} \cdot w_{k,l,j} + \beta_{k,l,j} \cdot B_{k,l,j}$. The following constraints can be added for the batch size.

$w_{k,l,j} \cdot B_{k,l,j}^{\min} \leq B_{k,l,j} \leq w_{k,l,j} \cdot B_{k,l,j}^{\max}$ $\forall k, j, l \in \mathbf{LKJ}_{k,j}$ (51a,b)

Similar to model **M1**, we can omit assumptions e) and f) by introducing two additional variables y_j^0 and z_j^0 to denote whether the unit is standby or switched off respectively at the beginning of the scheduling horizon. If the unit is in standby mode at the beginning of the scheduling horizon, then the initial standby energy consumption $\left(ES_j^0\right)$ is calculated as follows.

$$ES_j^0 \leq \left(T_{k,l} - T_j^{\mathrm{f0}}\right) \cdot PU_j \qquad \forall k, l \in \mathbf{L}_k, j \tag{52}$$

$$ES_j^0 \geq \left(T_{k,l} - T_j^{\mathrm{f0}}\right) \cdot PU_j - M \cdot \left(2 - XF_{k,l,j} - y_j^0\right) \qquad \forall k, l \in \mathbf{L}_k, j \tag{53}$$

Where $T_j^{f_0}$ the time that the last operation of unit j in scheduling horizon H finishes in comparison to the next scheduling horizon starts $|(H + 1)^{start}|$ and it is calculated as follows.

$$T_j^{f_0} = T_{k,l} + \alpha_{k,l,j} - |(H + 1)^{start}| \qquad \forall j, k, l \in \mathbf{L}_k, \sum_{k'} \sum_{l' \in \mathbf{L}_{k'}} x_{k,l,k',l',j} = 0 \wedge w_{k,l,j} = 1 \tag{54}$$

In this case the objective function is modified as follows.

$$z = \sum_j \sum_k \sum_{l \in \mathbf{LKJ}_{k,j}} \left(w_{k,l,j} \cdot \alpha_{k,l,j} \cdot PC_{k,l,j}\right) + \beta \cdot MS + \sum_k \sum_{l \in \mathbf{L}_k} ES_{k,l} + \\ + \sum_j \sum_k \sum_{l \in \mathbf{LKJ}_{k,j}} \left(EO_j \cdot z_{k,l,j}\right) + \sum_j \left(ES_j^0 + EO_j \cdot z_j^0\right) \tag{55}$$

3.3 Local sequence-based formulation (M2b)

In model **M2b** we also examine the immediate precedence of two operations. The main difference with model **M2a** is that **M2b** does not examine in which unit the two operations are processed. Therefore, we introduce a binary variable $x_{k',l',k,l}$to denote if an operation l belonging to a job k is processed immediately before another operation l' in a job k' as follows,

$$x_{k,l,k',l'} = \begin{cases} 1 & \text{if an operation } l \text{ in a job } k \text{ immidiately precedes operation } l' \text{ in job } k' \\ 0 & \text{otherwise} \end{cases}$$

Since in this local sequence-based model the defined binary variable does not examine the unit that the two operations are processed, constraints (56) and (57) are introduced to ensure that if an operation l immediately precedes another operation l' then both operations are processed in the same unit.

$$w_{k',l',j} \geq w_{k,l,j} + x_{k',l',k,l} + x_{k,l,k',l'} - 1 \quad \forall k, k', j, l \in \mathbf{LKJ}_{k,j}, l' \in \mathbf{LKJ}_{k',j}, l < l' \tag{56}$$

$$w_{k,l,j} \geq w_{k',l',j} + x_{k',l',k,l} + x_{k,l,k',l'} - 1 \quad \forall k, k', j, l \in \mathbf{LKJ}_{k,j}, l' \in \mathbf{LKJ}_{k',j}, l < l' \tag{57}$$

Similar to model **M2a**, if an operation l is processed in a processing unit, then it should be either be processed first or be immediately preceded by another operation l'.

$$\sum_{k'} \sum_{\substack{l' \in \mathbf{L}_{k'} \\ (k \neq k' \vee (k=k' \wedge l \neq l'))}} x_{k',l',k,l} + \sum_{j \in \mathbf{J}_{k,l}} XF_{k,l,j} = \sum_{j \in \mathbf{J}_{k,l}} w_{k,l,j}$$

$$\forall k, l \in \mathbf{L}_k, \tag{58}$$

Additionally, an operation l of job k can be processed first in unit j $(XF_{k,l,j} = 1)$ only if the operation is processed in this unit $(w_{k,l,j} = 1)$

$$XF_{k,l,j} \leq w_{k,l,j}$$

$$\forall k, j, l \in \mathbf{LKJ}_{k,j} \tag{59}$$

For different operations in different units, an operation l' belonging to a job k should start after its predecessor l' finishes plus the idle time.

$$T_{k',l'} \geq T_{k,l} + \sum_{j \in \mathbf{J}_{k,l}} (\alpha_{k,l,j} \cdot w_{k,l,j} + ST_{k,l,j}) - H(1 - x_{k,l,k',l'})$$

$$\forall k, k', k \neq k', l \in \mathbf{L}_k, l' \in \mathbf{L}_{k'} \tag{60}$$

$$T_{k',l'} \leq T_{k,l} + \sum_{j \in \mathbf{J}_{k,l}} (\alpha_{k,l,j} \cdot w_{k,l,j} + ST_{k,l,j}) + H(1 - x_{k,l,k',l'}) + H \cdot \sum_{j \in \mathbf{J}_{k,l}} z_{k,l,j}$$

$$\forall k, k', k \neq k', l \in \mathbf{L}_k, l' \in \mathbf{L}_{k'} \tag{61}$$

Mathematical model **M2b** consists of constraints 31-36, 39-46 and 55-61, with the 47 to be the objective.

4. Enhanced Rolling horizon decomposition approach

The rolling horizon decomposition approach proposed by Lin *et al.* (2002); Janak *et al.* (2004); Li *et al.* (2012) is often used to solve industrial-scale scheduling problems that are difficult to solve directly using the mathematical programming models. The key idea of the decomposition approach is to divide the entire scheduling problem into small-scale subproblems based on job or order due dates. Each subproblem is then solved using the mathematical programming model. However, it cannot directly solve this flexible job-shop scheduling problem due to the same due dates of all jobs. In this work, we develop a grouping strategy to enhance the rolling horizon decomposition algorithm (Lin *et al.* 2002; Janak *et al.* 2004; Li *et al.* 2012) using a mixed-integer linear programming model below in which assigns operations/tasks to several groups.

4.1. Mathematical formulation for grouping

We introduce two binary variables Y_g which is equal to 1 if a group g is selected and $Y_{i,g}$ which is equal to 1 if a task i is assigned to the group g respectively. A task i can be included to a group g only if the group g is selected.

$$Y_{i,g} \leq Y_g \qquad \forall i, g \qquad (62)$$

A task should be assigned to exactly one group.

$$\sum_{g} Y_{i,g} = 1 \qquad \forall i \qquad (63)$$

Furthermore, a task i belonging to a job k can be included in a group g only if the preceding task is included in same group g or in a previous group $g' < g$.

$$Y_{i',g} \leq Y_{i,g} + \sum_{g'<g} Y_{i,g'} \qquad \forall i, i' \in \mathbf{I}_k, i' = i + 1 \qquad (64)$$

If a group g is selected, then it should contain at least one operation/task. Constraint (65) is introduced to ensure such condition.

$$\sum_{i} Y_{i,g} \geq Y_g \qquad \forall g \qquad (65)$$

If a group g is not selected, then the next group $(g + 1)$ cannot be selected either.

$$Y_{g+1} \leq Y_g \qquad \forall g < G \qquad (66)$$

We introduce a continuous variable $TNI_{k,g}$ to denote the number of tasks in a job k that are included in a group g.

$$TNI_{k,g} = \sum_{i \in \mathbf{I}_k} Y_{i,g} \qquad \forall k, g \qquad (67)$$

The number of tasks from job k that are included in a group $(g + 1)$ should be less than the tasks from the same job included in the previous group g. In this case, we sequence the number of tasks of each job included in each group in a decreasing order.

$$TNI_{k,g+1} \leq TNI_{k,g} \qquad \forall k, g < G \qquad (68)$$

The total number of tasks included in a group g is monitored by using a continuous variable TNL_g.

$$TNL_g = \sum_{i} Y_{i,g} \qquad \forall g \qquad (69)$$

In order to avoid subproblems with many tasks that require excessive computational time to generate the optimum solution, we introduce a parameter L^{max} to denote the maximum number of tasks is allowed in a group g. The number of tasks included in each group must not exceed L^{max}.

$$TNL_g \leq L^{\max} \qquad \forall g \qquad (70)$$

Alternatively, we can also limit the model complexity in each group g through using the constraints (72)-(73). In constraint (72), the number of binary variables can be calculated if the number of tasks $\left(\left|\mathbf{I}_g\right|\right)$ and units $(|\mathbf{J}|)$ included in the subproblem as well as the number of event points (EN_g) are known.

$$B_g^{\mathrm{v}} = \left(\left|\mathbf{I}_g\right| \cdot |\mathbf{J}| \cdot \mathbf{I}_j + |\mathbf{J}|\right) \cdot EN_g \qquad \forall g \qquad (71)$$

A parameter $B^{\mathrm{v,max}}$ is introduced to denote the maximum number of binary variables allowed in each group.

$$B_g^{\mathrm{v}} \leq B^{\mathrm{v,max}} \qquad \forall g \qquad (72)$$

where B_g^{v} denotes total number of binary variables in each group.

Finally, we use two penalties $PEN1$ and $PEN2$ in order to minimize the difference in the total number of tasks included in each group g. By introducing such penalties, all groups are enforced to contain the same number of tasks of each job.

$$PEN1 \geq TNI_{k,g} \qquad \forall k, g \qquad (73)$$

$$PEN2 \leq TNI_{k,g} + |G| \cdot \left(1 - Y_g\right) \qquad \forall k, g \qquad (74)$$

The objective of this model is to minimize the number of groups selected. In this way, we minimize the number of subproblems that the main problem is divided.

$$obj = w_1 \cdot \sum_g Y_g + w_2(PEN1 - PEN2) \qquad (75)$$

where w_1 and w_2 are the two importance weight parameters.

For each subproblem, the number of event points required is equal to the maximum number of tasks that a unit j is able to process.

$$EN_g = \max_j\left(\sum_{i \in \mathbf{I}_j} Y_{i,g}\right) \qquad (76)$$

Figure 4 illustrates the improved rolling-horizon decomposition algorithm. In the beginning, the level-1 decomposition model from Lin *et al.* (2002), Janak *et al.* (2004)

and Li *et al.* (2012) determine the sub-horizons and tasks/operations in each sub-horizon based on the due dates of orders. In the next step, the proposed model for grouping in this work further decomposes the sub-horizon problem through the assignment of the operations/tasks in the sub-horizon into multiple groups. Note that for sub-horizon problems with small model complexity, the model for grouping includes all tasks/operations into one group. Operations/tasks that belong to a group are scheduled in the available processing units simultaneously using the short-term scheduling model. After the generation of the optimal schedule for a given group, this schedule is fixed and the time that the processing units are available to process new operations/tasks is calculated for the operations/tasks in the next group. The procedure continues until the approach assigns all operations/tasks in all groups to available processing units. Integer cuts are also introduced to the level-1 model or the proposed grouping model to generate a new combination of integer solutions if the current integration solution is not satisfactory after solving a grouping problem or a sub-horizon problem. Note that the energy consumption is calculated at the start time of the first event point in the current group or subhorizon, which depends on the finish time of each processing unit in the previous subproblems.

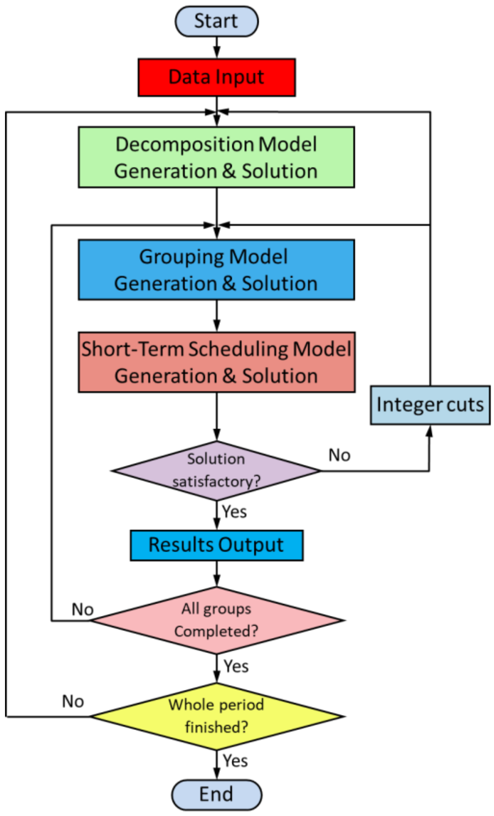

Figure 4 The enhanced rolling horizon decomposition algorithm

5. Hybrid algorithm

The GEP-based algorithm of Zhang *et al.* (2017) can generate several dispatching rules that can efficiently develop *good* feasible solutions even for large-scale examples, as demonstrated in Zhang *et al*. (2017). However, the solution obtained for this energy-efficient scheduling of job-shop problems is often a bit far from the optimal solution. The main reason may lay to the methodology of how the dispatching rules generate the schedule. More specifically, a dispatching rule only decides which is the next operation/task that will take place and in which processing unit is going to be processed (sequencing and allocation). If the operation/task and the processing unit is chosen, then the operation/task is assigned to start at the earliest time possible. Although such an approach can lead to the smallest possible makespan, it often leads to schedules with high standby energy consumption and switch off-on energy consumption. To further demonstrate this issue, let consider an example with ten units and ten jobs , and generate a schedule using the dispatching rule 8 from the GEP-based algorithm of Zhang *et al.* (2017), as illustrated in Figure 5. From Figure 5, it seems that most processing units remain idle for multiple times during the scheduling horizon. For instance, unit J10 does not process any task in five periods (398 min – 420 min, 503 min – 517 min, 604 min – 781 min, 829 min – 952 min, and 974 min –1031 min). While the unit remains in the standby mode in the first two periods, the unit switches off in the remaining periods. The total standby and switch off-on energy consumption for unit J10 is 91.8 kW.

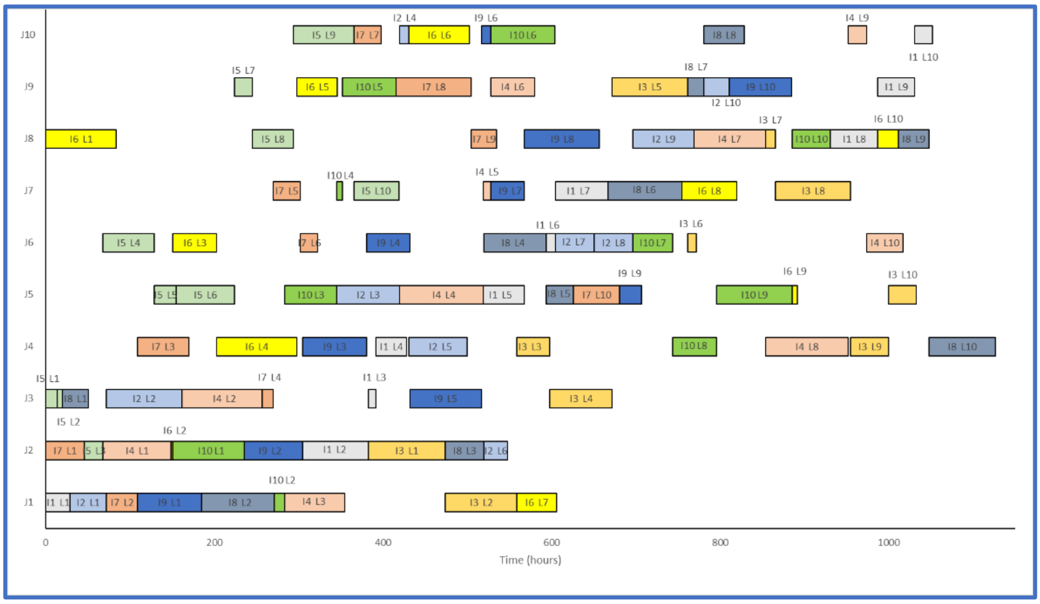

Figure 5 A schedule for the example with 10 units and 10 jobs using Rule 8 from Zhang et al. (2017)

To make the best trade-off between indirect energy consumption, standby energy consumption, and switch off-on energy consumption, we develop a hybrid algorithm through the combination of the eGEP and the mixed-integer linear programming approach. We first use the eGEP algorithm to generate efficient dispatching rules. These dispatching rules determine the allocation of operations/tasks and their sequence on a unit. After this step, the proposed local sequence-based models (i.e., **M2a**) determine the best operation/task timings and the best trade-off between indirect energy consumption and switching off-on energy consumption. Figure 6 illustrates the hybrid algorithm.

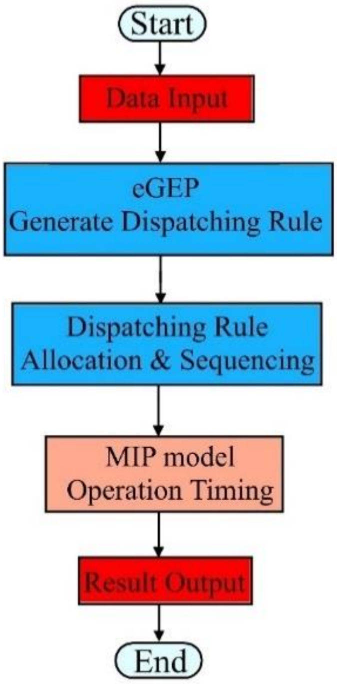

Figure 6 The proposed hybrid algorithm

6. Computational studies

We solve 58 examples from Zhang *et al.* (2017) to illustrate the capability of the proposed models **M1**, **M2a** and **M2b**. Examples 1-20 are small-size examples having from 2 to 3 jobs, and a total of 2 or 3 processing units. Each of those jobs includes from 2 to 3 operations. Examples 21-58 are large-size problems, where each job contains from 5 to 15 operations that can be processed on 5-15 processing units available. We also use the model of Zhang *et al.* (2017) and the best model (i.e., model 2) from Meng *et al.* (2019) to solve all examples for a fair comparison. We also solve the same examples by using the rolling horizon decomposition approaches **RH-M1** and **RH-M2** and the hybrid algorithm **eGEP-M2**. For the hybrid algorithm, we use the five most effective dispatching rules from Zhang *et al.* (2017). Table 1 depicts those dispatching rules. All examples are solved using CPLEX 12/GAMS 24.6.1. on a desktop computer with Intel® Core™ i5-2500 3.3 GHz and 8 GB RAM running Windows 7. The maximum computational time is set as one hour for all examples.

Table 1. Effective dispatching rules (Zhang *et al.* 2017)

ID	Dispatching rule
1	$PC - \frac{NR}{IT \cdot \alpha}$
5	$PC + IT + \frac{\alpha}{NR}$
7	$\frac{\alpha + IT \cdot \alpha}{NR} + PC + \alpha$
8	$2PU \cdot IT + \frac{\alpha}{\sqrt{IT} + NR} + PC$
9	$\frac{2\sqrt{NR} + PC + 2PU}{2NR \cdot \sqrt{NR}} - 2NR + PC$

PC: cutting power. *PU*: unloaded power. α: processing time. *IT*: idle time.

6.1. Small-size problems: Examples 1-20

Tables 2-3 present the computational results for Examples 1-20. From Tables 2-3, it seems that the model of Zhang et al. (2017) leads to significantly larger model sizes than the proposed models **M1**, **M2a** and **M2b** as well as the model of Meng *et al.* (2019). For instance, the model of Zhang *et al.* (2017) has 218 constraints for Example 1, which is 69% more than the model of Meng *et al.* (2019) and models **M1** and **M2b** (218 vs 68) and 74% more than model **M2a** (218 vs 57). It also requires 30 binary variables, which is 30% (30 vs 21), 37% (30 vs 19), 56.7% (30 vs 13) and 56.7% (30 vs 13) more than the number of binary variables from the models of Meng *et al.* (2019), **M1**, **M2a** and **M2b**, respectively. Furthermore, the model of Zhang *et al.* (2017) leads to much worse MILP relaxation than the other models for all examples. As a result, this model requires at least one order of magnitude more computational time, even for examples with three jobs, three operations and three processing units (Examples 16-20). For instance, the model of Zhang *et al.* (2017) requires 2.8 s to generate the optimum solution for Example 20, while the model of Meng *et al.* (2019) and models **M1**, **M2a** and **M2b** require 0.3 s, 0.05 s, 0.02 s, 0.05 s, respectively. In brief, the model of Zhang e*t al*. (2017) is the least efficient among all the models.

Table 2. Computational results for Examples 1-10 from different models

Example	Model	Event points	CPU time (s)	RMILP (kW)	TEC (kW)	Bin. Var.	Cont. Var.	Constr.
Ex1	ZTWW	3	0.19	7.33	63.03	30	45	218
	MZSR	3[a]	0.06	63.03	63.03	21	23	68
	M1	3	0.02	63.03	63.03	19	28	68
	M2a	-	0.02	63.03	63.03	13	30	57
	M2b	-	0.02	63.03	63.03	13	30	68
Ex2	ZTWW	2	0.09	13.98	122.44	20	31	128
	MZSR	3[a]	0.06	122.44	122.44	28	26	85
	M1	2	0.03	120.44	122.44	14	20	51
	M2a	-	0.02	122.44	122.44	18	34	65
	M2b	-	0.02	122.44	122.44	16	34	79
Ex3	ZTWW	4	0.44	7.71	75.74	40	59	324
	MZSR	4[a]	0.08	72.74	75.74	37	29	106
	M1	4	0.02	75.74	75.74	34	28	52
	M2a	-	0.02	75.74	75.74	25	38	72
	M2b	-	0.02	75.74	75.74	19	38	91
Ex4	ZTWW	2	0.05	13.22	146.63	20	31	128
	MZSR	3[a]	0.05	143.63	146.63	28	26	85
	M1	2	0.11	142.63	146.63	14	20	51
	M2a	-	0.02	143.63	146.63	18	34	65
	M2b	-	0.03	143.63	146.63	16	34	79
Ex5	ZTWW	3	0.20	7.90	78.40	30	45	218
	MZSR	4[a]	0.03	75.40	78.40	37	29	106
	M1	3	0.02	78.40	78.40	25	28	79
	M2a	-	0.02	78.40	78.40	25	36	72
	M2b	-	0.02	78.40	78.40	19	36	91
Ex6	ZTWW	3	0.19	24.58	220.74	63	84	596
	MZSR	4[a]	0.05	214.74	220.74	37	35	114
	M1	3	0.02	220.74	220.74	30	44	123
	M2a	-	0.03	220.74	220.74	24	46	94
	M2b	-	0.03	220.74	220.74	24	46	114
Ex7	ZTWW	3	0.20	10.95	97.54	63	84	596
	MZSR	4[a]	0.14	95.51	97.54	37	35	114
	M1	3	0.03	96.51	97.54	30	44	124
	M2a	-	0.02	96.51	97.54	24	46	95
	M2b	-	0.02	96.51	97.54	24	46	115
Ex8	ZTWW	2	0.08	9.99	146.81	42	57	323
	MZSR	5[a]	0.14	145.81	146.81	62	44	173
	M1	2	0.02	137.81	146.81	25	31	91
	M2a	-	0.03	145.81	146.81	43	58	129
	M2b	-	0.03	145.81	146.81	39	58	162
Ex9	ZTWW	3	0.22	16.86	230.66	63	84	596
	MZSR	3[a]	0.03	222.06	230.66	28	32	93
	M1	3	0.03	219.06	230.66	27	44	118
	M2a	-	0.03	222.06	230.66	17	42	79
	M2b	-	0.02	222.06	230.66	17	42	94

Ex10	ZTWW	3	0.19	11.20	161.06	63	84	596
	MZSR	4[a]	0.14	159.23	161.06	44	38	131
	M1	3	0.03	158.06	161.06	30	44	121
	M2a	-	0.03	159.06	161.06	24	46	95
	M2b	-	0.03	159.06	161.06	24	46	115

ZTWW is the model of Zhang et al. (2017 model), MZSR is the model of Meng et al. (2019) model. [a]Maximum number of positions of all units.

Table 3. Computational results for Examples 11-20 from different models

Example	Model	Event points	CPU time (s)	RMILP (kW)	TEC (kW)	Bin. Var.	Cont. Var.	Constr.
Ex11	ZTWW	3	0.20	20.00	166.23	42	57	292
	MZSR	4[a]	0.05	159.23	166.23	46	36	133
	M1	3	0.02	166.23	166.23	28	31	76
	M2a	-	0.03	166.23	166.23	32	46	103
	M2b	-	0.02	166.23	166.23	30	46	131
Ex12	ZTWW	3	0.27	17.00	176.75	42	57	292
	MZSR	6[a]	0.06	174.75	176.75	70	42	187
	M1	3	0.02	176.75	176.75	34	31	84
	M2a	-	0.03	176.75	176.75	52	54	126
	M2b	-	0.03	176.75	176.75	40	54	170
Ex13	ZTWW	5	1.03	13.44	121.30	70	93	608
	MZSR	5[a]	0.06	115.30	121.30	48	36	137
	M1	5	0.02	121.30	121.30	48	49	132
	M2a	-	0.02	121.30	121.30	34	46	107
	M2b	-	0.02	121.30	121.30	32	46	137
Ex14	ZTWW	4	0.15	16.97	156.86	56	75	438
	MZSR	4[a]	0.08	154.86	156.86	30	30	95
	M1	4	0.02	156.86	156.86	30	40	91
	M2a	-	0.03	156.86	156.86	20	38	79
	M2b	-	0.03	156.86	156.86	20	38	99
Ex15	ZTWW	3	0.20	18.00	163.20	42	57	292
	MZSR	4[a]	0.09	160.20	163.20	46	36	133
	M1	3	0.03	163.20	163.20	28	31	70
	M2a	-	0.03	163.20	163.20	32	46	103
	M2b	-	0.02	163.20	163.20	30	46	131
Ex16	ZTWW	5	2.9	19.79	219.46	150	183	1946
	MZSR	7[a]	0.19	212.97	219.46	90	56	244
	M1	5	0.11	218.97	219.46	77	80	274
	M2a	-	0.09	218.97	219.46	67	72	195
	M2b	-	0.03	218.97	219.46	65	72	254
Ex17	ZTWW	4	1.5	26.72	306.68	120	147	1340
	MZSR	4[a]	0.06	294.68	306.68	51	47	157
	M1	4	0.05	302.68	306.68	49	65	186
	M2a	-	0.03	302.68	306.68	34	60	128
	M2b	-	0.03	302.68	306.68	34	60	160

Ex18	ZTWW	4	1.5	14.74	210.60	120	147	1340
	MZSR	6[a]	0.17	207.60	210.60	84	66	232
	M1	4	0.05	203.60	210.60	61	65	206
	M2a	-	0.05	207.60	210.60	61	72	184
	M2b	-	0.05	207.60	210.60	59	72	237
Ex19	ZTWW	4	2.5	17.58	269.52	120	147	1340
	MZSR	6[a]	0.14	264.52	269.52	104	62	278
	M1	4	0.02	269.52	269.52	69	65	217
	M2a	-	0.06	269.52	269.52	77	80	199
	M2b	-	0.03	269.52	269.52	65	80	264
Ex20	ZTWW	6	2.8	25.70	274.94	180	219	2660
	MZSR	6[a]	0.30	260.94	274.94	84	56	232
	M1	6	0.05	274.94	274.94	93	95	315
	M2a	-	0.02	274.94	274.94	61	72	183
	M2b	-	0.05	274.94	274.94	59	72	236

ZTWW is the model of Zhang et al. (2017 model), MZSR is the model of Meng et al. (2019) model. [a] Maximum number of positions of all units.

We also compare the performance of the models **M1**, **M2a** and **M2b** with the model of Meng *et al.* (2019). From Tables 2-3, all these models can efficiently solve all small examples in less than one second. Model **M1** requires a smaller number of binary variables than the model of Meng *et al.* (2019) for Examples 1-20. For instance, the model of Meng *et al.* (2019) requires 90 binary variables to generate the optimal solution for Example 16, while model **M1** requires 77 only. Only for Example 20, the model **M1** requires more number of binary variables than the model of Meng *et al.* (2019) (93 vs 84). Models **M2a** and **M2b** lead to fewer binary variables than the model of Meng *et al.* (2019). Between **M1** and models **M2a** and **M2b** there is not a clear trend on which model requires fewer binary variables to generate the optimal solution. For instance, in Example 15 model **M1** requires fewer binary variables than **M2a** (28 vs 40) and **M2b** (28 vs 30), whilst in Example 17 it requires more binary variables than **M2a** and **M2b** (49 vs 34). For continuous variables and constraints, there is not a clear trend on which model requires the least either. As a result, it is not clear which of these three models is the most efficient by solving such small-scale examples. Despite that, all proposed models lead to slightly smaller model sizes for most cases, which can make them potentially more efficient than the model of Meng *et al.* (2019). The optimal schedule for Example 1 generated by model **M1** is depicted in Figure 7. From this schedule, we can observe that unit J1 switches off after task I1 finishes (4 h) and switched, on right before the time that task I3 starts (12 h). The switch off-on energy consumption, in this case, is 3.6 kW. If the unit J1 remains standby from 4 h to 12 h, then the standby energy consumption would be

11.1 kW. Therefore, considering switching off and on unit J1 during the period that it does not process any tasks, it leads to 68% energy savings. This example illustrates the benefit of switching off-on units that does not process any operation/task for long periods.

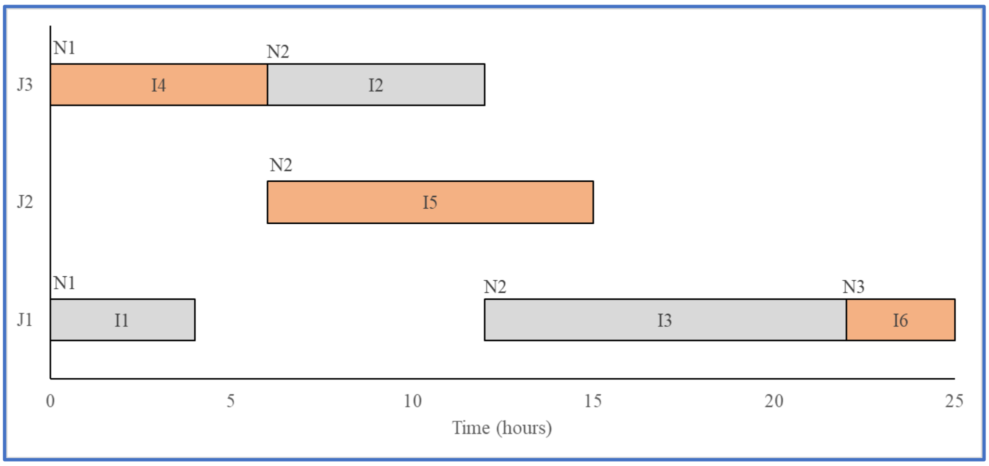

Figure 7 Optimal schedule for Example 1 from the model **M1**

6.2. Large-size problems: Examples 21-51

The computational results for Examples 21-51 are presented in Tables 4-5. From Tables 4-5, it is clear that the model of Zhang *et al.* (2017) is the least efficient as it can only generate a feasible solution for Example 21 after one hour. The main reason is that this model both leads to significantly larger model sizes and worse MILP relaxation compared to the other models. Similarly, the model of Meng *et al.* (2019) can only generate a feasible solution for Examples 21 and 24-28 after one hour. On the other hand, the proposed model **M1** and **M2a** and **M2b** can provide solutions for significantly more examples. More specifically, model **M2a** can generate a feasible solution in 19 out of the 38 examined examples (i.e., Examples 21-28, 30-34, 36, 38-43), while model **M2b** generates a solution for 16 out of 38 examined examples (i.e., Examples 21, 22, 24-33,39,41-43). Model **M1** can successfully solve 31 out of 38 tested examples (Examples 21-51). Therefore, the proposed model **M1** is the most general and efficient model among all examined models.

By comparing the performance of models **M1** and the models **M2a** and **M2b**, it still seems that it is not clear which of the proposed models requires the least number of binary variables. For instance, model **M1** requires more binary variables than **M2a** and **M2b** for Example 21 (426 vs 298 and 426 vs 296 respectively), but less binary variables for Example 25 (645 vs 734 and 645 vs 724 respectively). On the other hand, models **M2a** require a significantly smaller number of continuous variables and constraints. For instance, model **M2a** requires 59% and 30% less continuous variables and constraints for the same example than model **M1** (496 vs 1207 continuous variables and 3374 vs 4796 constraints) for Example 31. Model **M2b** requires the same number of variables than **M2a** but a significantly larger number of constraints, and as a result, is less robust. Among the three models, **M1** can solve more examples than model **M2**. For instance, model **M1** can generate schedules for Examples 35, 37 and 44-51, while models **M2a** and **M2b** fail to provide a feasible solution. Therefore, it is concluded that **M1** is more robust than models **M2a** and **M2b**. It should be noted, though, that models **M2a** and **M2b** can generate better solutions than model **M1** in some cases. For instance, model **M2a** can provide the significantly better solution of 3409.61 kW for Example 22 in comparison to the result of 3674.04 kW generated from model **M1**. Even though the proposed models **M1**, **M2a** and **M2b** are more efficient than the existing models, it seems that they still fail to generate the optimal solution within one hour for all examples. Additionally, for some cases, where more than ten jobs and fifteen operations have to be processed (Examples 52-58), none of the proposed models can generate a feasible solution after one hour. For other examples (Examples 44-51) the result from model **M1** seems to be far from the optimum, since the relative gap after one hour is up to 40 % as depicted in Table 5.

Table 4. Computational results for Examples 21-30 from different models

Ex	Model	Event points	CPU time (s)	RMILP (kW)	TEC (kW)	Bin. Var.	Cont. Var.	Constr.	Gap (%)
Ex21	ZTWW	6	3600	40.00	252.93	1332	1407	43244	84.2
		7	3600	39.76	189.90	1554	1641	58010	78.9
	MZSR	8[a]	3600	129.50	210.96	376	194	981	36.4
	M1	9	3600	123.65	182.49	426	440	1567	32.1
	M2a	-	3600	129.50	182.49	298	242	809	27.5
	M2b	-	3600	129.50	191.81	296	242	1095	28.2
Ex22	M1	14	3600	2729.60	3674.04	1712	1692	6499	25.6
	M2a	-	3600	2775.24	3409.61	1287	654	3137	18.6
	M2b	-	3600	2775.24	3743.42	1285	654	4398	25.9
Ex23	M1	23	3600	3115.90	3497.00	2709	2192	8841	10.6
	M2a	-	3600	3115.90	3719.05	2573	654	5649	16.9
Ex24	M1	11	1526	1776.14	1776.14	710	612	2405	0.0
	M2a	-	3600	1776.14	1792.95	937	342	1731	0.9
	M2b	-	3600	1776.14	1897.10	720	342	2433	6.4
Ex25	MZSR	15[a]	3600	1455.95	1715.22	853	277	2032	22.6
	M1	10	3600	1558.32	1789.95	645	557	2191	5.1
	M2a	-	3600	1710.62	1840.23	734	342	1739	7.0
	M2b	-	3600	1710.62	1846.45	724	342	2453	7.4
Ex26	MZSR	14[a]	3600	1304.53	1876.93	849	277	2024	22.4

	M1	11	3600	1710.62	1783.95	710	612	2405	3.0
	M2a	-	3600	1470.48	1624.92	699	338	1675	9.5
	M2b	-	3600	1470.48	1671.46	693	338	2356	12.0
Ex27	MZSR	13[a]	3600	1423.21	1789.42	814	274	1951	20.5
	M1	10	3600	1579.21	1684.29	635	557	2201	5.6
	M2a	-	3600	1579.21	1719.91	701	338	1671	8.2
	M2b	-	3600	1579.21	1687.79	691	338	2354	6.4
Ex28	MZSR	14[a]	3600	1336.72	1595.13	845	277	2016	16.2
	M1	11	3600	1460.73	1465.37	710	612	2427	0.2
	M2a	-	3600	1460.13	1528.72	730	342	1739	4.4
	M2b	-	3600	1460.13	1493.72	724	342	2449	2.2
Ex29	M1	17	336.1	2582.66	2583.71	1542	1282	5125	0.0
	M2b	-	3600	2582.66	2748.69	1476	496	4834	6.0
Ex30	M1	18	3600	2350.70	2388.63	1633	1357	5350	1.3
	M2a	-	3600	2350.70	2576.18	1492	496	3378	8.7
	M2b	-	3600	2350.70	2548.29	1480	496	4848	7.8

[a] Maximum number of positions of all units.

Table 5. Computational results for Examples 31-58 from different models

Example	Model	Event points	CPU time (s)	RMILP (kW)	TEC (kW)	Bin. Var.	Cont. Var.	Constr.	GAP (%)
Ex31	M1	16	3322	2456.82	2486.18	1451	1207	4796	0.0
	M2a	-	3600	2456.82	2814.41	1494	496	3374	12.7
	M2b	-	3600	2456.82	2826.72	1478	496	4846	13.1
Ex32	M1	20	3600	2606.72	2637.50	1815	1507	6016	0.3
	M2a	-	3600	2606.72	2807.47	1490	496	3366	7.1
	M2b	-	3600	2606.72	2893.30	1474	496	4834	9.9
Ex33	M1	18	3600	2518.27	2523.77	1651	1357	5476	<0.1
	M2a	-	3600	2518.12	2707.47	1537	500	3470	7.0
	M2b	-	3600	2518.12	3026.89	1525	500	4983	16.8
Ex34	M1	21	3600	3297.15	3420.67	2473	2002	8144	3.2
	M2a	-	3600	3297.15	4078.88	2567	654	5653	19.1
Ex35	M1	21	3600	3002.11	3035.98	2494	2002	8207	0.4
Ex36	M1	21	3600	3178.89	3196.92	2494	2002	8081	< 0.1
	M2a	-	3600	3610.51	3674.82	2624	658	5749	14.0
Ex37	M1	21	3600	3393.39	3477.73	2473	2002	8060	2.2
Ex38	M1	23	3600	3378.11	3459.03	2709	2192	8910	2.3
	M2a	-	3600	3378.11	3842.27	2567	654	5637	12.1
Ex39	M1	13	3600	2784.29	4041.88	1589	1572	5951	31.0
	M2a	-	3600	2880.97	3833.61	1266	654	3133	24.8
	M2b	-	3600	2880.97	3727.81	1260	650	4327	22.7
Ex40	M1	12	3600	2512.49	3648.90	1466	1452	5585	31.1
	M2a	-	3600	2590.43	3603.47	1285	654	3129	24.0
Ex41	M1	12	3600	2717.94	3589.61	1466	1452	5525	24.3
	M2a	-	3600	2757.94	3574.34	1289	654	3145	22.9
	M2b	-	3600	2757.94	3630.29	1289	654	4408	24.0

Ex42	M1	11	3600	2857.78	3703.47	1343	1332	5084	22.8
	M2a	-	3600	2857.78	3668.62	1285	654	3137	22.1
	M2b	-	3600	2857.78	3698.93	1285	654	4396	22.7
Ex43	M1	14	3600	2914.72	3782.58	1712	1692	6499	22.9
	M2a	-	3600	3004.51	3881.38	1285	654	3125	22.6
	M2b	-	3600	3004.51	3720.48	1279	654	4384	19.2
Ex44	M1	18	3600	4251.69	5374.11	3284	2982	12067	20.8
Ex45	M1	18	3600	3831.82	5195.68	3284	2982	12013	26.1
Ex46	M1	17	3600	4246.91	5501.46	3067	2817	11354	22.8
Ex47	M1	15	3600	4078.29	5916.36	2750	2487	10021	31.0
Ex48	M1	17	3600	3885.07	6704.23	2934	2652	11439	42.1
Ex49	M1	21	3600	5459.23	9654.12	4967	4422	17872	43.3
Ex50	M1	20	3600	5719.35	9953.75	4730	4212	17041	42.4
Ex51	M1	21	3600	5598.36	9603.38	4946	4422	17809	41.5

Figure 8 depicts the best schedule for Example 24 from model **M1**. From this schedule, no unit that remains idle during the scheduling horizon. Therefore, there is no standby energy or switch off-on energy consumed.

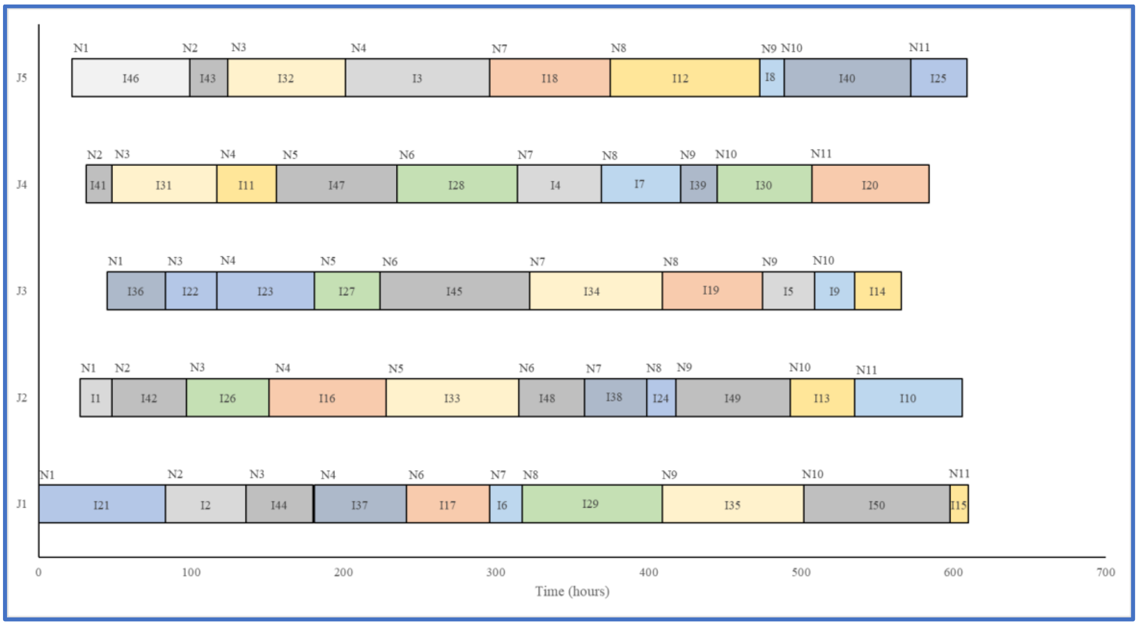

Figure 8 Best schedule obtained for Example 24 using model **M1**

We also compare the results for Examples 21-58 from model **M1** and the eGEP algorithm of Zhang *et al*. (2017). These comparative results are provided in Table 6. From Table 6, it seems that model **M1** can generate better solutions for examples with up to ten jobs and fifteen operations (Examples 21-46) than the eGEP algorithm by up to 26.9%. For instance, model **M1** can generate a schedule with TEC of 2523.77 kW for Example 33, which is approximately 20% less than the TEC of the solution provided using the eGEP algorithm (3036.47 kW). However, the eGEP algorithm can generate a better solution for examples with more than ten operations and ten jobs (Examples 47-58). For instance, the eGEP can generate a schedule with TEC of 7351.24 kW for Example 49, which contains ten jobs and twenty operations, while model **M1** provides a solution with 23.9 % more TEC (9654.12 kW). More interestingly, those dispatching rules, created using eGEP are even able to generate solutions for Examples 52-58, in contrast to model **M1** where it fails to develop a feasible solution for those examples within one hour.

Table 6 Comparative results for Examples 21-58 from model **M1** and **eGEP**

	eGEP	**M1**	Diff		**eGEP**	**M1**	Diff
Ex.	TEC (kW)	TEC (kW)	(%)	Ex.	TEC (kW)	TEC (kW)	(%)
Ex21	296.71	182.49	-38.5	Ex40	4082.95	3648.90	-10.6
Ex22	4047.03	3674.04	-9.2	Ex41	4059.53	3589.61	-11.6
Ex23	3924.75	3497.00	-10.9	Ex42	3937.63	3703.47	-5.9
Ex24	1914.55	1776.14	-7.2	Ex43	4311.92	3782.58	-12.3
Ex25	1975.75	1789.95	-9.4	Ex44	5708.72	5374.11	-5.9
Ex26	1964.55	1783.95	-9.2	Ex45	5756.06	5195.68	-9.7
Ex27	1939.76	1684.29	-13.2	Ex46	5987.00	5501.46	-8.1
Ex28	1859.41	1465.37	-21.2	Ex47	5763.51	5916.36	2.7
Ex29	2891.37	2583.71	-10.6	Ex48	6640.15	6704.23	1.0
Ex30	2761.52	2388.63	-13.5	Ex49	7351.24	9654.12	31.3
Ex31	2765.72	2486.18	-10.1	Ex50	7859.20	9953.75	26.7
Ex32	3046.21	2637.50	-13.4	Ex51	7173.32	9603.38	33.9
Ex33	3036.47	2523.77	-16.9	Ex52	7285.72	-	-
Ex34	3947.29	3365.35	-14.7	Ex53	7284.41	-	-
Ex35	3700.86	3035.98	-18.0	Ex54	10036.36	-	-
Ex36	3682.76	3196.92	-13.2	Ex55	10667.02	-	-
Ex37	3796.85	3477.73	-8.4	Ex56	10183.47	-	-
Ex38	3740.39	3459.03	-7.5	Ex57	9865.68	-	-
Ex39	4480.15	4041.88	-9.8	Ex58	10751.25	-	-

6.3. Computational results from the enhanced rolling horizon decomposition algorithm

We use the enhanced rolling horizon decomposition algorithm to solve all Examples 1-58. The important weight parameters are set to $w_1 = 0.8$ and $w_2 = 0.2$. The maximum number of tasks, allowed to be included for each group is ten for Examples 1-20 and thirty for Examples 21-58. The computational limit for solving each group subproblem is 100 s in the rolling-horizon decomposition algorithms **RH-M1**, **RH-M2**. Tables 8-10 presents the computational results from the enhanced rolling horizon decomposition algorithms **RH-M1** and **RH-M2** for Examples 1-58.

The comparative results of model **M1**, **M2**, **RH-M1**, **RH-M2** and **eGEP** dispatching rule 1 are also presented in Tables 7-8. From Table 7, it seems that both **RH-**

M1 and **RH-M2** generates the same optimal solutions as models **M1** and **M2** for Examples 1-20 due to only one group required in the improved rolling horizon decomposition, which is equivalent to directly solving models **M1** and **M2**. From Table 8, it is observed that **RH-M1** and **RH-M2** obtain worse solutions for most examples in Examples 21-44 compared to model **M1** within 1 h. These worse solutions is mainly due to no or very less units remaining idle during the scheduling horizon in the best schedule from model **M1** compared to that from **RH-M1**, as depicted in Figures 8 and 9. However, the computational time from **RH-M1** and **RH-M2** is significantly reduced by 88.5%-99.9%. More importantly, both rolling horizon decomposition algorithms generate better solutions for Examples 45-51 by up to 31.1% in less computational time compared to model **M1**. Additionally, both decompositions approaches provide *good* feasible solutions for Examples 52-58, whilst model **M1** fails to solve them.

Table 7. Computational results for Examples 1-20 from model **M1**, **M2a**, **RH-M1** and **eGEP** dispatching rule 1

	eGEP	**M1/M2a**	**RH-M1**		**RH-M2**		**Diff (%)**			
Ex	TEC (kW)	TEC (kW)	TEC (kW)	Time (s)	TEC (kw)	Time (s)	**RH-M1** vs. **M1**	**RH-M1** vs **eGEP**	**RH-M2** vs. **M1**	**RH-M2** vs. **eGEP**
Ex1	63.03	63.03	63.03	0.03	63.03	0.11	0.0	0.0	0.0	0.0
Ex2	138.16	122.44	122.44	0.03	122.44	0.14	0.0	-11.4	0.0	-11.4
Ex3	120.76	75.74	75.74	0.03	75.74	0.09	0.0	-37.3	0.0	-37.3
Ex4	161.73	146.63	146.63	0.03	146.63	0.20	0.0	-9.3	0.0	-9.3
Ex5	101.01	78.40	78.40	0.03	78.40	0.20	0.0	-22.4	0.0	-22.4
Ex6	279.84	220.74	220.74	0.02	220.74	0.17	0.0	-21.1	0.0	-21.1
Ex7	107.69	97.54	97.54	0.05	97.54	0.20	0.0	-9.4	0.0	-9.4
Ex8	205.32	146.81	146.81	0.08	146.81	0.14	0.0	-28.5	0.0	-28.5
Ex9	233.66	230.66	230.66	0.03	230.66	0.09	0.0	-1.3	0.0	-1.3
Ex10	191.68	161.06	161.06	0.05	161.06	0.13	0.0	-16.0	0.0	-16.0
Ex11	166.23	166.23	166.23	0.03	166.23	0.20	0.0	0.0	0.0	0.0
Ex12	176.75	176.75	176.75	0.03	176.75	0.19	0.0	0.0	0.0	0.0
Ex13	121.3	121.30	121.30	0.02	121.30	0.20	0.0	0.0	0.0	0.0
Ex14	156.86	156.86	156.86	0.03	156.86	0.11	0.0	0.0	0.0	0.0
Ex15	191.83	163.20	163.20	0.02	163.20	0.14	0.0	-14.9	0.0	-14.9
Ex16	297.59	219.46	219.46	2.30	219.46	0.16	0.0	-26.3	0.0	-26.3
Ex17	329.48	306.68	306.68	0.06	306.68	0.27	0.0	-6.9	0.0	-6.9
Ex18	284.72	210.60	210.60	0.30	210.60	0.22	0.0	-26.0	0.0	-26.0
Ex19	283.83	269.52	269.52	0.03	269.52	0.17	0.0	-5.0	0.0	-5.0
Ex20	335.58	274.94	274.94	0.05	274.94	0.23	0.0	-18.1	0.0	-18.1

We set the maximum computational time of 5 and 10 minutes for solving Examples 21-40 and 41-58 using model **M1** respectively, which are similar to that required by **RH-**

M1, the best solutions obtained from model **M1** are reported in the column denoted as **M1*** in Table 8. Decomposition algorithms can generate better solutions than model **M1*** for most examples in Examples 21-58. The energy consumption is reduced by up to 41.4%.

From Table 7, it seems that **RH-M1** and **RH-M2** cam generate the same or better solutions than **eGEP** for Examples 1-20. The reduction in energy consumption can reach up to 37.3%. Furthermore, both **RH-M1** and **RH-M2** can generate better solutions than **eGEP** for most Examples 21-58 (36 out of 38 examples for **RH-M1**, and 37 out of 38 examples for **RH-M2**). The improvement can be up to 35.3%. The comparative results of **RH-M1**, **RH-M2** and other **eGEP** dispatching rules are presented in Tables S1-S8 in **Supplementary material**. **RH-M1** and **RH-M2** can generate the same or better solutions for Examples 1-20 than **eGEP** dispatching rules 5, 7, 8 and by up to 21.1%, 21.3%, 31.6% and 37.0%, respectively. Furthermore, **RH-M1** and **RH-M2** can generate better solutions than **eGEP** dispatching rule 5 by up to 27.9% and 27.6% for Examples 21-22 and 24-58, dispatching rule 7 by up to 43.1% and 42.9% for Examples 21-37 and 39-58, dispatching rule 8 by up to 28.2% and 27.2% for Examples 21-22, 24-37 and 39-58, and dispatching rule 9 by up to 32.9% and 32.6% for Examples 21-37 and 39-58.

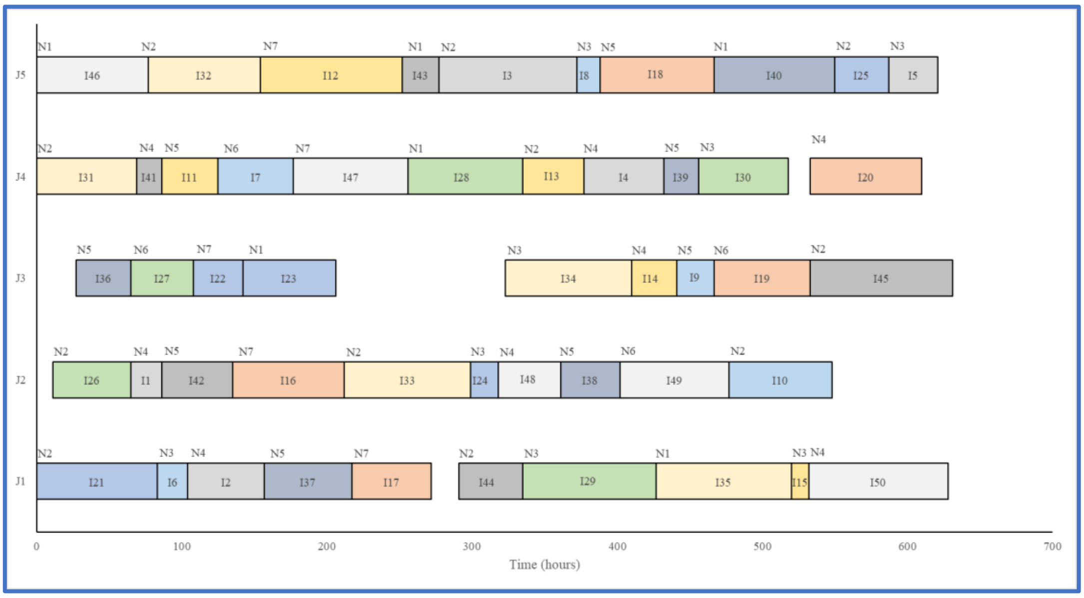

Figure 9 Best schedule for Example 24 using **RH-M1**

The schedule for Example 24 from **RH-M1** is depicted in Figure 9. From Figure 9, it seems that all operations are assigned to three small groups, as provided in Table 10. Comparing the schedule generated by **RH-M1** with those by **M1** (see Figure 8), we notice that the proposed methodology units J1, J3 and J4 remain idle once during the scheduling horizon. On the contrary, no unit remains idle during the scheduling horizon in Figure 8. As a result, a slightly worse solution is generated from **RH-M1** (1847.84 kW vs. 1776.14 kW).

Table 8. Computational results for Examples 21-58 from model **M1**, **M1***, **RH-M1**and **eGEP** dispatching rule 1

	eGEP	**M1**	**M1***	**RH-M1**		**Diff (%)**		
Ex	TEC (kW)	TEC (kW)	TEC (kW)	TEC (kW)	Time (s)	**RH-M1** vs. **M1**	**RH-M1** vs. **eGEP**	**RH-M1** vs. **M1***
Ex21	296.71	182.49	191.81	214.78	44.2	17.7	-35.3	12.0
Ex22	4047.03	3674.04	4563.69	4091.73	157.9	11.4	-13.3	-16.5
Ex23	3924.75	3497.00	4317.50	4192.16	1.0	19.9	6.8	-2.9
Ex24	1914.55	1776.14	1877.71	1847.84	100.1	4.0	-17.7	-1.3
Ex25	1975.75	1789.95	1953.76	1928.84	100.1	7.8	-17.2	-2.5
Ex26	1964.55	1783.95	1900.86	1712.62	101.2	-4.0	-22.5	-15.7
Ex27	1939.76	1684.29	1696.54	1914.79	16.7	13.7	-13.5	3.2
Ex28	1859.41	1465.37	1473.37	1644.57	1.4	12.2	-16.2	5.7
Ex29	2891.37	2583.71	2680.31	2851.87	0.1	10.4	-6.5	0.8
Ex30	2761.52	2388.63	2607.22	2820.98	0.2	18.1	-12.1	-0.6
Ex31	2765.72	2486.18	2544.93	2830.43	0.3	13.8	-14.9	7.4
Ex32	3046.21	2637.50	2684.47	2821.51	0.2	7.0	-9.2	4.9
Ex33	3036.47	2523.77	2974.33	2910.03	0.4	15.3	-17.8	-7.7
Ex34	3947.29	3365.35	4451.29	3632.30	0.6	7.9	-19.1	-18.4
Ex35	3700.86	3035.98	3414.20	3406.47	1.3	12.2	-13.8	-0.2
Ex36	3682.76	3196.92	3837.11	3272.25	0.2	2.4	-16.7	-14.7
Ex37	3796.85	3477.73	4295.25	3636.68	0.8	4.6	-12.8	-15.3
Ex38	3740.39	3459.03	3845.46	3987.97	0.9	15.3	3.5	3.7
Ex39	4480.15	4041.88	4928.97	4278.56	217.4	5.9	-18.8	-20.3
Ex40	4082.95	3648.90	4591.14	3608.97	129.8	-1.1	-26.4	-23.4
Ex41	4059.53	3589.61	3670.20	4023.78	202.2	12.1	-16.3	2.7
Ex42	3937.63	3703.47	4096.44	4039.19	330.4	9.1	-12.2	-7.0
Ex43	4311.92	3782.58	4257.14	4339.28	302.3	14.7	-19.6	-8.8
Ex44	5708.72	5374.11	9223.46	5984.42	0.8	11.4	-5.3	-41.4
Ex45	5756.06	5195.68	6311.21	5466.79	1.5	5.2	-15.0	-22.5
Ex46	5987.00	5501.46	-	5858.67	1.7	6.5	-13.3	-
Ex47	5763.51	5916.36	-	5578.43	0.6	-5.7	-12.8	-
Ex48	6640.15	6704.23	7404.23	6178.14	1.0	-7.8	-23.1	-31.0
Ex49	7351.24	9654.12	-	6946.16	1.6	-28.0	-5.5	12.0
Ex50	7859.20	9953.75	-	7434.35	1.3	-25.3	-5.4	-16.5
Ex51	7173.32	9603.38	-	6866.13	1.2	-28.5	-4.3	-2.9
Ex52	7285.72	-	-	7257.84	1.0	-	-0.4	-
Ex53	7284.41	-	-	7200.05	1.1	-	-1.2	-
Ex54	10036.36	-	-	8698.35	3.5	-	-13.3	-
Ex55	10667.02	-	-	9580.33	4.9	-	-10.2	-
Ex56	10183.47	-	-	8834.19	3.2	-	-13.2	-
Ex57	9865.68	-	-	8958.33	4.5	-	-9.2	-
Ex58	10751.25	-	-	9775.46	4.8	-	-9.1	-

Table 9. Computational results for Examples 21-58 from model **M1**, **M1***, **RH-M2** and **eGEP** dispatching rule 1

Ex	**eGEP** TEC (kW)	**M1** TEC (kW)	**M1*** TEC (kW)	**RH-M2** TEC (kW)	**RH-M2** Time (s)	**Diff (%)** **RH-M2** vs. **M1**	**RH-M2** vs. **eGEP**	**RH-M2** vs. **M1***
Ex21	296.71	182.49	191.81	215.87	2.6	18.3	-27.2	12.5
Ex22	4047.03	3674.04	4563.69	3679.99	112.8	0.2	-9.1	-19.4
Ex23	3924.75	3497.00	4317.50	3808.34	288.1	8.9	-3.0	-11.8
Ex24	1914.55	1776.14	1877.71	1907.12	5.7	7.4	-0.4	1.6
Ex25	1975.75	1789.95	1953.76	1941.73	163.3	8.5	-1.7	-0.6
Ex26	1964.55	1783.95	1900.86	1633.59	170.6	-8.4	-16.8	-14.1
Ex27	1939.76	1684.29	1696.54	1763.91	138.5	4.7	-9.1	4.0
Ex28	1859.41	1465.37	1473.37	1598	134.5	9.1	-14.1	8.5
Ex29	2891.37	2583.71	2680.31	2718.8	108.9	5.2	-6.0	1.4
Ex30	2761.52	2388.63	2607.22	2521.31	200.8	5.6	-8.7	-3.3
Ex31	2765.72	2486.18	2544.93	2645.48	101.1	6.4	-4.3	4.0
Ex32	3046.21	2637.50	2684.47	2685.13	51.2	1.8	-11.9	0.0
Ex33	3036.47	2523.77	2974.33	2657.98	106.4	5.3	-12.5	-10.6
Ex34	3947.29	3365.35	4451.29	3565.38	211.9	5.9	-9.7	-19.9
Ex35	3700.86	3035.98	3414.20	3523.2	262.4	16.0	-4.8	3.2
Ex36	3682.76	3196.92	3837.11	3417.27	205.2	6.9	-7.2	-10.9
Ex37	3796.85	3477.73	4295.25	3716.38	189.2	6.9	-2.1	-13.5
Ex38	3740.39	3459.03	3845.46	3787.02	204.0	9.5	1.2	-1.5
Ex39	4480.15	4041.88	4928.97	3884.04	17.2	-3.9	-13.3	-21.2
Ex40	4082.95	3648.90	4591.14	3562.7	10.2	-2.4	-12.7	-22.4
Ex41	4059.53	3589.61	3670.20	3754.31	0.9	4.6	-7.5	2.3
Ex42	3937.63	3703.47	4096.44	3717.92	3.7	0.4	-5.6	-9.2
Ex43	4311.92	3782.58	4257.14	3978.46	7.2	5.2	-7.7	-6.5
Ex44	5708.72	5374.11	9223.46	5475.11	210.4	1.9	-4.1	-40.6
Ex45	5756.06	5195.68	6311.21	4941.6	216.6	-4.9	-14.1	-21.7
Ex46	5987.00	5501.46	-	5185.61	55.5	-5.7	-13.4	-
Ex47	5763.51	5916.36	-	5257.31	253.9	-11.1	-8.8	
Ex48	6640.15	6704.23	7404.23	5527.5	233.6	-17.6	-16.8	-25.3
Ex49	7351.24	9654.12	-	6951.96	4.4	-28.0	-5.4	-28.0
Ex50	7859.20	9953.75	-	7560.55	105.5	-24.0	-3.8	-24.0
Ex51	7173.32	9603.38	-	6620.26	0.9	-31.1	-7.7	-31.1
Ex52	7285.72	-	-	7060.59	106.4	-	-3.1	-
Ex53	7284.41	-	-	6938.57	2.3	-	-4.7	-
Ex54	10036.36	-	-	9167.93	374.8	-	-8.7	-
Ex55	10667.02	-	-	9708.14	509.6	-	-9.0	-
Ex56	10183.47	-	-	8861.43	394.8	-	-13.0	-
Ex57	9865.68	-	-	9610.23	551.4	-	-2.6	-
Ex58	10751.25	-	-	10003.95	494.7	-	-7.0	-

Table 10. Tasks belonging to each job included in each group

Group	Tasks
G1	K1: I1-I2, K2: I6-I7, K3: I11-I12, K4: I16-I17, K5: I21-I22, K6: I26-I27, K7: I31-I32, K8: I36-I37, K9: I41-I42, K10: I46-I47
G2	K1: I3-I4, K2: I8-I9, K3: I13-I14, K4: I18-I19, K5: I23-I24, K6: I28-I29, K7: I33-I34, K8: I38-I39, K9: I43-I44, K10: I48-I49
G3	K1: I5, K2: I10, K3: I15, K4: I20, K5: I25, K6: I30, K7: I35, K8: I40, K9: I45, K10: I50

We also compare the results of **RH-M2** with those of **RH-M1** in Table 11. From the reported results, it seems that in many cases, **RH-M2** can generate a better solution than **RH-M1**. For instance, model **RH-M2** can provide better solutions, which lead from 1.3 to 10.6 % less energy consumption for examples with ten jobs and fifteen operations (Examples 41-48 and 51-53). On the other hand, **RH-M1** is more efficient for cases with thirty jobs (i.e., Examples 54-58), as it can generate solutions with up to 7.3% less energy consumption.

Table 11. Comparative results of **RH-M1** and **RH-M2**

Ex.	**RH-M1**	**RH-M2**	Diff (%)	Ex.	**RH-M1**	**RH-M2**	Diff (%)
	TEC (kW)	TEC (kW)			TEC (kW)	TEC (kW)	
Ex21	214.78	215.87	0.5	Ex40	3608.97	3562.7	-1.3
Ex22	4091.73	3679.99	-10.1	Ex41	4023.78	3754.31	-6.7
Ex23	4192.16	3808.34	-9.2	Ex42	4039.19	3717.92	-8.0
Ex24	1847.84	1907.12	3.2	Ex43	4339.28	3978.46	-8.3
Ex25	1928.84	1941.73	0.7	Ex44	5984.42	5475.11	-8.5
Ex26	1712.62	1633.59	-4.6	Ex45	5466.79	4941.6	-9.6
Ex27	1914.79	1763.91	-7.9	Ex46	5858.67	5185.61	-11.5
Ex28	1644.57	1598	-2.8	Ex47	5578.43	5257.31	-5.8
Ex29	2851.87	2718.8	-4.7	Ex48	6178.14	5527.5	-10.5
Ex30	2820.98	2521.31	-10.6	Ex49	6946.16	6951.96	0.1
Ex31	2830.43	2645.48	-6.5	Ex50	7434.35	7560.55	1.7
Ex32	2821.51	2685.13	-4.8	Ex51	6866.13	6620.26	-3.6
Ex33	2910.03	2657.98	-8.7	Ex52	7257.84	7060.59	-2.7
Ex34	3632.30	3565.38	-1.8	Ex53	7200.05	6938.57	-3.6
Ex35	3406.47	3523.2	3.4	Ex54	8698.35	9167.93	5.4
Ex36	3272.25	3417.27	4.4	Ex55	9580.33	9708.14	1.3
Ex37	3636.68	3716.38	2.2	Ex56	8834.19	8861.43	0.3
Ex38	3987.97	3787.02	-5.0	Ex57	8958.33	9610.23	7.3
Ex39	4278.56	3884.04	-9.2	Ex58	9775.46	10003.95	2.3

6.4. Computational results from the hybrid algorithm

The proposed hybrid algorithm **eGEP-M2** is also used to solve Examples 21-58. The computational results are provided in Tables 12-16, where several GEP-based dispatching rules from Zhang *et al.* (2017) are applied. From Tables 12-16, it seems that a significantly better solution by up to 20% less TEC from **eGEP-M2** is identified compared to that from **eGEP**. For instance, **eGEP-M2** with the dispatching rule 1 generates a schedule with TEC of 5792.41 kW for Example 48, which is 19.7% less energy consumption than that from **eGEP** using the dispatching rule 1. Furthermore, it seems that less than one minute is required to generate the optimal solution for most examples from **eGEP-M2**. For instance, **eGEP-M2** generates a schedule with TEC of 5792.41 kW for Example 48 in 31 s. Even for most large examples (i.e. Example 53-58), small computational time (i.e., within 5 minutes) is required to generate the best solution.

Figure 10 illustrates the schedule for Example 22 generated from the hybrid algorithm **GEP-M2** with dispatching rule 8. The schedule for this example from **eGEP** is depicted in Figure 5. Comparing those schedules in Figure 10 and Figure 5, we can see that units remain idle for fewer periods during the scheduling horizon in Figure 10. Additionally, for those periods, the units are switched off, since it is more energy-efficient. For instance, unit J10 is switched off at 590 min and switched on at 961 min. The switch off-on energy consumption during this period for this unit J10 is 19.8kW. As already discussed, **eGEP** leads to a combined standby and switch off-on energy consumption of 91.8kW for the same unit. As a result, **GEP-M2** generates a schedule with 78.4% less energy consumption for unit J10.

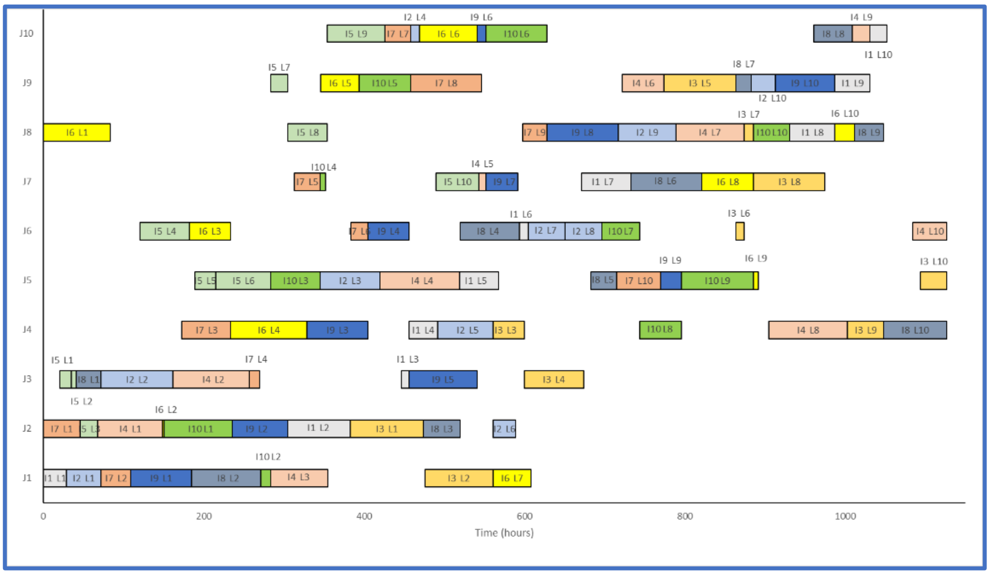

Figure 10 Best schedule for Example 22 using **eGEP-M2** with the dispatching rule 8

Table 12. Comparison results for Examples 21-58 using **eGEP** dispatching rule 1 and the hybrid algorithm **eGEP-M2**

	eGEP	**eGEP-M2**		
Ex	TEC (kW)	TEC (kW)	CPU Time (s)	Diff (%)
Ex21	331.78	273.44	0.031	
Ex22	4398.33	3973.63	1.0	-9.7
Ex23	3924.75	3808.29	0.047	-3.0
Ex24	2251.54	2150.22	0.031	-4.5
Ex25	2299.34	2139.78	0.032	-6.9
Ex26	2067.84	1814.60	0.047	-12.2
Ex27	2023.54	1873.24	0.046	-7.4
Ex28	1859.41	1784.59	0.047	-4.0
Ex29	2891.37	2772.52	0.031	-4.1
Ex30	2947.75	2746.31	0.031	-6.8
Ex31	3211.28	2971.75	0.047	-7.5
Ex32	3101.72	2901.06	0.046	-6.5
Ex33	3340.19	2829.80	0.062	-15.3
Ex34	4490.27	3779.94	0.046	-15.8
Ex35	3954.04	3493.97	0.109	-11.6
Ex36	3926.56	3412.03	0.063	-13.1
Ex37	4170.90	3873.96	0.078	-7.1
Ex38	3851.87	3711.79	0.047	-3.6
Ex39	4837.29	4253.91	0.69	-12.1
Ex40	4779.03	3950.16	1.0	-17.3
Ex41	4500.77	3838.06	1.4	-14.7
Ex42	4337.40	3867.97	0.89	-10.8
Ex43	4825.34	4329.41	0.98	-10.3
Ex44	5951.20	5372.26	1.7	-9.7
Ex45	6257.80	5348.49	1.7	-14.5
Ex46	6325.06	5569.00	2.0	-12.0
Ex47	5988.91	5376.65	2.9	-10.2
Ex48	7211.06	5792.41	31.0	-19.7
Ex49	7686.70	6777.31	32.6	-11.8
Ex50	8169.05	7153.02	153	-12.4
Ex51	7201.22	6687.08	3.6	-7.1
Ex52	7683.01	6637.30	142	-13.6
Ex53	7284.41	6711.15	0.94	-7.9
Ex54	10307.09	8975.42	157	-12.9
Ex55	11054.20	9815.61	219	-11.2
Ex56	10736.46	9246.65	95.6	-13.9
Ex57	10427.51	9161.15	761	-12.1
Ex58	11196.15	9813.16	3600[a]	-12.4

[a] Relative gap 0.07%

Table 13. Comparative results for Examples 21-58 using the **eGEP** dispatching rule 5 and the hybrid algorithm **eGEP-M2**

Ex	**eGEP** TEC (kW)	**eGEP-M2** TEC (kW)	CPU Time (s)	Diff (%)
Ex21	297.98	275.55	0.062	-7.5
Ex22	4047.03	3649.89	0.81	-9.8
Ex23	4088.49	3548.94	0.078	-13.2
Ex24	2070.50	2029.38	0.032	-2.0
Ex25	1975.75	1943.83	0.031	-1.6
Ex26	1964.55	1751.54	0.031	-10.8
Ex27	1939.76	1836.00	0.047	-5.3
Ex28	2006.52	1865.30	0.078	-7.0
Ex29	3060.93	2886.49	0.046	-5.7
Ex30	2835.82	2617.58	0.063	-7.7
Ex31	2807.84	2754.71	0.047	-1.9
Ex32	3046.21	2890.21	0.062	-5.1
Ex33	3271.49	2758.93	0.047	-15.7
Ex34	4064.60	3671.31	0.063	-9.7
Ex35	3700.86	3468.03	0.20	-6.3
Ex36	3682.76	3427.29	0.078	-6.9
Ex37	3983.02	3672.87	0.109	-7.8
Ex38	4018.18	3767.52	0.047	-6.2
Ex39	4982.54	4260.96	0.84	-14.5
Ex40	4300.04	3715.50	2.4	-13.6
Ex41	4059.53	3728.99	0.36	-8.1
Ex42	3937.63	3706.31	1.1	-5.9
Ex43	4396.11	3897.01	1.3	-11.4
Ex44	5708.72	5325.92	1.9	-6.7
Ex45	5876.62	5169.44	5.3	-12.0
Ex46	6322.86	5578.53	26.2	-11.8
Ex47	5763.51	5278.09	8.8	-8.4
Ex48	6640.15	5671.46	2.7	-14.6
Ex49	7550.94	6793.95	35.8	-10.0
Ex50	7859.20	7127.99	13.3	-9.3
Ex51	7201.43	6662.99	4.1	-7.5
Ex52	7287.90	6552.15	30.6	-10.1
Ex53	7332.79	6731.61	8.2	-8.2
Ex54	10108.14	9098.46	1106	-10.0
Ex55	10939.20	9726.36	159	-11.1
Ex56	10339.46	9112.79	145	-11.9
Ex57	10081.65	9142.43	3600[a]	-9.3
Ex58	10751.25	9459.67	586	-12.0

[a] Relative gap 0.80%

Table 14. Comparative results for Examples 21-58 using the **eGEP** dispatching rule 7 and the hybrid algorithm **eGEP-M2**

	eGEP	**eGEP-M2**		
Ex	TEC (kW)	TEC (kW)	CPU Time (s)	Diff (%)
Ex21	377.73	310.24	0.031	-17.9
Ex22	4342.12	4014.32	0.52	-7.5
Ex23	4402.20	3756.34	0.22	-14.7
Ex24	2172.22	2095.10	0.062	-3.6
Ex25	2033.00	1947.04	0.031	-4.2
Ex26	2021.92	1873.05	0.031	-7.4
Ex27	2059.48	1959.69	0.031	-4.8
Ex28	1976.26	1849.54	0.078	-6.4
Ex29	3044.67	2853.12	0.031	-6.3
Ex30	2826.03	2676.27	0.093	-5.3
Ex31	2765.72	2679.49	0.078	-3.1
Ex32	3136.43	2964.49	0.047	-5.5
Ex33	3036.47	2756.63	0.063	-9.2
Ex34	3947.29	3691.09	0.062	-6.5
Ex35	3731.46	3499.69	0.48	-6.2
Ex36	4061.80	3676.62	0.36	-9.5
Ex37	4463.81	3926.23	0.078	-12.0
Ex38	3740.39	3600.77	0.078	-3.7
Ex39	4480.15	4018.49	0.34	-10.3
Ex40	4482.24	3914.30	1.4	-12.7
Ex41	4160.37	3678.10	0.44	709.5
Ex42	4330.02	3815.93	1.2	-11.9
Ex43	4437.60	3974.99	1.4	-10.4
Ex44	5976.21	5560.87	2.6	-6.9
Ex45	5756.06	5079.62	12.8	-11.8
Ex46	5987.00	5383.50	13.1	-10.1
Ex47	6180.85	5394.63	3.6	-12.7
Ex48	7138.21	5899.35	64.3	-17.4
Ex49	7504.90	6784.20	3.3	-9.6
Ex50	8192.58	7206.61	3.9	-12.0
Ex51	7528.43	6698.02	3.5	-11.0
Ex52	7388.66	6647.78	37.7	-10.0
Ex53	7950.47	7080.96	34.1	-10.9
Ex54	10036.36	9062.42	100	-9.7
Ex55	10703.56	9799.30	1279	-8.4
Ex56	10194.85	9146.91	766	-10.3
Ex57	9884.19	9131.27	3600[a]	-7.6
Ex58	11269.35	9780.38	32.1	-13.2

[a] Relative gap 0.04%

Table 15. Comparative results for Examples 21-58 using the **eGEP** dispatching rule 8 and the hybrid algorithm **eGEP-M2**

	eGEP	**eGEP-M2**		
Ex	TEC (kW)	TEC (kW)	CPU Time (s)	Diff (%)
Ex21	296.71	255.75	0.124	-13.8
Ex22	4289.90	3786.76	0.717	-11.7
Ex23	4101.52	3627.59	0.125	-11.5
Ex24	2017.38	1996.38	0.094	-1.0
Ex25	2061.22	1994.99	0.047	-3.2
Ex26	2077.87	1871.63	0.218	-9.9
Ex27	1940.73	1836.00	0.219	-5.4
Ex28	2015.52	1842.34	0.156	-8.6
Ex29	2921.23	2851.49	0.109	-2.4
Ex30	2761.52	2668.89	0.218	-3.4
Ex31	2866.40	2716.39	0.093	-5.2
Ex32	3122.96	2911.21	0.094	-6.8
Ex33	3164.21	2830.84	0.078	-10.5
Ex34	3954.61	3581.06	0.296	-9.4
Ex35	3751.06	3496.18	1.435	-6.8
Ex36	3698.97	3456.04	0.266	-6.6
Ex37	3796.85	3687.75	0.187	-2.9
Ex38	3876.12	3632.99	0.234	-6.3
Ex39	4698.71	4073.37	0.624	-13.3
Ex40	4328.51	3851.69	2.184	-11.0
Ex41	4219.24	3764.29	1.123	-10.8
Ex42	4264.51	3820.54	1.388	-10.4
Ex43	4311.92	3905.98	0.999	-9.4
Ex44	5972.97	5362.92	2.153	-10.2
Ex45	6157.10	5136.97	15.6	-16.5
Ex46	6307.20	5554.23	4.462	-11.9
Ex47	5981.64	5275.81	1.482	-11.8
Ex48	7113.48	5832.81	3.946	-18.0
Ex49	7351.24	6807.56	5.865	-7.4
Ex50	8244.86	7111.49	43.0	-13.7
Ex51	7396.69	6720.92	3.432	-9.1
Ex52	7285.72	6536.35	111	-10.3
Ex53	7999.57	7147.95	25.9	-10.6
Ex54	10344.35	9515.27	779	-8.0
Ex55	10680.83	9763.51	576	-8.5
Ex56	10183.47	9056.31	1584	-11.1
Ex57	9865.68	9080.37	3600[a]	-8.0
Ex58	10832.23	9556.34	134	-11.8

[a] Relative gap 0.05%

Table 16. Comparative results for Examples 21-58 using the **eGEP** dispatching rule 9 and the hybrid algorithm **eGEP-M2**

	eGEP	**eGEP-M2**		
Ex	TEC (kW)	TEC (kW)	CPU Time (s)	Diff (%)
Ex21	320.25	286.63	0.047	-10.5
Ex22	4465.27	3860.23	1.2	-13.5
Ex23	4355.59	3763.03	0.047	-13.6
Ex24	1914.55	1881.38	0.062	-1.7
Ex25	2195.64	2101.69	0.046	-4.3
Ex26	2294.79	1931.57	0.031	-15.8
Ex27	2121.88	1925.57	0.062	-9.3
Ex28	1864.79	1685.43	0.031	-9.6
Ex29	3314.21	2909.79	0.047	-12.2
Ex30	2917.68	2700.44	0.047	-7.4
Ex31	3034.12	2828.43	0.047	-6.8
Ex32	3220.20	2964.46	0.047	-7.9
Ex33	3338.57	2888.41	0.063	-13.5
Ex34	4584.44	3668.83	0.063	-20.0
Ex35	4070.23	3485.15	0.171	-14.4
Ex36	3979.86	3333.06	0.047	-16.3
Ex37	4144.10	3780.81	0.078	-8.8
Ex38	3841.63	3625.34	0.062	-5.6
Ex39	4823.96	4183.65	1.4	-13.3
Ex40	4082.95	3516.39	1.5	-13.9
Ex41	4117.93	3639.54	0.67	-11.6
Ex42	4084.68	3853.95	0.64	-5.6
Ex43	4491.61	3979.13	9.2	-11.4
Ex44	6116.02	5408.99	25.4	-11.6
Ex45	5923.10	5061.75	2.4	-14.5
Ex46	6095.66	5428.28	190	-10.9
Ex47	6051.80	5191.31	19.3	-14.2
Ex48	6672.93	5522.70	30.1	-17.2
Ex49	7727.28	6637.33	133	-14.1
Ex50	8163.12	7011.05	26.5	-14.1
Ex51	7173.32	6447.07	14.5	-10.1
Ex52	7650.43	6489.99	53.7	-15.2
Ex53	7589.11	6554.44	15.1	-13.6
Ex54	10070.27	8938.16	417	-11.2
Ex55	10667.02	9436.44	222	-11.5
Ex56	10683.11	8802.82	945	-17.6
Ex57	9939.02	8785.05	142	-11.6
Ex58	10963.14	9530.19	134	-13.1

Tables 17-21 demonstrate the comparative results among **eGEP** and **eGEP-M2**, as well as **RH-M1** and **RH-M2**. From Tables 17-21, it seems that **eGEP-M2** and **RH-M1** are the most efficient since they can generate the best solution for most cases among these approaches. For instance, if we use dispatching rule 1, **eGEP-M2** and **RH-M1** provide the best schedule for 26 out of the 38 large-size examples. From those 26 examples, **eGEP-M2** can generate the best solution for 10 of them, while **RH-M1** leads to a better solution than all approaches for the rest examples. **RH-M2** generates the best schedule for 12 cases, while **eGEP** do not provide the best solution for any of the examples. However, the superiority of each approach highly depends on the dispatching rule used. For instance, **eGEP-M2** using dispatching rule 9 can generate better solutions for 22 examples out of the 38 large-size examples. On the contrary, **RH-M2** can generate the best solutions only for 11 examples.

Table 17. Comparative results for Examples 21-58 using the **eGEP** dispatching rule 1, **eGEP-M2**, **RH-M1** and **RH-M2**

Example	**eGEP**	**eGEP-M2**	**RH-M1**	**RH-M2**
Ex21	331.78	273.44	**214.78**	215.87
Ex22	4398.33	3973.63	4091.73	**3682.85**
Ex23	3924.75	**3808.29**	4192.16	3808.34
Ex24	2251.54	2150.22	**1847.84**	1862.22
Ex25	2299.34	2139.78	**1928.84**	1981.00
Ex26	2067.84	1814.60	**1712.62**	1807.29
Ex27	2023.54	**1873.24**	1914.79	1928.91
Ex28	1859.41	1784.59	**1644.57**	1648.33
Ex29	2891.37	2772.52	2851.87	**2771.20**
Ex30	2947.75	**2746.31**	2820.98	2822.26
Ex31	3211.28	2971.75	2830.43	**2787.95**
Ex32	3101.72	2901.06	**2821.51**	2878.99
Ex33	3340.19	**2829.80**	2910.03	2831.50
Ex34	4490.27	3779.94	3632.30	**3565.38**
Ex35	3954.04	3493.97	**3406.47**	3523.20
Ex36	3926.56	3412.03	**3272.25**	3417.27
Ex37	4170.90	3873.96	**3636.68**	3716.38
Ex38	3851.87	**3711.79**	3987.97	3787.02
Ex39	4837.29	**4253.91**	4278.56	4307.73
Ex40	4779.03	3950.16	**3608.97**	3782.58
Ex41	4500.77	3838.06	4023.78	**3754.30**
Ex42	4337.40	**3867.97**	4039.19	3893.04
Ex43	4825.34	4329.41	4339.28	**4047.04**
Ex44	5951.20	**5372.26**	5984.42	5630.35
Ex45	6257.80	5348.49	5466.79	**5122.95**
Ex46	6325.06	**5569.00**	5858.67	5638.42
Ex47	5988.91	5376.65	5578.43	**5263.17**
Ex48	7211.06	**5792.41**	6178.14	5935.52
Ex49	7686.70	**6777.31**	6946.16	6951.96
Ex50	8169.05	**7153.02**	7434.35	7560.55
Ex51	7201.22	6687.08	6866.13	**6620.26**
Ex52	7683.01	**6637.30**	7257.84	7060.59

Ex53	7284.41	**6711.15**	7200.05	6938.57
Ex54	10307.09	8975.42	**8698.35**	9167.93
Ex55	11054.20	9815.61	**9580.33**	9708.14
Ex56	10736.46	9246.65	**8834.19**	8861.43
Ex57	10427.51	9161.15	**8958.33**	9610.23
Ex58	11196.15	9813.16	**9775.46**	10004.00

Table 18. Comparative results for Examples 21-58 using the **eGEP** dispatching rule 5, **eGEP-M2**, **RH-M1** and **RH-M2**

Example	**eGEP**	**eGEP-M2**	**RH-M1**	**RH-M2**
Ex21	297.98	275.55	**214.78**	215.87
Ex22	4047.03	**3649.89**	4091.73	3682.85
Ex23	4088.49	**3548.94**	4192.16	3808.34
Ex24	2070.50	2029.38	**1847.84**	1862.22
Ex25	1975.75	1943.83	**1928.84**	1981.00
Ex26	1964.55	1751.54	**1712.62**	1807.29
Ex27	1939.76	**1836.00**	1914.79	1928.91
Ex28	2006.52	1865.30	**1644.57**	1648.33
Ex29	3060.93	2886.49	2851.87	**2771.20**
Ex30	2835.82	**2617.58**	2820.98	2822.26
Ex31	2807.84	**2754.71**	2830.43	2787.95
Ex32	3046.21	2890.21	**2821.51**	2878.99
Ex33	3271.49	**2758.93**	2910.03	2831.50
Ex34	4064.60	3671.31	3632.30	**3565.38**
Ex35	3700.86	3468.03	**3406.47**	3523.20
Ex36	3682.76	3427.29	**3272.25**	3417.27
Ex37	3983.02	3672.87	**3636.68**	3716.38
Ex38	4018.18	**3767.52**	3987.97	3787.02
Ex39	4982.54	**4260.96**	4278.56	4307.73
Ex40	4300.04	3715.50	**3608.97**	3782.58
Ex41	4059.53	**3728.99**	4023.78	3754.30
Ex42	3937.63	**3706.31**	4039.19	3893.04
Ex43	4396.11	**3897.01**	4339.28	4047.04
Ex44	5708.72	**5325.92**	5984.42	5630.35
Ex45	5876.62	5169.44	5466.79	**5122.95**
Ex46	6322.86	**5578.53**	5858.67	5638.42
Ex47	5763.51	5278.09	5578.43	**5263.17**
Ex48	6640.15	**5671.46**	6178.14	5935.52
Ex49	7550.94	**6793.95**	6946.16	6951.96
Ex50	7859.20	**7127.99**	7434.35	7560.55
Ex51	7201.43	6662.99	6866.13	**6620.26**
Ex52	7287.90	**6552.15**	7257.84	7060.59
Ex53	7332.79	**6731.61**	7200.05	6938.57
Ex54	10108.14	9098.46	**8698.35**	9167.93
Ex55	10939.20	9726.36	**9580.33**	9708.14
Ex56	10339.46	9112.79	**8834.19**	8861.43
Ex57	10081.65	9142.43	**8958.33**	9610.23
Ex58	10751.25	**9459.67**	9775.46	10004.00

Table 19. Comparative results for Examples 21-58 using the **eGEP** dispatching rule 7, **eGEP-M2**, **RH-M1** and **RH-M2**

Example	**eGEP**	**eGEP-M2**	**RH-M1**	**RH-M2**
Ex21	377.73	310.24	**214.78**	215.87
Ex22	4342.12	4014.32	4091.73	**3682.85**
Ex23	4402.20	**3756.34**	4192.16	3808.34
Ex24	2172.22	2095.10	**1847.84**	1862.22
Ex25	2033.00	1947.04	**1928.84**	1981.00
Ex26	2021.92	1873.05	**1712.62**	1807.29
Ex27	2059.48	1959.69	**1914.79**	1928.91
Ex28	1976.26	1849.54	**1644.57**	1648.33
Ex29	3044.67	2853.12	2851.87	**2771.20**
Ex30	2826.03	**2676.27**	2820.98	2822.26
Ex31	2765.72	**2679.49**	2830.43	2787.95
Ex32	3136.43	2964.49	**2821.51**	2878.99
Ex33	3036.47	**2756.63**	2910.03	2831.50
Ex34	3947.29	3691.09	3632.30	**3565.38**
Ex35	3731.46	3499.69	**3406.47**	3523.20
Ex36	4061.80	3676.62	**3272.25**	3417.27
Ex37	4463.81	3926.23	**3636.68**	3716.38
Ex38	3740.39	**3600.77**	3987.97	3787.02
Ex39	4480.15	**4018.49**	4278.56	4307.73
Ex40	4482.24	3914.30	**3608.97**	3782.58
Ex41	4160.37	**3678.10**	4023.78	3754.30
Ex42	4330.02	**3815.93**	4039.19	3893.04
Ex43	4437.60	**3974.99**	4339.28	4047.04
Ex44	5976.21	**5560.87**	5984.42	5630.35
Ex45	5756.06	**5079.62**	5466.79	5122.95
Ex46	5987.00	**5383.50**	5858.67	5638.42
Ex47	6180.85	5394.63	5578.43	**5263.17**
Ex48	7138.21	**5899.35**	6178.14	5935.52
Ex49	7504.90	**6784.20**	6946.16	6951.96
Ex50	8192.58	**7206.61**	7434.35	7560.55
Ex51	7528.43	6698.02	6866.13	**6620.26**
Ex52	7388.66	**6647.78**	7257.84	7060.59
Ex53	7950.47	7080.96	7200.05	**6938.57**
Ex54	10036.36	9062.42	**8698.35**	9167.93
Ex55	10703.56	9799.30	**9580.33**	9708.14
Ex56	10194.85	9146.91	**8834.19**	8861.43
Ex57	9884.19	9131.27	**8958.33**	9610.23
Ex58	11269.35	9780.38	**9775.46**	10004.00

Table 20. Comparative results for Examples 21-58 using the **eGEP** dispatching rule 8, **eGEP-M2**, **RH-M1** and **RH-M2**

Example	**eGEP**	**eGEP-M2**	**RH-M1**	**RH-M2**
Ex21	296.71	255.75	**214.78**	215.87
Ex22	4289.90	3786.76	4091.73	**3682.85**
Ex23	4101.52	**3627.59**	4192.16	3808.34
Ex24	2017.38	1996.38	**1847.84**	1862.22
Ex25	2061.22	1994.99	**1928.84**	1981.00
Ex26	2077.87	1871.63	**1712.62**	1807.29
Ex27	1940.73	**1836.00**	1914.79	1928.91
Ex28	2015.52	1842.34	**1644.57**	1648.33
Ex29	2921.23	2851.49	2851.87	**2771.20**
Ex30	2761.52	**2668.89**	2820.98	2822.26
Ex31	2866.40	**2716.39**	2830.43	2787.95
Ex32	3122.96	2911.21	**2821.51**	2878.99
Ex33	3164.21	**2830.84**	2910.03	2831.50
Ex34	3954.61	3581.06	3632.30	**3565.38**
Ex35	3751.06	3496.18	**3406.47**	3523.20
Ex36	3698.97	3456.04	**3272.25**	3417.27
Ex37	3796.85	3687.75	**3636.68**	3716.38
Ex38	3876.12	**3632.99**	3987.97	3787.02
Ex39	4698.71	**4073.37**	4278.56	4307.73
Ex40	4328.51	3851.69	**3608.97**	3782.58
Ex41	4219.24	3764.29	4023.78	**3754.30**
Ex42	4264.51	**3820.54**	4039.19	3893.04
Ex43	4311.92	**3905.98**	4339.28	4047.04
Ex44	5972.97	**5362.92**	5984.42	5630.35
Ex45	6157.10	5136.97	5466.79	**5122.95**
Ex46	6307.20	**5554.23**	5858.67	5638.42
Ex47	5981.64	5275.81	5578.43	**5263.17**
Ex48	7113.48	**5832.81**	6178.14	5935.52
Ex49	7351.24	**6807.56**	6946.16	6951.96
Ex50	8244.86	**7111.49**	7434.35	7560.55
Ex51	7396.69	6720.92	6866.13	**6620.26**
Ex52	7285.72	**6536.35**	7257.84	7060.59
Ex53	7999.57	7147.95	7200.05	**6938.57**
Ex54	10344.35	9515.27	**8698.35**	9167.93
Ex55	10680.83	9763.51	**9580.33**	9708.14
Ex56	10183.47	9056.31	**8834.19**	8861.43
Ex57	9865.68	9080.37	**8958.33**	9610.23
Ex58	10832.23	**9556.34**	9775.46	10004.00

Table 21. Comparative results for Examples 21-58 using the **eGEP** dispatching rule 9, **eGEP-M2**, **RH-M1** and **RH-M2**

Example	**eGEP**	**eGEP-M2**	**RH-M1**	**RH-M2**
Ex21	320.25	286.63	**214.78**	215.87
Ex22	4465.27	3860.23	4091.73	**3682.85**
Ex23	4355.59	**3763.03**	4192.16	3808.34
Ex24	1914.55	1881.38	**1847.84**	1862.22
Ex25	2195.64	2101.69	**1928.84**	1981.00
Ex26	2294.79	1931.57	**1712.62**	1807.29
Ex27	2121.88	1925.57	**1914.79**	1928.91
Ex28	1864.79	1685.43	**1644.57**	1648.33
Ex29	3314.21	2909.79	2851.87	**2771.2**
Ex30	2917.68	**2700.44**	2820.98	2822.26
Ex31	3034.12	2828.43	2830.43	**2787.95**
Ex32	3220.20	2964.46	**2821.51**	2878.99
Ex33	3338.57	2888.41	2910.03	**2831.5**
Ex34	4584.44	3668.83	3632.3	**3565.38**
Ex35	4070.23	3485.15	**3406.47**	3523.2
Ex36	3979.86	3333.06	**3272.25**	3417.27
Ex37	4144.10	3780.81	**3636.68**	3716.38
Ex38	3841.63	**3625.34**	3987.97	3787.02
Ex39	4823.96	**4183.65**	4278.56	4307.73
Ex40	4082.95	**3516.39**	3608.97	3782.58
Ex41	4117.93	3639.54	4023.78	**3754.3**
Ex42	4084.68	**3853.95**	4039.19	3893.04
Ex43	4491.61	**3979.13**	4339.28	4047.04
Ex44	6116.02	**5408.99**	5984.42	5630.35
Ex45	5923.10	**5061.75**	5466.79	5122.95
Ex46	6095.66	**5428.28**	5858.67	5638.42
Ex47	6051.80	**5191.31**	5578.43	5263.17
Ex48	6672.93	**5522.70**	6178.14	5935.52
Ex49	7727.28	**6637.33**	6946.16	6951.96
Ex50	8163.12	**7011.05**	7434.35	7560.55
Ex51	7173.32	**6447.07**	6866.13	6620.26
Ex52	7650.43	**6489.99**	7257.84	7060.59
Ex53	7589.11	**6554.44**	7200.05	6938.57
Ex54	10070.27	8938.16	**8698.35**	9167.93
Ex55	10667.02	**9436.44**	9580.33	9708.14
Ex56	10683.11	**8802.82**	8834.19	8861.43
Ex57	9939.02	**8785.05**	8958.33	9610.23
Ex58	10963.14	**9530.19**	9775.46	10004.00

7. Conclusions

In this work, we developed efficient approaches to generate energy-efficient schedules for flexible job-shops. Two MILP models based on unit-specific event-based and sequence-based representations have been presented. The proposed models are significantly more efficient than the existing mathematical models since they can generate solutions even for large-size examples. Between the proposed models, the unit-specific event-based model is more efficient since it can provide schedules for more examples than the sequence-based model. However, this model may fail to generate a feasible solution after one hour for cases with more than 20 jobs. By enhancing the rolling horizon decomposition approach, where the operations are divided into different groups using mixed-integer programming, both models can generate schedules for all examples, and they can generate up to 27.6% better solution than the best-reported solution generated with GEP. The proposed decomposition algorithm with the sequence-based model as short-term model scheduling model is more efficient since it can generate schedules with a better solution for examples with up to 30 jobs. Furthermore, we examined the combinations of mixed-integer programming approach and genetic evolutionary programming approach. By combining these approaches significantly better solutions with up to 20% less energy consumption, in comparison to the dispatching rules generated by using GEP can be generated.

Acknowledgements

Nikolaos Rakovitis would like to acknowledge financial support from the postgraduate award by The University of Manchester.

References

Al-Hinai Nasr, ElMekkawy, An efficient hybridized genetic algorithm architecture for the flexible job shop scheduling problem, Flexible services and manufacturing journal, 2011, 23(1), 64-85 https://doi.org/10.1007/s10696-010-9067-y

Bagheri A., Zandieh M., Mahdavi I., Yazdani M., An artificial immune algorithm for the flexible job-shop scheduling problem, Future generation computer systems, 2010, 26(4), 533-541 https://doi.org/10.1016/j.future.2009.10.004

Bowman, E. H., 1959. The schedule-sequence problem, Operations research, 7(5), 621-624.

Brandimarte Paolo, Routing and scheduling problem in a flexible job shop by tabu search, Annals of operations research, 1993, 41(3), 157-183 https://doi.org/10.1007/BF02023073

Chaudhry Imran Ali, Khan Abid Ali, 2016, A research survey: review of flexible job shop scheduling techinques, International transactions in operational research, 23(3), 551-591 https://doi.org/10.1111/itor.12199

Chen Haoxun, Ihlow Jiirgen, Lehmann Carsten, A genetic algorithm for flexible job-shop scheduling, Proceedings of the 1999 IEEE international conference on robotics & automation, 1999, 2, 1120-1125 https://doi.org/10.1109/ROBOT.1999.772512

Choi In-Chan, Choi Dae-Sik, A local search algorithm for jobshop scheduling problems with alternative operations and sequence-dependent setups, Computers & industrial engineering, 2002, 42(1), 43-58 https://doi.org/10.1016/S0360-8352(02)00002-5

Dai Min, Tang Dunbing, Giret Adriana, Salido Miguel, Li W., Energy-efficient scheduling for a flexible flow shop using improved genetic-simulated annealing algorithm, Robotics and computer-integrated manufacturing, 2013, 29(5), 418-429 https://doi.org/10.1016/j.rcim.2013.04.001

Fattahi Parviz, Saidi Mehrabad Mohammad, Jolai Fariborz, Mathematical modelling and heuristic approaches to flexible job shop scheduling problems, Journal of Intelligent manufacturing, 2007, 18(3), 331-342 https://doi.org/10.1007/s10845-007-0026-8

Gao Jie, Gen Mitsuo, Sun Linyan, Scheduling jobs and maintenances in flexible job shop with a hybrid genetic algorithm, Journal of intelligent manufacturing, 2006, 17(4), 493-507 https://doi.org/10.1007/s10845-005-0021-x

Gao Jie, Sun Linyan, Gen Mitsuo, A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems, Computers & operations research, 2008, 35(9), 2892-2907 https://doi.org/10.1016/j.cor.2007.01.001

Hurink Johann, Jurish Bernd, Thole Monika, Tabu search for the job-shop scheduling problem with multipurpose machines, Operations-research-spectrum, 1994, 15(4), 205-215 https://doi.org/10.1007/BF01719451

Janak Stacy, Floudas Christodoulos, Kallrath Josef, Vormbrock Norbert, Production scheduling of a large-scale industrial batch plant. I. short-term and medium-term scheduling, Industrial and Engineering Chemistry Research, 2006, 45(25), 8234-8252 https://doi.org/10.1021/ie0600588

Kacem Imed, Hammadi Slim, Borne Pierre, Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems, IEEE transactions on systems, 2002, 32(1), 1-13 https://doi.org/10.1109/TSMCC.2002.1009117

Karimi Sajad, Ardalan Zaniar, Naderi B., Mohammadi M., Scheduling flexible job-shops with transportation times: Mathematical models and a hybrid imperialist competitive algorithm, Applied mathematical modelling, 2017, 41, 667-682 https://doi.org/10.1016/j.apm.2016.09.022

Kondili, E., Pantelides, C. C., Sargent, R. W. H., 1993. A general algorithm for short-term scheduling of batch operations-I MILP formulation. Computers & Chemical Engineering. 17(2). 211-227. https://doi.org/10.1016/0098-1354(93)80015-F.

Lei Deming, Zheng Youlian, Guo Xiuping, 2017, A shuffled frogleaping algorithm for flexible job shop scheduling with the consideration of energy consumption, International Journal of Production Research, 55:11, 3126-3140, https://doi.org/10.1080/00207543.2016.1262082

Li, J., Sursarla, N., Karimi, I. A., Shaik, M. A., Floudas, C. A., 2010. An Analysis of Some Unit-Specific Event-Based Models for the Short-Term Scheduling of Noncontinuous Processes. Industrial and Engineering Chemistry Research. 49(2). 633-647. https://doi.org/ 10.1021/ie801879n.

Li Jie, Xiao Xin, Tang Qiuhua, Floudas Christodoulos, Production scheduling of a large-scale steelmaking continuous casting process via unit-specific event-based continuous-time models: short-term and medium-term scheduling, Industrial and Engineering Chemistry Research. 2012, 51(21), 7300-7319 https://doi.org/10.1021/ie2015944

Lin Xiaoxia, Floudas Christodoulos, Modi Sweta, Juhasz Nikola, Continuous-time optimization approach for medium-range production scheduling of a multiproduct batch plant, Industrial & engineering chemistry research, 2002, 41(16), 3884-3906, https://doi.org/10.1021/ie011002a

Liouane Noureddine, Saad Ihsen, Hammadi Slim, Borne Pierre, Ant systems and local search optimization for flexible job shop scheduling production, International journal of computers, communications & control, 2007, 2(2), 174-184 https://doi.org/10.15837/ijccc.2007.2.2350

Mastrolilli Monaldo, Gambardella Luca Maria, Effective neighborhood functions for the flexible job shop problem, Journal of scheduling, 2000, 3, 3-20 https://doi.org/10.1002/(SICI)1099-1425(200001/02)3:1<3::AID-JOS32>3.0.CO;2-Y

May Gökan, Stahl Bojan, Taishu Marco, Prabhu Vittal, Multi-objective genetic algorithm for energy-efficient job shop scheduling, International journal of production research, 2015, 53(23), 7071-7089 https://doi.org/10.1080/00207543.2015.1005248

Méndez C.A., Cerdá J., Optimal scheduling of a resource-constrained multiproduct batch plant supplying intermediates to nearby end-product facilities. Computers and Chemical Engineering, 2000, 24(2-7), 369-376 https://doi.org/10.1016/S0098-1354(00)00482-8

Meng Leilei, Zhang Chaoyong, Shao Xinyu, Ren Yaping, MILP models for energy-aware flexible job-shop scheduling problem, Journal of cleaner production, 2019, 210, 710-723 https://doi.org/10.1016/j.jclepro.2018.11.021

Mokhtari Hadi, Hasani Aliakbar, 2017, An energy-efficient multi-objective optimization for flexible job-shop problem, Computers and chemical engineering, 104, 339-352 https://doi.org/10.1016/j.compchemeng.2017.05.004

Mouzon Gilles, Yildirim Mehmet, Twomey Janet, Operational methods for minimization of energy consumption of manufacturing equipment, International journal of production research, 2007, 45, 4247-4271 https://doi.org/10.1080/00207540701450013

Özgüven Cemal, Özbakir Lale, Yavuz Yasemin, Mathematical models for job-shop scheduling problems with routing and process plan flexibility, Applied mathematical modelling, 2010, 34(6), 1539-1548 https://doi.org/10.1016/j.apm.2009.09.002

Paulli 1995, A hierarchical approach for the FMS scheduling problem, European journal of operational research, 1995, 86(1), 32-42 https://doi.org/10.1016/0377-2217(95)00059-Y

Pezzella F., Morganti G., Ciaschetti G., A genetic algorithm for the flexible job-shop scheduling problem, Computers & operations research, 2008, 35(10), 3202-3212 https://doi.org/10.1016/j.cor.2007.02.014

Rakovitis N., Zhang N., Li J., Zhang L., A new approach for scheduling of multipurpose batch processes with unlimited intermediate storage policy, Frontiers of chemical science and engineering, 2019, 13, 784-802 https://doi.org/10.1007/s11705-019-1858-4

Rakovitis N., Zhang N., Li J., A novel unit-specific event-based formulation for short-term scheduling of multitasking processes in scientific service facilities, 2020, 133, 106626 https://doi.org/10.1016/j.compchemeng.2019.106626

Roshanaei V., Azab Ahmed, ElMaraghy H., Mathematical modelling and a meta-heuristic for flexible job shop scheduling, International journal of production research, 2013, 51(20), 6247-6274 https://doi.org/10.1080/00207543.2013.827806

Roy, S., Sussman, B., 1964. Les problémes d'ordonnancement avec constraintes disjonctives, SEMA, 9

Saidi-Mehrabad Mohammad, Fattahi Parviz, Flexible job shop scheduling with tabu search algorithms, The international journal of advanced manufacturing technology, 2007, 32(5-6), 563-570 https://doi.org/10.1007/s00170-005-0375-4

Shaik, M. A., Floudas, C. A., 2009. Novel Unified Modeling Approach for Short-Term Scheduling. Industrial & Engineering Chemistry Research. 48(6). 2947-2964. https://doi.org/10.1021/ie8010726.

Shaik Munawar, Floudas Christodoulos, Kallrath Josef, Pitz Hans-Joachim, Production scheduling of a large-scale industrial continuous plant: short-term and medium-term scheduling, Computers and chemical engineering, 2009, 31(3), 670-686 https://doi.org/10.1016/j.compchemeng.2008.08.013

Wagner Harvey, An integer linear-programming model for machine scheduling, Naval research logistics quarterly, 1959, 6(2), 131-140 https://doi.org/10.1002/nav.3800060205

Wang Han, Jiang Zhigang, Wang Yan, Zhang Hua, Wang Yanhong, A two-stage optimization method for energy-efficient flexible job-shop scheduling based on energy dynamic characterization, Journal of cleaner production, 2018, 188(1), 575-588 https://doi.org/10.1016/j.jclepro.2018.03.254

Xie Jin, Gao Liang, Peng Kunkun, Li Xinyu, Li Haoran, 2019, Review on flexible job shop scheduling, IET Collaborative Intelligent Manufacturing, 1(3), 67-77 http://dx.doi.org/10.1049/iet-cim.2018.0009

Yazdani M., Amiri M., Zandieh M., Flexible job-shop scheduling with parallel variable neighborhood search algorithm, Expert systems with applications, 2010, 37(1), 678-687 https://doi.org/10.1016/j.eswa.2009.06.007

Zhang Guohui, Gao Liang, Shi Yang, An effective genetic algorithm for the flexible job-shop scheduling problem, Expert systems with apprications, 2011, 38(4), 3563-3573 https://doi.org/10.1016/j.eswa.2010.08.145

Zhang Liping, Tang Qiuhua, Wu Zhengjia, Wang Fang, Mathematical modeling and evolutionary generation of rule sets for energy-efficient flexible job shops, Energy, 2017, 138, 210-227 https://doi.org/10.1016/j.energy.2017.07.005

Zhang Zhongwei, Wu Lihui, Peng Tao, Jia Shun, An improved scheduling approach for minimizing total energy consumption and makespan in a flexible job shop environment, Sustainability, 2018, 11(1), 1-21

Nomenclature

Sets

G: groups

I: tasks

$\mathbf{I}_j$: units that can process task i

$\mathbf{I}_k$: tasks that belong to job k

$\mathbf{I}_j^e$: tasks that can be exclusively processed in unit j

$\mathbf{I}_s^C$: tasks that consume state s

$\mathbf{I}_s^P$: tasks that produce state s

J: processing units/machines

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

$\mathbf{J}_{k,l}$: units that can process operation l which belongs to job k

$\mathbf{JP}_{k,j}$: jobs that is forbidden to be assigned in unit j

K: jobs

$\mathbf{KI}_k$: tasks that belong to job k

$\mathbf{KLJ}_{k,l,j}$: operations that can be processed in unit j

$\mathbf{KLKL}_{k,l,k',l'}$: operations that can succeed operation l which belogs to job k

L: operations

$\mathbf{L}_k$: operations that belong to job k

$\mathbf{LD}_k$: jobs with due dates

$\mathbf{LR}_k$: jobs with a non-zero release date

N: event points

S: states

$\mathbf{S}^R$: raw material states

$\mathbf{S}^{IN}$: intermediate states

$\mathbf{S}^P$: product states

Indicies

g, g': groups

i, i': tasks

j, j': units

k, k': jobs

l, l': operations

s: states

n, n', n'': event points

Parameters

d_k: due date for job k

EO_j: switch off-on energy consumption of unit j

EN_g: number of event points for group g

ES_j^0: initial standby energy consumption of unit j

H: scheduling horizon

L^{max}: maximum number of tasks that can be included in a group g

M: large positive number

$N_j^{\min}$: minimum number of tasks/operations that can be processed in unit j

$N_j^{\max}$: maximum number of tasks/operations that can be processed in unit j

$PC_{i,j}$: cutting power of task i in unit j

$PEN1$, $PEN2$: penalty coefficients

PU_j: unload power of unit j

r_k: release time for job k

w_1, w_2: importance weight parameters

$\alpha_{i,j}$: processing time of task i in unit j

$\alpha_{k,l,j}$: processing time of operation l that belong in job k in unit j

β: coefficient of indirect energy consumption

Δn: maximum number of event points that a task i is allowed to span

$\rho_{i,s}$: indicator of whether state s is consumed ($\rho_{i,s}$ = -1) or produced ($\rho_{i,s}$ = 1) by task i.

Binary variables

$w_{i,j,n,n'}$: 1 if task i is processed in unit j from event point n to event point n'

$x_{j,n}$: 1 if unit j is in standby mode at event point n

$x_{k,l,k',l',j}$: 1 if operation l of job k precedes operation l' of job k' in unit j

$z_{k,l,j}$: 1 if unit j is switched off after processing operation l of job k

z_j^0: parameter to denote if unit j is switched-off at the beginning of the scheduling horizon

Y_g: 1 if group g is selected

$Y_{i,g}^{\mathrm{i}}$: 1 if a task i is included to group g

$Y_{k,l,g}^{\mathrm{i}}$: 1 if an operation l belonging to a job k is included to group g

Continuous variables

$CT_{k,l,j}$: time that unit *j* remains idle after finishing processing operation *l* of job *k*

$ES_{j,n}$: standby energy consumption of unit *j* at event point *n*

$ES_{k,l}$: standby energy consumption of the unit after processing operation *l* of job *k*

MS: makespan

Obj: Objective value (rolling horizon decomposition approach)

TEC: total energy consumption

$T^{\mathrm{s}}_{j,n}$: start time of unit *j* at event point *n*

$T^{\mathrm{f}}_{j,n}$: end time of unit *j* at event point *n*

$T_{k,l}$: start time of operation *l* that belongs to job *k*

$T_{s,n}$: time that state *s* is available to be consumed at event point *n*

$TNI_{k,g}$: number of tasks from each job *k* that are included in a group *g*

TNL_g: total number of tasks included in a group *g*

$w_{k,l,j}$: 0-1 continuous variable, 1 if operation *l* of job *k* is processed in unit *j*

$XF_{k,l,j}$: 0-1 continuous variable, 1 if operation *l* of job *k* is the first being processed in unit *j*

$y_{k,l,j}$: 0-1 continuous variable, 1 if unit remains in standby mode after processing operation *l* of job *k*

y^0_j: parameter to denote if unit *j* is idle at the beginning of the scheduling horizon

Appendix A Proof that $w_{i,l,j}$, $XF_{i,l,j}$ can only take 0 and 1 values

Let's consider two tasks l and l' of job k and k' respectively. Both tasks l and l' are able to be processed in the same unit j. We can distinct 2 different cases

<u>Case 1</u>: Task l precedes task l' in processing unit j

In this case $x_{k,l,k',l',j} = 1$. As a result we have that from (20).

$$(20) \Rightarrow 1 \leq w_{k,l,j}$$

Since $w_{k,l,j}$ cannot take values greater than 1, it is concluded that it can only take the value 1. Similarly, from (19) it is concluded that $XF_{k,l,j}$ can only be zero.

<u>Case 2</u>: Task l does not precede task l'

In this case $x_{k,l,k',l',j} = 0$. Therefore, according to (19) we have that $w_{k,l,j} = XF_{k,l,j}$. As a result $w_{k,l,j}$ and $XF_{k,l,j}$ can take any value between 0 and 1 as far as they are equal. If we assume that $0 < w_{k,l,j} < 1$ then according to (22) there should be at least one more variable of the same set that $0 < w_{k,l,j'} < 1$. However, according to (20).

$$(20) \Rightarrow x_{k,l,k',l',j'} \leq w_{k,l,j}$$

And as a result $x_{k,l,k',l',j'} = 0$ since $x_{k,l,k',l',j'}$ is defined as binary variable. This means that no other task can succeed operation l. Based on the results presented on Case 1, any other operation cannot precede operation l in any unit j or j' ($x_{k,l,k',l',j} = 1$ or $x_{k,l,k',l',j'} = 1$) since in such case it should be $w_{k,l,j'} = 1$.

In conclusion, for Case 2 if $w_{k,l,j}$ and $XF_{k,l,j}$ take any value other than 0 and 1, it means that at least two processing units (unit j plus one or more units j' where $0 < w_{k,l,j'} < 1$) can only process this operation. Such assignment in most cases will be infeasible, since in most cases there are operations that can be processed in these unit j or j', or to significantly worse solutions that the optimal solution.

Appendix B Rolling horizon decomposition approach for sequence-based model

We use the same rolling horizon decomposition approach for the sequence-based model. Since this model is using the definition operations instead of tasks, we slightly modify the decomposition model. More specifically, we introduce a binary variable $Y^{\mathrm{i}}_{k,l,g}$, which is equal to 1 if an operation l belonging to a job k is included in group g. The decomposition model for the sequence-based model is modified as follows.

$$Y^{\mathrm{l}}_{k,l,g} \leq Y_g \qquad \forall k, l \in \mathbf{KL}_{k,l}, g \qquad \text{(B.1)}$$

$$\sum_{g} Y^{\mathrm{l}}_{k,l,g} = 1 \qquad \forall k, l \in \mathbf{KL}_{k,l} \qquad \text{(B.2)}$$

$$Y^{\mathrm{l}}_{k,l',g} \leq Y^{\mathrm{l}}_{k,l,g} + \sum_{g'<g} Y^{\mathrm{l}}_{k,l,g'} \qquad \forall l' = l-1, \mathbf{KL}_{k,l}, \mathbf{KL}_{k,l'} \qquad \text{(B.3)}$$

$$\sum_{k} \sum_{l \in \mathbf{KL}_{k,l}} Y^{\mathrm{l}}_{k,l,g} \geq Y_g \qquad \forall\, g \qquad \text{(B.4)}$$

$$Y_{g+1} \leq Y_g \qquad \forall\, g \qquad \text{(B.5)}$$

$$TNI_{k,g} = \sum_{l \in \mathbf{KL}_{k,l}} Y^{\mathrm{l}}_{k,l,g} \qquad \forall\, k, g \qquad \text{(B.6)}$$

$$TNI_{k,g+1} = TNI_{k,g} \qquad \forall\, k, g \qquad \text{(B.7)}$$

$$TNL_k = \sum_{k} \sum_{l \in \mathbf{KL}_{k,l}} Y^{\mathrm{l}}_{k,l,g} \qquad \forall\, g \qquad \text{(B.8)}$$

$$TNL_g \leq L^{\max} \qquad \forall\, g \qquad \text{(B.9)}$$

$$PEN1 \geq TNI_{k,g} \qquad \forall\, k, g \tag{B.10}$$

$$PEN2 \leq TNI_{k,g} + |G| \cdot \left(1 - Y_g\right) \qquad \forall\, k, g \tag{B.11}$$

$$obj = w_1 \cdot \sum_g Y_g + w_2(PEN1 - PEN2) \tag{B.12}$$

Chapter 8: Conclusions and Future Work

8.1 Conclusions

Even though many mathematical models for scheduling of process industry were proposed in the past three decades, it still seems that they are not efficient, since they require excessive computational time to generate the optimal solution. In some cases, they even fail to provide an optimal solution. In this Thesis, multiple different features were examined to improve model efficiency. For instance, in research contribution 1, the feature of allowing related tasks to take place at the same event point was examined. Such a feature was implemented into two new mathematical models for scheduling of multipurpose batch processes. A new definition for recycling tasks was also presented, to avoid generating suboptimal solutions. In both proposed models all related non-recycling production and consumption tasks can take place at the same event point. While the first model was based on task-based timing variables, the second model uses timing variables based on units. The computational results demonstrated that both models can generate the optimal solution for all examples, and they both reduce the number of event points required. As a result, they led to smaller model sizes in comparison to existing formulations, where related production and consumption tasks are not allowed to take place at the same event point (Shaik and Floudas 2009). The model with unit-based timing variables was the most efficient since it required the least possible computational time which can reach up to one magnitude in most cases.

The feature of allowing related tasks at the same event point was also implemented to a generic and efficient framework for process scheduling in research contribution 2. In summary, except for this feature, this framework also included the following features:

- Related production and consumption tasks are sequenced, only if there is an indirect transfer between the units processing those tasks
- Related production and consumption tasks are aligned, only if there is a direct transfer between the units processing those tasks
- A unit can store materials that produced for more than one event points.

The proposed framework was first implemented in the multipurpose batch process problem. By solving several motivating and benchmark results, it was shown that the proposed framework can always generate the optimal solution, even for those examples that existing formulations can only generate a suboptimum solution. Additionally, the formulation required a smaller number of binary variables in most cases compared to the existing mathematical formulation, especially when a processing unit can process multiple tasks. Furthermore, it did not need to allow a task to span over multiple event points to generate the optimal solution, which significantly reduced the model size. As a result, the computational time was significantly reduced by one order of magnitude in most cases.

In research contributions 3, 4 and 5, the approach presented in research approach 2, is implemented in the continuous, multitasking and flexible job-shop processes, respectively. The results demonstrated that the proposed framework can be successfully implemented for all those different processes. In all cases, same or better solutions than existing formulations were generated. For some examples, the proposed framework was even able to provide solutions, that existing formulations fail to generate after a specified time (i.e. one hour). As a result, it was concluded that the proposed framework is both generic and efficient since it can solve different types of scheduling problems in significantly less computational time.

For large-scale scheduling problems, where it is impossible to generate a good solution in small computational time, the rolling horizon decomposition approach was enhanced. Such a formulation can decompose the problem even if all orders/operations have the same due dates. The proposed decomposition approach grouped different orders/operations by using mixed-integer programming. To successfully decompose a large-scale problem, the complexity of each subproblem was previously determined. Such decomposition approach was successfully implemented in multitasking and flexible job-shop problems in Chapter 4 and research contribution 5, where up to 99.9% less computational time is required to generate near optimum solutions. The proposed decomposition approach can even generate good schedules for examples, where all mathematical models fail to solve.

Finally, in the last part of this thesis, a hybrid evolutionary programming and mathematical programming approach were developed for the flexible job-shop scheduling problem. In the first stage of this approach, the dispatching rules generated by GEP were used to provide the allocation and sequencing of tasks into units. In the second stage, the mathematical model is used to generate the optimum timing of operations into units for the given allocation and sequencing. Such a hybrid approach was able to generate schedules with up to 20% less energy consumption than the efficient dispatching rules generated by using eGEP. Additionally, the mathematical model only requires up to 15 minutes for most of the examples, which is acceptable for large-scale problems.

8.2 Future work

As discussed, a generic and efficient mathematical framework for scheduling of process industry was presented. Even though the proposed framework can solve different types of process scheduling problems and outperforms existing formulations, some limitations need to be handled in the future.

- The proposed framework was considered for unlimited and finite intermediate storage policy. Even though it can also solve examples with no intermediate storage policy (NIS), the performance of the formulation can be further improved by including several additional constraints only for states with this policy. For instance, if a unit processes a task that produces a state with NIS policy, then this unit can only directly transfer materials to another unit. In this case, the model size can be further decreased, by reducing the number of binary variables required to generate the optimal solution. This approach could improve the performance of the model.
- For unit wait policies, the proposed framework only considers cases with unlimited unit wait. As a result, the current formulation cannot solve examples, where several unstable intermediate products are produced within the facility. Such case can be easily handled, by using several duration constraints, where the duration of each task is limited, for states with limited unit wait policy, or it is equal to the processing time, for states with no unit wait policy. Even though such unit wait policies have already been considered in unit-specific event-based formulations with timing variables based on tasks, there is no unit-specific model with unit-based timing variables that consider such policies.
- In the proposed framework, all resources such as raw materials, utilities and manpower are unlimited. Even though the facilities are supplied with up to three months of supplies, the market may lack a specific raw material, or there may be a significant increase in the price of this material. In such cases, the facility should

carefully consider the final products that will schedule to produce based on the availability. Additionally, in most cases, a processing facility produces the necessary utilities (i.e. steam). As a result, the processing facility may not be available to produce all the required utilities based on the resulted schedule, which can lead them to buy such utilities in a significantly higher cost. Even if such utilizes can be produced in large amounts, it is desirable to only produce the amount required to reduce the costs. To tackle this limitation, several resource constraints, should be imposed, where they limit the number of tasks that can be simultaneously produced based on the available resources.

As also presented in this PhD thesis, hybrid GEP and mathematical modelling approaches can generate significantly better solutions than GEP approach in acceptable computational time. More specifically, the hybrid algorithm can provide better solutions than by just using the efficient dispatching rules developed by eGEP. However, such an approach is only implemented in the flexible job-shop scheduling problem. Therefore, for future work, the use of such hybrid methods in different types of the process industry, including multipurpose and continuous processes should be considered. Similar to the flexible job-shop scheduling problem, the eGEP can develop several rules that can generate allocation and sequencing decisions for multipurpose batch processes and continuous processes. The proposed framework can then develop optimal timing and batching decisions. By using such a hybrid approach, optimum or near optimum solutions for small-scale problems and good solutions in significantly less computational time than mathematical modelling approaches for large-scale problems can be generated. As a result, the proposed hybrid approach will be able to generate good solutions without decomposing the problem in smaller subproblems.

References

Al-Hinai, N., ElMekkawy, T. Y., 2011. An efficient hybridized genetic algorithm architecture for the flexible job shop scheduling problem. Flexible services and manufacturing journal. 23. 64-85.

Ackermann, S., Fumero, Y., Montagra, J. M. 2018. Optimisation framework for the simultaneous batching and scheduling of multisite production environments. Industrial and Engineering Chemistry Research. 57(48). 16395-16406.

Bagheri, A., Zandieh, M., Mahdavi, I., Yazdani, M., 2010. An artificial immune algorithm for the flexible job-shop scheduling problem. Future generation computer systems. 26(4). 533-541.

Balas, E., 1969. Machine sequencing via disjunctive graphs: an implicit enumeration algorithm. Operations research. 17(6). 927-1092.

Bowman, E. H., 1959. The schedule-sequence problem. Operations research. 7(5). 621-624.

Brandimarte, P., 1993. Routing and scheduling in a flexible job shop by tabu search. Annals of operations research. 41. 157-183.

Castro, P. M., Barbosa-Póvoa, A. P. F. D., Matos, H., 2001. An improved RTN continuous-time formulation for the short-term scheduling of multipurpose batch plants. Industrial and Engineering Chemistry research. 40(9). 2059-2068.

Castro, P. M., Barbosa-Póvoa, A. P., Matos H. A., Novais, A. Q. 2004. Simple Continuous-Time Formulation for Short-Term Scheduling of Batch and Continuous Processes. Industrial Engineering and Chemistry Research. 43(1). 105-118.

Castro, P. M., Grossmann I. E., 2005. New continuous-time MILP model for the short-term scheduling of multistage batch plants. Industrial Engineering and Chemistry Research. 44(24). 9175-9190.

Castro, P. M., Grossmann I. E., 2006. An efficient MILP model for the short-term scheduling of single stage batch plants. Computers and chemical engineering. 30(6-7). 1003-1018.

Cerdá, J., Henning, G. P., Grossmann, I. E., 1997. A mixed-integer linear programming model for short-term scheduling of single-stage multiproduct batch plants with parallel lines. Industrial and engineering chemistry research. 36(5). 1695-1707.

Chan, F. T. S., Chung, S. H., Chan, P. L. Y., 2006. Application of genetic algorithms with dominant genes in a distributed scheduling problem in flexible manufacturing systems. International journal of production research. 44(3). 523-543.

Chen, H., Ihlow, J., Lehmann, C., 1999. A genetic algorithm for flexible job-shop scheduling. Computers and operations research. 35(10). 3202-3212.

Colorni, A., Dorigo, M., Maniezzo, V., Trubian, M., 1994. Ant system for job-shop scheduling. Statistics and computer science. 1-15.

Dai, M., Tang, D., Giret, A., Salido, M. A., Li W. D., 2013. Energy-efficient scheduling for a flexible flow shop using an improved genetic-simulated annealing algorithm. Robotics and computer-integrated manufacturing. 29(5). 418-429.

Edgar, T. and Himmelblau, D., 1989. Optimization of chemical processes. 1st ed. Singapore:McGraw-Hill

Falkenauer, E., Bouffouix, S., 1991. A genetic algorithm for job shop. Proceedings of 1991 IEEE International Conference on Robotics and Automation. Sacramento, CA, USA. 1. 824-829

Fattahi, P., Saidi Mehrabad, M., Jolai, F., 2007. Mathematical modelling and heuristic approaches to flexible job shop scheduling problems. Journal of intelligent manufacturing. 18. 331-342.

Fisher, M. L., Lageweg, B. J., Lenstra, J. K., Rinnoy Kan, A. H. G., 1983. Surrogate duality relaxation for job-shop scheduling. Discrete applied mathematics. 5. 65-75.

Florian, M., Trepant P., McMahon G., 1971. An implicit enumeration algorithm for the machine sequencing problem. management science, 17(12). B782-B792

Floudas, C. A., 1995. Nonlinear and mixed-integer optimization. 1st ed. New York:Oxford University press

Fumero, Y., Corsano, G., Montagna, J. M. 2012 Scheduling of multistage multiproduct batch plants operating in campaign-mode. Industrial and Engineering Chemistry Research. 51(10). 3988-4001.

Gao, J., Gen, M., Sun, L., 2006. Scheduling jobs and maintenances in flexible job shop with hybrid genetic algorithm. Journal of intelligent manufacturing. 17. 493-507.

Gao, J., Sun, L., Gen, M., 2008. A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems. Computers and operations research. 35(9). 2892-2907.

Garey, M. R., Johnson D. S., Sethi R., 1976. The complexity of flowshop and jobshop scheduling. Mathematics of operation research. 1976. 1(2). 117-129.

Giannelos, N. F., Georgiadis, M. C., 2002. A Simple New Continuous-Time Formulation for Short-Term Scheduling of Multipurpose Batch Processes. Industrial and Engineering Chemistry Research. 41(9). 2178-2184.

Glover, F., 1975. Improved linear integer programming formulations of non-linear integer problems. Management science. 22(4). 455-460.

Greenberg, H. H., 1966. A branch-bound solution to the general scheduling problem. Operations research. 16(2). 353-361.

Gupta, S., Karimi, I. A., 2003a. An improved MILP formulation for scheduling multiproduct multistage batch plants. Industrial and Engineering Chemistry Research. 42(11). 2365-2380

Gupta, S., Karimi, I. A., 2003b. Scheduling a two-stage multiproduct process with limited product shelf life in intermediate storage. Industrial and Engineering Chemistry Research. 42(3). 490-508

Harjunkoski, I., Grossmann, I. E., 2002. Decomposition techniques for multistage scheduling problems using mixed-integer and constraint programming methods. Computers and chemical engineering. 26(11). 1533-1552.

He, Y., Hui, C., 2010. A binary coding genetic algorithm for multi-purpose process scheduling: A case study. Chemical engineering science. 65. 4816-4828.

Ho, N. B., Tay, J. C., 2004. GENACE: An efficient cultural algorithm for solving the flexible job-shop problem. Proceedings of the 2004 Congress on Evolutionary Computation. Portland, OR, USA. 2. 1759-1766.

Hui, C., Gupta, A., 2000. A novel MILP formulation for short-term scheduling of multistage multi-product batch plants. Computers and chemical engineering. 24(12). 1611-1617.

Hui, C., Gupta, A., van der Meulen, H. A. J., 2000. A novel MILP formulation for short-term scheduling of multi-stage multi-product batch plants with sequence-dependent constraints. Computers and chemical engineering. 24(12). 2705-2717.

Hurink, J., Jurisch, B., Thole, M., 1994. Tabu search for the job-shop scheduling problem with multi-purpose machines. OR Spektrum. 15. 205-215.

Hussain, M. F., Joshi, S. B., 1998. A genetic algorithm for job shop scheduling problems with alternative routing. Proceedings of 1998 IEEE International Conference on Systems. Man, and Cybernetics, San Diego, CA, USA, 3, 2225-2230.

Ierapetritou, M. G., Floudas, C. A., 1998a. Effective continuous-time formulation for short-term scheduling. 1. Multipurpose batch processes. Industrial & Engineering Chemistry. 37(11). 4341-4359.

Ierapetritou, M. G., Floudas, C. A., 1998b. Effective Continuous-Time Formulation for Short-Term Scheduling 2. Continuous and Semicontinuous Processes. Industrial & Engineering Chemistry. 37(11). 4360-4374.

Janak, S. L., Lin, X., Floudas, C. A., 2004. Enhanced continuous-time unit-specific event-based formulation for short-term scheduling of multipurpose batch processes Resource constraints and mixed storage policies. Industrial and engineering Chemistry research. 43(10). 2516-2533.

Jia, H. Z., Nee, A. Y. C., Fuh, J. Y. H., Zhang, Y. F., 2003. A modified genetic algorithm for distributed scheduling problem. Journal of Intelligent Manufacturing. 14. 351-362.

Kacem, I., Hammadi, S., 2002. Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems. I*EEE Transactions on Systems, Man, and Cybernetics.* 32(1). 1-13.

Karimi, I. A., McDonald, C. M., 1997. Planning and scheduling of parallel semicontinuous processes 2. Short term scheduling. Industrial and engineering chemistry research. 36(7). 2701-2714.

Karimi, S., Ardalan, Z., Naderi, B., Mohammadi, M., 2017. Scheduling flexible job-shops with transportation times: mathematical models and a hybrid imperialist competitive algorithm. Applied mathematical modelling, 41, 667-682.

Kondili, E., Pantelides, C. C., Sargent, R. W. H., 1993. A general algorithm for short-term scheduling of batch operations-I MILP formulation. Computers & Chemical Engineering. 17(2). 211-227.

Ku, H., Karimi, L. A., 1988. Scheduling in serial multiproduct batch processes with finite interstage storage: A mixed integer linear pro-gramming formulation. Industrial and Engineering Chemistry Research. 27. 1840–1848.

Lagzi, S., Fukasawa, R., Ricardez-Sandoval, L., 2017a. A multitasking continuous time formulation for short-term scheduling of operations in multipurpose plants. Computers & Chemical Engineering. 97. 135-146.

Lagzi, S., Lee, D. Y., Fukasawa, R., Ricardez-Sandoval, L., 2017b. A computational study of continuous and discrete time formulations for a class of short-term scheduling problems for multipurpose plants. Industrial and Engineering Chemistry Research. 56(31). 8940-8953.

Lamba, N., Karimi, I. A., 2002. Scheduling parallel production lines with resource constraints. 1. Model formulation. Industrial and Engineering Chemistry research. 41(4). 779-789.

Lee, H., Maravelias, C. T., 2017. Discrete-time mixed-integer programming models for short-term scheduling in multipurpose environments. Computers and Chemical Engineering. 107. 171-183.

Lee, H., Maravelias, C. T., 2018. Combining the advantages of discrete- and continuous-time scheduling models: Part 1. Framework and mathematical formulations. Computers and Chemical Engineering. 116. 176-190.

Lee, K., Park, H. I., Lee, I., 2001. A Novel Nonuniform Discrete Time Formulation for Short-Term Scheduling of Batch and Continuous Processes. Industrial and Engineering Chemistry research. 40(22). 4902-4911.

Lee, S. Y., Fukasawa, R., Ricardez-Sandoval, L., 2019. Bi-objective short-term scheduling in a rolling horizon framework: a priori approaches with alternative operational objectives. Computers and Operations research. 111.141-154.

Li, J., Floudas, C.A., 2010. Optimal Event Point Determination for Short-Term Scheduling of Multipurpose Batch Plants via Unit-Specific Event-Based Continuous-Time Approaches. Industrial & Engineering Chemistry Reshearch. 49(16). 7446-7469.

Li, J., Karimi, I. A., Srinivasan, R., 2010. Recipe determination and scheduling of gasoline blending operations. AIChE journal. 56(2). 441-465.

Li, J., Xiao, X., Tang, Q., Floudas C. A., 2012. Production scheduling of a large-scale steelmaking continuous casting process via unit-specific event-based continuous-time models: short-term and medium-term scheduling. Industrial and engineering chemistry research. 51(21). 7300-7319.

Lin, X., Floudas C. A., Modi, S., Juhasz, M., 2002. Continuous-time optimization approach for medium-range production scheduling of a multiproduct batch plant. Industrial and engineering chemistry research. 41(16). 3884-3906.

Liouane, N., Saad, I., Hammadi, S., Borne, P., 2007. Ant systems & local search optimization for flexible job shop scheduling production. International journal of computers. communications and control. 2. 174-184.

Liu, Y., Karimi, I. A., 2007. Scheduling multistage, multiproduct batch plants with nonidentical parallel units and unlimited intermediate storage. Chemical engineering science. 62(6). 1549-1566.

Majozi, T., Zhu, X. X., 2001. A Novel Continuous-Time MILP Formulation for Multipurpose batch plants. 1. Short-term scheduling. Industrial and Engineering Chemistry. 40(25). 5935-5949.

Manne, A. S., 1960. On the job-shop scheduling problem. Operations research. 8(2). 219-223.

Maravelias, C. T., Grossmann, I. E., 2003. New General Continuous-Time State-Task Network Formulation for Short-Term Scheduling of Multipurpose Batch Plants. Industrial Engineering and Chemistry research. 42(13). 3056-3074.

Mastrolilli, M., Gambardella, L. M., 2000. Effective neighborhood functions for the flexible job shop problem. Journal of scheduling. 3(1). 3-20.

May, G., Stahl, B., Taisch, M., Prabhu, V., 2015. Multi-objective genetic algorithm for energy-efficient job shop scheduling. International journal of production research. 53(23). 7071-7089.

Méndez, C. A., Cerdá J., 2000. Optimal scheduling of resource-constrained multiproduct batch plant supplying intermediates to nearby end-product facilities. Computers and chemical engineering. 24(2-7), 369-376.

Méndez, C. A., Cerdá J., 2002. An efficient MILP continuous-time formulation for short-term scheduling of multiproduct continuous facilities. Computers and chemical engineering. 26(4-5). 687-695.

Méndez, C. A., Cerdá J., 2003. Short-term scheduling of multistage batch processes subject to limited finite resources. Computer aided chemical engineering. 15. 984-989.

Méndez, C. A., Henning G. P., Cerdá J., 2000. Optimal scheduling of batch plants satisfying multiple product orders with different due-dates. Computers and chemical engineering. 24(9-10). 2223-2245.

Méndez, C. A., Henning G. P., Cerdá J., 2001. An MILP continuous-time approach to short-term scheduling of resource-constrained multistage flowshop batch facilities. Computers and chemical engineering. 25(4-6). 701-711.

Meng, L., Zhnag, C., Shao, X., Ren, Y., 2019. MILP models for energy-aware flexible job shop scheduling problem. Journal of cleaner production. 210. 710-723.

Mesghouni, K., Hammadi, S., Borne, P., 1997. Evolution programs for job-shop scheduling, *1*997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation. Orlando, FL, USA. 1. 720-725.

Mockus, L., Reklaitis, G. V., 1997. Mathematical programming formulation for scheduling of batch operations based on nonuniform time discretization. Computers and Chemical Engineering. 21(10). 1147-1156.

Mockus, L., Reklaitis, G. V., 1999a. Continuous time representation approach to batch and continuous process scheduling. 1. MINLP formulation. Industrial and engineering chemistry research. 38(1). 197-203.

Mockus, L., Reklaitis, G. V., 1999b. Continuous time representation approach to batch and continuous process scheduling. 2. Computational issues. Industrial and engineering chemistry research. 38(1). 204-210.

Mohammadi, M., Poursabzi, O., 2014. A rolling horizon-based heuristic to solve a multi-level general lot sizing and scheduling problem with multiple machines (MLGLSP_MM) in job shop manufacturing system. Uncertain supply chain management. 2. 167-178.

Novara, F. M., Novas, J. M., Henning. G. P. 2016. A novel constraint programming model for large-scale scheduling problems in multiproduct multistage batch plants: Limited resources and campaign-based operation. Computers and Chemical Engineering. 93. 101-117.

Office for National Statistics. 2016. GDP(O) low level aggregates. Available at: https://www.ons.gov.uk/economy/grossdomesticproductgdp/datasets/gdpolowlevelaggregates (Accessed: 4 August 2020)

Özgüven, C., Özbakır, L., Yavuz, Y., 2010. Mathematical models for job-shop scheduling problems with routing and process plan flexibility. Applied mathematical modelling. 34(6). 1539-1548.

Pantelides, C., 1994. Unified frameworks for optimal process planning and scheduling. Proceedings of the Second Conference on Foundations of Computer Aided Operations. 253–274.

Panwalkar, S. S., Iskander, W., 1977. A survey of scheduling rules. Operations research. 25(1). 45-61.

Park, B. J., Choi, H. R., Kim, H. S., 2003. A hybrid genetic algorithm for the job shop scheduling problems. Computers and industrial engineering. 45. 597-613.

Patil, B. P., Fukasawa, R., Ricardez-Sandoval, L. A., 2015. Scheduling of operations in a large-scale Scientific services facility via multicommodity flow and an optimization-based algorithm. Industrial and Engineering Chemistry Research. 54(5). 1628-1639.

Pauli, J., 1995. A hierarchical approach for the FMS scheduling problem. European journal of operational research. 86(1). 32-42.

Pezzella, F., Morganti, G., Ciaschetti, G., 2008. A genetic algorithm for the flexible job-shop scheduling problem. Computers and operations research. 35(10). 3202-3212.

Pinto, J. M., Grossmann, I. E. 1994. Optimal cyclic scheduling of multistage continuous multiproduct plants. Computers and Chemical Engineering. 18. 797–816.

Pinto, J. M., Grossmann, I. E., 1995. A continuous time mixed integer linear programming model for short-term scheduling of multistage batch plants. Industrial and Engineering Chemistry research. 34(9). 3037-3051.

Reklaitis, G. V., Mockus, L., 1995. Mathematical programming formulation for scheduling of batch operations based on nonuniform time discretization. Acta Chimica Slovenica. 42. 81-86.

Roy, S., Sussman, B., 1964. Les problémes d'ordonnancement avec constraintes disjonctives. SEMA. 9

Saidi-Mehrabad, M., Fattahi, P., 2006. Flexible job shop scheduling with tabu search algorithms. The international journal of advanced manufacturing technology, 32, 563-570.

Santos F. Fukasawa, R., Ricardez-Sandoval, L., 2018. An integrated personnel allocation and machine scheduling problem for industrial size multipurpose plants. IFAC-PapersOnLine. 51(18). 156-161.

Schilling, G., Pantelides, C. C., 1996. A simple continuous-time process scheduling formulation and a novel solution algorithm. Computers and Chemical Engineering. 20(2). S1221-S1226.

Schrage, L., 1969. Solving resource-constrained network problems by implicit enumeration – nonpreemptive case. Operations research. 18(2). 263-278.

Seid, R., Majozi, T., 2012. A robust mathematical formulation for multipurpose batch plants. Chemical Engineering Science. 68(1). 36-53.

SelectUSA. 2020. Chemical Spotlight. The Chemical Industry in the United States. Available at: https://www.selectusa.gov/chemical-industry-united-states (Accessed: 4 August 2020)

Sha, D. Y., Hsu, C. Y., 2006. A hybrid particle swarm optimization for job shop scheduling problem. Computers and industrial engineering. 51(4). 791-808.

Shah, N., Pantelides, C. C., Sargent, R. W. H., 1993. A general algorithm for short-term scheduling of batch operations-II. Computational issues. Computers & Chemical Engineering. 17(2). 229-244.

Shaik, M. A., Floudas, C. A., 2007. Improved unit-specific event-based continuous-time model for short-term scheduling of continuous processes: Rigorous treatment of storage requirements. Industrial & Engineering Chemistry Research. 46(6). 1764-1779.

Shaik, M. A., Floudas, C. A., 2008. Unit-specific event-based continuous-time approach for short-term scheduling of batch plants using RTN framework. Computers and Chemical Engineering. 32(1-2). 260-274.

Shaik, M. A., Floudas, C. A., 2009. Novel Unified Modeling Approach for Short-Term Scheduling. Industrial and Engineering Chemistry Research. 48(6). 2947-2964.

Shaik, M. A., Floudas, C. A., Kallarth, J., Pitz, H. J., 2009. Production scheduling of a large-scale industrial continuous plant: Short-term and medium-term scheduling. Computers and chemical engineering. 33(3). 670-686.

Singer, M., 2001. Decomposition methods for large job-shops. Computers and operations research. 28(3). 193-207.

Stadler, H., Kigler, C, Meyr H., 2015. Supply chain management and advanced planning. 3rd edition. Berlin: Springer

Sundaramoorthy, A., Karimi, I. A., 2005. A simpler better slot-based continuous-time formulation for short-term scheduling in multipurpose batch plants. Chemical engineering science. 60(10). 2679-2702.

Sundaramoorthy, A., Maravelias C. T., 2008. Simultaneous batching and scheduling in multistage multiproduct processes. Industrial and engineering chemistry research. 47(5). 1546-1555.

Sundaramoorthy, A., Maravelias C. T., Prasad, P., 2009. Scheduling of multistage batch processes under utility constraints. Industrial and engineering chemistry research. 48(13). 6050-6058.

Sung, C., Maravelias, C. T., 2007. An attainable region approach for production planning and multiproduct processes. AIChE journal. 53(5). 1298-1315.

Tay, J. C., Wibowo, D., 2004. An effective chromosome representation for evolving flexible job shop schedules. Lecture notes in computer science. 3103. 210-221.

Vooradi, R., Shaik, M. A., 2012. Improved three-index unit-specific event-based model for short-term scheduling of batch plants. Computers and Chemical Engineering. 43(10). 148-172.

Vooradi, R., Shaik, M. A., 2013. Rigorous unit-specific event based model for short term scheduling of batch plants using conditional sequencing and unit-wait times. Industrial and Engineering research. 52(36) .12950-12792.

Wagner, H. M., 1959. An integer linear-programming model for machine scheduling. Naval research logistics quarterly. 6(2). 131-140.

Wang, H., Jiang, Z., Wang, Y., Zhang, H., Wang, Y., 2018. A two-stage optimization method for energy-saving flexible job-shop scheduling based on energy dynamic characterization. Journal of cleaner production. 188. 575-588.

Watanabe, M., Ida, K., Gen, M., 2005. A genetic algorithm with modified crossover operator and search area adaptation for the job-shop scheduling problem. Computers and industrial engineering. 48(4). 743-752.

Woolway M., Majozi T., 2018. A novel metaheuristic framework for the scheduling of multipurpose batch plants. Chemical engineering science. 192. 678-687.

Woolway, M., Majozi T., 2019. On the application of a metaheuristic suite with parallel implementations for the scheduling of multipurpose batch plants. Computers and Chemical engineering. 126. 371-390.

Wu, X., Sun, Y., 2018. A green scheduling algorithm for flexible job shop with energy-saving measures. Journal of cleaner production. 172. 3249-3264.

Yan, Z., Anke, X., Xiaohui, Z., Baofeng, G., 2013. A rolling horizon procedure with optimal operation assignment. Proceedings of the 23nd Chinese control conference. Xi'an, China. 2580-2584.

Yazdani, M., Amiri, M., Zandieh, M., 2010. Flexible job-shop scheduling with parallel variable neighborhood search algorithm. Expert systems with applications. 37(1). 678-687.

Zhang, G., Gao, L., Shi, Y., 2011. An effective genetic algorithm for the flexible job-shop scheduling problem. Expert systems with applications. 38(4). 3563-3573.

Zhang, L., Tang, Q., Wu, Z., Wang, F., 2017. Mathematical modelling and evolutionary generation of rule sets for energy-efficient flexible job shops. Energy. 138, 210-227.

Zhang, X., Sargent, R. W. H., 1996. The optimal operation of mixed production facilities-A general formulation and some approaches for the solution. Computers and Chemical Engineering. 20(6-7). 897-904.

Zhang, X., Sargent, R. W. H., 1996. The optimal operation of mixed production facilities – a general formulation and some approaches for the solution. Computers and chemical engineering. 20(6-7). 897-904.

Zhang, Z., Wu, L., Peng, T., Jia, S., 2019. An improved scheduling approach for minimizing total energy consumption and makespan in a flexible job shop environment. Sustainability. 11(1). 179.

Publications and presentations

Journal Publications

Nikolaos Rakovitis, Nan Zhang, Jie Li, Liping Zhang, A new approach for scheduling of multipurpose batch processes with unlimited intermediate storage policy, Frontiers of Chemical Engineering, 2019, 13, 784-802

Nikolaos Rakovitis, Nan Zhang, Jie Li, A novel unit-specific event-based formulation for short-term scheduling of multitasking processes in scientific service facilities, Computers and Chemical engineering, 2020, 133, 106626

Nikolaos Rakovitis, Yueting Pan, Nan Zhang, Jie Li, Giorgos Kopanos, Generic mathematical formulations for scheduling of multipurpose batch plants, AIChE Journal, 2020, under review

Nikolaos Rakovitis, Nan Zhang, Jie Li, Liping Zhang, Novel Approaches for Energy-Efficient Scheduling of Flexible Job-Shop Problems. to be submitted in energy

Nikolaos Rakovitis, N., Wan Mohd Azril bin Wan Hasnuddin, Nan Zhang, Jie Li, A Generic Approach for Scheduling of Semi-continuous and Continuous Processes. to be submitted to European journal of operational research

Conference publications

Nikolaos Rakovitis, Jie Li, Nan Zhang, New approaches for scheduling of multitasking multipurpose batch processes in scientific service facilities, Computer Aided Chemical Engineering, 2018, 43, 1033-1038

Nikolaos Rakovitis, Jie Li, Nan Zhang, A novel modelling approach to scheduling of multipurpose batch processes, Computer Aided Chemical Engineering, 2018, 44, 1333-1338

Nikolaos Rakovitis, Jie Li and Nan Zhang, A new approach for scheduling of multipurpose batch processes with unlimited intermediate storage policy, *2018 5th* International Conference on Control, Decision and Information Technologies (CoDIT), Thessaloniki, 2018, pp. 779-784, doi: 10.1109/CoDIT.2018.8394914.

Nikolaos Rakovitis, Jie Li, Nan Zhang, An improved approach to scheduling multipurpose batch processes with conditional sequencing, Computer Aided Chemical Engineering, 2019, 46, 1387-1392

Nikolaos Rakovitis, Dan Li, Nan Zhang, Jie Li, Liping Zhang, Xin Xiao, Novel Approaches for Energy-Efficient Flexible Job-Shop Scheduling Problems, Chemical engineering transactions, in press.

Conference presentations

Nikolaos Rakovitis, Jie Li, Nan Zhang, A new approach for scheduling of operations in scientific services facilities via multi-commodity flow, Presented at AIChE Annual Meeting, 29 October–3 November 2017, Minneapolis, Minnesota

Nikolaos Rakovitis, Jie Li, Nan Zhang, A Novel Mathematical Model for Short-Term and Medium-Term Scheduling of Multipurpose Batch Plants, Presented at AIChE Annual Meeting, 28 October–2 November 2018, Pittsburgh, Pennsylvania

Nikolaos Rakovitis, Jie Li, Nan Zhang, Liping Zhang, Novel Approach to Scheduling of Energy-Efficient Flexible Job Shops Presented at AIChE Annual Meeting, 10–15 November 2019, Orlando, Florida

Nikolaos Rakovitis, Jie Li, Nan Zhang, A novel approach for scheduling of operations in scientific services facility, Presented at ChemEngDayUK 2017 conference, 27-28 March 2017, Birmingham, United Kingdom

Nikolaos Rakovitis, Jie Li, Nan Zhang, A novel approach for optimal scheduling of multipurpose batch processes, Presented at ChemEngDayUK 2018 conference, 27-28 March 2018, Leeds, United Kingdom

Supplementary materials for Advances on mathematical modelling and optimization framework for process scheduling

List of Supplementary materials

Supplementary material 1: supplementary material for research contribution 2

Supplementary material 2: supplementary material for research contribution 4

Supplementary material 3: supplementary materials A, B for research contribution 5

Blank Page

Supplementary material 1: supplementary material for research contribution 2

Rakovitis, N., Pan Y, Zhang, N., Li, J. Kopanos, G. Generic mathematical formulations for scheduling of multipurpose batch plants, AIChE journal, submitted

Blank Page

Supplementary Material for

Generic mathematical formulations for scheduling of multipurpose batch plants

Nikolaos Rakovitis,[1] Yueting Pan,[1] Nan Zhang,[1] Jie Li,[1,**] and Giorgos Kopanos[2]

[1]Centre for Process Integration, Department of Chemical Engineering and Analytical Science, The University of Manchester, Manchester, M13 9PL, United Kingdom
[2]Flexciton Limited, London, 145 City Rd, Hoxton, London EC1V 1AZ

List of Tables

** To whom correspondence should be addressed. Email: jie.li-2@manchester.ac.uk. Tel: +44 (0) 161 306 8622

List of Figures

List of Appendices

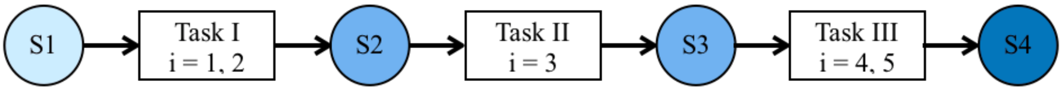

Figure S1 STN representation of Example 1

Table S1 Data for processing units for Example 1

Task	Processing Unit	α_i	β_i	B_i^{min}	B_i^{max}
I1	J1	1.333	0.01333	0	100
I2	J2	1.333	0.01333	0	150
I3	J3	1.000	0.00500	0	200
I4	J4	0.667	0.00445	0	150
I5	J5	0.667	0.00445	0	150

Table S2 Initial amount and maximum capacities (FIS) for Example 1

State	$ST0_s$	ST_s^{max}
S1	∞	∞
S2	0	200
S3	0	300
S4	0	∞

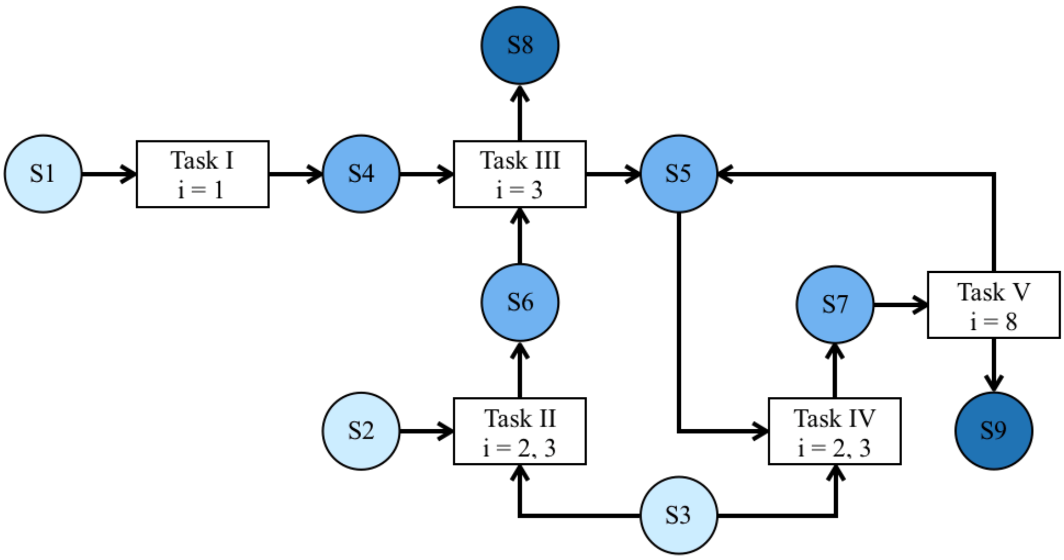

Figure S2 STN representation of Example 2

Table S3 Data for processing units for Example 2

Task	Processing Unit	α_i	β_i	B_i^{min}	B_i^{max}
I1	J1	0.667	0.00667	0	100
I2	J2	1.334	0.02664	0	50
I3	J3	1.334	0.01665	0	80
I4	J2	1.334	0.02664	0	50
I5	J3	1.334	0.01665	0	80
I6	J2	0.667	0.01332	0	50
I7	J3	0.667	0.008325	0	80
I8	J4	1.334	0.00666	0	200

Table S4 Initial amount and maximum capacities (FIS) for Example 2

State	$ST0_s$	ST_s^{max}
S1	∞	∞
S2	∞	∞
S3	∞	∞
S4	0	100
S5	0	200
S6	0	150
S7	0	200
S8	0	∞
S9	0	∞

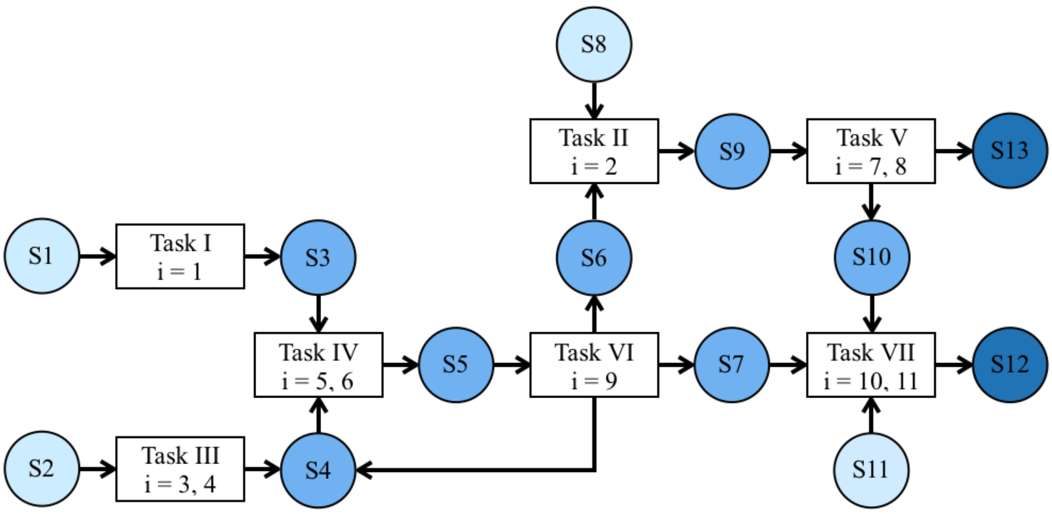

Figure S3 STN representation of Example 3

Table S5 Data for processing units for Example 3

Task	Processing Unit	α_i	β_i	B_i^{min}	B_i^{max}
I1	I1	0.667	0.00667	0	100
I2	I1	1.000	0.01000	0	100
I3	I2	1.333	0.01333	0	100
I4	I3	1.333	0.00889	0	150
I5	I2	0.667	0.00667	0	100
I6	I3	0.667	0.00445	0	150
I7	I2	1.333	0.01330	0	100
I8	I3	1.333	0.00889	0	150
I9	I4	2.000	0.00667	0	300
I10	I5	1.333	0.00667	20	200
I11	I6	1.333	0.00667	20	200

Table S6 Initial amount and maximum capacities (FIS) for Example 3

State	$ST0_s$	ST_s^{max}
S1	∞	∞
S2	∞	∞
S3	0	100
S4	0	100
S5	0	300
S6	50	150
S7	50	150
S8	∞	∞
S9	0	150
S10	0	150
S11	∞	∞
S12	0	∞
S13	0	∞

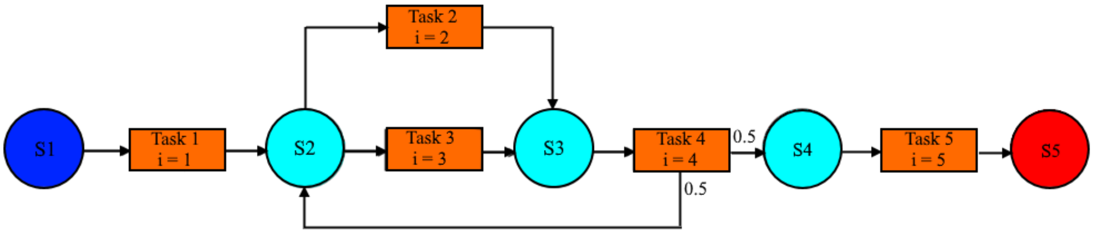

Figure S4 STN representation of Example 4

Table S7 Data for processing units for Example 4

Task	Processing Unit	α_i	β_i	B_i^{min}	B_i^{max}
I1	J1	0.9500	0.010000	0	6
I2	J2	2.9400	0.020000	0	3
I3	J3	2.4800	0.010000	0	2
I4	J4	4.4666	0.006680	0	6
I5	J5	1.9663	0.013348	0	8

Table S8 Initial amount and maximum capacities (FIS) for Example 4

State	$ST0_s$	ST_s^{max}
S1	∞	∞
S2	0	3
S3	0	4
S4	0	8
S5	∞	∞

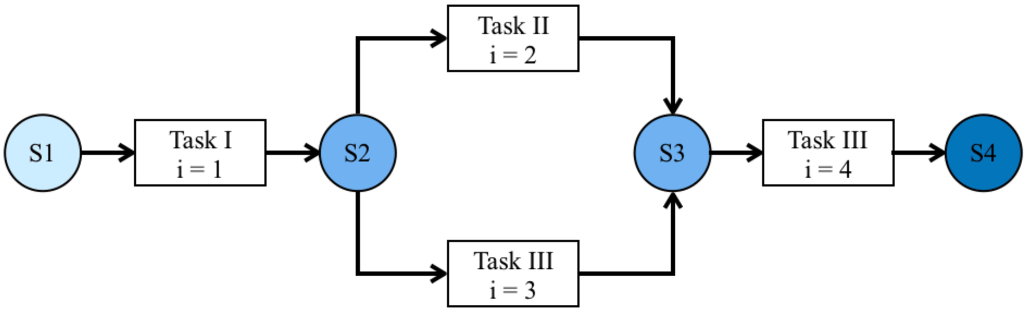

Figure S5 STN representation of Example 5

Table S9 Data for processing units for Example 5

Task	Processing Unit	α_i	β_i	B_i^{min}	B_i^{max}
1	1	1.000	0	0	10
2	2	3.000	0	0	4
3	3	1.000	0	0	2
4	4	2.000	0	0	10

Table S10 Initial amount and maximum capacities (FIS) for Example 5

State	$ST0_s$	ST_s^{max}
S1	∞	∞
S2	0	6
S3	0	4
S4	∞	∞

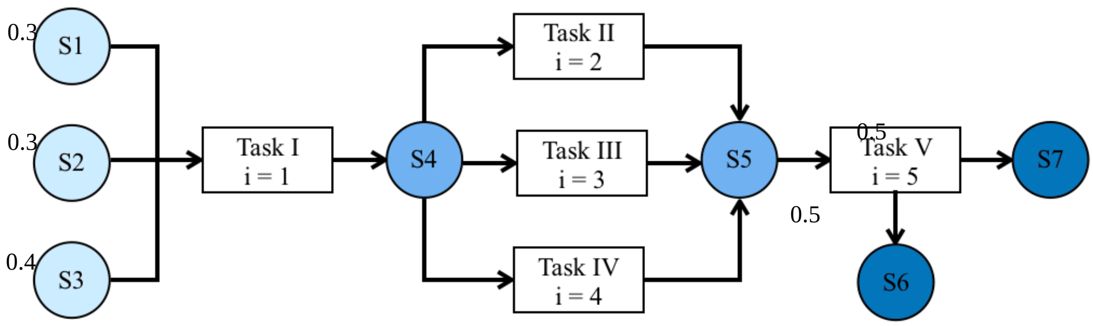

Figure S6 STN representation of Example 6

Table S11 Data for processing units for Example 6

Task	Processing Unit	α_i	β_i	B_i^{min}	B_i^{max}
1	1	1.500	0	0	150
2	2	4.500	0	0	60
3	3	1.500	0	0	30
4	4	1.500	0	0	30
5	5	3.000	0	0	150

Table S12 Initial amount and maximum capacities (FIS) for Example 6

State	$ST0_s$	ST_s^{max}
S1	∞	∞
S2	∞	∞
S3	∞	∞
S4	0	60
S5	0	60
S6	∞	∞
S7	∞	∞

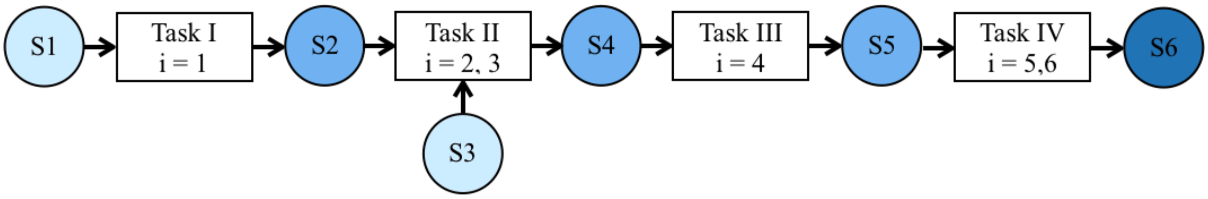

Figure S7 STN representation of Example 7

Table S13 Data for processing units for Example 7

Task	Processing Unit	α_i	β_i	B_i^{min}	B_i^{max}
1	1	17.3333	0.866	0	20
2	2	2.667	0.133	0	20
3	3	2.667	0.133	0	20
4	4	4.000	0.200	0	20
5	5	5.333	0.266	0	20
6	6	5.333	0.266	0	20

Table S14 Initial amount and Maximum capacities (FIS) for Example 7

State	$ST0_s$	ST_s^{max}
S1	∞	∞
S2	0	100
S3	∞	∞
S4	0	100
S5	0	100
S6	∞	∞

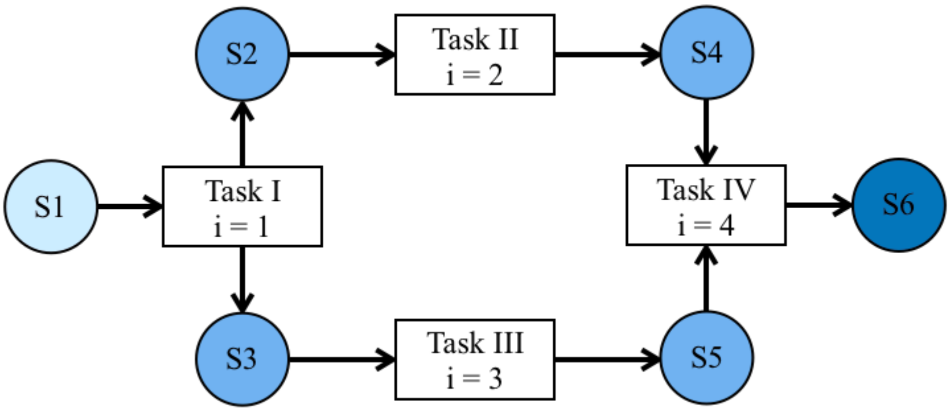

Figure S8 STN representation of Example 8, 9

Table S15 Data for processing units for Example 8

Task	Processing Unit	α_i	β_i	B_i^{min}	B_i^{max}
1	1	1.666	0.03335	0	40
2	2	2.333	0.08335	0	20
3	3	0.333	0.06800	0	2.5
4	4	2.667	0.008325	0	40

Table S16 Initial amount and Maximum capacities (FIS) for Example 8

State	$ST0_s$	ST_s^{max}
S1	∞	∞
S2	0	10
S3	0	17.5
S4	0	10
S5	0	18
S6	∞	∞

Table S17 Data for processing units for Example 9

Task	Processing Unit	α_i	β_i	B_i^{min}	B_i^{max}
1	1	1.666	0.03335	0	40
2	2	2.333	0.08335	0	20
3	3	0.333	0.06800	0	2.5
4	4	2.667	0.008325	0	40

Table S18 Initial amount and maximum capacities (FIS) for Example 9

State	$ST0_s$	ST_s^{max}
S1	∞	∞
S2	0	10
S3	0	17.5
S4	0	10
S5	0	18
S6	∞	∞

Appendix A. Nomenclature

Indices

i, i': tasks

j, j': units

n, n', n'': event points

s: states

Sets

I: tasks

$\mathbf{I}_j$: tasks that can be performed in unit j

$\mathbf{I}_s$: tasks that produce/consume state s

$\mathbf{I}_s^c$: tasks that consume state s

$\mathbf{I}_s^P$: tasks that produce state s

$\mathbf{I}^R$: tasks considered as recycling tasks

J: units

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

$\mathbf{J}_s$: units that produce/consume state s

N: event points

S: states

$\mathbf{S}^{FIS}$: states with unlimited intermediate storage policy

$\mathbf{S}^P$: states that are final products

$\mathbf{S}^{IN}$: states that are intermediate products

$\mathbf{S}^{\mathrm{R}}$: states that are raw materials

$\mathbf{S}^{UIS}$: states with unlimited intermediate storage policy

Parameters

B_{ij}^{max}: maximum batch size of task i processed in unit j

B_{ij}^{min} : minimum batch size of task i processed in unit j

D_s: demand of state s

H: scheduling horizon

M: big-M value

P_s: price of state s

$ST0_s$: initial amount of state s

$ST_s^{\max}$: maximum capacity of state s (for states with FIS policy)

α_{ij}: coefficient of constant term of processing time of task i in unit j

β_{ij}: coefficient of variable term of processing time of task i in unit j

Δn: maximum number of event points that task i is allowed to be active

ρ_{sij}: portion of state s consumed/produced by task i processed in unit j

Binary variables

$w_{ijnn'}$: binary variable which takes the value 1 if task i is processed in unit j from event point n to $n' \geq n$

ys_{ijn}: binary variable which takes the value 1 if there is any amount of materials stored in unit j at event point n, which were previously produced by task i processed in unit j at event point $n' < n$

$zI_{jj'\mathrm{n}}$: binary variable which takes the value 1 if there is indirect material transfer between unit j and j'

$zD_{jj'\mathrm{n}}$: binary variable which takes the value 1 if there is indirect material transfer between unit j and j'

Continuous variables

$b_{ijnn'}$: amount of materials that are processed in unit j processing task i from time event point n to time event point $n' \geq n$

bs_{ijn}: amount of materials stored in unit j at event point n, which were previously produced by task i processed in unit j at event point $n' < n$

$bTi_{iji'j'n}$: amount of materials, which produced by task i processed in unit j, were indirectly transferred to unit j' which consumes task i' at event point n

$bTd_{iji'j'n}$: amount of materials, which produced by task i processed in unit j, were indirectly transferred to unit j' which consumes task i' at event point n

ST_{sn}: amount of state s that has to be stored at time event point n

T_{sjn}: time that state s produced in unit j is available to be consumed at event point n

T_{jn}^{s}: start time of unit j at time event point n

T_{jn}^{f}: end time of unit j at time event point n

Appendix B. Modified short-term model and Rolling horizon decomposition approach of Janak *et al.* (2006)

B1. Rolling-horizon Level 1 Formulation

Indices

d: days

s, s': states

Sets

D: days

I: tasks

$\mathbf{I}_s$: tasks that produce/consume state s

J: units

$\mathbf{J}_i$: units that process task i

$\mathbf{S}^F$: final products (materials produced by a task type 6)

$\mathbf{S}^{IN}$: intermediate products

$\boldsymbol{S}^{PIN}$: states which are either final or intermediate products

Parameters

a_{sd}: amount of ahead time required to fulfil demand of state s which is due of day d

D^f: last day of the previous sub-horizon. Zero for the first sub-horizon

$fill_{ss'}$: parameter to relate final product s with an intermediate product s' before being processes in a type 6 task

$last_s$: parameter to indicate whether state s was still being produced (the relevant task is still being processed by a unit) at the end of the previous sub-horizon

LI_1, LI_2: weights for the objective function of the level 1 decomposition formulation

$nepd$: number of event point used for each day

$praw_{ss'}$: parameter to relate final product s with raw material or intermediate product s'

r_{sd}: demand for product s due on day d

$ubin$: maximum number of binary variables in each sub-problem

$uprd$: upper amount of production in the current sub-horizon

wt_s: weight of product s in the objective function

Y_{sd}: amount of state s that has to be processed in a type 5 task within the period $[d,d+3]$

Binary Variables

day_d: 1 if day d is included in the current horizon

pr_s: 1 if state s is included in the current horizon

Continuous variables

slbin: slack variable to allow extra days to be included in the current sub-horizon

$prday_{sd}$: bilinear term $pr_s \cdot day_d$

slackbin: Slack variable to allow extra days to be added in the current horizon

z: Objective value

Constraints

$$day_d \geq day_{(d+1)} \qquad \forall d,\, d > D^f,\, d < |D| \tag{B.1}$$

$$pr_s \geq day_d \qquad \forall d,\, d > D^f,\, d < |D|,\, s \in D^f,\, r_{sd} > 0 \tag{B.2}$$

$$pr_s \geq day_{(d-a_{s,d})} \qquad \forall d,\, d > D^f,\, d < |D|,\, s \in \mathbf{S}^F,\, a_{sd} > 0,\, d > a_{sd} \tag{B.3}$$

$$pr_s \leq \sum_{d > D^f, r_{s,d} > 0} day_d + \sum_{d > D^f, a_{s,d}, r_{s,d} > 0} day_{(d-a_{s,d})} + \sum_{d > D^f, Y_{s,d} > 0} day_d \qquad \forall s \in S^f,\, \text{last}_s = 0 \tag{B.4}$$

$$pr_{s'} \geq pr_s \qquad \forall s \in \boldsymbol{S}^F,\; s' \in \boldsymbol{S}^{IN},\; \text{praw}_{s,s'} > 0 \tag{B.5}$$

$$pr_{s'} \leq \sum_{s \in S^f, fill_{s,s'} > 0} pr_s + \sum_{s \in S^p, praw_{s,s'} > 0} pr_s \qquad \forall s' \in \boldsymbol{S}^{IN} \quad last_{s'} > 0 \tag{B.6}$$

$$pr_s \geq last_s \qquad \forall s \in \boldsymbol{S}^{PIN} \tag{B.7}$$

$$nepd \cdot \left(\sum_{d > D^f} \sum_{s \in S^f} \sum_{i \in I^s} \sum_{j \in J^i} pr_s \cdot day_d \right) \leq ubin + slbin \tag{B.8}$$

$$\sum_{d > D^f} \sum_{s \in \mathbf{S}^F} \sum_{i \in \mathbf{I}_s} \sum_{j \in \mathbf{J}_i} pr_s \cdot day_d \cdot r_{sd} \leq uprd \tag{B.9}$$

$$prday_{sd} \geq pr_s + day_d + 1 \qquad \forall s, d \tag{B.10}$$

$$prday_{sd} \leq pr_s \qquad \forall s, d \tag{B.11}$$

$$prday_{sd} \geq day_d \qquad \forall s, d \tag{B.12}$$

$$nepd \cdot \left(\sum_{d > D^f} \sum_{s \in \mathbf{S}^f} \sum_{i \in \mathbf{I}_s} \sum_{j \in \mathbf{J}_i} prday_{sd} \right) \leq upper + slackbin \tag{B.13}$$

$$\sum_{d > D^f} \sum_{s \in \mathbf{S}^F} \sum_{i \in \mathbf{I}_s} \sum_{j \in \mathbf{J}_i} prday_{sd} \cdot r_{sd} \leq uprd \tag{B.14}$$

$$z = \sum_{d > D^f} day_d + LI_1 \sum_{s \in \mathbf{S}^{PIN}} wt_s \cdot pprod_s + LI_2 slackbin \tag{B.15}$$

B2. Rolling horizon Level 2 Formulation

Indices

i: tasks

s: states

Sets

$\mathbf{I}_j$: tasks that can be processed in unit j

$\mathbf{I}^{T1}$: set which includes the type 1 tasks

$\mathbf{J}^{T1}$: units that process type 1 tasks

$\mathbf{S}^{cat1}$: final product type 1

$\mathbf{S}^{cat2}$: final product type 2

$\mathbf{S}_i$: states that were produced/consumed by task i

$\boldsymbol{S}^{PIN}$: states which are either final or intermediate products

Parameters

$B_{ij}^{\max}$: maximum capacity of task i in unit j

Dem_s: demand of state s

$last_s$: parameter to indicate whether state s was still being produced (the relevant task is still being processed by a unit) at the end of the previous sub-horizon

lower: lower bound on the utilization level of units processing type 1 tasks

$pack_{ss'}$: parameter that relates final products of type 1 with final products of type 2

$pt_i^{\min}$: minimum processing time of task i in all available processing units

sl_s: parameter to indicate whether state s is included in the current horizon because it has demands in the horizon or needs to be processed ahead of time to fulfill demands at a later horizon

$ST_s^{\max}$: maximum capacity of state s (for states with FIS policy)

Binary variables

$react_i$: 1 if type 1 task i is included in the scheduling model

Continuous variables

z: objective value

Constraints

$$react_i \le sl_s \qquad \forall\ i' \in \mathbf{I}^{T1},\ s \in \mathbf{S}^{PIN}, \mathbf{S}_i \qquad \text{(B.16)}$$

$$react_i \ge last_s \qquad \forall\ i' \in \mathbf{I}^{T1},\ s \in \mathbf{S}^{PIN}, \mathbf{S}_i \qquad \text{(B.17)}$$

$$\sum_{i\in\mathbf{I}^{T1}} react_i \cdot pt_i^{\min} \frac{\left|\mathbf{J}_i\right|}{\sum_{j\in\mathbf{J}_i} B_{ij}^{\max}} \cdot \left[\sum_{\substack{s\in\left(\mathbf{S}^{cat\,2}\cap\mathbf{S}_i\right)\\ sl_s+last_s=1\\ D_s>0}} \sum_{\substack{s'\in\left(\mathbf{S}^{cat1}\cap\mathbf{S}_i\right)\\ sl_s+last_s=0\\ pack_{s's}>0}} Dem_s + \sum_{\substack{s'\in\left(\mathbf{S}^{cat1}\cap S_i\right)\\ sl_s+last_s=0\\ D_s>0}} st_s^{\max} + \sum_{\substack{s\in\left(\mathbf{S}^{cat\,2}\cap\mathbf{S}_i\right)\\ sl_s+last_s=0\\ D_s>0}} \sum_{\substack{s'\in\left(\mathbf{S}^{cat1}\cap\mathbf{S}_i\right)\\ sl_s+last_s=0\\ pack_{s's}>0}} st_s^{\max} \right] \geq$$

$$\geq lower \cdot H \cdot \left|\mathbf{J}^{T1}\right| \tag{B.18}$$

$$z = \sum_{i\in\mathbf{I}^{T1}} react_i \tag{B.19}$$

B3. Modified short-term model of Janak et al. (2006)

Indices

i, i': tasks

j, j': units

n: event points

Sets

D: days

$\mathbf{D}^{in}$: days that included in the current horizon

I: tasks

$\mathbf{I}_j$: tasks that can be performed in unit j

$\mathbf{I}_s^C$: tasks that consume state s

$\mathbf{I}^{in}$: tasks included in the current horizon

$\mathbf{I}_k$: tasks related with order k

$\mathbf{I}_s^P$: tasks that produce state s

$\mathbf{I}^R$: tasks considered as recycling tasks

$\mathbf{I}^{T6b}$: type 6 tasks that produce category 1 products

J: units

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

$\mathbf{J}_s$: units that produce/consume state s

$\mathbf{J}^{T4}$: units that process task types 4a and 4b

$\mathbf{J}^{T6}$: units that process task type 6

K:orders

$\mathbf{K}^{in}$: orders included in the current horizon

$\mathbf{K}_i$: orders related with task i

$\mathbf{K}_s$: orders that are related with state s

N: event points

S: states

$\mathbf{S}^{cat1}$: category 1 final products

$\mathbf{S}^{FIS}$: states with finite intermediate storage policy

$\mathbf{S}^{IN}$: states included in the current horizon

$\mathbf{S}^{st}$: states with no intermediate storage policy

$\mathbf{S}^{unl}$: states with unlimited intermediate storage policy

Parameters

B_s^{max}: maximum available batch that process state s

B_s^{min}: minimum available batch that process state s

B_{ij}^{max}: maximum capacity of task i in unit j

B_{ij}^{min}: minimum capacity of task i in unit j

Dem_s: demand of state s

Dem_s^{raw}: demand of raw material state s

$duek_{ksd}$: due date of order k for state s on day d

H: scheduling horizon

M: big-M value

mtasks: minimum number of tasks that are allowed to be active in the units processing type 1 tasks

$N^{\max}$: event points within the current scheduling horizon

$prior_s$: priority of state s

$prior_s^{raw}$: priority of state s

rk_{ksd}: amount of order k for state s on day d

sl_s: indicator that state s is included in the current horizon because there is demand in this horizon or in the ahead of time

$ST_s^{\max}$: maximum storage capacity for state s

$ST_s^{\min}$: minimum amount of state s that should be stored at any point

$ST0_s$: initial amount of state s at the beginning of the current scheduling horizon

α, β, γ, δ, φ, ξ, λ, η, w, w_4: weights for objective function

α_{ij} : coefficient of constant term of processing time of task i in unit j

β_{ij} : coefficient of variable term of processing time of task i in unit j

ρ_{sij} : portion of state s consumed/produced by task i processed in unit j

Binary Variables

wv_{ijn}: 1 if task i is processed in unit j at event point n

y_{ikn}: binary variable which assigns the delivery of order k through task i

ys_{ijn}: binary variable which takes the value 1 if there is any amount of materials stored in unit j at event point n, which were previously produced by task i processed in unit j at event point $n' < n$

$zI_{jj'n}$: binary variable which takes the value 1 if there is indirect material transfer between unit j and j'

$zD_{j,j',n}$: binary variable which takes the value 1 if there is indirect material transfer between unit j and j'

Continuous variables

b_{ijn}: amount of materials processed in unit j processing task i at event point n

bs_{ijn} : amount of materials stored in unit j at event point n, which were previously produced by task i processed in unit j at event point $n' < n$

$bTi_{iji'j'n}$: amount of materials, which produced by task i processed in unit j, were indirectly transferred to unit j' which consumes task i' at event point n

$bTd_{iji'j'n}$: amount of materials, which produced by task i processed in unit j, were indirectly transferred to unit j' which consumes task i' at event point n

D_{sn}: amount of state s delivered at event point n

D_{sn}^{f}: amount of state s delivered after the last event point

kD_{ksn}: amount of state s delivered at event point n in order k

kD_{ksn}^{f}: amount of state s delivered after the last event point in order k

$sla1_{ksd}$: amount of state s due on day d of order k that is not delivered

$sla2_{ksd}$: amount of state s due on day d of order k that is overdelivered

$slcap_{sn}$: amount of state s that cannot be stored in storage tanks

sll_s: amount of state s due in the current horizon but not made

sll_s^{raw}: amount of raw material state s due in the current horizon but not made

$slt1_{ksd}$: amount of time state s is due on day d for order k is late

$slt2_{ksd}$: amount of time state s is due on day d for order k is early

sl_{sn}^{cap}: Amount of materials of state s required to fulfil the minimum amount requirement at event point n

sl_k^{order}: 0-1 continuous variable that indicates whether order k is fulfilled

ST_{sn}: amount of state s stored during event point n

$ST0_s$: amount of state s at the end of the current scheduling horizon

T_{sjn} : time that state s produced in unit j is available to be consumed at event point n

T_{jn}^{s}: start time of unit j at time event point n

T_{jn}^{f}: end time of unit j at time event point n

tt_{jn}^{s}: start time of unit j processing an active task i at time event point n

term1-term9: objective terms

z: objective values

Constraints

Allocation constraints

$$\sum_{i\in\mathbf{I}_j} wv_{ijn} \le 1 \qquad \forall\, j, n \le N^{max} \qquad \text{(B.20)}$$

Capacity constraints

$$B_{ij}^{\min}\cdot wv_{ijn} \le b_{ijn} \le B_{ij}^{\max}\cdot wv_{ijn} \qquad \forall\, j,\, i\in\mathbf{I}_j,\, n\le N^{max} \qquad \text{(B.21)}$$

Storage constraints

$$ST_{sn} \le ST_s^{\max} \qquad \forall\, s\in\mathbf{S}^{FIS},\ \mathrm{n} \qquad \text{(B.22)}$$

$$ST_{sn} \ge ST_s^{\min} + sl_{sn}^{cap} \qquad \forall\, s\in\mathbf{S}^{FIS},\ \mathrm{n} \qquad \text{(B.23)}$$

Material balance constraints

$$ST_{sn} = ST_{sn-1} + \sum_{i\in\mathbf{I}_s^P\setminus\mathbf{I}^R}\ \sum_{j\in(\mathbf{J}_s\cap\mathbf{J}_i)} \rho_{sij}\cdot b_{ijn} + \sum_{i\in(\mathbf{I}_s^P\cap\mathbf{I}^R)}\ \sum_{j\in(\mathbf{J}_s\cap\mathbf{J}_i)} \rho_{sij}\cdot b_{ij(n-1)} + \sum_{i\in\mathbf{I}_s^C}\ \sum_{j\in(\mathbf{J}_s\cap\mathbf{J}_i)} \rho_{sij}\cdot b_{ijn} - D_{sn}$$

$$\forall s,\, n>1 \qquad \text{(B.24)}$$

$$ST_{sn} = ST0_s + \sum_{i\in\mathbf{I}_s^P\setminus\mathbf{I}^R}\ \sum_{j\in(\mathbf{J}_s\cap\mathbf{J}_i)} \rho_{sij}\cdot b_{ijn} + \sum_{i\in\mathbf{I}_s^C}\ \sum_{j\in(\mathbf{J}_s\cap\mathbf{J}_i)} \rho_{sij}\cdot b_{ijn} - D_{sn} \qquad \forall s,\, n=1 \qquad \text{(B.25)}$$

Duration constraints

$$T_{jn}^{\mathrm{f}} \ge T_{jn}^{\mathrm{s}} + \sum_{i\in\mathbf{I}_j}\left(\alpha_{ij}\cdot wv_{ijn} + \beta_{ij}\cdot b_{ijn}\right) \qquad \forall j,\, n\le N^{max} \qquad \text{(B.26)}$$

$$T_{jn}^{\mathrm{f}} = H \qquad \forall s\in(\mathbf{S}^{in}\cap\mathbf{S}^{st})\setminus\mathbf{S}^{unl},\, j\in\mathbf{J}^{T4},\ \sum_{i\in(\mathbf{I}_j\cap\mathbf{I}_s^P)} \rho_{sij} > 0\ \ n = N^{max} \qquad \text{(B.27)}$$

$$T_{sjn} \ge T_{jn}^{\mathrm{f}} - M\left(1 - \sum_{i\in(\mathbf{I}_j\cap\mathbf{I}_s^P)} wv_{ijn}\right) \qquad \forall s\in\mathbf{S}^{in},\, j\in\mathbf{J}_s,\ \sum_{i\in(\mathbf{I}_j\cap\mathbf{I}_s^P)} \rho_{sij} > 0,\, n \qquad \text{(B.28)}$$

$$T_{sjn} \le T_{j'n}^{\mathrm{s}} + M\left(2 - \sum_{i'\in(\mathbf{I}_{j'}\cap\mathbf{I}_s^C)} wv_{i'j'n} - zI_{jj'n}\right)$$

$$\forall s\in\mathbf{S}^{in},\, j\in\mathbf{J}_s,\ \sum_{i\in(\mathbf{I}_j\cap\mathbf{I}_s^P)\setminus\mathbf{I}^R} \rho_{sij} > 0,\, j\ne j',\, j'\in\mathbf{J}_s,\ \sum_{i'\in(\mathbf{I}_{j'}\cap\mathbf{I}_s^C)} \rho_{si'j'} < 0,\, n \qquad \text{(B.28)}$$

$$T_{sjn} \le T_{j'(n+1)}^{\mathrm{s}} + M\left(2 - \sum_{i'\in(\mathbf{I}_{j'}\cap\mathbf{I}_s^C)} wv_{i'j'(n+1)} - zI_{jj'n}\right)$$

$$\forall s\in\mathbf{S}^{IN},\, j\in\mathbf{J}_s,\ \sum_{i\in(\mathbf{I}_j\cap\mathbf{I}_s^P\cap\mathbf{I}^R)} \rho_{sij} > 0,\, j\ne j',\, j'\in\mathbf{J}_s,\ \sum_{i'\in(\mathbf{I}_{j'}\cap\mathbf{I}_s^C)} \rho_{si'j'} < 0,\, n < N^{\max} \qquad \text{(B.29)}$$

$$-\sum_{j'\in\mathbf{J}_s}\ \sum_{i'\in(\mathbf{I}_s^C\cap\mathbf{I}_{j'})} \rho_{si'j'}\cdot b_{i'j'n} \le ST_{s(n-1)} + \sum_{j\in\mathbf{J}_s}\ \sum_{j'\in\mathbf{J}_s}\ \sum_{i\in(\mathbf{I}_j\cap\mathbf{I}_s^P)}\ \sum_{i'\in(\mathbf{I}_{j'}\cap\mathbf{I}_s^C)} bTi_{iji'j'n} \qquad \forall s\in\mathbf{S}^{IN},\, n \qquad \text{(B.30)}$$

$$\rho_{sij}\cdot b_{ijn} \ge \sum_{j'\in\mathbf{J}_s}\ \sum_{i'\in(\mathbf{I}_{j'}\cap\mathbf{I}_s^C)} bTi_{iji'j'n} \qquad \forall s\in\mathbf{S}^{IN},\, j\in\mathbf{J}_s,\, i\in(\mathbf{I}_j\cap\mathbf{I}_S^P)\setminus\mathbf{I}^R,\, n \qquad \text{(B.31)}$$

$$\rho_{sij} \cdot b_{ij(n-1)} \geq \sum_{j' \in \mathbf{J}_s} \sum_{i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C)} bTi_{iji'j'n} \qquad \forall s \in \mathbf{S}^{IN}, j \in \mathbf{J}_s, i \in (\mathbf{I}_j \cap \mathbf{I}_s^P \cap \mathbf{I}^R), n > 1 \tag{B.32}$$

$$-\rho_{si'j'} \cdot \sum_{n \leq n' \leq n+\Delta n} b_{i'j'nn'} \geq \sum_{j} \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P)} bTi_{iji'j'n} \qquad \forall s \in \mathbf{S}^{IN}, j' \in \mathbf{J}_s, i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C), n \tag{B.33}$$

$$\sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P)} \sum_{i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C)} bTi_{iji'j'n} \leq \min\left[B_j^{\max}, B_{j'}^{\max}\right] \cdot zI_{jj'n} \qquad \forall s \in \mathbf{S}^{IN}, j \neq j', j \in \mathbf{J}_s, j' \in \mathbf{J}_s, n \tag{B.34}$$

where $B_j^{max} = \max_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P)} [B_{ij}^{max}]$ and $B_{j'}^{max} = \max_{i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C)} [B_{ij'}^{max}]$.

$$T_{sjn} \geq T_{jn}^{\mathrm{f}} - M\left(1 - \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P)} wv_{ijn}\right) \qquad \forall s \in \mathbf{S}^{IN}, j \in \mathbf{J}_s, \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P)} \rho_{sij} > 0, n \tag{B.35}$$

$$T_{sjn} \leq T_{j'n}^{\mathrm{s}} + M\left(1 - zI_{jj'n}\right)$$

$$\forall s \in \mathbf{S}^{IN}, j \in \mathbf{J}_s, \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P) \backslash \mathbf{I}_R} \rho_{sij} > 0, j \neq j', j' \in \mathbf{J}_s, \sum_{i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C)} \rho_{si'j'} < 0, n \tag{B.36}$$

$$T_{sjn} \leq T_{j'(n+1)}^{\mathrm{s}} + M\left(1 - zI_{jj'n}\right)$$

$$\forall s \in \mathbf{S}^{IN}, j \in \mathbf{J}_s, \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P \cap \mathbf{I}_R)} \rho_{sij} > 0, j \neq j', j' \in \mathbf{J}_s, \sum_{i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C)} \rho_{si'j'} < 0, n < N \tag{B.37}$$

$$T_{sjn} \leq T_{j'(n+1)}^{\mathrm{s}} + M\left(1 - \sum_{i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C)} wv_{i'j'(n+1)}\right)$$

$$\forall s \in \mathbf{S}^{IN}, j \in \mathbf{J}_s, \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P) \backslash \mathbf{I}_R} \rho_{sij} > 0, j \neq j', j' \in \mathbf{J}_s, \sum_{i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C)} \rho_{si'j'} < 0, n < N \tag{B.38}$$

$$T_{sjn} \leq T_{j'(n+2)}^{\mathrm{s}} + M\left(1 - \sum_{i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C)} wv_{i'j'(n+2)}\right)$$

$$\forall s \in \mathbf{S}^{IN}, j \in \mathbf{J}_s, \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P \cap \mathbf{I}^R)} \rho_{sij} > 0, j \neq j', j' \in \mathbf{J}_s, \sum_{i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C)} \rho_{si'j'} < 0, n < N-1 \tag{B.39}$$

$$T_{sj(n+1)} \geq T_{sjn} \qquad \forall s \in \mathbf{S}^{IN}, j \in \mathbf{J}_s, n < N \tag{B.40}$$

$$bs_{ijn} \leq \sum_{n-1-\Delta n \leq n' \leq n} \left(\rho_{sij} \cdot b_{ijn'(n-1)}\right) + bs_{ij(n-1)}$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}), j \in \mathbf{J}_s, i \in (\mathbf{I}_j \cap \mathbf{I}_s^P), n > 1 \tag{B.41}$$

$$bs_{ijn} \geq bs_{ij(n-1)} - \sum_{j' \in \mathbf{J}_s} \sum_{i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C)} bTd_{iji'j'n}$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}), j \in \mathbf{J}_s, i \in (\mathbf{I}_j \cap \mathbf{I}_s^P) \backslash \mathbf{I}^R, n > 1 \tag{B.42}$$

$$bs_{ijn} \geq bs_{ij(n-1)} - \sum_{j' \in \mathbf{J}_s} \sum_{i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C)} bTd_{iji'j'(n+1)}$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}),\ j \in \mathbf{J}_s,\ i \in (\mathbf{I}_j \cap \mathbf{I}_s^P \cap \mathbf{I}^R),\ 1 < n < N \tag{B.43}$$

$$bs_{ijn} \le B_{ij}^{\max} \cdot ys_{ijn} \qquad \forall j,\ i \in \mathbf{I}_j,\ n \tag{B.44}$$

$$\sum_{i \in \mathbf{I}_j} ys_{ijn} \le 1 - \sum_{i \in \mathbf{I}_j} w_{ijn} \qquad \forall j,\ n \tag{B.45}$$

$$\sum_{j \in \mathbf{J}_s} \sum_{i \in \left(\mathbf{I}_j \cap \mathbf{I}_s^P\right) \setminus \mathbf{I}^R} \left(\rho_{sij} \cdot b_{ijn}\right) + ST_{s(n-1)} \le ST_s^{\max} + \sum_{j \in \mathbf{J}_s} \sum_{j' \in \mathbf{J}_s} \sum_{i \in \left(\mathbf{I}_{j'} \cap \mathbf{I}_s^P\right)} \sum_{i' \in \left(\mathbf{I}_{j'} \cap \mathbf{I}_s^C\right),} bTd_{iji'j'n} + \sum_{j \in \mathbf{J}_s} \sum_{i \in \left(\mathbf{I}_j \cap \mathbf{I}_s^P\right) \setminus \mathbf{I}^R} bs_{ij(n+1)}$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}),\ n \tag{B.46}$$

$$\sum_{j \in \mathbf{J}_s} \sum_{i \in \left(\mathbf{I}_j \cap \mathbf{I}_s^P \cap \mathbf{I}^R\right)} \left(\rho_{sij} \cdot b_{ij(n-1)}\right) + ST_{s(n-1)} \le ST_s^{\max} + \sum_{j \in \mathbf{J}_s} \sum_{j' \in \mathbf{J}_s} \sum_{i \in \left(\mathbf{I}_j \cap \mathbf{I}_s^P\right)} \sum_{i' \in \left(\mathbf{I}_{j'} \cap \mathbf{I}_s^C\right)} bTd_{iji'j'n} + \sum_{j \in \mathbf{J}_s} \sum_{i \in \left(\mathbf{I}_j \cap \mathbf{I}_s^P \cap \mathbf{I}^R\right)} bs_{ijn}$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}),\ n > 1 \tag{B.47}$$

$$\sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P)} \sum_{i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C)} bTd_{iji'j'n} \le \min\left[B_j^{\max}, B_{j'}^{\max}\right] \cdot zD_{jj'n}$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}),\ j \neq j',\ j \in \mathbf{J}_s,\ j' \in \mathbf{J}_s,\ n \tag{B.48}$$

where $B_j^{max} = \max_{i \in (\mathbf{I}_j \cap \mathbf{I}_S^P)} [B_{ij}^{max}]$ and $B_{j'}^{max} = \max_{i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_S^C)} [B_{ij'}^{max}]$.

$$\rho_{sij} \cdot \sum_{n - \Delta n \le n' \le n} b_{ijn'n} + bs_{ijn} \ge \sum_{j' \in \mathbf{J}_s} \sum_{i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C)} bTd_{iji'j'n}$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}),\ j \in \mathbf{J}_s,\ i \in (\mathbf{I}_j \cap \mathbf{I}_S^P) \setminus \mathbf{I}^R,\ n \tag{B.49}$$

$$\rho_{sij} \cdot \sum_{n-1-\Delta n \le n' \le n-1} b_{ijn'(n-1)} + bs_{ij(n-1)} \ge \sum_{j' \in \mathbf{J}_s} \sum_{i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C)} bTd_{iji'j'n}$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}),\ j \in \mathbf{J}_s,\ i \in (\mathbf{I}_j \cap \mathbf{I}_S^P \cap \mathbf{I}^R),\ n > 1 \tag{B.50}$$

$$-\rho_{si'j'} \cdot \sum_{n \le n' \le n + \Delta n} b_{i'j'nn'} \ge \sum_{j} \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P)} bTd_{iji'j'n}$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}),\ j' \in \mathbf{J}_s,\ i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_S^C),\ n \tag{B.51}$$

$$T_{j'(n-1)}^{\mathrm{f}} \le T_{jn}^{\mathrm{f}} + M\left(1 - zD_{jj'n}\right)$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}),\ j \in \mathbf{J}_s,\ \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P) \setminus \mathbf{I}^R} \rho_{sij} > 0,\ j \neq j',\ j' \in \mathbf{J}_s,\ \sum_{i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C)} \rho_{si'j'} < 0,\ n > 1 \tag{B.52}$$

$$T_{j'n}^{\mathrm{f}} \le T_{jn}^{\mathrm{f}} + M\left(1 - zD_{jj'(n+1)}\right)$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}),\ j \in \mathbf{J}_s,\ \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P \cap \mathbf{I}^R)} \rho_{sij} > 0,\ j \neq j',\ j' \in \mathbf{J}_s,\ \sum_{i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C)} \rho_{si'j'} < 0,\ n < N \tag{B.53}$$

$$T_{jn}^{\mathrm{f}} \ge T_{j'(n-1)}^{\mathrm{s}} - M\left(1 - wv_{ijn}\right)$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}),\ j \in \mathbf{J}_s,\ \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}_s^P) \setminus \mathbf{I}^R} \rho_{sij} > 0,\ j \neq j',\ j' \in \mathbf{J}_s,\ \sum_{i' \in (\mathbf{I}_{j'} \cap \mathbf{I}_s^C)} \rho_{si'j'} < 0,\ n > 1 \tag{B.54}$$

$$T^{\mathrm{f}}_{jn} \geq T^{\mathrm{s}}_{j'n} - M\left(1 - wv_{ijn}\right)$$

$$\forall s \in (\mathbf{S}^{IN} \cap \mathbf{S}^{FIS}),\ j \in \mathbf{J}_s,\ \sum_{i \in (\mathbf{I}_j \cap \mathbf{I}^P_s \cap \mathbf{I}^R)} \rho_{sij} > 0,\ j \neq j',\ j' \in \mathbf{J}_s,\ \sum_{i' \in (\mathbf{I}_{j'} \cap \mathbf{I}^C_s)} \rho_{si'j'} < 0,\ n \tag{B.55}$$

$$\sum_{i \in \mathbf{I}^{in}, \mathbf{I}_k, \mathbf{I}^{T6b}} \sum_{n \leq N^{\max}} y_{ikn} + sl^{order}_k \geq 1 \qquad \forall\ k \in \mathbf{K}^{in} \tag{B.56}$$

$$\sum_{i \in \mathbf{I}^{in}, \mathbf{I}_k, \mathbf{I}^{T6b}} \sum_{n \leq N^{\max}} y_{ikn} \leq \left\lceil \sum_{s \in \left(\mathbf{S}^{in} \cup sl_s \cup \mathbf{S}^{cat1}\right)} \frac{\sum_d rk_{ksd}}{B^{\min}_s} \right\rceil \qquad \forall\ k \in \mathbf{K}^{in} \tag{B.57}$$

$$\sum_{k \in \left(\mathbf{K}^{in} \cup \mathbf{K}_i\right)} \sum_{j \in \mathbf{J}_i} \left|\mathbf{I}_j\right| \cdot y_{ikn} \geq \sum_{j \in \left(\mathbf{J}_i \cup \mathbf{J}^{T6}\right)} wv_{ijn} \qquad \forall\ i \in (\mathbf{I}^{in} \cap \mathbf{I}^{T6b}),\ n \leq N^{\max} \tag{B.58}$$

$$\sum_{k \in \left(\mathbf{K}^{in} \cup \mathbf{K}_i\right)} y_{ikn} \leq \sum_{j \in \left(\mathbf{J}_i \cup \mathbf{J}^{T6}\right)} wv_{ijn} \qquad \forall\ i \in (\mathbf{I}^{in} \cap \mathbf{I}^{T6b}),\ n \leq N^{\max} \tag{B.59}$$

$$D_{sn} = \sum_{k \in \left(\mathbf{K}^{in} \cap \mathbf{K}_s\right)} kD_{ksn} \qquad \forall\ s \in (\mathbf{S}^{in} \cap \mathbf{S}^{cat1}),\ n \leq N^{\max} \tag{B.60}$$

$$D^{\mathrm{f}}_{sn} = \sum_{k \in \left(\mathbf{K}^{in} \cap \mathbf{K}_s\right)} kD^{\mathrm{f}}_{ksn} \qquad \forall\ s \in (\mathbf{S}^{in} \cap \mathbf{S}^{cat1}),\ n \leq N^{\max} \tag{B.61}$$

$$kD_{ksn} + kD^{\mathrm{f}}_{ksn} = \sum_{j \in \mathbf{J}_i} b_{ij(n-1)} - M \cdot \left(1 - y_{ik(n-1)}\right)$$

$$\forall\ k \in (\mathbf{K}^{in} \cap \mathbf{K}_s),\ i \in (\mathbf{I}_k \cap \mathbf{I}^{T6b}),\ 1 < n \leq N^{\max} \tag{B.62}$$

$$\sum_{n \leq N^{\max}} \left(kD_{ksn} + kD^{\mathrm{f}}_{ksn}\right) + sla1_{ksd} \geq rk_{ksd}$$

$$\forall\ s \in (\mathbf{S}^{in} \cap \mathbf{S}^{cat1}),\ k \in (\mathbf{K}^{in} \cap \mathbf{K}_s),\ d \in \mathbf{D}^{in},\ rk_{ksd} > 0 \tag{B.63}$$

$$\sum_{n \leq N^{\max}} \left(kD_{ksn} + kD^{\mathrm{f}}_{ksn}\right) + STf_s + sla2_{ksd} \leq rk_{ksd}$$

$$\forall\ s \in (\mathbf{S}^{in} \cap \mathbf{S}^{cat1}),\ k \in (\mathbf{K}^{in} \cap \mathbf{K}_s),\ d \in \mathbf{D}^{in},\ rk_{ksd} > 0 \tag{B.64}$$

$$t^{\mathrm{f}}_{jn} - slt1_{ksd} \leq duek_{ksd} + H \cdot \left(2 - wv_{ijn} - y_{ikn}\right)$$

$$\forall\ s \in (\mathbf{S}^{in} \cap \mathbf{S}^{cat1}),\ k \in (\mathbf{K}^{in} \cap \mathbf{K}_s),\ i \in (\mathbf{I}_k \cap \mathbf{I}^{T6b}),\ j \in \mathbf{J}_i,\ d \in \mathbf{D}^{in},\ rk_{ksd} > 0 \tag{B.65}$$

$$t^{\mathrm{f}}_{jn} + slt2_{ksd} \geq \left(duek_{ksd} - 24\right) + H \cdot \left(2 - wv_{ijn} - y_{ikn}\right)$$

$$\forall\ s \in (\mathbf{S}^{in} \cap \mathbf{S}^{cat1}),\ k \in (\mathbf{K}^{in} \cap \mathbf{K}_s),\ i \in (\mathbf{I}_k \cap \mathbf{I}^{T6b}),\ j \in \mathbf{J}_i,\ d \in \mathbf{D}^{in},\ rk_{k,s,d} > 0 \tag{B.66}$$

$$\sum_{n \leq N^{\max}} D_{sn} + D^{\mathrm{f}}_{sn} + sll_s \geq Dem_s \qquad \forall\ s \in (\mathbf{S}^{in} \cap \mathbf{S}^{cat1}) \tag{B.67}$$

$$tot_s + sll_s \geq Dem_s \qquad \forall\ s \in (\mathbf{S}^{in} \cap \mathbf{S}^{cat2}) \tag{B.68}$$

$$\sum_{i \in \left(\mathbf{I}^{in} \cap \mathbf{I}^p_s\right)} \sum_{j \in \mathbf{J}_i} \sum_{n \leq N^{\max}} b_{ijn} + sll^{raw}_s \geq Dem^{raw}_s$$

$\forall\ s \in (\mathbf{S}^{in} \cap \mathbf{S}^{rw}),\ s' \in (\mathbf{S}^{in} \cap \mathbf{S}^{f}),\ \mathrm{praw}_{s,s'} > 0,\ \mathrm{D}em_{s'} > 0,\ Dem_s^{\mathrm{raw}} > 0$ (B.69)

$$tt_{jn}^{\mathrm{s}} \leq t_{jn}^{\mathrm{s}} \qquad \forall\ j,\ n \leq N^{\max} \tag{B.70}$$

$$tt_{jn}^{\mathrm{s}} \geq t_{jn}^{\mathrm{s}} - H \cdot \left(1 - \sum_{i \in \mathbf{I}_j} wv_{ijn}\right) \qquad \forall\ j,\ n \leq N^{\max} \tag{B.71}$$

$$tt_{jn}^{\mathrm{s}} \leq H \cdot \left(1 - \sum_{i \in \mathbf{I}_j} wv_{ijn}\right) \qquad \forall\ j,\ n \leq N^{\max} \tag{B.72}$$

$$\sum_{j \in \mathbf{J}^{T1} \setminus \mathbf{J}^{T5}} \sum_{i \in \mathbf{I}_j} \sum_{n \leq N^{\max}} wv_{i,j,n} \geq mtasks \tag{B.73}$$

Objective function

$$z = \sum_{l} term_l \tag{B.74}$$

$$term_1 = \gamma \sum_{s \in \mathbf{S}^f} prior_s \cdot sll_s \tag{B.75}$$

$$term_2 = \varphi \sum_{k} slorder_k \tag{B.76}$$

$$term_3 = \alpha \sum_{k \in \mathbf{K}^{in}} \sum_{s \in \mathbf{S}^{cat1}} \sum_{d \in \mathbf{D}^{in}} sla1_{ksd} + w \cdot sla2_{ksd} \tag{B.77}$$

$$term_4 = \beta \sum_{k \in \mathbf{K}^{in}} \sum_{s \in \mathbf{S}^{cat1}} \sum_{d \in \mathbf{D}^{in}} \sum_{n \leq N^{\max}} \left(w_4 \cdot slt1_{ksdn} + slt2_{ksdn}\right) \tag{B.78}$$

$$term_5 = \xi \sum_{s \in \mathbf{S}^{rw}} prior_s^{raw} \cdot sll_s^{raw} \tag{B.79}$$

$$term_6 = \delta \sum_{s \in S^{cpm}} \sum_{n \leq N^{\max}} slcap_{sn} \tag{B.80}$$

$$term_7 = \lambda \sum_{j} \sum_{n \leq N^{\max}} tt_{jn}^{\mathrm{s}} \tag{B.81}$$

$$term_8 = n \sum_{i} \sum_{j \in \mathbf{J}_i} \sum_{n \leq N^{\max}} wv_{ijn} + \sum_{k} \sum_{i} \sum_{n \leq N^{\max}} y_{ikn} \tag{B.82}$$

$$term_9 = \lambda \sum_{j} \sum_{n \leq N^{\max}} t_{jn}^{\mathrm{f}} \tag{B.83}$$

Blank Page

Supplementary material 2: supplementary material for research contribution 4

Rakovitis, N., Zhang, N., Li, J. A novel unit-specific event-based formulation for short-term scheduling of multitasking processes in scientific service facilities, Computers and Chemical Engineering, 133(2), (2020) doi: doi.org/10.1016/j.compchemeng.2019.106626.

Blank Page

Supplementary Material for

A novel Unit-Specific Event-Based Formulation for Short-Term Scheduling of Multitasking Processes in Scientific Service Facilities

Nikolaos Rakovitis, Nan Zhang, Jie Li[6]

Centre for Process Integration, School of Chemical Engineering and Analytical Science,

The University of Manchester, Manchester, M13 9PL, United Kingdom

List of Tables

[6] To whom correspondence should be addressed. jie.li-2@manchester.ac.uk. Tel: +44 (0) 161 306 8622

Table S1 Sample group data for Examples 2-5

Sample group	Samples	Processing path	Sample group	Samples	Processing path
Example 2			Example 3		
1	73	$P_2 - P_4$	1	57	$P_1 - P_2 - P_3 - P_4$
2	67	$P_1 - P_2 - P_4$	2	59	$P_1 - P_2 - P_3 - P_4$
3	57	$P_1 - P_2 - P_4$	3	54	$P_1 - P_2 - P_3 - P_4$
4	68	$P_1 - P_2 - P_4$	4	71	$P_1 - P_2 - P_3 - P_4$
5	72	$P_2 - P_3 - P_4$	5	58	$P_1 - P_2 - P_3$
6	51	$P_1 - P_2 - P_3 - P_4$	6	77	$P_1 - P_2 - P_3 - P_4$
7	52	$P_1 - P_2 - P_3$	7	70	$P_2 - P_3 - P_4$
8	63	$P_1 - P_2 - P_3$	8	73	$P_1 - P_2 - P_3 - P_4$
9	52	$P_1 - P_2 - P_4$	9	59	$P_2 - P_3 - P_4$
10	79	$P_1 - P_2 - P_3 - P_4$	10	55	$P_1 - P_2 - P_4$
Example 4			Example 5		
1	64	$P_1 - P_2 - P_3$	1	53	$P_2 - P_3 - P_4$
2	64	$P_1 - P_2 - P_3 - P_4$	2	79	$P_1 - P_3 - P_4$
3	68	$P_1 - P_2 - P_4$	3	72	$P_1 - P_2 - P_4$
4	55	$P_1 - P_2 - P_3 - P_4$	4	52	$P_3 - P_4$
5	75	$P_1 - P_2 - P_3 - P_4$	5	66	$P_1 - P_2 - P_3 - P_4$
6	69	$P_1 - P_3 - P_4$	6	74	$P_1 - P_2 - P_4$
7	65	$P_1 - P_2 - P_3 - P_4$	7	72	$P_1 - P_2 - P_3$
8	69	$P_1 - P_2 - P_3$	8	77	$P_2 - P_3 - P_4$
9	51	$P_1 - P_2 - P_3 - P_4$	9	59	$P_1 - P_2 - P_3 - P_4$
10	73	$P_1 - P_2 - P_4$	10	63	$P_1 - P_2$

Table S2 Sample group data for Example 6

Sample group	Samples	Processing path
1	61	$P_1 - P_2 - P_3 - P_4$
2	80	$P_1 - P_2 - P_3 - P_4$
3	71	$P_1 - P_2 - P_3 - P_4$
4	56	$P_1 - P_2 - P_3 - P_4$
5	70	$P_1 - P_2 - P_3 - P_4$
6	72	$P_1 - P_2$
7	67	$P_1 - P_2 - P_3 - P_4$
8	70	$P_3 - P_4$
9	76	$P_1 - P_3 - P_4$
10	76	$P_1 - P_3 - P_4$

Table S3 Sample group data for Examples 7-10

Sample group	Samples	Processing path	Sample group	Samples	Processing path
Example 7			Example 8		
1	64	$P_1-P_2-P_3-P_4$	1	68	$P_1-P_2-P_1-P_4$
2	61	$P_1-P_2-P_3$	2	52	$P_1-P_2-P_3-P_4$
3	69	$P_1-P_2-P_3-P_4$	3	56	$P_1-P_2-P_3-P_4$
4	51	$P_1-P_2-P_3-P_4$	4	73	$P_1-P_2-P_4$
5	50	$P_1-P_2-P_3-P_2-P_4$	5	63	$P_1-P_2-P_3-P_1-P_4$
Example 9			Example 10		
1	63	$P_1-P_2-P_3-P_4$	1	57	P_1
2	74	P_1-P_4	2	77	P_3-P_4
3	78	P_3-P_4	3	76	$P_1-P_2-P_3-P_4$
4	71	P_1-P_4	4	58	$P_1-P_2-P_4$
5	71	$P_1-P_2-P_3$	5	77	P_2-P_4

Table S4 Sample group data for Examples 11-16

Sample group	Samples	Processing path	Sample group	Samples	Processing path
Example 11			Example 12		
1	63	$P_1 - P_2 - P_3 - P_4$	1	56	$P_1 - P_2 - P_3$
2	68	$P_2 - P_3$	2	68	$P_1 - P_2 - P_3$
3	71	$P_1 - P_2 - P_3 - P_4$			
4	53	$P_1 - P_2 - P_3 - P_4$			
5	75	$P_1 - P_2 - P_3$			
Example 13			Example 14		
1	50	$P_1 - P_2$	1	50	$P_1 - P_2 - P_3$
2	71	$P_1 - P_2 - P_3$	2	57	$P_1 - P_2 - P_3$
Example 15			Example 16		
1	68	$P_1 - P_2 - P_3$	1	59	$P_1 - P_2$
2	60	$P_1 - P_3 - P_4$	2	51	$P_1 - P_2$
3	55	$P_1 - P_2 - P_3 - P_4$	3	73	$P_2 - P_3 - P_4$

Table S5 Sample group data for Example 17

Sample groups	Samples	Processing path
1	60	$P_1 - P_3 - P_4 - P_5$
2	71	$P_1 - P_2 - P_3 - P_4 - P_5$
3	77	$P_2 - P_3 - P_4 - P_5$
4	80	$P_1 - P_2 - P_4 - P_5$

Table S6 Sample group data for Example 18

Sample groups	Samples	Processing path
1	76	$P_1 - P_2 - P_4 - P_5 - P_6$
2	53	$P_1 - P_2 - P_3 - P_4 - P_5 - P_6$
3	59	$P_1 - P_2 - P_3 - P_5 - P_6 - P_7$
4	67	$P_1 - P_2 - P_3 - P_4 - P_5 - P_6 - P_7$
5	63	$P_1 - P_2 - P_4 - P_5 - P_6$
6	72	$P_1 - P_2 - P_3 - P_5 - P_6 - P_7$
7	73	$P_3 - P_4 - P_6 - P_7$
8	75	$P_1 - P_2 - P_3 - P_4 - P_5 - P_7$
9	65	$P_1 - P_2 - P_4 - P_5 - P_6 - P_7$
10	51	$P_1 - P_2 - P_3 - P_4 - P_5 - P_6$
11	73	$P_1 - P_2 - P_3 - P_4 - P_5 - P_6 - P_7$
12	56	$P_1 - P_2 - P_3 - P_4 - P_6$

Table S7 Sample group data for Example 19

Sample groups	Samples	Processing path
1	75	$P_1 - P_2 - P_3 - P_4 - P_5 - P_8$
2	53	$P_1 - P_2 - P_3 - P_4 - P_5 - P_6 - P_7 - P_8$
3	50	$P_1 - P_2 - P_3 - P_4 - P_6 - P_7 - P_8$
4	74	$P_1 - P_3 - P_4 - P_5 - P_6 - P_7 - P_8$
5	72	$P_1 - P_4 - P_5 - P_6 - P_7 - P_8$
6	55	$P_1 - P_3 - P_4 - P_5 - P_6 - P_8$
7	56	$P_1 - P_2 - P_3 - P_5 - P_6 - P_7 - P_8$
8	76	$Pr_1 - Pr_2 - Pr_3 - Pr_4 - Pr_5 - Pr_6 - Pr_8$
9	51	$P_1 - P_2 - P_3 - P_4 - P_5 - P_6 - P_7 - P_8$
10	76	$P_1 - P_3 - P_4 - P_5 - P_6 - P_7 - P_8$
11	74	$P_1 - P_2 - P_4 - P_5 - P_6 - P_7 - P_8$
12	72	$P_1 - P_2 - P_3 - P_4 - P_5 - P_6 - P_7 - P_8$
13	64	$P_2 - P_3 - P_4 - P_5 - P_6 - P_7 - P_8$
14	62	$P_1 - P_2 - P_3 - P_4 - P_5 - P_6 - P_8$
15	72	$P_2 - P_3 - P_4 - P_5 - P_6 - P_7 - P_8$
16	58	$P_3 - P_4 - P_5 - P_6 - P_7 - P_8$

Table S8 Processing unit data for Example 20

Sample group	Samples	Processing path	Sample group	Samples	Processing path	Sample group	Samples	Processing path
1	295	8	35	279	1	68	295	9
2	271	5	36	276	5	69	248	7
3	220	1	37	224	7	70	284	7
4	286	7	38	275	6	71	298	3
5	210	4	39	278	1	72	267	6
6	212	3	40	241	4	73	283	3
7	200	2	41	211	6	74	228	10
8	236	11	42	253	9	75	251	4
9	234	11	43	261	6	76	237	4
10	292	2	44	280	3	77	265	2
11	250	8	45	272	8	78	220	9
12	225	7	46	252	8	79	247	9
13	297	11	47	228	4	80	239	4
14	220	10	48	222	10	81	298	10
15	263	8	49	276	4	82	200	1
16	235	9	50	239	11	83	248	3
17	288	5	51	287	3	84	217	7
18	204	8	52	221	7	85	214	8
19	268	2	53	288	8	86	272	4
20	292	4	54	241	9	87	208	4
21	222	10	55	204	10	88	238	11
22	250	5	56	266	8	89	285	7
23	206	3	57	276	3	90	245	7
24	223	7	58	252	9	91	229	8
25	262	9	59	242	7	92	225	9
26	287	10	60	221	2	93	270	1
27	244	4	61	262	6	94	267	6
28	252	7	62	205	10	95	281	6
29	255	3	63	254	5	96	254	3
30	284	5	64	203	10	97	206	9
31	239	5	65	296	8	98	209	2
32	280	10	66	232	11	99	253	11
33	257	5	67	254	7	100	271	7
34	267	4						

Table S9 Processing unit data for Examples 21-29

Sample group	Samples	Processing path	Sample group	Samples	Processing path	Sample group	Samples	Processing path
Example 21			Example 22			Example 23		
1	66	8	1	54	3	1	57	6
2	78	1	2	62	1	2	72	1
3	66	9	3	79	10	3	77	9
4	73	5	4	76	4	4	74	2
5	50	4	5	70	5	5	51	2
Example 24			Example 25			Example 26		
1	65	10	1	71	10	1	75	5
2	58	8	2	50	1	2	68	2
3	77	1	3	75	5	3	67	7
4	69	9	4	78	5	4	59	7
5	56	11	5	70	8	5	72	4
						6	55	5
						7	52	10
						8	66	7
						9	76	10
						10	50	9
Example 27			Example 28			Example 29		
1	79	2	1	65	11	1	77	3
2	53	4	2	71	4	2	58	7
3	54	2	3	71	3	3	56	4
4	72	8	4	50	8	4	51	8
5	68	1	5	70	3	5	70	10
6	61	4	6	62	4	6	51	10
7	58	9	7	61	10	7	61	2
8	56	3	8	52	3	8	54	7
9	70	2	9	78	5	9	56	4
10	71	1	10	50	8	10	60	5

Table S10 Processing unit data for Example 30

Sample group	Samples	Processing path
1	68	8
2	68	1
3	78	10
4	69	9
5	56	2
6	80	3
7	72	9
8	65	4
9	60	11
10	58	2

Table S11 Processing unit data for Example 31

Sample group	Samples	Processing path	Sample group	Samples	Processing path	Sample group	Samples	Processing path
1	219	4	35	274	11	68	245	5
2	299	6	36	224	10	69	225	4
3	215	10	37	207	1	70	275	4
4	230	7	38	296	6	71	289	2
5	295	2	39	289	8	72	241	7
6	220	2	40	211	2	73	238	2
7	292	9	41	218	3	74	223	11
8	250	10	42	225	8	75	207	4
9	276	5	43	230	3	76	202	9
10	288	7	44	216	7	77	235	6
11	261	10	45	232	7	78	232	1
12	290	9	46	256	10	79	295	9
13	281	10	47	207	6	80	265	8
14	218	9	48	270	9	81	298	1
15	260	1	49	236	10	82	298	1
16	258	5	50	261	2	83	229	6
17	291	4	51	200	3	84	236	9
18	206	8	52	268	8	85	249	1
19	278	5	53	275	10	86	238	4
20	211	7	54	255	10	87	293	4
21	298	7	55	275	5	88	220	9
22	296	6	56	264	8	89	292	11
23	288	6	57	250	11	90	206	3
24	212	5	58	267	5	91	259	2
25	214	7	59	223	5	92	279	8
26	251	3	60	298	4	93	269	8
27	276	8	61	223	5	94	279	8
28	219	2	62	283	9	95	247	2
29	216	9	63	292	10	96	237	8
30	222	5	64	289	4	97	248	8
31	263	2	65	266	5	98	298	11
32	243	5	66	223	10	99	243	8
33	282	7	67	276	4	100	206	5
34	252	2						

Table S12 Processing unit data for Example 32

Sample group	Samples	Processing path	Sample group	Samples	Processing path	Sample group	Samples	Processing path
1	283	5	35	292	7	68	330	11
2	317	6	36	321	11	69	298	2
3	330	5	37	282	6	70	295	2
4	309	9	38	279	6	71	256	8
5	314	9	39	321	8	72	277	1
6	340	3	40	255	6	73	269	3
7	337	6	41	256	4	74	316	9
8	307	3	42	260	6	75	323	5
9	298	6	43	317	5	76	261	10
10	268	1	44	308	7	77	250	4
11	327	10	45	288	7	78	333	7
12	328	5	46	300	5	79	290	5
13	294	7	47	256	11	80	316	2
14	292	8	48	294	6	81	333	1
15	278	10	49	273	8	82	256	7
16	322	10	50	325	3	83	280	6
17	343	1	51	268	2	84	345	3
18	254	4	52	288	11	85	310	4
19	333	8	53	306	2	86	335	8
20	332	2	54	305	9	87	341	9
21	276	2	55	274	5	88	272	4
22	341	7	56	257	4	89	260	9
23	323	9	57	338	4	90	342	8
24	276	10	58	334	8	91	325	2
25	303	8	59	266	6	92	320	11
26	266	4	60	329	5	93	299	5
27	316	1	61	265	6	94	287	3
28	329	10	62	329	7	95	265	7
29	280	5	63	335	4	96	287	8
30	288	4	64	338	5	97	322	6
31	325	3	65	297	11	98	330	5
32	276	2	66	319	2	99	314	11
33	289	4	67	277	8	100	264	10
34	320	6						

Table S13 Processing unit data for Examples 1-13 and 16-17

Property	Unit	Capacity (cu)	Processing time (min)	Process	Unit	Capacity (cu)	Processing time (min)
Examples 1-6, 16, 17				Example 7			
1	1	140	50	1	1	187	24
2	2	70	30	2	2	106	59
	3	70	30	3	3	105	191
3	4	50	60				
	5	50	60				
4	6	120	195				
Example 8				Example 9			
1	1	73	70	1	1	50	160
2	2	61	20	2	2	50	78
3	3	193	179		3	50	78
				3	4	50	165
					5	50	165
				4	6	50	199
Example 10				Example 11			
1	1	50	136	1	1	96	26
2	2	50	147	2	2	168	150
	3	50	147		3	138	39
3	4	50	82	3	4	171	143
	5	50	82		5	185	17
4	6	50	55	4	6	114	53
Example 12				Example 13			
1	1	16	55	1	1	60	26
2	2	34	99	2	2	49	99
	3	34	99	3	3	44	20
3	4	150	219	4	4	122	205

Table S14 Processing unit data for Examples 14-15 and 18-19

Property	Unit	Capacity (cu)	Processing time (min)	Property	Unit	Capacity (cu)	Processing time (min)
		Example 14				Example 18	
1	1	200	30	1	1	14	19
2	2	165	11	2	2	122	40
	3	165	11	3	3	180	63
3	4	179	15	4	4	182	22
4	5	123	220	5	5	68	217
		Example 15				Example 19	
1	1	15	33	1	1	15	22
2	2	182	12	2	2	132	48
3	3	104	56	3	3	164	61
4	4	102	19	4	4	19	69
5	5	106	75	5	5	41	69
6	6	176	25	6	6	176	43
7	7	196	204	7	7	113	22
				8	8	88	215

Table S15 Process data for Example 20

Property	Total capacity (cu)	No. of units	Processing time (min)
1	950	2	190
2	1000	3	250
3	1135	2	118
4	803	4	589
5	354	2	222
6	1873	2	958
7	1504	2	382
8	696	1	1259
9	1140	1	188
10	1965	1	268
11	1054	3	1021
12	282	2	675
13	652	4	1020
14	95	4	297
15	1405	1	952
16	819	1	637
17	569	1	401
18	1386	10	1372
19	1622	10	1219
20	373	8	1111
21	534	8	1332
22	694	1	670
23	760	4	1096
24	2025	6	1552
25	1039	1	537

Table S16 Process data for Examples 21-31

Property	Total capacity (cu)	No. of units	Processing time (min)
1	500	2	15
2	60	3	60
3	2250	2	1440
4	50	4	375
5	420	2	40
6	216	2	300
7	21	2	150
8	48	1	615
9	7	1	1440
10	150	1	240
11	480	3	180
12	440	2	240
13	216	4	120
14	440	4	220
15	1	1	10
16	180	1	390
17	240	1	1440
18	720	10	735
19	480	10	471
20	112	8	1256
21	135	8	1141
22	22	1	60
23	440	4	1620
24	10	6	10
25	10	1	10

Blank Page

Supplementary material 3: supplementary materials for research contribution 5

Rakovitis, N., Zhang, N., Li, J. Zhang, L. Novel Approaches for Energy-Efficient Scheduling of Flexible Job-Shop Problems, to be submitted to European Journal of Operational Research

Blank Page

Supplementary Material for

Novel Approach to Energy-Efficient Scheduling of Flexible Job-Shop Problems

Nikolaos Rakovitis[1], Nan Zhang[1], Jie Li[1,7] and Liping Zhang[2]

[1]Centre for Process Integration, Department of Chemical Engineering and Analytical Science, The University of Manchester, Manchester, M13 9PL, United Kingdom

[2]Department of Industrial Engineering, School of Machinery and Automation, Wuhan University of Science and Technology, Wuhan, Hubei, 430081 P. R China

List of Tables

---

[7] To whom correspondence should be addressed. jie.li-2@manchester.ac.uk. Tel: +44 (0) 161 306 8622

Table S12 Computational results for Examples 21-58 from **M1**, **RH-M2** and **eGEP** dispatching rule 9

Table S1 Computational results for Examples 1-20 from **M1**, **RH-M1**, **RH-M2** and **eGEP** dispatching rule 5

	eGEP	**M1/M2a**	**RH-M1**		**RH-M2**		**Diff (%)**			
Ex	TEC (kW)	TEC (kW)	TEC (kW)	Time (s)	TEC (kw)	Time (s)	**RH-M1** vs. **M1**	**RH-M1** vs **eGEP**	**RH-M2** vs. **M1**	**RH-M2** vs. **eGEP**
Ex1	65.03	63.03	63.03	0.03	63.03	0.11	0.0	-3.1	0.0	-3.1
Ex2	126.04	122.44	122.44	0.03	122.44	0.14	0.0	-2.9	0.0	-2.9
Ex3	75.74	75.74	75.74	0.03	75.74	0.09	0.0	0.0	0.0	0.0
Ex4	161.73	146.63	146.63	0.03	146.63	0.20	0.0	-9.3	0.0	-9.3
Ex5	78.40	78.40	78.40	0.03	78.40	0.20	0.0	0.0	0.0	0.0
Ex6	279.84	220.74	220.74	0.02	220.74	0.17	0.0	-21.1	0.0	-21.1
Ex7	107.69	97.54	97.54	0.05	97.54	0.20	0.0	-9.4	0.0	-9.4
Ex8	184.44	146.81	146.81	0.08	146.81	0.14	0.0	-20.4	0.0	-20.4
Ex9	233.66	230.66	230.66	0.03	230.66	0.09	0.0	-1.3	0.0	-1.3
Ex10	191.68	161.06	161.06	0.05	161.06	0.13	0.0	-16.0	0.0	-16.0
Ex11	166.23	166.23	166.23	0.03	166.23	0.20	0.0	0.0	0.0	0.0
Ex12	176.75	176.75	176.75	0.03	176.75	0.19	0.0	0.0	0.0	0.0
Ex13	121.30	121.30	121.30	0.02	121.30	0.20	0.0	0.0	0.0	0.0
Ex14	167.46	156.86	156.86	0.03	156.86	0.11	0.0	-6.3	0.0	-6.3
Ex15	174.85	163.20	163.20	0.02	163.20	0.14	0.0	-6.7	0.0	-6.7
Ex16	245.44	219.46	219.46	2.30	219.46	0.16	0.0	-10.6	0.0	-10.6
Ex17	321.80	306.68	306.68	0.06	306.68	0.27	0.0	-4.7	0.0	-4.7
Ex18	216.86	210.60	210.60	0.30	210.60	0.22	0.0	-2.9	0.0	-2.9
Ex19	283.83	269.52	269.52	0.03	269.52	0.17	0.0	-5.0	0.0	-5.0
Ex20	327.30	274.94	274.94	0.05	274.94	0.23	0.0	-16.0	0.0	-16.0

Table S2 Computational results for Examples 21-58 from model **M1**, **RH-M1** and **eGEP** dispatching rule 5

	eGEP	**M1**	**RH-M1**		**Diff (%)**	
Ex	TEC (kW)	TEC (kW)	TEC (kW)	CPU Time (s)	**M1** vs. **eGEP**	**RH-M1** vs. **eGEP**
Ex21	297.98	182.49	214.78	46.9	-38.8	-27.9
Ex22	4047.03	3674.04	3812.87	301.4	-9.2	-5.8
Ex23	4088.49	3497.00	4192.16	1.0	-14.5	2.5
Ex24	2070.50	1776.14	1852.97	7.1	-14.2	-10.5
Ex25	1975.75	1789.95	1904.05	102.3	-9.4	-3.6
Ex26	1964.55	1783.95	1602.78	100.3	-9.2	-18.4
Ex27	1939.76	1684.29	1750.05	100.8	-13.2	-9.8
Ex28	2006.52	1465.37	1557.80	101.0	-27.0	-22.4
Ex29	3060.93	2583.71	2703.05	6.2	-15.6	-11.7
Ex30	2835.82	2388.63	2591.91	6.7	-15.8	-8.6
Ex31	2807.84	2486.18	2732.67	1.9	-11.5	-2.7
Ex32	3046.21	2637.50	2814.93	100.3	-13.4	-7.6
Ex33	3271.49	2523.77	2744.54	43.9	-22.9	-16.1
Ex34	4064.60	3365.35	3632.30	0.7	-17.2	-10.6
Ex35	3700.86	3035.98	3406.47	1.3	-18.0	-8.0
Ex36	3682.76	3196.92	3272.25	0.2	-13.2	-11.1
Ex37	3983.02	3477.73	3636.68	0.8	-12.7	-8.7
Ex38	4018.18	3459.03	3987.98	0.9	-13.9	-0.8
Ex39	4982.54	4041.88	3930.09	113.7	-18.9	-21.1
Ex40	4300.04	3648.90	3517.77	223.8	-15.1	-18.2
Ex41	4059.53	3589.61	3767.92	400.1	-11.6	-7.2
Ex42	3937.63	3703.47	3809.58	400.1	-5.9	-3.3
Ex43	4396.11	3782.58	3880.84	312.8	-14.0	-11.7
Ex44	5708.72	5374.11	5405.87	414.7	-5.9	-5.3
Ex45	5876.62	5195.68	4890.87	308.1	-11.6	-16.8
Ex46	6322.86	5501.46	5190.55	404.1	-13.0	-17.9
Ex47	5763.51	5916.36	5027.49	400.7	2.7	-12.8
Ex48	6640.15	6704.23	5107.14	400.3	1.0	-23.1
Ex49	7550.94	9654.12	6946.16	1.6	27.9	-8.0
Ex50	7859.20	9953.75	7434.35	1.3	26.7	-5.4
Ex51	7201.43	9603.38	6866.13	1.2	33.4	-4.7
Ex52	7287.90	-	7257.84	1.0	-	-0.4
Ex53	7332.79	-	7200.05	1.2	-	-1.8
Ex54	10108.14	-	8698.35	3.5	-	-13.9
Ex55	10939.20	-	9580.33	4.9	-	-12.4
Ex56	10339.46	-	8834.19	3.2	-	-14.6
Ex57	10081.65	-	8958.33	4.5	-	-11.1
Ex58	10751.25	-	9775.46	4.8	-	-9.1

Table S3 Computational results for Examples 21-58 from model **M1**, **RH-M2** and **eGEP** dispatching rule 5

	eGEP	**M1**	**RH-M2**		**Diff (%)**	
Ex	TEC (kW)	TEC (kW)	TEC (kW)	CPU Time (s)	**M1** vs. **eGEP**	**RH-M2** vs. **eGEP**
Ex21	297.98	182.49	215.87	2.6	-38.8	-27.6
Ex22	4047.03	3674.04	3679.99	112.8	-9.2	-9.1
Ex23	4088.49	3497.00	3808.34	288.1	-14.5	-6.9
Ex24	2070.50	1776.14	1907.12	5.7	-14.2	-7.9
Ex25	1975.75	1789.95	1941.73	163.3	-9.4	-1.7
Ex26	1964.55	1783.95	1633.59	170.6	-9.2	-16.8
Ex27	1939.76	1684.29	1763.91	138.5	-13.2	-9.1
Ex28	2006.52	1465.37	1598	134.5	-27.0	-20.4
Ex29	3060.93	2583.71	2718.8	108.9	-15.6	-11.2
Ex30	2835.82	2388.63	2521.31	200.8	-15.8	-11.1
Ex31	2807.84	2486.18	2645.48	101.1	-11.5	-5.8
Ex32	3046.21	2637.50	2685.13	51.2	-13.4	-11.9
Ex33	3271.49	2523.77	2657.98	106.4	-22.9	-18.8
Ex34	4064.60	3365.35	3565.38	211.9	-17.2	-12.3
Ex35	3700.86	3035.98	3523.2	262.4	-18.0	-4.8
Ex36	3682.76	3196.92	3417.27	205.2	-13.2	-7.2
Ex37	3983.02	3477.73	3716.38	189.2	-12.7	-6.7
Ex38	4018.18	3459.03	3787.02	204.0	-13.9	-5.8
Ex39	4982.54	4041.88	3884.04	17.2	-18.9	-22.0
Ex40	4300.04	3648.90	3562.7	10.2	-15.1	-17.1
Ex41	4059.53	3589.61	3754.31	0.9	-11.6	-7.5
Ex42	3937.63	3703.47	3717.92	3.7	-5.9	-5.6
Ex43	4396.11	3782.58	3978.46	7.2	-14.0	-9.5
Ex44	5708.72	5374.11	5475.11	210.4	-5.9	-4.1
Ex45	5876.62	5195.68	4941.6	216.6	-11.6	-15.9
Ex46	6322.86	5501.46	5185.61	55.5	-13.0	-18.0
Ex47	5763.51	5916.36	5257.31	253.9	2.7	-8.8
Ex48	6640.15	6704.23	5527.5	233.6	1.0	-16.8
Ex49	7550.94	9654.12	6951.96	4.4	27.9	-7.9
Ex50	7859.20	9953.75	7560.55	105.5	26.7	-3.8
Ex51	7201.43	9603.38	6620.26	0.9	33.4	-8.1
Ex52	7287.90	-	7060.59	106.4	-	-3.1
Ex53	7332.79	-	6938.57	2.3	-	-5.4
Ex54	10108.14	-	9167.93	374.8	-	-9.3
Ex55	10939.20	-	9708.14	509.6	-	-11.3
Ex56	10339.46	-	8861.43	394.8	-	-14.3
Ex57	10081.65	-	9610.23	551.4	-	-4.7
Ex58	10751.25	-	10003.95	494.7	-	-7.0

Table S4 Computational results for Examples 1-20 from model **M1**, **RH-M1**, **RH-M2** and **eGEP** dispatching rule 7

	eGEP	**M1/M2a**	**RH-M1**		**RH-M2**		**Diff (%)**			
Ex	TEC (kW)	TEC (kW)	TEC (kW)	Time (s)	TEC (kw)	Time (s)	**RH-M1** vs. **M1**	**RH-M1** vs **eGEP**	**RH-M2** vs. **M1**	**RH-M2** vs. **eGEP**
Ex1	67.03	63.03	63.03	0.03	63.03	0.11	0.0	-6.0	0.0	-6.0
Ex2	126.04	122.44	122.44	0.03	122.44	0.14	0.0	-2.9	0.0	-2.9
Ex3	75.74	75.74	75.74	0.03	75.74	0.09	0.0	0.0	0.0	0.0
Ex4	161.73	146.63	146.63	0.03	146.63	0.20	0.0	-9.3	0.0	-9.3
Ex5	78.4	78.40	78.40	0.03	78.40	0.20	0.0	0.0	0.0	0.0
Ex6	263.14	220.74	220.74	0.02	220.74	0.17	0.0	-16.1	0.0	-16.1
Ex7	98.57	97.54	97.54	0.05	97.54	0.20	0.0	-1.0	0.0	-1.0
Ex8	186.44	146.81	146.81	0.08	146.81	0.14	0.0	-21.3	0.0	-21.3
Ex9	233.66	230.66	230.66	0.03	230.66	0.09	0.0	-1.3	0.0	-1.3
Ex10	162.89	161.06	161.06	0.05	161.06	0.13	0.0	-1.1	0.0	-1.1
Ex11	166.23	166.23	166.23	0.03	166.23	0.20	0.0	0.0	0.0	0.0
Ex12	176.75	176.75	176.75	0.03	176.75	0.19	0.0	0.0	0.0	0.0
Ex13	121.3	121.30	121.30	0.02	121.30	0.20	0.0	0.0	0.0	0.0
Ex14	167.46	156.86	156.86	0.03	156.86	0.11	0.0	-6.3	0.0	-6.3
Ex15	171.8	163.20	163.20	0.02	163.20	0.14	0.0	-5.0	0.0	-5.0
Ex16	253.41	219.46	219.46	2.30	219.46	0.16	0.0	-13.4	0.0	-13.4
Ex17	321.8	306.68	306.68	0.06	306.68	0.27	0.0	-4.7	0.0	-4.7
Ex18	216.86	210.60	210.60	0.30	210.60	0.22	0.0	-2.9	0.0	-2.9
Ex19	269.52	269.52	269.52	0.03	269.52	0.17	0.0	0.0	0.0	0.0
Ex20	304.98	274.94	274.94	0.05	274.94	0.23	0.0	-9.8	0.0	-9.8

Table S5 Computational results for Examples 21-58 from model **M1**, **RH-M1** and **eGEP** dispatching rule 7

Ex	**eGEP**	**M1**	**RH-M1**		**Diff (%)**	
	TEC (kW)	TEC (kW)	TEC (kW)	CPU Time (s)	**M1** vs. **eGEP**	**RH-M1** vs. **eGEP**
Ex21	377.73	182.49	214.78	46.9	-51.7	-43.1
Ex22	4342.12	3674.04	3812.87	301.4	-15.4	-12.2
Ex23	4402.20	3497.00	4192.16	1.0	-20.6	-4.8
Ex24	2172.22	1776.14	1852.97	7.1	-18.2	-14.7
Ex25	2033.00	1789.95	1904.05	102.3	-12.0	-6.3
Ex26	2021.92	1783.95	1602.78	100.3	-11.8	-20.7
Ex27	2059.48	1684.29	1750.05	100.8	-18.2	-15.0
Ex28	1976.26	1465.37	1557.80	101.0	-25.9	-21.2
Ex29	3044.67	2583.71	2703.05	6.2	-15.1	-11.2
Ex30	2826.03	2388.63	2591.91	6.7	-15.5	-8.3
Ex31	2765.72	2486.18	2732.67	1.9	-10.1	-1.2
Ex32	3136.43	2637.50	2814.93	100.3	-15.9	-10.3
Ex33	3036.47	2523.77	2744.54	43.9	-16.9	-9.6
Ex34	3947.29	3365.35	3632.30	0.7	-14.7	-8.0
Ex35	3731.46	3035.98	3406.47	1.3	-18.6	-8.7
Ex36	4061.80	3196.92	3272.25	0.2	-21.3	-19.4
Ex37	4463.81	3477.73	3636.68	0.8	-22.1	-18.5
Ex38	3740.39	3459.03	3987.98	0.9	-7.5	6.6
Ex39	4480.15	4041.88	3930.09	113.7	-9.8	-12.3
Ex40	4482.24	3648.90	3517.77	223.8	-18.6	-21.5
Ex41	4160.37	3589.61	3767.92	400.1	-13.7	-9.4
Ex42	4330.02	3703.47	3809.58	400.1	-14.5	-12.0
Ex43	4437.60	3782.58	3880.84	312.8	-14.8	-12.5
Ex44	5976.21	5374.11	5405.87	414.7	-10.1	-9.5
Ex45	5756.06	5195.68	4890.87	308.1	-9.7	-15.0
Ex46	5987.00	5501.46	5190.55	404.1	-8.1	-13.3
Ex47	6180.85	5916.36	5027.49	400.7	-4.3	-18.7
Ex48	7138.21	6704.23	5107.14	400.3	-6.1	-28.5
Ex49	7504.90	9654.12	6946.16	1.6	28.6	-7.4
Ex50	8192.58	9953.75	7434.35	1.3	21.5	-9.3
Ex51	7528.43	9603.38	6866.13	1.2	27.6	-8.8
Ex52	7388.66	-	7257.84	1.0	-	-1.8
Ex53	7950.47	-	7200.05	1.2	-	-9.4
Ex54	10036.36	-	8698.35	3.5	-	-13.3
Ex55	10703.56	-	9580.33	4.9	-	-10.5
Ex56	10194.85	-	8834.19	3.2	-	-13.3
Ex57	9884.19	-	8958.33	4.5	-	-9.4
Ex58	11269.35	-	9775.46	4.8	-	-13.3

Table S6 Computational results for Examples 21-58 from model **M1**, **RH-M2** and **eGEP** dispatching rule 7

	eGEP	**M1**	**RH-M2**		**Diff (%)**	
Ex	TEC (kW)	TEC (kW)	TEC (kW)	CPU Time (s)	**M1** vs. **eGEP**	**RH-M2** vs. **eGEP**
Ex21	377.73	182.49	215.87	2.6	-51.7	-42.9
Ex22	4342.12	3674.04	3679.99	112.8	-15.4	-15.2
Ex23	4402.20	3497.00	3808.34	288.1	-20.6	-13.5
Ex24	2172.22	1776.14	1907.12	5.7	-18.2	-12.2
Ex25	2033.00	1789.95	1941.73	163.3	-12.0	-4.5
Ex26	2021.92	1783.95	1633.59	170.6	-11.8	-19.2
Ex27	2059.48	1684.29	1763.91	138.5	-18.2	-14.4
Ex28	1976.26	1465.37	1598	134.5	-25.9	-19.1
Ex29	3044.67	2583.71	2718.8	108.9	-15.1	-10.7
Ex30	2826.03	2388.63	2521.31	200.8	-15.5	-10.8
Ex31	2765.72	2486.18	2645.48	101.1	-10.1	-4.3
Ex32	3136.43	2637.50	2685.13	51.2	-15.9	-14.4
Ex33	3036.47	2523.77	2657.98	106.4	-16.9	-12.5
Ex34	3947.29	3365.35	3565.38	211.9	-14.7	-9.7
Ex35	3731.46	3035.98	3523.2	262.4	-18.6	-5.6
Ex36	4061.80	3196.92	3417.27	205.2	-21.3	-15.9
Ex37	4463.81	3477.73	3716.38	189.2	-22.1	-16.7
Ex38	3740.39	3459.03	3787.02	204.0	-7.5	1.2
Ex39	4480.15	4041.88	3884.04	17.2	-9.8	-13.3
Ex40	4482.24	3648.90	3562.7	10.2	-18.6	-20.5
Ex41	4160.37	3589.61	3754.31	0.9	-13.7	-9.8
Ex42	4330.02	3703.47	3717.92	3.7	-14.5	-14.1
Ex43	4437.60	3782.58	3978.46	7.2	-14.8	-10.3
Ex44	5976.21	5374.11	5475.11	210.4	-10.1	-8.4
Ex45	5756.06	5195.68	4941.6	216.6	-9.7	-14.1
Ex46	5987.00	5501.46	5185.61	55.5	-8.1	-13.4
Ex47	6180.85	5916.36	5257.31	253.9	-4.3	-14.9
Ex48	7138.21	6704.23	5527.5	233.6	-6.1	-22.6
Ex49	7504.90	9654.12	6951.96	4.4	28.6	-7.4
Ex50	8192.58	9953.75	7560.55	105.5	21.5	-7.7
Ex51	7528.43	9603.38	6620.26	0.9	27.6	-12.1
Ex52	7388.66	-	7060.59	106.4	-	-4.4
Ex53	7950.47	-	6938.57	2.3	-	-12.7
Ex54	10036.36	-	9167.93	374.8	-	-8.7
Ex55	10703.56	-	9708.14	509.6	-	-9.3
Ex56	10194.85	-	8861.43	394.8	-	-13.1
Ex57	9884.19	-	9610.23	551.4	-	-2.8
Ex58	11269.35	-	10003.95	494.7	-	-11.2

Table S7 Computational results for Examples 1-20 from **M1**, **RH-M1**, **RH-M2** and **eGEP** dispatching rule 8

	eGEP	**M1/M2a**	**RH-M1**		**RH-M2**		**Diff (%)**			
Ex	TEC (kW)	TEC (kW)	TEC (kW)	Time (s)	TEC (kw)	Time (s)	**RH-M1** vs. **M1**	**RH-M1** vs **eGEP**	**RH-M2** vs. **M1**	**RH-M2** vs. **eGEP**
Ex1	65.03	63.03	63.03	0.03	63.03	0.11	0.0	-3.1	0.0	-3.1
Ex2	126.04	122.44	122.44	0.03	122.44	0.14	0.0	-2.9	0.0	-2.9
Ex3	78.38	75.74	75.74	0.03	75.74	0.09	0.0	-3.4	0.0	-3.4
Ex4	147.12	146.63	146.63	0.03	146.63	0.20	0.0	-0.3	0.0	-0.3
Ex5	114.62	78.40	78.40	0.03	78.40	0.20	0.0	-31.6	0.0	-31.6
Ex6	279.84	220.74	220.74	0.02	220.74	0.17	0.0	-21.1	0.0	-21.1
Ex7	107.69	97.54	97.54	0.05	97.54	0.20	0.0	-9.4	0.0	-9.4
Ex8	170.17	146.81	146.81	0.08	146.81	0.14	0.0	-13.7	0.0	-13.7
Ex9	230.66	230.66	230.66	0.03	230.66	0.09	0.0	0.0	0.0	0.0
Ex10	191.68	161.06	161.06	0.05	161.06	0.13	0.0	-16.0	0.0	-16.0
Ex11	166.23	166.23	166.23	0.03	166.23	0.20	0.0	0.0	0.0	0.0
Ex12	176.75	176.75	176.75	0.03	176.75	0.19	0.0	0.0	0.0	0.0
Ex13	121.3	121.30	121.30	0.02	121.30	0.20	0.0	0.0	0.0	0.0
Ex14	156.86	156.86	156.86	0.03	156.86	0.11	0.0	0.0	0.0	0.0
Ex15	174.85	163.20	163.20	0.02	163.20	0.14	0.0	-6.7	0.0	-6.7
Ex16	245.44	219.46	219.46	2.30	219.46	0.16	0.0	-10.6	0.0	-10.6
Ex17	315.08	306.68	306.68	0.06	306.68	0.27	0.0	-2.7	0.0	-2.7
Ex18	216.86	210.60	210.60	0.30	210.60	0.22	0.0	-2.9	0.0	-2.9
Ex19	283.83	269.52	269.52	0.03	269.52	0.17	0.0	-5.0	0.0	-5.0
Ex20	325.86	274.94	274.94	0.05	274.94	0.23	0.0	-15.6	0.0	-15.6

Table S8 Computational results for Examples 21-58 from **M1**, **RH-M1** and **eGEP** dispatching rule 8

Ex	**eGEP**	**M1**	**RH-M1**		**Diff (%)**	
	TEC (kW)	TEC (kW)	TEC (kW)	CPU Time (s)	**M1** vs. **eGEP**	**RH-M1** vs. **eGEP**
Ex21	296.71	182.49	214.78	46.9	-38.5	-27.6
Ex22	4289.90	3674.04	3812.87	301.4	-14.4	-11.1
Ex23	4101.52	3497.00	4192.16	1.0	-14.7	2.2
Ex24	2017.38	1776.14	1852.97	7.1	-12.0	-8.1
Ex25	2061.22	1789.95	1904.05	102.3	-13.2	-7.6
Ex26	2077.87	1783.95	1602.78	100.3	-14.1	-22.9
Ex27	1940.73	1684.29	1750.05	100.8	-13.2	-9.8
Ex28	2015.52	1465.37	1557.80	101.0	-27.3	-22.7
Ex29	2921.23	2583.71	2703.05	6.2	-11.6	-7.5
Ex30	2761.52	2388.63	2591.91	6.7	-13.5	-6.1
Ex31	2866.40	2486.18	2732.67	1.9	-13.3	-4.7
Ex32	3122.96	2637.50	2814.93	100.3	-15.5	-9.9
Ex33	3164.21	2523.77	2744.54	43.9	-20.2	-13.3
Ex34	3954.61	3365.35	3632.30	0.7	-14.9	-8.2
Ex35	3751.06	3035.98	3406.47	1.3	-19.1	-9.2
Ex36	3698.97	3196.92	3272.25	0.2	-13.6	-11.5
Ex37	3796.85	3477.73	3636.68	0.8	-8.4	-4.2
Ex38	3876.12	3459.03	3987.98	0.9	-10.8	2.9
Ex39	4698.71	4041.88	3930.09	113.7	-14.0	-16.4
Ex40	4328.51	3648.90	3517.77	223.8	-15.7	-18.7
Ex41	4219.24	3589.61	3767.92	400.1	-14.9	-10.7
Ex42	4264.51	3703.47	3809.58	400.1	-13.2	-10.7
Ex43	4311.92	3782.58	3880.84	312.8	-12.3	-10.0
Ex44	5972.97	5374.11	5405.87	414.7	-10.0	-9.5
Ex45	6157.10	5195.68	4890.87	308.1	-15.6	-20.6
Ex46	6307.20	5501.46	5190.55	404.1	-12.8	-17.7
Ex47	5981.64	5916.36	5027.49	400.7	-1.1	-16.0
Ex48	7113.48	6704.23	5107.14	400.3	-5.8	-28.2
Ex49	7351.24	9654.12	6946.16	1.6	31.3	-5.5
Ex50	8244.86	9953.75	7434.35	1.3	20.7	-9.8
Ex51	7396.69	9603.38	6866.13	1.2	29.8	-7.2
Ex52	7285.72	-	7257.84	1.0	-	-0.4
Ex53	7999.57	-	7200.05	1.2	-	-10.0
Ex54	10344.35	-	8698.35	3.5	-	-15.9
Ex55	10680.83	-	9580.33	4.9	-	-10.3
Ex56	10183.47	-	8834.19	3.2	-	-13.2
Ex57	9865.68	-	8958.33	4.5	-	-9.2
Ex58	10832.23	-	9775.46	4.8	-	-9.8

Table S9 Computational results for Examples 21-58 from model **M1**, **RH-M2** and **eGEP** dispatching rule 8

	eGEP	**M1**	**RH-M2**		**Diff (%)**	
Ex	TEC (kW)	TEC (kW)	TEC (kW)	CPU Time (s)	**M1** vs. **eGEP**	**RH-M2** vs. **eGEP**
Ex21	296.71	182.49	215.87	2.6	-38.5	-27.2
Ex22	4289.90	3674.04	3679.99	112.8	-14.4	-14.2
Ex23	4101.52	3497.00	3808.34	288.1	-14.7	-7.1
Ex24	2017.38	1776.14	1907.12	5.7	-12.0	-5.5
Ex25	2061.22	1789.95	1941.73	163.3	-13.2	-5.8
Ex26	2077.87	1783.95	1633.59	170.6	-14.1	-21.4
Ex27	1940.73	1684.29	1763.91	138.5	-13.2	-9.1
Ex28	2015.52	1465.37	1598	134.5	-27.3	-20.7
Ex29	2921.23	2583.71	2718.8	108.9	-11.6	-6.9
Ex30	2761.52	2388.63	2521.31	200.8	-13.5	-8.7
Ex31	2866.40	2486.18	2645.48	101.1	-13.3	-7.7
Ex32	3122.96	2637.50	2685.13	51.2	-15.5	-14.0
Ex33	3164.21	2523.77	2657.98	106.4	-20.2	-16.0
Ex34	3954.61	3365.35	3565.38	211.9	-14.9	-9.8
Ex35	3751.06	3035.98	3523.2	262.4	-19.1	-6.1
Ex36	3698.97	3196.92	3417.27	205.2	-13.6	-7.6
Ex37	3796.85	3477.73	3716.38	189.2	-8.4	-2.1
Ex38	3876.12	3459.03	3787.02	204.0	-10.8	-2.3
Ex39	4698.71	4041.88	3884.04	17.2	-14.0	-17.3
Ex40	4328.51	3648.90	3562.7	10.2	-15.7	-17.7
Ex41	4219.24	3589.61	3754.31	0.9	-14.9	-11.0
Ex42	4264.51	3703.47	3717.92	3.7	-13.2	-12.8
Ex43	4311.92	3782.58	3978.46	7.2	-12.3	-7.7
Ex44	5972.97	5374.11	5475.11	210.4	-10.0	-8.3
Ex45	6157.10	5195.68	4941.6	216.6	-15.6	-19.7
Ex46	6307.20	5501.46	5185.61	55.5	-12.8	-17.8
Ex47	5981.64	5916.36	5257.31	253.9	-1.1	-12.1
Ex48	7113.48	6704.23	5527.5	233.6	-5.8	-22.3
Ex49	7351.24	9654.12	6951.96	4.4	31.3	-5.4
Ex50	8244.86	9953.75	7560.55	105.5	20.7	-8.3
Ex51	7396.69	9603.38	6620.26	0.9	29.8	-10.5
Ex52	7285.72	-	7060.59	106.4	-	-3.1
Ex53	7999.57	-	6938.57	2.3	-	-13.3
Ex54	10344.35	-	9167.93	374.8	-	-11.4
Ex55	10680.83	-	9708.14	509.6	-	-9.1
Ex56	10183.47	-	8861.43	394.8	-	-13.0
Ex57	9865.68	-	9610.23	551.4	-	-2.6
Ex58	10832.23	-	10003.95	494.7	-	-7.6

Table S10 Computational results for Examples 1-20 from model **M1**, **RH-M1**, **RH-M2** and **eGEP** dispatching rule 9

	eGEP	**M1/M2a**	**RH-M1**		**RH-M2**		**Diff (%)**			
Ex	TEC (kW)	TEC (kW)	TEC (kW)	Time (s)	TEC (kw)	Time (s)	**RH-M1** vs. **M1**	**RH-M1** vs **eGEP**	**RH-M2** vs. **M1**	**RH-M2** vs. **eGEP**
Ex1	63.03	63.03	63.03	0.03	63.03	0.11	0.0	0.0	0.0	0.0
Ex2	162.28	122.44	122.44	0.03	122.44	0.14	0.0	-24.6	0.0	-24.6
Ex3	99.06	75.74	75.74	0.03	75.74	0.09	0.0	-23.5	0.0	-23.5
Ex4	147.12	146.63	146.63	0.03	146.63	0.20	0.0	-0.3	0.0	-0.3
Ex5	124.44	78.40	78.40	0.03	78.40	0.20	0.0	-37.0	0.0	-37.0
Ex6	279.84	220.74	220.74	0.02	220.74	0.17	0.0	-21.1	0.0	-21.1
Ex7	107.69	97.54	97.54	0.05	97.54	0.20	0.0	-9.4	0.0	-9.4
Ex8	225.78	146.81	146.81	0.08	146.81	0.14	0.0	-35.0	0.0	-35.0
Ex9	248.81	230.66	230.66	0.03	230.66	0.09	0.0	-7.3	0.0	-7.3
Ex10	172.56	161.06	161.06	0.05	161.06	0.13	0.0	-6.7	0.0	-6.7
Ex11	166.23	166.23	166.23	0.03	166.23	0.20	0.0	0.0	0.0	0.0
Ex12	182.69	176.75	176.75	0.03	176.75	0.19	0.0	-3.3	0.0	-3.3
Ex13	121.3	121.30	121.30	0.02	121.30	0.20	0.0	0.0	0.0	0.0
Ex14	156.86	156.86	156.86	0.03	156.86	0.11	0.0	0.0	0.0	0.0
Ex15	191.83	163.20	163.20	0.02	163.20	0.14	0.0	-14.9	0.0	-14.9
Ex16	240.83	219.46	219.46	2.30	219.46	0.16	0.0	-8.9	0.0	-8.9
Ex17	315.08	306.68	306.68	0.06	306.68	0.27	0.0	-2.7	0.0	-2.7
Ex18	251.86	210.60	210.60	0.30	210.60	0.22	0.0	-16.4	0.0	-16.4
Ex19	296.71	269.52	269.52	0.03	269.52	0.17	0.0	-9.2	0.0	-9.2
Ex20	393.83	274.94	274.94	0.05	274.94	0.23	0.0	-30.2	0.0	-30.2

Table S11 Computational results for Examples 21-58 from **M1**, **RH-M1** and **eGEP** dispatching rule 9

Ex	eGEP	M1	RH-M1		Diff (%)	
	TEC (kW)	TEC (kW)	TEC (kW)	CPU Time (s)	**M1** vs. **eGEP**	**RH-M1** vs. **eGEP**
Ex21	320.25	182.49	214.78	46.9	-43.0	-32.9
Ex22	4465.27	3674.04	3812.87	301.4	-17.7	-14.6
Ex23	4355.59	3497.00	4192.16	1.0	-19.7	-3.8
Ex24	1914.55	1776.14	1852.97	7.1	-7.2	-3.2
Ex25	2195.64	1789.95	1904.05	102.3	-18.5	-13.3
Ex26	2294.79	1783.95	1602.78	100.3	-22.3	-30.2
Ex27	2121.88	1684.29	1750.05	100.8	-20.6	-17.5
Ex28	1864.79	1465.37	1557.80	101.0	-21.4	-16.5
Ex29	3314.21	2583.71	2703.05	6.2	-22.0	-18.4
Ex30	2917.68	2388.63	2591.91	6.7	-18.1	-11.2
Ex31	3034.12	2486.18	2732.67	1.9	-18.1	-9.9
Ex32	3220.20	2637.50	2814.93	100.3	-18.1	-12.6
Ex33	3338.57	2523.77	2744.54	43.9	-24.4	-17.8
Ex34	4584.44	3365.35	3632.30	0.7	-26.6	-20.8
Ex35	4070.23	3035.98	3406.47	1.3	-25.4	-16.3
Ex36	3979.86	3196.92	3272.25	0.2	-19.7	-17.8
Ex37	4144.10	3477.73	3636.68	0.8	-16.1	-12.2
Ex38	3841.63	3459.03	3987.98	0.9	-10.0	3.8
Ex39	4823.96	4041.88	3930.09	113.7	-16.2	-18.5
Ex40	4082.95	3648.90	3517.77	223.8	-10.6	-13.8
Ex41	4117.93	3589.61	3767.92	400.1	-12.8	-8.5
Ex42	4084.68	3703.47	3809.58	400.1	-9.3	-6.7
Ex43	4491.61	3782.58	3880.84	312.8	-15.8	-13.6
Ex44	6116.02	5374.11	5405.87	414.7	-12.1	-11.6
Ex45	5923.10	5195.68	4890.87	308.1	-12.3	-17.4
Ex46	6095.66	5501.46	5190.55	404.1	-9.7	-14.8
Ex47	6051.80	5916.36	5027.49	400.7	-2.2	-16.9
Ex48	6672.93	6704.23	5107.14	400.3	0.5	-23.5
Ex49	7727.28	9654.12	6946.16	1.6	24.9	-10.1
Ex50	8163.12	9953.75	7434.35	1.3	21.9	-8.9
Ex51	7173.32	9603.38	6866.13	1.2	33.9	-4.3
Ex52	7650.43	-	7257.84	1.0	-	-5.1
Ex53	7589.11	-	7200.05	1.2	-	-5.1
Ex54	10070.27	-	8698.35	3.5	-	-13.6
Ex55	10667.02	-	9580.33	4.9	-	-10.2
Ex56	10683.11	-	8834.19	3.2	-	-17.3
Ex57	9939.02	-	8958.33	4.5	-	-9.9
Ex58	10963.14	-	9775.46	4.8	-	-10.8

Table S12 Computational results for Examples 21-58 from model **M1**, **RH-M2** and **eGEP** dispatching rule 9

	eGEP	**M1**	**RH-M2**		**Diff (%)**	
Ex	TEC (kW)	TEC (kW)	TEC (kW)	CPU Time (s)	**M1** vs. **eGEP**	**RH-M2** vs. **eGEP**
Ex21	320.25	182.49	215.87	2.6	-43.0	-32.6
Ex22	4465.27	3674.04	3679.99	112.8	-17.7	-17.6
Ex23	4355.59	3497.00	3808.34	288.1	-19.7	-12.6
Ex24	1914.55	1776.14	1907.12	5.7	-7.2	-0.4
Ex25	2195.64	1789.95	1941.73	163.3	-18.5	-11.6
Ex26	2294.79	1783.95	1633.59	170.6	-22.3	-28.8
Ex27	2121.88	1684.29	1763.91	138.5	-20.6	-16.9
Ex28	1864.79	1465.37	1598.00	134.5	-21.4	-14.3
Ex29	3314.21	2583.71	2718.8	108.9	-22.0	-18.0
Ex30	2917.68	2388.63	2521.31	200.8	-18.1	-13.6
Ex31	3034.12	2486.18	2645.48	101.1	-18.1	-12.8
Ex32	3220.20	2637.50	2685.13	51.2	-18.1	-16.6
Ex33	3338.57	2523.77	2657.98	106.4	-24.4	-20.4
Ex34	4584.44	3365.35	3565.38	211.9	-26.6	-22.2
Ex35	4070.23	3035.98	3523.2	262.4	-25.4	-13.4
Ex36	3979.86	3196.92	3417.27	205.2	-19.7	-14.1
Ex37	4144.10	3477.73	3716.38	189.2	-16.1	-10.3
Ex38	3841.63	3459.03	3787.02	204.0	-10.0	-1.4
Ex39	4823.96	4041.88	3884.04	17.2	-16.2	-19.5
Ex40	4082.95	3648.90	3562.7	10.2	-10.6	-12.7
Ex41	4117.93	3589.61	3754.31	0.9	-12.8	-8.8
Ex42	4084.68	3703.47	3717.92	3.7	-9.3	-9.0
Ex43	4491.61	3782.58	3978.46	7.2	-15.8	-11.4
Ex44	6116.02	5374.11	5475.11	210.4	-12.1	-10.5
Ex45	5923.10	5195.68	4941.6	216.6	-12.3	-16.6
Ex46	6095.66	5501.46	5185.61	55.5	-9.7	-14.9
Ex47	6051.80	5916.36	5257.31	253.9	-2.2	-13.1
Ex48	6672.93	6704.23	5527.5	233.6	0.5	-17.2
Ex49	7727.28	9654.12	6951.96	4.4	24.9	-10.0
Ex50	8163.12	9953.75	7560.55	105.5	21.9	-7.4
Ex51	7173.32	9603.38	6620.26	0.9	33.9	-7.7
Ex52	7650.43	-	7060.59	106.4	-	-7.7
Ex53	7589.11	-	6938.57	2.3	-	-8.6
Ex54	10070.27	-	9167.93	374.8	-	-9.0
Ex55	10667.02	-	9708.14	509.6	-	-9.0
Ex56	10683.11	-	8861.43	394.8	-	-17.1
Ex57	9939.02	-	9610.23	551.4	-	-3.3
Ex58	10963.14	-	10003.95	494.7	-	-8.7