A Study of Time Representation in a Class of Short Term Scheduling Problems

by

Saman Lagzi

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master Mathematics
in
Combinatorics and Optimization

Waterloo, Ontario, Canada, 2016

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

The problem of scheduling operations has received significant attention from academia and industrial practitioners in the past few decades. A key decision in various scheduling operations problems is when to perform an operation and thus the quality of the final schedule can be seriously affected by the choice of how to model the times at which such a decision may take place. The two most commonly used approaches for modeling these times are: Discrete time approaches, which pre-specify a finite set of time points when any decision may be taken, and continuous time approaches, in which the optimization model determines, through the use of continuous decision variables, at which point in time the operation will be performed.

The focus of this thesis will be to study the benefits and limitations of each of these approaches within the context of an analytical services facility. Such a facility receives a large number of samples that need to be analyzed/processed through a specific sequence of limited resources/machines before its analysis is completed. The results of these analyses form a basis for many of the decisions made in their client industries (e.g. oil and mining), which in turn indicates the economic importance of the analytical services sector. The operations of such facilities have several particular conditions that need to be modeled and a particularly important one is called *multitasking*. If analyzing each type of samples is regarded as a task, then the machines in such facilities have the ability to perform multiple tasks at the same time as they are able to analyze different types of samples together at the same (as long as their capacity is not overloaded). The above mentioned study will be performed through an empirical comparison of the discrete and continuous approaches that take into account all the conditions in such facilities, including multitasking.

While discrete and continuous approaches have often been independently employed, few studies have considered a comparison between them [37, 28, 39]. In addition, none of these studies consider the operational conditions that are present in short-term scheduling of operations in an analytical services facility.

Since the continuous time formulations in the literature are not capable of accounting for multitasking, this thesis presents a novel continuous time mixed-integer linear programming (MILP) formulation that is capable of accommodating such feature and several other operational constraints present at analytical services facilities. The performance of the presented formulation is studied in comparison with a singletasking formulation. The results show that, while the multitasking formulation is not more costly in terms of solution time, it is capable of producing significantly better solutions. Furthermore, this thesis extends the idea of flexible time discretization for discrete time formulations, previously proposed by Velez and Maravelias [40], to be able to account for the operational constraints of an analytical services facility.

To proceed with the desired comparison between the state of the art continuous and discrete time formulations, it is shown, by experimenting on various instances that properly reflect the operations of the facility, that discrete formulations are typically better than their continuous counterparts. This is in contrast with previously reported results, which were performed with different operational conditions and show that such a comparison must be made within the right context to allow for proper conclusions to be taken.

Acknowledgements

I would like to thank my supervisors Dr. Ricardo Fukasawa and Dr. Luis Ricardez-Sandoval for their valuable guidance.

Dedication

This is dedicated to the one I love.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Industries that need to perform a set of operations using a set of limited resources, within a time frame, need suitable scheduling of these operations to increase efficiency, decrease costs and eventually maximize profit. In general, the problem of scheduling operations may be defined as deciding for each operation, to be performed at what time and at which resource. These decisions will be made in such a way to satisfy the operational and demand constraints of the facility, while bearing in mind the objectives of the facility, e.g., maximizing profit, increasing efficiency or minimizing cost.

As with every optimization problem, there are two sides to the problem of scheduling operations. On the one hand, an optimization model that properly reflects the conditions of the facility, including its objectives and operational constraints, is required and on the other hand, it is required to find the optimal solution to the optimization model. The modeling aspect is of particular importance since it determines the accuracy of the final solution, meaning that a model that more accurately reflects the reality of the conditions of the facility will lead to more useful solutions in practice. However, as with most problems in operations research, there is a trade-off between accuracy and tractability of the model; typically, more accurate models become harder to solve.

This thesis will focus on deterministic Mixed Integer Linear Programming (MILP) models, where all the information about the operations of the facility are known *a priori* and there is no uncertainty in the operations of the facility. For further readings on MILP and its solution methods please refer to [7]

Since scheduling inherently involves making decisions about when an event happens, e.g., the arrival of an operation or turning a machine on or off, some sort of time representation is needed. To this regards, there are two main types of models. If an event is allowed

to happen at any point in time and the model can decide at which point it will happen, then the model is called a continuous time model. However, if a finite set of choices is given to the model and events can happen only at those given choices, then the model is referred to as a discrete time model. An example will help better explaining the difference between these two approaches. Assume there is an 8 hours scheduling horizon; operation A, which will take 2 hours, needs to be performed on machine M. If the scheduling model needs a set of choices for the time that operation A will start being performed on machine M, e.g., 4 choices: hour 0, 2, 4 and 6, then, the model is a discrete time model. If the scheduling model does not require a set of choices to be given to it and can decide to perform operation A at machine M, at whatever time it considered suitable, then the model is continuous time model.

Based on the above, it is obvious that the choice of time representation affects both the accuracy and the computational cost of solving a scheduling model. Therefore, prior to modeling a scheduling of operations problem, it is essential to wisely decide upon the approach to represent time. It is a very reasonable question to ask which of the two types of time representation approaches is better. It would be extremely hard to categorically answer this question and decide one of the two approaches must always be chosen instead of the other. It seems to be much more fitting to answer this question in the context of case studies and analyze what are the properties of the scheduling problems that make each of these two approaches a better fit for the problem. Furthermore, such case studies could highlight the pros and cons of these approaches and could shed more light on the path of improving the current scheduling of operations. This thesis aims at comparing the performance of discrete time and continuous time MILP models for scheduling operations in the context of a proper case study.

The case study that is of interest to this thesis is the case of short-term scheduling of operations in an analytical services facility. The analytical services industry forms a major sector in which various types of analysis are carried out on a set of samples in order to determine its properties and chemical composition, which can be used by end-customers in the decision-making process, .e.g., Mining, Health, Petroleum and Food industries. Such facilities receive samples in the order of thousands to be processed on a daily basis. Therefore, devising an efficient scheduling algorithm for such facilities is both challenging and economically attractive. Furthermore, scheduling of operations for such facilities has not been widely studied. To the best of the author's knowledge, Patil et al [34], is the only work reported that has presented an optimal scheduling framework for this type of facilities.

The main characteristics of such facilities are as follows:

An analytical services facility receives a set of tasks, where each task is composed of

a specific number of samples. Each sample in a task needs to visit a specific sequence of processing units, called a path. Paths are associated to tasks, meaning that different tasks may have different paths, but all the samples in a task must go through the same path. Each processing unit consists of a set of machines that are identical in terms of capacity and processing time. A sample in a task needs to be processed only at one of the machines in each processing unit before it can proceed to the next processing unit in the path of its task. Machines are assumed to run continuously, i.e., without disruptions, and accordingly, samples processed in the machines cannot be removed from the machines while in operation.[1] This scheduling problem can be represented as a network of interconnected processing units, where each machine has the ability to *multitask*, i.e., they can process samples from multiple tasks simultaneously, provided the samples in the tasks are available and the machine has enough capacity to process them. Therefore, flow conservation of samples in each task through the network is necessary, meaning that it is essential, at each time point, to keep track of the number of samples in each task that are available to be processed at each processing unit in the task sequence. Furthermore, it is necessary to make sure that the same number of samples in a task that starts being processed in a machine, leaves the machine, after completing its processing.

An additional operational constraint is that some machines or samples may only become available sometime in the near future. This is because, in some cases, processing some samples at a machine may not be completed within the scheduling horizon and therefore, their processing must be completed in a later scheduling horizon.

The goal of this thesis is to perform a comprehensive comparison between the performance of discrete and continuous time MILP models for the scheduling of operations problem discussed above. Analytical services facilities can be categorized in a larger type of facilities, called multipurpose facilities, where a set of tasks need to be processed within a pre-determined scheduling horizon using a set of shared resources. Hence, in the next section, the literature on short-term scheduling of operations in multipurpose facilities will be thoroughly studied to find the discrete or continuous time MILP formulations in the literature that could be readily applied to model the scheduling of operations problem that is of interest to this work. Furthermore, such discrete and continuous time formulations are found, it is interesting to see if there has ever been a proper comparison of their performances in any context.

[1] This is sometimes referred to as a non-preemptive schedule.

1.1 Literature Review

In this section, we will explore the literature on MILP formulations of the problem of short-term scheduling of operations in multipurpose plants. The first attempts to model the operations of a multipurpose plant were done by Bowman [4] and Manne [24], in the context of the job shop problem. These studies were restricted to the situation where each task could be performed on a single machine, and therefore the main problem was to schedule a set of tasks on the machine that could perform them. The machines were prohibited from performing multiple tasks at the same time and performing a new task could only start if the machine was not performing any other tasks. As for the time representation, both studies had used a discrete time approach. They had pre-specified a finite set of time points at which the event of starting a task at a machine could happen. In their approach, the time elapsed between two consecutive time points was always the same, meaning that the time points were uniformly distributed along the axis of time. Furthermore, the locations of the time points were shared by all the machines.

This situation could be described as having a universal clock where the locations of the time points were marked, with an even distance between each two consecutive time points, on the clock. The main advantage of such a uniform time representation was its simplicity. Since the locations of the time points were known in advance, evenly distributed along the axis of time, and shared by all the machines, it was very simple to model various situations using this approach. More recent discrete time formulations [17, 36, 33, 35, 43, 9, 34] had used the same time representation approach to model various operational conditions and objectives in different multipurpose plants. One issue to notice is that a very fine discretization (for example every minute) allows for much more flexibility in the solutions and thus possibly leads to improved solutions. However, such a fine discretization also leads to an increased number of decision variables, thus resulting in larger (and more intensive) optimization problems to solve. Thus, there exists a trade-off between computational time and solution quality.

Most of the later attempts to reduce the difficulty in solving the large MILP problems resulting from the discrete-time models, were concentrated on developing better solution algorithms for these formulations, rather than trying to alleviate the modeling issues that resulted in such large MILP problems in the first place. Yee and Shah [42] attempted reformulating the discrete time formulations to reduce the gap between the LP relaxation and the optimal solution. They also added several cuts to reduce the integer infeasibility region. Dedopoulos and Shah [8] proposed intervening in the branch and bound procedure by fixing variables to values implied to the values of other variables fixed during the branch and bound procedure, which also reduced the dimensionality of the problem. Basset et al.

[2] proposed some decomposition techniques based on the time points, to decompose the large MILP problems into several smaller problems.

A common drawback in the previously cited discrete time formulations is that the discretization is the same for all the machines, meaning that, for example, the third time point happens at the same time for all the machines. To address this issue, Velez and Maravelias [40] proposed a *flexible discrete time* formulation, that is capable of assigning different discretizations to different machines. That is, a time point can happen at different times for different machines, effectively allowing many decisions to be made for some of the machines without defining unnecessary events for the rest of the machines. Such an approach has the advantage of obtaining a better balance between solution quality and computational time, since finer discretizations are only used when needed. However, the MILP formulation presented by Velez and Maravelias [40] could not be readily applied to the case study we are interested in since it had not considered the case where the machines were able to perform multiple tasks at the same time.

Continuous time models were developed much later than discrete time formulations. The main motivation behind developing these methods was the degree of inaccuracy entangled with the nature of discrete time formulations. Discrete time formulations predetermine the location of the time points, and therefore, they need a very fine discretization, e.g., one time point every minute, to have a very accurate and high quality optimal solution, which in return results in such very large problems that solving them are challenging in practice. But, if the locations of the time points could be left to the optimization model to decide, then, it is possible to reach high quality solutions without the need to introduce a time point every minute.

To address this issue, continuous time formulations have been proposed to address several scheduling problems. Furthermore, this type of continuous time formulations are capable of modeling some operational features, e.g., variable machine processing times or non identical machines in a processing unit, that are challenging for discrete time formulations, with relative ease. However, this type of formulations have not been used to address multipurpose plants with multitasking machines. The existing continuous time formulations for multipurpose plants have only considered the case where each machine is capable of processing one task at a time and therefore, they cannot be readily used for the present problem. As was mentioned before, an analytical services facility could be represented as a network of inter-connected processing units and according to Floudas and Lin [10], the continuous time formulations for network represented multipurpose facilities can be classified into two sub-classes, namely Unit-specific event based formulations and Global event based formulations.

Unit-specific event based formulations [14, 5, 23, 31] partition the planning horizon

into a sequence of time points. The location of each time point along the time axis is not specified *a priori*. Each time point specifies the beginning of a task or utilization of a unit (alternatively a machine). The time points are defined on a unit basis which allows tasks assigned to the same time point at different units to occur at different times, meaning that, for example, the second time point of machine M could happen at hour 2 while the second time point of machine W could happen at hour 3 of the scheduling horizon. Although Unit-specific event based formulations have a relatively flexible nature, according to Sundaramoorthy and Karimi [38], flow conservation of samples cannot be easily addressed using these approaches. This is because the same event points can occur at different times on two consecutive units, making it challenging to keep track, at what time, materials processed in the previous processing unit would be available for the next processing unit.

Global event based formulations [44, 16, 21, 5, 29, 41, 11, 38, 22] partition the planning horizon into several blocks of time for a predetermined number of blocks. However, each block has a length that will be determined by the optimization model itself, and blocks are common among all tasks and units. A task could start being processed at a machine only at the beginning of a time block, which means the beginning of a time block marks a time point that decisions can be taken. Although flow conservation of materials is relatively easier to achieve in these approaches, the extension of these approaches to the case of multitasking is not trivial. One key challenge is that tasks that start being processed in a machine simultaneously, need to be released simultaneously, which becomes particularly challenging since the length of the blocks of time is not known *a priori*.

A special type of multipurpose plants are called sequential processes plants. For the purpose of completeness, and since some of the continuous time formulations for this type of plants are widely used, we provide a short review of the literature on the continuous time formulations for short-term scheduling of operations in such facilities. In these plants, the tasks are not allowed to be split along the production line, meaning that the tasks need to be treated as a single block. Because of this specific assumption, continuous time models for such facilities, [6, 26, 27, 30, 13, 12, 32, 20] were able to assign tasks to machines without the use of time points. This is because they could use a prioritizing binary variable indicating which task will be performed before another task on each machine. In the context of an analytical services facility, not being able to split a task means that the samples in a task need to go through every machine together, no matter how many of them are in the task, as if they are one block. Considering the fact that some of the tasks arriving at an analytical services facility might have a very large number of samples in them, more than the capacity of the machines in some of the processing units, using these formulations means that the larger tasks must be broken into smaller tasks that could fit the machines

in all the processing units along their paths, prior to performing any optimization, which is suboptimal and very time consuming. Furthermore, these formulations have not considered the case where the machines can perform multiple tasks together.

Although there is a thick body of literature on each side of the debate of discrete versus continuous time formulations, there are actually very few studies that have compared the performance of a continuous time formulation versus a discrete time formulation. To the best of the author's knowledge, there are only 3 such studies [37, 39, 28], which are discussed next. Stefansson et al. [37] performed a comparison between the performance of continuous and discrete time formulations in the context of scheduling the operations of a pharmaceutical production facility. The continuous time formulation used in their study was based on the widely used formulation of Méndez et al. [27] which does not allow for splitting the tasks. Furthermore, their discrete time formulation, based on the work by Shah et al [36], was not capable of accommodating flexible time discretization. They had reported that the continuous time formulation had outperformed the discrete time formulation. The work of Sundaramoorthy and Maravelias [39] performed a thorough comparison between continuous and discrete time formulations in the literature. Their continuous time formulation was based on the formulation by Sundaramoorthy and Karimi [38], which does not consider the multitasking feature in the machines of the facility. The discrete time formulation that Sundaramoorthy and Maravelias [39] used was also based on the work by Shah et al [36] which does not allow for flexible time discretization. Their results showed that discrete time formulations could not be ruled as inferior to continuous time formulations, specifically when the size of the problems grow. Finally, the work of Merchan et al [28] performed a comparison between the performance of flexible discrete time formulations and a continuous time formulation. The continuous time formulation used in their study was based on the work by Méndez et al [27] which does not allow for tasks to be split along the production line.

In summary, none of the existing comparison studies in the literature consider comparing the performance of the newly developed flexible discrete time formulations versus continuous time formulations that allow for splitting tasks along the production line, simultaneously. Furthermore, none of the formulations used in these comparisons have considered the multitasking feature of the machines in the facility.

In the next section, we will discuss the existing gaps in the literature of continuous and discrete MILP formulations for short-term scheduling of operations in multipurpose plants, and how these gaps manifest themselves in the context of the considered case study, scheduling operations in an analytical services facility. Furthermore, the next subsection will highlight the goal of the current thesis and its contributions to filling the gaps in the literature through fulfilling its goal.

1.2 Thesis objectives and contributions

One of the main gaps in the literature of MILP formulations for scheduling operations in multipurpose plants is the fact that currently there is no continuous time formulation that can account for multitasking feature of the machines in an analytical services facility. That is, none of the existing formulations in the literature can be readily applied to model the operation of an analytical services facility without a major compromise, which consists of forcing the machines in the facility to process samples from only one task at any time. One of the goals of this thesis is to develop a continuous time formulation to accurately model the operations of the analytical services facilities, including the multitasking feature of the machines. To achieve this goal, Global event based continuous time formulations for multipurpose plants are extended by adding features that allow multitasking in the facility and thus enable them to produce better schedules in terms of resource utilization. We chose to extend Global event based formulations over other existing formulations in the literature, primarily because in these formulations events are shared between all the processing units, which makes flow conservation of samples through the network of the processing units much easier. In addition, the proposed formulation is able to model the extra operational condition that processing some samples at some machines may not be completed within the current scheduling horizon and thus they need to be considered in the next schedule. The proposed formulation can be readily used to model the problem of scheduling operations in analytical services facilities.

The next goal of this thesis is to extend the discrete time formulation presented by Patil et al [34] for modeling the operations of an analytical services facility, to be able to account for flexible time discretization, using the idea of flexible time discretization previously proposed by Velez and Maravelias [40]. This is very important since the idea of flexible time discretization is new and it needs to be applied to various situations before judging its merits and understanding its pros and cons. Furthermore, the problem of scheduling operations in analytical services facilities is substantially understudied and proper flexible discrete time formulations will enrich the literature on this problem.

Furthermore, the major goal of this thesis is to perform a thorough computational study of the choice of time representation in the context of optimal scheduling of the operations of an analytical services facility. The desired study will be conducted through a comprehensive comparison between the results obtained through both the flexible discrete and continuous time formulations that have been developed in this thesis. Such a study will fill the existing gap in the literature for comparisons between flexible discrete time formulations and those formulations that are able to split the samples in tasks along the production line and can account for multitasking feature of the machines in the facility.

Thus, the main contributions of this thesis are as follows:

- Developing a multitasking continuous time formulation that can handle the multitasking feature of the machines in an analytical services facility. It was shown through an extensive computational study that the multitasking continuous time formulation outperforms its singletasking counterpart, both in term of quality of the objective function and computational complexity, and therefore, substantiating the need to account the multitasking feature of machines in the facility.
- Extending the discrete time formulation by Patil et al [34] to be able to account for flexible time discretization.
- Performing a thorough comparison between the performance of the developed flexible discrete and continuous time formulations, to shed more light on the specific advantages and disadvantages of each of the two main time representation approaches in the literature of short-term scheduling of operations.

In the next section, the problem of scheduling of operations in an analytical services facility is described.

1.3 Problem definition

The problem that will be studied in this work can be described as follows: An analytical services facility receives a set of tasks, I, and needs to process them within a scheduling horizon, H, using a set of processing units, P. Each processing unit, $p \in P$, consists of a set of identical machines, J_p, that perform a specific process. The set of all machines in the facility is denoted by J, where J is the disjoint union of J_p's. Task $i \in I$, consists of a_i number of samples that need to visit a specific sequence of processing units, called a path. The path of task i, denoted by S_i, is a sequence of $n(i)$ distinct processing units $(p^i_1, \ldots, p^i_{n(i)})$, where $p^i_k \in P$ for all $i \in I$, $k = 1, \ldots, n(i)$. The samples in task i must visit every processing unit in a pre-specified sequence defined by S_i, that is, p^i_k in S_i can only be visited if p^i_{k-1} has already been visited. A sample is considered to have visited processing unit p if it has been processed by one of the machines in J_p.

Machines in a processing unit p have a specific capacity, denoted as β_p, and an associated processing time, denoted as $\tau(p)$. This means that machine $j \in J_p$ can be loaded with at most β_p number of samples from potentially different tasks. Once a machine has been turned on to process the samples, it will run without interruption for a time $\tau(p)$. After this

time, the machine is considered to be available; also, the samples are considered to have visited the corresponding processing unit and they are ready to visit the next processing unit in their path. Since machines are assumed to be identical, the processing time of the machines in a processing unit can be referred to as the processing time of the processing unit. Furthermore, there is no minimum working capacity for any machine, that is, the machines can be turned on with any number between 0 and β_p samples.

Samples in task $i \in I$ will be available to start visiting the first processing unit in their path, p_1^i, at time TA_i; machine $j \in J$ will become available to start processing samples at the facility at time TM_j. It is possible for TA_i to happen after the beginning of the current scheduling horizon, if the samples in task i were backlogged from the previous scheduling horizon, meaning that they were part of another task in the previous scheduling horizon and did not visit all the processing units in their path. In such cases, S_i is the sequence of the remaining processing units that have not been visited. If machine j is processing samples when the current scheduling horizon begins, TM_j will represent the time that j will finish such processing. Otherwise, TM_j coincides with the beginning of the current scheduling horizon H.

It is assumed that the information described above is available and known *a priori*. There are three objectives for the schedule to meet. The first objective is to maximize the total number of samples that need to complete their processing in the facility. The second objective is to maximize the task throughput of the machines in the facility. The throughput of task i for machine j, is the total number of samples from task i processed at machines $j \in J_p$ for every $p \in S_i$, throughout the scheduling horizon. The third objective is to minimize the turnaround time of the operations in the facility, i.e., minimizing the latest time that the samples from a task leave the facility. Therefore, the objective function considered in this work is a weighted summation of these three different objectives. The total number of finished samples from task i is added to the objective function with weight f_i. The throughput of task i for machine j is added to the objective function with weight c_{ij} and we assume $c_{ij} = c_{ij'}$ if $j, j' \in J_p$ for some $p \in P$. Also, to simplify notation, we assume $c_{ij} = 0$ if $j \notin \bigcup_{p \in S_i} J_p$ so that c_{ij} is defined for any $i \in I$, $j \in J$. Minimizing turnaround time is achieved by minimizing the latest time that a machine from the k-th processing unit in the path of task i finishes processing a sample from it. Thus, the objective function includes a weight d_{ik} that has a negative value and represents the latest time that samples from task i are processed at a machine in processing unit $p_k^i, \forall\ k = 1, \ldots, n(i)$. Note that in fact to minimize turnaround time, the model should only have such penalty for $k = n(i)$. However, the penalty terms were included for all $k = 1, \ldots, n(i)$ so that the optimization model gives preference to finishing a particular task as early as possible.

1.4 Thesis Structure

The rest of this thesis is structured as follows:

Chapter 2 presents the multitasking continuous time formulation that can be readily applied to model the operations of an analytical services facility. Later in the chapter, an illustrative instance is presented with the Gantt chart of the schedule obtained through the proposed model. Furthermore, an extensive comparative study between the continuous time formulation and its singletasking counterpart, the widely used continuous time formulation previously proposed by [38], will be carried out through solving several instances.

Chapter 3 will present the flexible discrete time formulation as well as an illustrative instance that will be solved through the flexible discrete time formulation along with its scheduling Gantt chart.

Chapter 4 will present a comprehensive comparison between the results obtained through both the continuous time formulation and the flexible discrete time formulation. The quality of the objective function and the computational complexity of the models will be explored through solving several instances ranging from small to relatively large instances. Also, the merits of flexible time discretization, as opposed to the old fashioned uniform time discretization approach, will be explored. Later in the chapter, some insight will be given upon the possible reasons behind the behaviors of each of the formulations.

Chapter 5 will present the conclusions that can be taken from the research presented in this thesis along with some further insight on the possible tracks that can be taken for future research.

Chapter 2

Continuous Time Formulation

In this chapter, a continuous time formulation for the problem that ws defined in the previous chapter, is presented. Our proposed formulation extends Global event based continuous time formulations to be able to account for the multitasking feature of the machines in the facility. There are a few reasons behind why these types of formulations were chosen over the other formulations in the literature.

Recall that there are two other types of formulations in the literature, which were explained in the previous chapter: Sequential processes formulations and Unit-specific event based formulations. The former is only applicable to those facilities where the samples in a task move as one single block throughout the facility, and is not applicable to analytical services facilities where splitting the samples in a task along the facility might happen very often. The Unit-specific event based formulations are built upon the idea that the same event point can happen at different times from two different machines. That means that, for example, event 2 for machine 1 can happen at hour 1 and it can happen at hour 2 for machine 2. This becomes a substantial challenge if we have a task whose samples need to visit machine 1 and after finishing their processing there, they must visit machine 2. In such a case, it would be very hard to coordinate the samples that leave machine 1 with the samples that visit machine 2. Even with these potential drawbacks, attempts were made in this work to extend Unit-specific event based formulations to the case of interest. However, the resulting formulation had an extremely large number of binary variables, making it quickly impractical, so it was chosen not to present such effort here.

Next, the continuous-time formulation developed from the global event point formulations is presented. Later in this chapter, we present a thorough computational comparison between the presented formulation and one of the most widely used continuous time formulations in the literature.

2.1 Formulation

To derive a continuous-time formulation, the time domain has been partitioned into a pre-determined number of time points, where the location of each time point along the time axis will be obtained from the optimization model. The time points are universal and shared by all the tasks and machines, meaning that a time point occurs at the same time for all machines and tasks. In the present formulation, the locations of two particular time points have been fixed *a priori*, i.e., the first time point, denoted as 0, has its location fixed at the beginning of the scheduling horizon, while the last time point, denoted by N, has its location fixed at the end of the scheduling horizon, H. The time between two consecutive time points is denoted as a time slot.
Accordingly, let $T_n \in [0, H] \ \forall \ n = 0, \ldots, N$, be the decision variable representing the location of time point n and $SL_n \in [0, H]$ be the decision variable representing the length of time slot n, where time slot n is the time between T_n and T_{n-1}, for all $n \in 1, \ldots, N$. Constraints (2.1)-(2.3) model the relationship between T_n and SL_n consistent with the above description, and ensure that the time points happen in order (time point $i-1$ happens no latter than time point i, since SL_n are nonnegative).

$$\sum_{n=1}^{N} SL_n = H, \tag{2.1}$$

$$T_n - T_{n-1} = SL_n; \quad \forall \ n = 1, \ldots, N, \tag{2.2}$$

$$T_0 = 0. \tag{2.3}$$

A time point specifies an alternative to start or finish processing samples from a set of tasks at a machine. At each time point, multiple machines from different processing units could either start or finish processing samples. That is, if a sample in task i is scheduled to start being processed at machine j at time point n, the machine is available to be used and the task has been accepted at the facility before time point n. To account for these conditions, the following binary variable is introduced:

$$Y_{ijn} = \begin{cases} 1, & \text{If a sample from task } i \text{ is assigned to machine } j \text{ to start being processed at time point } n; \ \forall \ j \in J, \ i \in I, \ n = 0, \ldots, N. \\ 0, & \text{Otherwise.} \end{cases}$$

Accordingly, Constraint (2.4) is introduced to ensure that a sample in a task will be

assigned to a machine for processing, only after the task is accepted at the facility, while Constraint (2.5) ensures that a sample from task i will be processed at machine j only after the machine has become available to be used in the facility. For every $p \in P$ and $j \in J_p$, we denote by I_j, the subset of the tasks $i \in I$ where $p \in S_i$.

$$T_n \geq TA_i Y_{ijn}; \quad \forall\ j \in J,\ i \in I_j,\ n = 0, \ldots, N. \tag{2.4}$$

$$T_n \geq TM_j Y_{ijn}; \quad \forall\ j \in J,\ i \in I_j,\ n = 0, \ldots, N. \tag{2.5}$$

The number of time points is an input to the model that requires tuning. N might have been chosen too small, meaning that, increasing N will result in better solutions. Furthermore, if the choice of N is excessively large, it will result in one or more time points being unused, which means some machines may idle. To deal with this condition, similar to the approach proposed by [38], an idle task is introduced. The idle task is a task that does not have a specific processing sequence, i.e., it does not have a specific path. If a machine is not processing samples from a real task, then it is processing samples from the idle task. Furthermore, the idle task does not have a processing time, i.e., a machine can finish processing samples in the idle task at any time and begin processing samples from a set of real tasks. In addition, a machine can potentially stay idle throughout the scheduling horizon. Moreover, the samples in the idle task are ready to be processed at any time during the scheduling horizon, at every machine in all the processing units. That is, a machine will become idle at any time during the scheduling horizon if it is not processing samples from other tasks. Furthermore, samples from the idle task cannot be processed in a machine together with samples from the real tasks at any time. In the present formulation, the idle task is denoted as task number 0, whereas $I = 1, ..., |I|$, is the set of the real tasks. Accordingly, the following notation is introduced in relation to the concept of an idle task:

$$Y_{0jn} = \begin{cases} 1, & \text{If machine } j \text{ starts processing samples from the idle task} \\ & \text{at time point } n,\ \forall\ j \in J,\ \ n = 0, \ldots, N. \\ 0, & \text{Otherwise.} \end{cases}$$

Since a time point specifies an alternative to start processing samples at a machine, it is necessary to ensure that the time points are being used consistently with the starting of the machines. For that purpose, the following binary variable is introduced:

$$Z_{jn} = \begin{cases} 1, & \text{If machine } j \text{ starts processing samples from a set of tasks} \\ & \text{at time point } n. \quad \forall\ j \in J, \quad n = 0, \ldots, N. \\ \\ 0, & \text{Otherwise.} \end{cases}$$

Constraint (2.6) ensures that, if samples from a set of tasks will start being processed in machine j at time point n, then machine j will be turned on at that time point. Constraint (2.7) ensures that machine j will not start processing at time point n if no samples are scheduled to start being processed in machine j at that time point.

$$Z_{jn} \geq\ Y_{ijn}; \quad \forall\ j \in J,\ i \in I_j \cup \{0\},\ n = 0, \ldots, N. \tag{2.6}$$

$$Z_{jn} \leq \sum_{i \in I_j \cup \{0\}} Y_{ijn}; \quad \forall\ j \in J,\ n = 0, \ldots, N. \tag{2.7}$$

After a machine starts processing samples from a set of tasks, the processing might continue for several consecutive time slots before it is completed at a later time point. Therefore, at each time point, it is necessary to keep track of the samples that continue to be processed at each machine, as well as the samples that complete their processing. To deal with these conditions the following notation is introduced:

$$YE_{ijn} = \begin{cases} 1, & \text{If a sample from task } i \text{ completed its processing at machine} \\ & j \text{ at time point } n; \ \forall\ j \in J,\ i \in I \cup \{0\},\ n = 0, \ldots, N. \\ \\ 0, & \text{Otherwise.} \end{cases}$$

$$YR_{ijn} = \begin{cases} 1, & \text{If a sample from task } i \text{ continues being processed at ma-} \\ & \text{chine } j \text{ at time point } n; \ \forall\ j \in J,\ i \in I \cup \{0\},\ n = 0, \ldots, N. \\ \\ 0, & \text{Otherwise.} \end{cases}$$

Constraint (2.8) assures that, if a sample in task i started or continued to be processed at machine j at time point $n-1$, then, at the next time point the samples will either

complete their processing or continue being processed in machine j.

$$YR_{ijn} = YR_{ij(n-1)} + Y_{ij(n-1)} - YE_{ijn};\ \ \forall\ j \in J,\ i \in I_j \cup \{0\},\ n = 1, \ldots, N. \tag{2.8}$$

Constraint (2.9) considers that, if a sample in task i completes its processing at machine j at time point n, then machine j will immediately start processing a new set of samples, either from the same tasks or from a new set of tasks (including the idle task).

$$Z_{jn} \geq YE_{ijn};\ \ \forall\ j \in J,\ i \in I_j \cup \{0\},\ n = 0 \ldots, N. \tag{2.9}$$

Constraints (2.10)-(2.11) ensure that machine j starts processing a new set of samples from a set of tasks at time point n, if and only if it is not continuing processing samples from any set of tasks at that time point.

$$Z_{jn} \leq 1 - YR_{ijn};\ \ \forall\ j \in J,\ i \in I_j \cup \{0\},\ n = 0, \ldots, N. \tag{2.10}$$

$$(1 - \sum_{i \in I_j \cup \{0\}} YR_{ijn}) \leq Z_{jn};\ \ \forall\ j \in J,\ n = 0, \ldots, N. \tag{2.11}$$

The samples cannot be inserted or removed from a machine while the machine is running. Therefore, it is required to ensure that samples from a set of tasks may continue to be processed at machine j at time point n if and only if the machine is not scheduled to complete processing any samples at that time point. This is achieved through Constraint (2.9) along with Constraint (2.10). Also, since a machine cannot both continue processing a set of samples and start processing a new set of samples at the same time point, Constraint (2.6) and Constraint (2.10) ensure that if samples from a set of tasks are scheduled to continue being processed at machine j at time point n, then machine j cannot start processing a new set of samples at that time point.

Constraint (2.12) ensures that a machine cannot process samples from a real task while the machine is processing the idle task.

$$Y_{ijn} \leq 1 - Y_{0jn};\ \ \forall\ j \in J,\ i \in I_j, n = 0, \ldots, N. \tag{2.12}$$

Capacity constraints are required to determine how many samples from each task will be processed in each machine at each time point. Accordingly, the following notation is introduced:
B_{ijn}: A nonnegative integer variable, representing the number of samples from task i that begin processing at machine j at time point n; $\forall\ j \in J,\ i \in I_j \cup \{0\},\ n = 0, \ldots, N$.
BE_{ijn}: A nonnegative integer variable, representing the number of samples from task i that complete their processing at machine j at time point n; $\forall\ j \in J,\ i \in I_j \cup \{0\},\ n = 0, \ldots, N$.

BR_{ijn}: A nonnegative integer variable, representing the number of samples from task i that continue their processing at machine j at time point n; $\forall\, j \in J,\ i \in I_j \cup \{0\},\ n = 0, \ldots, N$.

Constraint (2.13) considers that, if machine j is not assigned to process samples from task i at time point n, then the set of samples in task i assigned to machine j to start being processed at that time point, is empty. Constraint (2.14) is placed to ensure that the total number of all the samples from different tasks, scheduled to start being processed at machine j at time point n, does not exceed the capacity of the machine.

$$B_{ijn} \leq \beta_p Y_{ijn};\quad \forall\, p \in P,\ j \in J_p,\ i \in I_j \cup \{0\},\ n = 0, \ldots, N. \tag{2.13}$$

$$\sum_{i \in I_j \cup \{0\}} B_{ijn} \leq \beta_p Z_{jn};\quad \forall\, p \in P,\ j \in J_p,\ n = 0, \ldots, N. \tag{2.14}$$

To extend the capacity constraints to the situation where processing samples from task i is set to be continued or completed at machine j at time point n, Constraints (2.15)-(2.18) are introduced.

$$BR_{ijn} \leq \beta_p Y R_{ijn};\quad \forall\, p \in P,\ j \in J_p,\ i \in I_j \cup \{0\},\ n = 0, \ldots, N. \tag{2.15}$$

$$\sum_{i \in I_j \cup \{0\}} BR_{ijn} \leq \beta_p (1 - Z_{jn});\quad \forall\, p \in P,\ j \in J_p,\ n = 0, \ldots, N. \tag{2.16}$$

$$BE_{ijn} \leq \beta_p Y E_{ijn};\quad \forall\, p \in P,\ j \in J_p,\ i \in I_j \cup \{0\},\ n = 0, \ldots, N. \tag{2.17}$$

$$\sum_{i \in I_j \cup \{0\}} BE_{ijn} \leq \beta_p Z_{jn};\quad \forall\, p \in P,\ j \in J_p,\ n = 0, \ldots, N. \tag{2.18}$$

Constraint (2.19) is considered to ensure that the total number of samples in task i that is processed in the facility, throughout the scheduling horizon, does not exceed the total number of samples in task i, i.e., a_i.

$$\sum_{1 \leq n \leq N} \sum_{j \in J_p:\ p = p_1^i} B_{ijn} \leq a_i;\quad \forall\, i \in I. \tag{2.19}$$

Most facilities require conservation of materials for feasible operation of the plant. Therefore, Constraint (2.20) is considered to ensure that the amount of samples from task i that start being processed or continue to be processed at machine j at time point n, is

equal to the amount of samples from task i that complete or continue their processing at machine j at the next time point.

$$BR_{ijn} + BE_{ijn} = BR_{ij(n-1)} + B_{ij(n-1)}; \quad \forall\ j \in J,\ i \in I_j,\ n = 1, \ldots, N. \tag{2.20}$$

The next constraint is introduced to ensure that a sample from task i can visit processing unit p_k^i in S_i at time point n, only if it has already visited the previous processing unit, p_{k-1}^i, in S_i. Note that a sample from task i is considered to have visited processing unit p_k^i in S_i, if it has been processed by one of the machines in $J_{p_k^i}$. To account for this condition, the following notation is introduced:

W_{ikn}: is a nonnegative integer variable representing the number of samples from task i that have visited p_{k-1}^i and are waiting to visit processing unit p_k^i at time point n; $\forall\ i \in I,\ k = 2, \ldots, n(i),\ n = 0, \ldots, N$.

Constraint (2.21) ensures that a sample from task i can visit processing unit p_k^i in S_i at time point n, if it has already visited processing unit p_{k-1}^i in S_i before time point n.

$$\sum_{j \in J_p:\ p = p_k^i} B_{ijn} + W_{ikn} = W_{ik(n-1)} + \sum_{j \in J_p:\ p = p_{k-1}^i} BE_{ijn}; \quad \forall\ i \in I, \quad k = 2, \ldots, n(i),\ n = 1, \ldots, N. \tag{2.21}$$

Constraints (2.20) and (2.21) are referred to as flow conservation constraints, since they can be interpreted as corresponding samples to a flow that, at any time point, either are waiting to be put in a machine, are set to be put in a machine or remain in a machine, i.e. no samples are lost or created.

The next step in the formulation is to ensure that, if a number of samples start to be processed at a machine, it takes an amount of time equal to the processing time of the machine, before the samples complete their processing at that machine. For this purpose we define the following auxiliary decision variables.
TR_{ijn}: is a nonnegative continuous variable, representing the amount of time remaining to complete processing samples from task i that continue to be processed at machine j at time point n; $\forall\ j \in J,\ i \in I_j \cup \{0\},\ n = 0, \ldots, N$.

Constraint (2.22) ensures that the amount of time remaining to complete processing samples from task i at machine j at time point n can have a positive value, if and only if machine j is set to continue processing the samples at time point n. If the machine is set to start processing a new set of samples or finish processing a set of samples, i.e., Y_{ijn} or YE_{ijn} variables take a non-zero value, then this remaining time would take a value of

zero. To better understand this, assume that either of Y_{ijn} or YE_{ijn} have taken a value of 1. Then due to constraints (2.6) and (2.9), the variable Z_{jn} would take a value of 1 which in return, due to constraint (2.10), variables YR_{ijn} will take value of zero. Then, due to constraint (2.22), TR_{ijn} will take a value of zero.

Constraint (2.23) is considered to relate the remaining time to complete processing samples from task i at machine j at time point n, to the processing time of machine j and the amount of time spent on processing these samples at machine j prior to time point n.

$$TR_{ijn} \leq \tau(p) YR_{ijn}; \quad \forall\, p \in P,\ j \in J_p,\ i \in I_j \cup \{0\},\ n = 0, \ldots, N. \tag{2.22}$$

$$TR_{ij(n+1)} \geq TR_{ijn} + \tau(p) Y_{ijn} - SL_{n+1}; \quad \forall\, p \in P,\ j \in J,\ i \in I_j,\ n = 0, \ldots, N. \tag{2.23}$$

Constraint (2.23) is not considered for the idle task, since a machine can stop being idle at any time if samples from a real task are assigned to start being processed at the machine at that time.

To account for the minimization of the turnaround time of the operations in the objective function, an extra variable and additional constraint are needed.

Tf_{ik}: A nonnegative continuous variable representing the latest time that a sample from task i finishes visiting processing unit p^i_k; $\forall\, i \in I \cup \{0\},\ k = 1, \ldots, n(i)$.

Constraint (2.24) sets the value for the latest time that a sample from task i finishes visiting processing unit p^i_k.

$$Tf_{ik} \geq T_n - (1 - YE_{ijn})H; \quad \forall\, i \in I,\ k = 1, \ldots, n(i),\ j \in J_{p^i_k},\ n = 0, \ldots, N. \tag{2.24}$$

The throughput of task i for each machine j along its path, is defined as follows:

$$\sum_{n=0}^{N} B_{ijn} \quad \forall\, i \in I,\ j \in J_{p^i_k},\ k = 1, \ldots, n(i).$$

Likewise the total number of samples from task i that have finished their processing in the facility can be expressed as follows:

$$\sum_{n=0}^{N} \sum_{j \in J_p:\ p = p^i_{n(i)}} BE_{ijn}$$

Hence, the objective function for the continuous time formulation is:

$$\sum_{n=0}^{N}\sum_{i\in I}\sum_{j\in J} c_{ij}B_{ijn} + \sum_{i\in I}\sum_{n=0}^{N}\sum_{j\in J_p:p=p^i_{n(i)}} f_i BE_{ijn} + \sum_{i\in I}\sum_{k=1}^{n(i)} d_{ik}Tf_{ik}. \tag{2.25}$$

The presented formulation aims at maximizing the objective (2.25), subject to constraints (2.1)-(2.24).

We end this section by commenting on which parts of the model have been developed in previous works and which parts needed to be added or modified (and why) to account for multitasking and other operational conditions. Constraints (2.4)-(2.5) were added to the original formulation by [38] to account for the arrival of the samples and availability of the machines after the beginning of the scheduling horizon. The original formulation had no counterpart for these constraints since it had not considered such a situation. Constraints (2.6)-(2.7) needed to be added to the original formulation in their current format, to account for the case where samples from multiple tasks needed to start being processed at the same time in a machine. These constraints replaced a more simple constraint in the original formulation that would have triggered turning on machine j at time point n if and only if a task was assigned to the machine at that time point, with the assumption that at most one task could be assigned to the machine at any time point, which clearly cannot account for the situation considered in the current work.

Constraints (2.9)-(2.11) were added to the original formulation to account for the situation where samples from multiple tasks were scheduled to finish their processing or continue their processing at a machine. They also replaced two simpler constraints that were built based on the assumption that at any time at most one task could finish its processing or continue its processing at any machine. Constraint (2.12) was added to the original formulation to prevent the idle task to be performed along with a real task in a machine. It was not present in the original formulation since, by nature, in that formulation multiple tasks could not be processed together at the same time in one machine.

Capacity constraints, Constraints (2.14), (2.16) and (2.18), needed to be added to the original formulation to account for the capacity of the machines while processing multiple tasks at the same time. The rest of the capacity constraints, Constraints (2.13), (2.15) and (2.17), were enough for the original formulation since the assumption in that work was that only one task could be performed at a machine at any time, so forcing the capacity constraint on that task was enough. Constraint (2.19) was added to the original formulation to bound the number of samples in each task, and was not present in the original formulation since, by nature, the tasks considered in that work were only limited to the capacity of the machines processing them.

Constraints (2.22)-(2.23) along with TR_{ijn} variables needed to be adapted so that the tasks that start together at a machine will finish their processing together. TR_{ijn} had no i index in the original formulation, since because of the singletasking assumption, it was only necessary to monitor the single ongoing task on each machine as opposed to the set of tasks assigned to that machine. Furthermore, since the original formulation had not considered minimizing the latest finishing time of the tasks, Tf_{ik} variables and Constraint (2.24) were added to the original formulation as well.

The rest of the constraints, Constraints (2.1)-(2.3), (2.8), and (2.20)-(2.21) were already present in the original formulation.

2.2 Illustrative Instance

To better illustrate the schedules that can be obtained through the presented model, a small instance is presented and solved using the model; the corresponding schedule is depicted through a Gantt chart. The number of time points used is determined through increasing the number of time points by 1 and solving the instance iteratively until no further improvement in the objective function can be achieved through increasing the number of time points by 1 or 2. This method is heuristic and the objective function that is reported is only optimal for the chosen number of time points and it can not be guaranteed that a better objective function cannot be achieved with a much larger number of time points. This heuristic was previously used in [5], which reported that, although increasing the number of time points by 1 might not result in an improvement in the objective function, the objective function might improve with an increase of two or more in the number of time points. Table 2.1 shows the information related to the processing units and their corresponding machines. Please note that the processing times are provided in terms of minutes.

Processing Unit	J_p	β_p	$\tau(p)$	TM_j
1	1	140	50	0.0
2	2,3	70	30	0.0,0.0
3	4,5	50	60	0.0,0.0
4	6	120	195	0.0

Table 2.1: Processing Units Information

Likewise, Table 2.2, lists the information related to the tasks that need to be performed in the facility within an 8 hours scheduling horizon.

Task	S_i	a_i	TA_i
0	N/A	unlimited	0.0
1	1,3,4	120	0.0
2	1,2,3,4	100	0.0

Table 2.2: Tasks Information

As for the weights in the objective function, $\forall\, i \in I$, f_i is arbitrarily chosen to be equal to 10 while for all the machines in every processing unit, p, and every task, $i \in I_j$, the weight of the throughput of task i for machine $j \in J_p$ is chosen to be 1. The weight of the latest finishing time of a sample from task $i \in I$ at processing unit $p^i_{n(i)}$, $d_{in(i)}$, is arbitrarily chosen to be equal to $-.05$ while if $k \neq n(i)$, then d_{ik} is chosen to be $-.025$ for all $i \in I$.

The total number of time points is 8 and the objective function value is 1915.87; the solution time is 10.25 seconds and does not include the time spent on tuning the number of time points. Figure 2.1, shows the schedule obtained from the model. Note that only the actual tasks have been depicted. In each box, the first number represents the task whereas the second number is the number of samples from the task being processed in the corresponding machine, while the end of the box indicates the end of a time interval, i.e., T variables. The horizontal axis of the Gantt chart (time) is marked with hours.

The Gantt chart shows that the model behaves as expected, by scheduling as many samples as possible on the machines and enforcing multitasking in the machines at points were it is beneficial to process samples from both of the actual tasks on a machine at the same time. Furthermore, the schedule tries to minimize the latest time that the samples from a task visit the processing units along their path.

In addition, this example also can be used to highlight the importance of having the ability to model the fact that some machines and/or samples will only be available sometime in the near future. To illustrate this, note that some of the samples in tasks 1 and 2 do not finish their processing in the facility within the 8 hours scheduling horizon.

Suppose the instance studied is modified a little so that the samples in task 1 need to visit another processing unit, e.g., processing unit 5, after visiting processing unit 4. The staff of the facility are allowed to work overtime to finish processing the samples in task 1. To accommodate processing the samples in task 1, another scheduling horizon is required. To circumvent this condition, the rolling-horizon technique can be employed where the

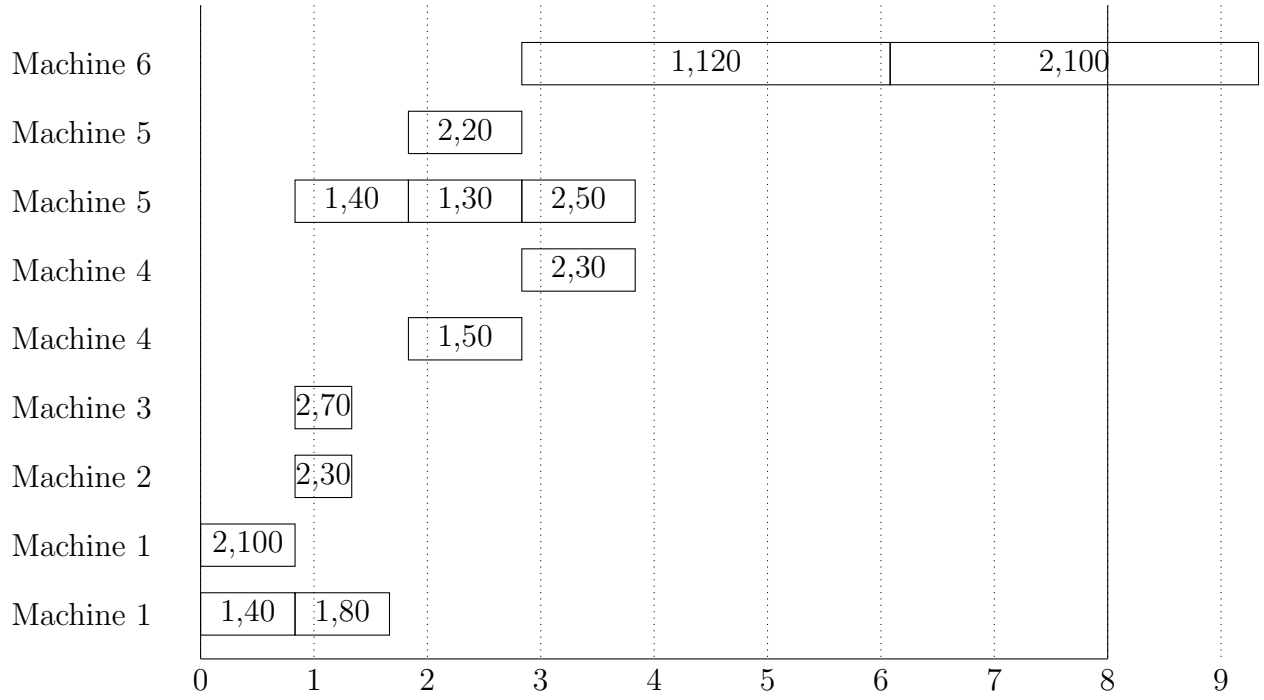

Figure 2.1: Illustrative Example, Objective Value =1915.88, N=7

remaining tasks will be performed in the later scheduling horizons (e.g. from hours 8-16). However, most of the conventional continuous time formulations either need to disregard the unavailable samples in task 1 in the next schedule, or they need to wait until the end of hour 9.33, which is the time that part of the samples in task 1 will finish their processing at processing unit 3, before they can start the next scheduling horizon. This is because they need all the samples in all the tasks to be ready for being processed in the facility at the beginning of the scheduling horizon. The present model is capable of starting the next scheduling horizon right at the end of hour 8, as opposed to the end of hour 9.33, and accommodate the fact that the samples in some of the tasks will be available to be processed in the facility after the beginning of the scheduling horizon. This capability will reduce the idleness of the machines in the facility, which may result in reducing the turnaround time of the facility as well as increasing its overall throughput.

2.3 Comparative Study of Multitasking

One of the main differences between the proposed continuous time formulation and the traditional continuous time formulations available in the literature is the ability of the current formulation to accommodate the multitasking feature in the machines i.e., machines in the facility have the ability to process samples from multiple tasks at the same time. Overlooking this feature may result in worse solutions. To cast more light onto this issue, 6 different instances have been designed and solved using the presented continuous time formulation and a version of the formulation that prohibits multitasking in the machines, previously proposed by [38].

In that study, they had only considered maximizing throughput as the objective function, and therefore, they had no Tf_{ik} variables, representing the latest finishing times. To have a fair comparison between the current approach and their approach, we only consider maximizing throughput of the facility as the objective function for both of the multitasking model and their single tasking model. Hence, the objective, for the comparative study can be expressed as follows:

$$max \sum_{n=0}^{N} \sum_{i \in I} \sum_{j \in J} c_{ij} B_{ijn}.$$

Furthermore, Tf_{ik} variables and Constraint 2.24 that regulates the values for those variables, will also not be considered in the multitasking formulation. The 6 instances used for the comparative study were inspired by the operations of an actual analytical services facility. The goal is to evaluate the performance of the models through instances that share many features with real instances from an actual facility.

In the considered facility, for the samples in a task, visiting the last processing unit in the path of the task has priority over visiting the rest of the processing units in the path. To better illustrate this, assume that the samples of two different tasks, e.g., task 1 and task 2, are competing for the machines of a processing unit, e.g., processing unit 6. Also assume that the samples in task 1 can finish their processing in the facility by visiting processing unit 6, while the samples in task 2 need to visit other processing units after visiting processing unit 6. Then, processing the samples in task 1 at the machines of processing unit 6, $j \in J_6$, has priority over processing the samples in task 2. Moreover, all the tasks in the facility have the same first and last processing units in their paths. This is because all the tasks need to visit a specific processing unit (e.g., preparation unit) before they can be admitted into the facility and they need to visit a specific processing unit (e.g., quality control) before they can leave the facility.

The same objective function is considered for all the instances. To properly reflect the conditions of the facility, c_{ij} is arbitrarily chosen to be 5 for machine j and task i if $j \in J_p$ for $p = p^i_{n(i)}$ while, c_{ij} is chosen to be 1 otherwise. Note that the weights are different than the ones used in the illustrative instance in the previous section, since here a different objective function is undertaken. The staff of the facility is assumed to work 8 hours per day. Since most of the working staff would prefer to know what the schedule throughout the day would be before the starting of the day, shorter than 8 hours schedules would be less convenient while longer than 8 hours schedules means that there needs to be an overtime working for the staff, which is inconvenient. Therefore, a convenient schedule should have a scheduling horizon of 8 hours, which is the scheduling horizon for all the studied instances.

All the computational experiments were performed on a machine that has 250 GB RAM and 4 CPUs, each with 12 cores and a processing speed of 2.4 GHz using CPLEX [15]. Each instance is solved within a time window of 20 minutes and each instance is solved for multiple numbers of time points. The number of time points has been increased to a point where increasing the number of time points by one or two does not improve the objective function within the 20 minutes solution time for any of the two models.

Table 2.3 presents the data related to the size of each instance. The first number in the name of each instance indicates the number of the processing units, while the second number indicates the number of actual tasks considered in the instance. Note that instances P4-3-1 and P-4-3-2 have the same number of processing units and tasks but are completely different instances (the number of samples in each task, the number of machines in each processing unit and the machines specifications such as processing times and capacities are different). Table 2.4 presents the computational results. The columns in Table 2.4 represent the following: *Obj. Val.* stands for the largest feasible objective function value found, *Bin. Var.* and *Gen. Var.* stand for the number of binary and general variables, respectively, while *T/Gap* stands for either the solution time, if the problem is solved to optimality, or the optimality gap after 20 minutes of running time. Note that the solution time is provided in terms of seconds while the optimality gap is reported in terms of percentage; thus, any number in that column appearing without a "%" sign represents solution time, whereas numbers appearing with a "%" sign represent optimality gap. The optimality gap is calculated using the following formula:

$$\frac{UB - Obj.Val.}{UB}$$

where UB denotes the smallest upper bound on the objective function value obtained from the Branch-and-bound tree search.

Instance	Number of Processing Units	Number of Actual Tasks
P3-2	3	2
P4-3-1	4	3
P4-3-2	4	3
P5-3	5	3
P5-4	5	4
P6-10	6	10

Table 2.3: Instances Information

The computational results show that, if solved to optimality, the multitasking model outperforms the singletasking model in every instance in terms of the objective function value. This result is in accordance with the expectations since the multitasking model is a generalization of the singletasking model; therefore, it can produce equal or better optimal objective function under any circumstances.

To better illustrate the gains of the multitasking versus the singletasking model, the optimal schedule of the actual tasks obtained for instance P3-2 through both singletasking and multitasking formulations with $N = 5$ is depicted through Gantt charts in Figure 2.2 and Figure 2.3. Table 2.5 represents the information related to the processing units and their corresponding machines (the processing times are provided in terms of minutes), while Table 2.6 shows the tasks information. As it can be observed from the figures, the multitasking feature allows machines 1 and 2 to process samples of different tasks simultaneously, which results in a better optimal solution for the multitasking formulation compared to the singletasking formulation.

The singletasking model is essentially solving a much smaller problem as opposed to the multitasking model, and, therefore, it usually requires a smaller number of time points to reach its global optimal solution. However, that would not be a disadvantage for the multitasking model since with the same instance and the same number of time points, if solved to optimality, the multitasking model can achieve an objective value that is always at least as good as the objective value from the singletasking model.

		Multitasking				Singletasking			
Instance	N	Obj. Val.	Bin. Var.	Gen. Var.	T/Gap	Obj.Val.	Bin. Var.	Gen. Var.	T/Gap
P3-2	4	3200	86	62	.08	800	103	30	.11
P3-2	5	3200	126	90	.15	2000	139	54	.13
P3-2	6	3200	166	118	.31	2000	175	82	.27
P3-2	7	3200	206	146	.44	2000	211	110	.35
P4-3-1	7	3900	293	231	.23	3400	318	211	.24
P4-3-1	8	4500	349	275	.91	4100	369	254	.81
P4-3-1	9	4500	405	318	4.7	4110	420	297	2.69
P4-3-1	10	4500	461	363	91.37	4400	471	340	52.69
P4-3-1	11	4500	517	406	178	4400	522	383	90.91
P4-3-1	12	4500	571	476	458	4400	568	426	309.91
P4-3-2	19	4300	1588	1169	3.02%	3360	1506	1123	25.26%
P4-3-2	20	4390	1680	1237	6.59%	3090	1590	1190	40.97%
P4-3-2	21	4340	1772	1304	13.51%	3065	1674	1257	46.19%
P4-3-2	22	4479	1864	1372	15.69%	3215	1578	1324	41.81%
P4-3-2	23	4435	1956	1440	22.29%	2900	1842	1391	60.90%
P4-3-2	24	4539	2046	1511	25.09%	2750	1938	1467	72.32%
P4-3-2	25	4393	2140	1580	34.37%	3120	2022	1534	54.23%
P4-3-2	26	4579	2232	1647	34.29%	2847	2106	1601	70.00%
P4-3-2	27	4264	2324	1715	49.55%	2935	2190	1668	66.77%
P4-3-2	28	4308	2416	1783	53.78%	2750	2274	1735	78.78%
P5-3	6	2450	285	226	48.75	2000	377	203	6.51
P5-3	7	2600	364	288	147.95	2100	449	264	34.76
P5-3	8	2850	443	350	759.72	2150	521	325	489
P5-3	9	3100	522	412	765.40	2150	593	386	7.45%
P5-3	10	3100	601	474	22.56%	2150	665	447	27.71%
P5-3	11	3100	680	536	35.48%	2150	737	508	44.52%
P5-4	9	4990	646	559	493.28	4160	768	525	64.34
P5-4	10	5150	746	644	7.70%	4160	861	610	525
P5-4	11	5144	846	730	7.88%	4160	954	695	1.72%
P5-4	12	5345	946	816	3.84%	4160	1047	780	20.37%
P5-4	13	5050	1046	902	9.90%	4160	1140	865	25.58%
P5-4	14	5345	1146	988	3.84%	3980	1233	950	27.86%
P6-10	10	4063	2002	1998	21.83%	1500	2420	1910	22.95%
P6-10	11	4099	2276	2272	20.76%	1500	2684	2183	39.87%
P6-10	12	3677	2550	2546	34.62%	1300	2948	2456	69.46%
P6-10	13	3364	2824	2820	47.15%	1500	3212	2729	51.11%

Table 2.4: Computational Results

Processing Unit	J_p	β_p	$\tau(p)$	TM_j
1	1	400	110	0.0
2	2	540	120	0.0
3	3,4	260	150	0.0,0.0

Table 2.5: P3-2: Processing Units Information

Task	S_i	a_i	TA_i
0	1,2,3	unlimited	0.0
1	1,2,3	200	0.0
2	1,2,3	200	0.0

Table 2.6: P3-2: Tasks Information

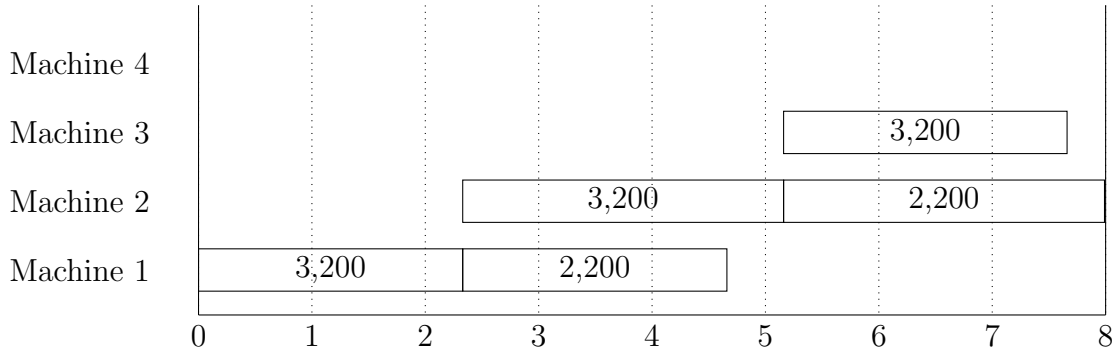

Figure 2.2: Instance P3-2 with singletasking, Objective Value =1900, N=5

One possible drawback of the multitasking model would be the computational cost. If the solution time for the multitasking model is considerably larger than its singletasking counterpart, then, within a fixed amount of solution time, the multitasking model might not reach optimality and therefore the singletasking model might be able to produce better quality solutions. Therefore, it is essential to monitor the solution time for both models as well as the quality of the objective function. The results presented in Table 2.4 suggest that, for the same instance with the same number of time points, there is not a considerable

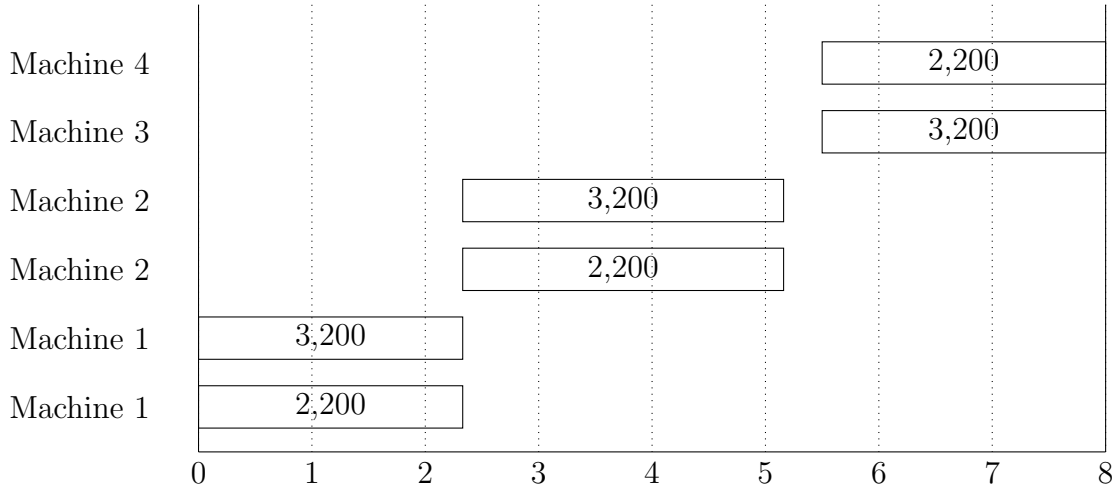

Figure 2.3: Instance P3-2 with multitasking, Objective Value =2100, N=5

difference in the solution time of the two models. It is worth noticing that some of the instances studied, e.g., P5-4, P4-3-2 and P6-10, are as large as some of the large instances solved in the continuous time formulation studies reported in the literature [38, 22].

2.4 Chapter Summary

In this chapter, a continuous time formulation was presented that can be readily used to model the operations of an analytical services facility. The formulation was capable of accommodating the multitasking feature of the machines in the facility and it could also take care of other operational conditions present in such facilities. Later on, a thorough comparison is performed between the performance of the proposed model and one of the most widely used formulations in the literature that is not capable of accommodating the multitasking feature, and it is shown that the multitasking formulation consistently provides significantly better solutions compared to the singletasking formulation.

In the next chapter, we present a flexible discrete time formulation, that could also be readily applied to model the operations of an analytical services facility.

Chapter 3

Flexible Discrete Time Formulation

This chapter presents a flexible discrete time formulation for the problem that was defined in section 1.3. Developing such formulations is very important since discrete time formulations that are capable of accommodating flexible time discretization are recently developed and it is yet to be seen if there is any particular advantage in using such formulations. The problem introduced in section 1.3 provides a good venue to test the performance of flexible discrete time formulations against other conventional approaches.

A flexible discrete formulation, while is a discrete time approach, by nature, has the capability to assign a specific clock to each processing unit. This means that, while the locations of the time points are determined *a priori* and given to the optimization model, these locations need not be the same for different processing units. For instance, the third time point can happen at hour 1 for the machines in the first processing unit, and can happen at hour 3 for the machines in the second processing unit. It is worth noticing that the conventional discrete time formulations in the literature, where the time points happen at the same time for all the machines in the facility, are a specific case of flexible discrete time formulations, where all the individual clocks are synchronized.

Furthermore, the chapter will provide the Gantt chart of the schedule obtained through the flexible discrete time formulation for the illustrative example depicted in section 2.2.

3.1 Formulation

The flexible discrete time formulation presented in this study has been adapted from the idea of non-uniform time discretization, previously proposed by [40]. To derive such a

formulation, extra notation and assumptions are required. The time domain for each individual processing unit will be discretized into a predetermined series of time points. Each $p \in P$ will have a time step $\Delta(p)$, which represents the time elapsed between two consecutive time points for processing unit p. Let $\mathcal{E}(p) = (0, \Delta(p), 2\Delta(p), \ldots, H)$ represent the increasing sequence of time points of processing unit p along the axis of time. Note that, $\forall\ p \in P$, $\mathcal{E}(p)$ is known *a priori*; also the machines in the processing unit can only be turned on at the beginning of a time point of the processing unit. In this formulation it is assumed that the time steps are fixed throughout the scheduling horizon for each processing unit; however, this condition can be relaxed to consider variable time steps. We choose not to present it here since it overly complicates the notation and it is not used in any of the tests. Since the machines in a processing unit are assumed to be identical, in the flexible discrete time formulation it is not needed to consider the machines J_p of processing unit p on an individual basis, rather they are considered as resources of the processing units. Accordingly, $R_{pt}, \forall\ t \in \mathcal{E}(p)$, represents the number of machines in processing unit p, for all $p \in P$, available to the facility at time point t of the processing unit. R_{pt} is defined with a time index since some of the machines in processing unit p might not be available at the beginning of the scheduling horizon because they are busy processing samples from the previous scheduling horizon, and they will only become available to the facility at time TM_j. Furthermore, the processing time of the machines in a processing unit can be referred to as the processing time of the processing unit. Similarly for the capacities.

The first set of decision variables for the flexible discrete time formulation are as follows. B_{ikt}: A nonnegative integer variable representing the number of samples from task i that are set to start being processed at the k-th processing unit in the path of task i, $p_k^i \in S_i$, at the time point t, $\forall\ i \in I,\ 1 \leq k \leq n(i),\ t \in \mathcal{E}(p_k^i)$.

X_{pt}: The number of machines from processing unit p that are being used at time point t, $\forall\ p \in P,\ t \in \mathcal{E}(p)$.

Constraints (3.1)-(3.2) ensure that the machines in the facility are not overloaded with samples. Constraint (3.1) enforces the capacity of the machines while Constraint (3.2) ensures that the number of machines in use in each processing unit and at each time point are not more than the total number of machines available to the processing unit at that time point, therefore the time subscript, θ, is chosen such that all the machines in each processing unit that are in use at each time point are accounted for.

$$\sum_{i,k:p=p_k^i} B_{ikt} \leq X_{pt}\beta_p; \quad \forall\ p \in P,\ t \in \mathcal{E}(p). \tag{3.1}$$

$$\sum_{\theta\in\mathcal{E}(p):t-\tau(p)<\theta\leq t} X_{p\theta} \leq R_{pt;} \quad \forall\ p \in P,\ t \in \mathcal{E}(p). \tag{3.2}$$

As mentioned above, a sample from task i can visit processing unit p_k^i in S_i only if it has already visited the previous processing unit, p_{k-1}^i, in S_i. Also, a sample from task i is considered to have visited processing unit p_k^i in S_i, if it has been processed by one of the machines in J_p. To account for these conditions, the following notation is considered:

W_{ikt}: A nonnegative integer variable representing the number of samples from task i that have visited p_{k-1}^i and are ready to visit processing unit p_k^i at time point t , $\forall\ i \in I \cup \{0\},\ k = 2, \ldots, n(i),\ t \in \mathcal{E}(p_k^i)$.

Constraint (3.3) ensures that a set of samples from task i can visit processing unit p_k^i in S_i at time point t, if it has already visited processing unit p_{k-1}^i in S_i before time point t. Furthermore, it assures that the same number of samples from task i that enter a processing unit, leave the processing unit after completing their processing, i.e., constraint (3.3) is a flow conservation constraint, preventing samples from being created or lost. Because of its structure, constraint (3.3) cannot be extended to the first processing unit in the path of a task and the first time point of the processing unit; hence, constraint (3.4) is introduced to extend constraint (3.3) to the first process in the path of the tasks while constraints (3.5)-(3.6) set up the number of samples from task i that are ready to visit p_1^i, in accordance to the number of samples in task i and the time these samples will become available to the facility, TA_i. Notice that in constraint (3.5), t' is defined to be the smallest $t \in \mathcal{E}(p_1^i)$ such that $t' \geq TA_i$.

$$B_{ikt} + W_{ikt} = W_{i,k,t-\Delta(p_{k-1}^i)} + \sum_{\substack{\theta\in\mathcal{E}(p_{k-1}^i):\\ t-\Delta(p_{k-1}^i)<\theta+\tau(p_{k-1}^i)\leq t}} B_{i(k-1)\theta}; \tag{3.3}$$

$$\forall\ i \in I,\ k = 2, \ldots, n(i),\ t \in \mathcal{E}(p_k^i) : t > 0;$$

$$B_{i1t} + W_{i1t} = W_{i1(t-\Delta(p_1^i))}; \quad \forall\ i \in I,\ t \in \mathcal{E}(p_k^i) : t > 0; \tag{3.4}$$

$$W_{i1t'} = a_i; \quad \forall\ i \in I,\ t' = \Delta(p_1^i)\left\lceil \frac{TA_i}{\Delta(p_1^i)} \right\rceil; \tag{3.5}$$

$$W_{i1t} = 0; \quad \forall\ i\ \in I,\ \forall\ t \in \mathcal{E}(p_1^i) : t < TA_i. \tag{3.6}$$

Since minimizing the turnaround time of the operations is part of the objective function,

it is necessary to monitor the latest time that a sample from a task, $i \in I$, finishes its processing at a unit, $p \in S_i$. To that avail, the following notation is introduced.

$$Y_{ipt} = \begin{cases} 1, & \text{If samples from task } i \text{ are assigned to be processed at a machine in processing unit } p \text{ at time point } t \text{ of the processing unit; } \forall\ p \in P,\ i \in I,\ t \in \mathcal{E}(p). \\ 0, & \text{Otherwise.} \end{cases}$$

Constraints (3.7)-(3.8) correlate the variables Y_{ipl} and B_{ikl}. Constraint (3.7) ensures that Y_{ipl} will take a value of 0 if $X_{ikl} = 0,\ p = p^i_k$ while constraint (3.8) assures that Y_{ipl} will take a value of 1 if $X_{ikl} \neq 0,\ p = p^i_k$.

$$Y_{ip^i_k t} \leq B_{ikt};\quad \forall\ i \in I, k = 1, \ldots, n(i),\ t \in \mathcal{E}(p^i_k). \tag{3.7}$$

$$B_{ikt} \leq R_{p^i_k} \beta_{p^i_k} Y_{ip^i_k t};\quad \forall\ i \in I,\ k = 1, \ldots, n(i),\ t \in \mathcal{E}(p^i_k). \tag{3.8}$$

Constraint (3.9) specifies the value for the latest time that a sample from task i finishes visiting a processing unit.

$$Tf_{ik} \geq t + \tau(p^i_k) - H(1 - Y_{ip^i_k t});\quad \forall\ i \in I,\ k = 1, \ldots, n(i),\ t \in \mathcal{E}(p^i_k). \tag{3.9}$$

For the present discrete formulation, the throughput of task i for all the machines in processing unit $p \in S_i$, $j \in J_p$, can be defined as the follows:

$$\sum_{t \in \mathcal{E}(p^i_k)} B_{ikt}\quad \forall\ k = 1, \ldots, n(i).$$

Similarly, the total number of samples from task i that have finished their processing in the facility can be calculated as follows:

$$\sum_{i \in I} \sum_{\substack{t \in \mathcal{E}(p^i_{n(i)}): \\ t + \tau(p^i_{n(i)}) \leq H}} B_{in(i)t}.$$

Hence, the objective function in the present discrete time formulation can be expressed as

follows:

$$\max \sum_{i \in I} \sum_{k=1}^{n(i)} \sum_{t \in \mathcal{E}(p_k^i)} c_{ij} B_{ikt} + \sum_{i \in I} \sum_{\substack{t \in \mathcal{E}(p_{n(i)}^i): \\ t+\tau(p_{n(i)}^i) \leq H}} f_i B_{in(i)t} + \sum_{i \in I} \sum_{k=1}^{n(i)} d_{ik} T f_{ik}. \tag{3.10}$$

The formulation aims at maximizing (3.10) subject to constraints (3.1)-(3.9)

3.2 Illustrative Instance

In this section we will solve the illustrative instance explained in Section 2.2, using the flexible discrete time formulation. The weights of the objective function are the same as the ones used in Section 2.2 and the experiment is performed on the same machine. Hence, a direct comparison could be driven from the schedules obtained through both of the formulations. For the purpose of illustration, we will use two different time discretization approaches for solving the instance through the flexible discrete time formulation. In the first approach, at the beginning of each hour, an event can happen. This essentially means that all the processing units will have the same time step. Since the scheduling horizon is 8 hours, there will be 9 time points, where the location of the first time point is hour 0 and the last time point is located at hour 8 while the distance of each two consecutive time points on a universal clock is 1 hour. Figure 3.1 depicts the Gantt chart of the schedule obtained for the illustrative instance using this approach. The solution time was .29 seconds. As for the previous charts, only the actual tasks are depicted, while the first number in each box is the task number and the second number indicates the total number of samples from that task that are being performed in the corresponding machine.

The second approach of time discretization is going to assign a specific time step to the machines of each processing unit, i.e., assign a specific clock to each processing unit. On the clock of each processing unit, the first time point happens at hour 0 while the time step of the processing unit is equal to its processing time. This means that, for example, for the second processing unit, which has a processing time of 30 minutes, there will be 17 time points, where the first time point happens at hour 0, while the last time point happens at hour 8, and the distance between two consecutive time points is 30 minutes. Figure 3.2 depicts the Gantt chart of the schedule obtained for the illustrative instance using this approach. The solution time was .27 seconds.

The results indicate that the continuous time formulation could generate a better result than the flexible discrete time formulation with either of the discretization approaches. The

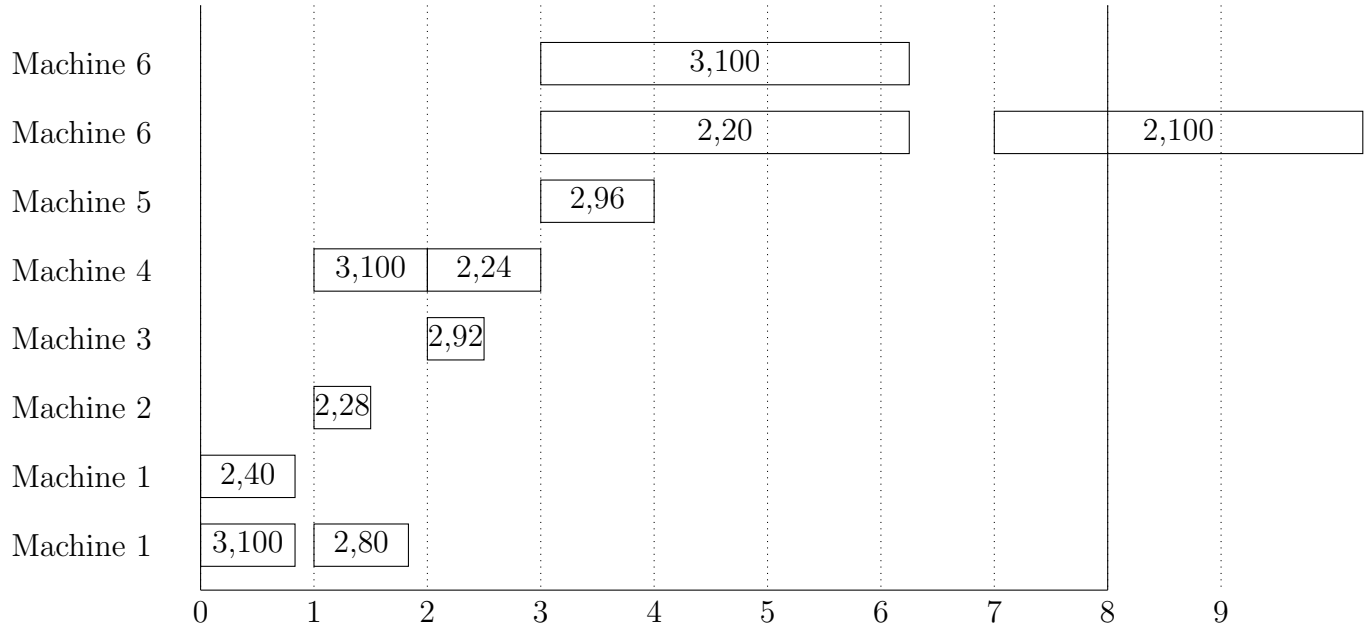

Figure 3.1: Illustrative Instance with first approach, Objective Value =1889

second discretization approach, where each processing unit had its own specific time step, was able to generate a slightly better solution, which is mostly due to the fact that it can put the second processing unit to a better use. Remember that the second processing unit has a processing time of 30 minutes, while the first approach can make a decision every hour, which means there would be a delay in the operations of the machines in the second processing unit. But the second approach assigns a specific time step to the second processing unit, and every 30 minutes a decision can be made for the machines in that processing unit, which puts them to a better use.

This illustrative instance provides a great venue to ask the main question of this thesis once again. Which one of these formulations is better for the problem we discussed in Section 1.3? A continuous time formulation or a flexible discrete time formulation? Furthermore, for the flexible discrete formulation, which one of the two discretization approaches is better? To have a single time step for all the machines in all the processing units or have a specific time step for the machines of each processing unit? The next chapter aims at answering this question through studying several instances that are inspired by the operations of an actual analytical services facility.

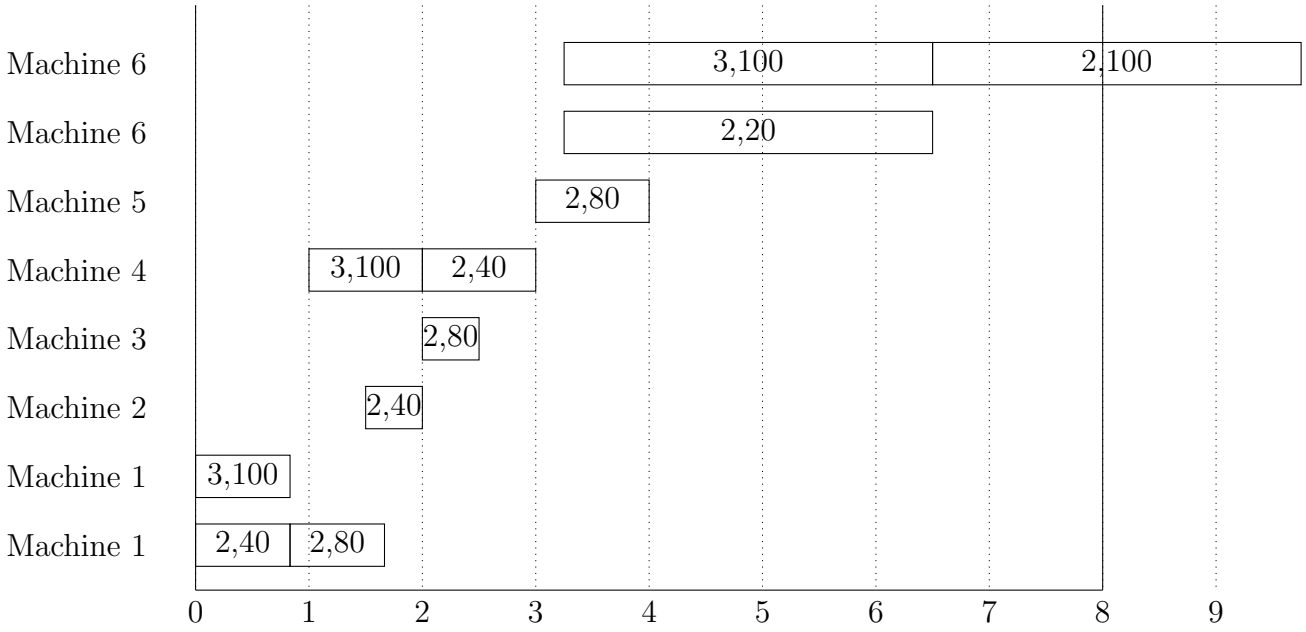

Figure 3.2: Illustrative Instance with first approach, Objective Value =1890.31

3.3 Chapter Summary

In this chapter a flexible discrete time formulation was presented that could be readily used to model the problem introduced in Section 1.3. Furthermore, the schedule that could be obtained through the presented formulation using different time discretization approaches was depicted. In the next section, the performance of the continuous and flexible discrete time formulations is fully compared and we will provide meaningful insights into the advantages and disadvantages of each of these formulations as well as, explaining to some extent, the reasons behind these advantages and disadvantages.

Chapter 4

Comparison of the Formulations

In this chapter, the performance of the flexible discrete and the continuous time formulations will be empirically evaluated. To do so, several instances have been designed and solved using the formulations described before. For each formulation, the most suitable parameter settings, among a set of predefined settings, have been specified for each instance. Thus, the best performance of each instance while using the different formulations will be obtained and used for comparison purposes.

The instances selected for this study were inspired by the operations of an actual analytical services facility. In the facility, completing the processing of samples is more important than merely maximizing the task throughput of the facility. Therefore, for every task $i \in I$, the weight of finishing task i, f_i, needs to overcome the weights of the throughputs of the tasks for any machine. Hence, $\forall\ i \in I$, f_i is arbitrarily chosen to be equal to 10 while for all the machines in every processing unit, p, and every task, $i \in I_j$, the weight of the throughput of task i for machine $j \in J_p$ is chosen to be 1. Since minimizing the turnaround time of the facility is of the least priority, the weight of the latest finishing time of a sample from task $i \in I$ at processing unit $p^i_{n(i)}$, $d_{in(i)}$, is arbitrarily chosen to be equal to $-.05$ while if $k \neq n(i)$, then d_{ik} is chosen to be $-.025$ for all $i \in I$. These weights have been used for every instance studied in this chapter. Note that these weights are the same as the weights used in Section 2.2, but they are different from the weights used in the comparative study for multitasking in Section 2.3, because the objective function that the formulations aim to optimize in the current chapter is different from the objective function used in Section 2.3.

The present facility includes processes whose processing times range from minutes to hours. An example of a process with short processing time is weighing samples, which is done within a minute, but the capacities of the weighing machines are also limited.

Therefore, any task whose samples require to be weighed will have a processing unit along its path that has a small processing time and a relatively small capacity. On the other hand, there are processing units with processing times larger than an hour, e.g, a fire assay process requires the samples to be heated to more than 1200°Celsius, and be kept in that temperature for around 80 minutes. To reflect this characteristic of the operations and its implications, half of the instances considered in this chapter included tasks that require processing units with both large and small processing times, referred from heretofore as *type I* instances, while the other half of the instances only have large processing times and are referred from heretofore as *type II* instances. Furthermore, the scheduling horizon for all the studied instances was set to 8 hours.

To fully evaluate the performance of the two formulations, the instances include small, medium range and large instances in terms of number of binary variables. Specifically, some of the larger instances are comparable in size with relatively large instances studied in the literature for continuous time formulations [38, 22]. Table 4.1 includes the information for the size and characteristics of all the 12 instances considered in this analysis. The *Processing Time Variability* entry indicates whether the instance includes tasks that have in their path processing units with small and large processing times or not. The first number in the name of the instance entry indicates the number of processing units, the second number indicates the number of tasks considered in the instance, while the last number is an indicative of whether the instance belongs to *type I* or *type II* instances.

Instance	Number of Processing Units	Number of Actual Tasks	Processing Time Variability
3-2-I	3	2	Yes
3-2-II	3	2	No
4-3-I	4	3	Yes
4-3-II	4	3	No
5-4-I	5	4	Yes
5-4-II	5	4	No
6-6-I	6	6	Yes
6-6-II	6	6	No
7-8-I	7	8	Yes
7-8-II	7	8	No
8-9-I	8	9	Yes
8-9-II	8	9	No

Table 4.1: Instances Information

The conventional approach in the literature for discrete time formulations is to discretize

time uniformly, meaning that the distance between two time points, the time step, will be the same for all the processing units in the facility, i.e., to have a universal clock that all the machines abide by the time points located on that clock. On the other hand, the flexible discrete time formulation provides a platform to discretize time in a very flexible fashion. Therefore, it is possible to use a specific time step for each processing unit, i.e., a specific clock for each processing unit, which would essentially be a non-uniform time discretization. Note that the uniform discretization can be implemented using the flexible discrete time formulation, by setting $\Delta(p)$ to be the same for all $p \in P$.

In order to analyze if there is a benefit in using non-uniform discretization as opposed to a conventional uniform discretization, in this work the flexible discrete time formulation is used with both methods of discretization. From now on, the flexible discrete time formulation with a uniform time discretization is referred to as a uniform discrete time formulation and the flexible discrete time formulation with a non-uniform time discretization is referred to as the non-uniform discrete time formulation.

In the next subsection, the parameters of each formulation, in solving each instance, have been set and the results for these settings are reported. All the computational experiments were performed on a machine that has 250 GB RAM and 4 CPUs, each with 12 cores and a processing speed of 2.4 GHz using CPLEX [15].

4.1 Computational Results

The main parameter that needs to be set for both non-uniform discrete and uniform discrete time formulations is the discretization of time. Time discretization for the non-uniform discrete time formulation can be done in multiple ways [40]. Since it is impractical to test all the possible discretization schemes, it was decided to discretize time for the non-uniform discrete time formulation in a way that captures the processing time of each processing unit, since having processing units with both large and small processing times is one of the key characteristics of the facility under consideration. Therefore, the time step for each processing unit is chosen to be equal to the processing time of the processing unit.

A runtime limit of 20 minutes is imposed on solving each instance. This limit is prespecified for all the instances and all the formulations considered in this work. Table 4.2 presents the computational results for the non-uniform discrete time formulation. The Columns in Table 4.2 are as follows: *Ins.* represents the name of the instance, *OBJ* stands for the objective function value of the best feasible solution found within the time limit, whereas *B.V.* stands for the number of binary variables. *T/G* represents either the solution time, if the problem is solved to optimality, or the optimality gap after 20 minutes

of running time. Note that the solution time is reported in terms of seconds while the optimality gap is reported in terms of percentage.

Ins.	OBJ	B.V.	T/G
3-2-I	2548.8	993	16s
4-3-I	3553.1	1146	241s
5-4-I	5659.8	1899	1.2%
6-6-I	8662.1	2375	2.1%
7-8-I	10273.8	4052	4%
8-9-I	12825	6362	2.5%
3-2-II	2550.3	69	13s
4-3-II	3984.9	104	38s
5-4-II	3872.5	126	37s
6-6-II	6213.8	233	560s
7-8-II	7957.7	539	2.7%
8-9-II	6730.3	409	1.7%

Table 4.2: Computational Results for Non-Uniform Discrete Time Formulation

In the uniform discrete time formulation, the time steps of two different processing units would be the same, regardless of the processing time of their machines. For that purpose, time steps of 1, 10, 30 and 60 minutes have been employed to capture a wide range of small and large processing times in the processing units of the facility. Table 4.3 presents the computational results for the uniform discrete time formulation. Note that if the optimality gap reported at the end of the 20 minutes runtime was larger than 75%, the optimality gap has been reported as *N/R*.

	Uniform: 1m			Uniform: 10m			Uniform: 30m			Uniform: 60m		
Ins.	OBJ	B.V.	T/G	OBJ	B.V.	T/G	OBJ	B.V	T/G	OBJ	B.V.	T/G
3-2-I	2559.4	3219	1.5%	2546.4	315	74s	2258.7	99	5s	1039.2	41	1s
4-3-I	-53.5	4642	N/R	3091.9	466	1.5%	1107	151	78s	600.8	71	1s
5-4-I	131.4	8432	N/R	5490.5	829	2.3%	1982.7	264	40s	900.8	111	16s
6-6-I	-34.7	12532	N/R	7993.3	1240	11%	3838.8	398	1.2%	2262	174	255s
7-8-I	2.5	19583	N/R	6364.4	1919	6.1%	3821.2	571	.8%	1883.1	243	.1%
8-9-I	2.9	23861	N/R	9797.7	2310	11.7%	2911.8	706	1.1%	998.4	298	21.8s
3-2-II	2551.4	3094	1.82%	2560.2	306	246s	2548	96	46s	2534.5	41	33s
4-3-II	2632.5	5048	55.9%	4002.4	500	1.6%	3975.1	156	218s	3962.4	72	22s
5-4-II	-11.3	7578	N/R	4606	756	2.9%	3404.7	239	.3%	3125	110	32s
6-6-II	2	12438	N/R	7042.5	1242	4.7%	6210	386	.3%	5126	173	51s
7-8-II	1.3	18592	N/R	8245.4	1844	6.4%	8252.7	574	2.8%	4443.1	250	1142s
8-9-II	2.1	20834	N/R	8000.7	2048	12.6%	5171.6	603	4%	4178.5	284	1.2%

Table 4.3: Computational Results for Discrete Time Formulation

Because of its nature, and the fact that it does not involve a predetermined time for the decisions in the scheduling algorithm to be made, the continuous time formulation is more accurate than its discrete time counterparts. Hence, provided that there are enough time points considered in the analysis for any instance, the continuous time formulation is capable of providing a solution that is at least as good as the solutions from the non-uniform and uniform discrete time formulations. Therefore, the main parameter of the continuous time formulation that requires to be set for each instance is the number of time points. For that purpose, a small number of time points was initially chosen; this parameter was then systematically increased until a *break point* was reached. The break point is the smallest number of time points such that, after 20 minutes of runtime, the upper bound on the optimal objective function value derived from the LP relaxation of the instance, U.B., is at least as large as the best objective function value obtained through either the non-uniform discrete or uniform discrete time formulations. After reaching the break point, the number of time points has been increased to a point where increasing the number of time points by one or two does not improve the objective function, within the 20 minutes runtime. This approach of tuning the number of event points for comparing continuous time formulations performance against a discrete time formulation was previously proposed by [39].

Table 4.4 depicts the results obtained for the continuous time formulation. *Num. T. Points* represents the number of time points. Note that for each instance, the first number of time points presented is the break point and is marked with '*'.

Ins.	Num. T. Points	OBJ	UB	B.V.	T/G
3-2-I	20*	2560.15	2586.22	880	1.02%
3-2-I	21	2561.6	2591.3	930	1.16%
3-2-I	22	2561.1	2592.4	980	1.22%
3-2-I	23	2561.4	2600	1030	1.51%
4-3-I	48*	0	4099.2	6477	N/R
4-3-I	49	0	4100	7097	N/R
4-3-I	50	0	3815.3	5857	N/R
5-4-I	44 *	0	5694.2	5497	N/R
5-4-I	45	0	5795.5	5632	N/R
5-4-I	46	0	2	5800	N/R
6-6-I	39*	0	8712.8	8323	N/R
6-6-I	40	0	8866.2	8557	N/R
6-6-I	41	0	8982	8791	N/R
7-8-I	81*	0	10344.6	24538	N/R
7-8-I	82	0	10428.6	24855	N/R
7-8-I	83	0	10512.6	25172	N/R
8-9-I	45*	0	13016.9	18154	N/R
8-9-I	46	0	13332.8	18598	N/R
8-9-I	47	0	13619.9	19042	N/R
3-2-II	8*	2563.1	2563.1	344	14s
3-2-II	9	2564.1	2564.1	404	33s
3-2-II	10	2564.5	2564.5	464	845s
3-2-II	11	2564.5	2572.83	524	.32%
3-2-II	12	2564.5	2577.8	584	.52%
4-3-II	9*	3997.9	4035.7	604	.95%
4-3-II	10	4003.9	4070.3	705	1.66%
4-3-II	11	4003.9	4086.7	806	2.07%
4-3-II	12	4003.8	4089.5	907	2.14%
5-4-II	11*	4027	4697.8	1074	16.66%
5-4-II	12	3957.5	5258.3	1209	32.87%
5-4-II	13	4241	5759.3	1344	35.8%
5-4-II	14	4240	5780.8	1479	36.34%
5-4-II	14	4072	5800	1614	42.44%

Ins.	Num. T. Points	OBJ	UB	B.V.	T/G
6-6-II	11*	6805.5	8036.6	1602	18.09%
6-6-II	12	6756	8958.2	1811	32.6%
6-6-II	13	6594	8998.7	2020	36.47%
7-8-II	13*	7487.2	8887.1	2589	18.7%
7-8-II	14	7018.1	9690.2	2875	38.1%
7-8-II	15	68	10362.2	3161	N/R
8-9-II	11*	6295.4	8080.3	2785	28.35%
8-9-II	12	5733	9430.7	3185	64.5%
8-9-II	13	0	10715	3585	N/R%

Table 4.4: Continuous Time Formulation: Computational Results

The next subsection provides a comparison between the results obtained for the non-uniform discrete, uniform discrete and continuous time formulations, in terms of the quality of the objective function, number of binary variables and the optimality gap after the designated solution time of 20 minutes.

4.2 Results: Comparison and Discussion

Only the best performance of the formulations, in terms of the objective function value, has been compared against each other. For the uniform discrete formulation, in solving each instance, among the 4 different discretization steps, the one is chosen that had resulted in the highest objective function value, within the solution runtime. e.g., the time step for the uniform discrete formulation for instance *3-2-I* is set to be 1 minute while the time step for instance *4-3-I* is set to 10 minutes. For the continuous time formulation, in solving each instance, the smallest number of event points that resulted in the highest objective function value is chosen, e.g., for instance *3-2-II* the number of event points is set to be 10. It was noted that, the time spent to tune the continuous time formulation is significantly more expensive than the time spent to tune the discrete time formulations. However, the time that was spent on tuning each formulation to yield its best result have not been included in the comparison, since this highly depends on the method used for the tuning of each formulation, and currently there is no well studied method in the literature on tunning either of these formulations. We chose to do so in order to see if it would be worth while to pursue a faster way to tune the continuous time formulation.

Figure 4.1 depicts the best objective function value for each of the 12 instances studied in this analysis, obtained through each of the three aforementioned formulations. As the figure suggests, the non-uniform discrete time formulation usually performs better than the other two formulations when dealing with *type I* instances. When dealing with *type II* instances, the best method depends on the size of the instance. If the instance is small, such as *3-2-II* and *4-3-II*, provided that it does not require too many time points to reach global optimality, then the continuous time formulation is the method that performs the best. Notice that, although from the figure it seems that the objective functions for the three approaches are the same, the numbers are actually different and the continuous time formulation actually results in the best objective function value for both instances *3-2-II* and *4-3-II*. The values for the objective function of instance *3-2-II* through continuous, uniform and non-uniform discrete time approaches are, respectively, 2564.5, 2560.2, 2550.3, while 4003.9, 4002.4, 3984,9 are the objective function values for instance *4-3-II*. If the instance is larger and requires more time points, e.g., instance *8-9-II*, then it is the uniform discrete formulation that provides better solutions.

In the case of the uniform discrete time formulation, in the presence of processing units with a processing time smaller than the time step, e.g., a processing unit with 1 minute processing time, while the time step is 10 minutes, these processing units cannot be put to effective use through the scheduling algorithm, e.g., instance *4-3-I*. In the case of the example, the scheduling algorithm can only schedule operations every 10 minutes while the processing unit has a much smaller processing time, which severely hampers the effective usage of the machines in that processing unit. If the scheduling algorithm could make a decision every minute, for the processing unit with 1 minute processing time, then the machines in the processing unit will be used more effectively, thereby generating better solutions. Nonetheless, this approach requires a 1 minute time discretization for the uniform discrete time formulation, which results in a drastic increase in the computational cost. Since the non-uniform discrete formulation can use different time steps for different processing units, it can reduce the time step for a processing unit to capture accurately its processing time without severely increasing the computational costs. Furthermore, if a similar objective function value is required from the continuous time formulation, it will require many time points, which in return will increase the computational costs to a point where even finding a feasible solution will be computationally prohibitive.

On the other hand, when all the processing units have processing times in the same order of magnitude, e.g., hours, then the non-uniform discrete formulation, with time step equal to processing time for each processing unit, loses its key advantage gained through non-uniform time discretization. Therefore, with a relatively small time step, e.g, 10 minutes, when the processing times are in the order of hours, the uniform discrete formu-

lation is able to produce better solutions compared to its non-uniform discrete counterpart. Furthermore, such a situation usually requires less time points for the continuous time formulation and therefore, it is able to generate better solutions. e.g., instances *5-4-II, 7,8,II* where the continuous time formulation produced a better solution than the non-uniform discrete time formulation.

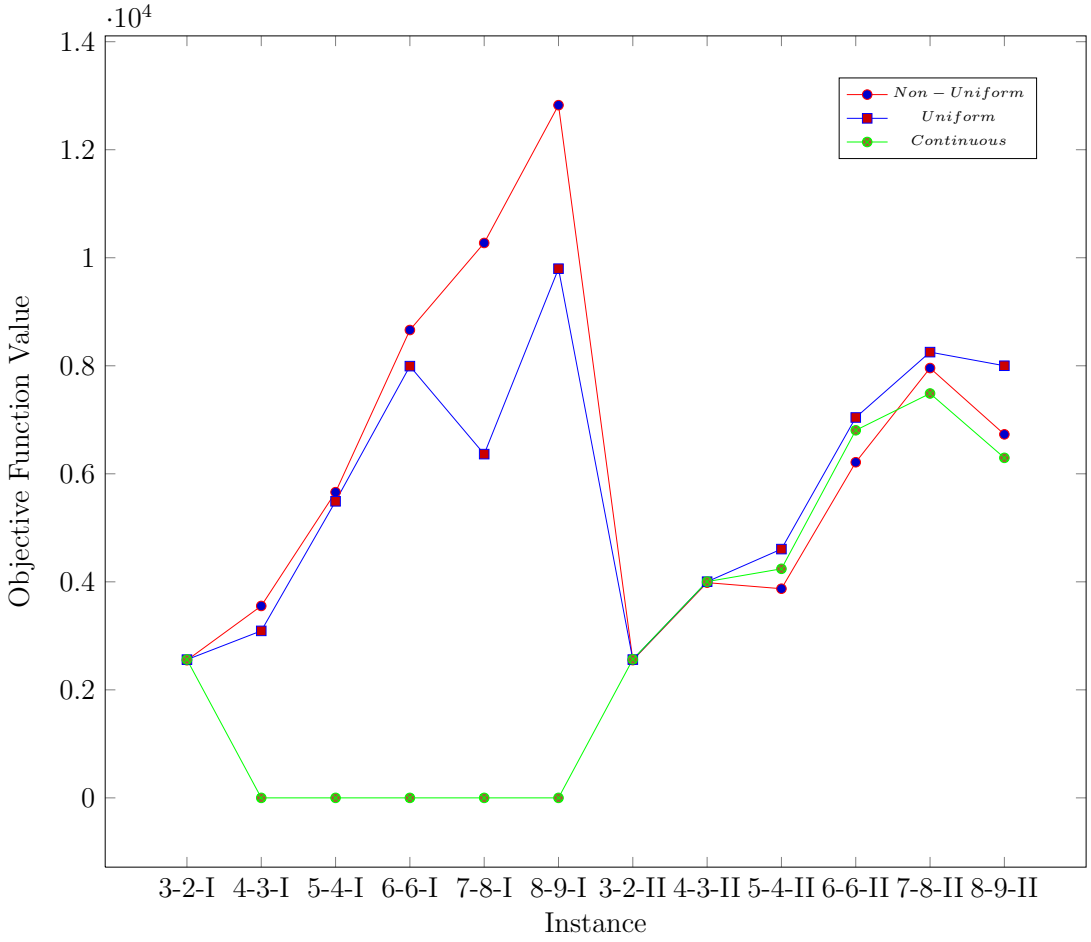

Figure 4.1: Objective Function Value Comparison

Figure 4.2 presents the number of binary variables that each formulation needed in solving each instance, with the settings specified at the beginning of the current subsection. Figure 4.2 suggests that the continuous time formulation requires the largest number of binary variables. This is not surprising since the model structure of the continuous

time formulation requires more binary variables than the other formulations. However, it is noticeable that when dealing with *type I* instances, the continuous time formulation requires a surprisingly high number of binary variables compared to the non-uniform discrete formulation. This observation may explain, to some extent, the reason as to why the continuous time formulation struggles with generating reasonably good feasible solutions for that type of instances.

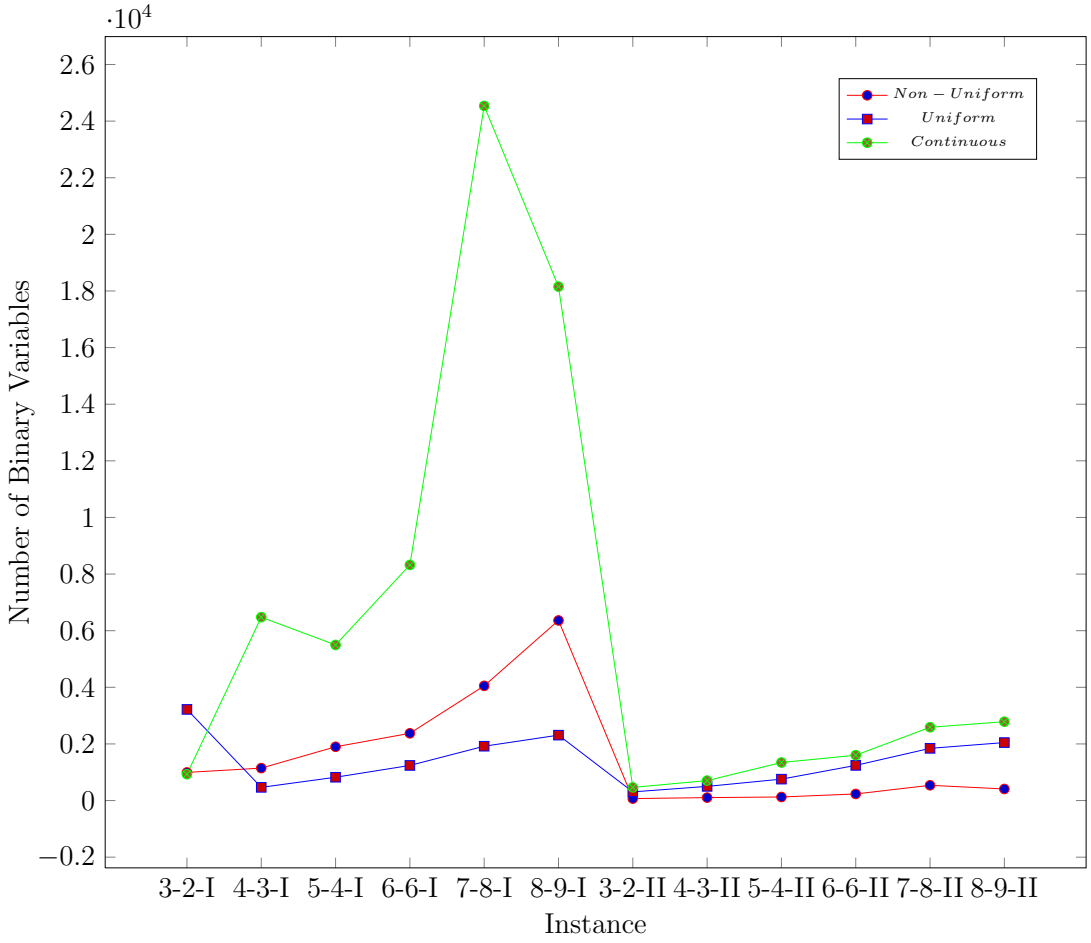

Figure 4.2: Number of binary Variables Comparison

The uniform discrete time formulation shows the least amount of fluctuation in the number of binary variables when dealing with the two types of instances. This is not surprising since a uniform time discretization treats different instances in a similar way, regardless of

the specific characteristics of each instance. This indifference to the characteristics of the instances hampers the effectiveness of the formulation in dealing with the instances that include processing units with small processing times, namely *type I* instances.

As shown in Figure 4.2, the number of binary variables used in the non-uniform discrete formulation suggests that in dealing with *type II* instances, it uses the least number of binary variables compared to the other two formulations. Moreover, Figure 4.1 shows that the performance of the non-uniform discrete formulation is inferior to the uniform discrete time formulation in dealing with *type II* instances. This further supports the need for more sophisticated time discretization schemes for the non-uniform discrete time formulation.

The number of binary variables used in a formulation not only affects the accuracy of the formulation, but it also affects the solution time. Figure 4.3 presents the optimality gap, reported at the end of the maximum allowed solution time, for specific settings specified for each formulation at the beginning of the subsection.

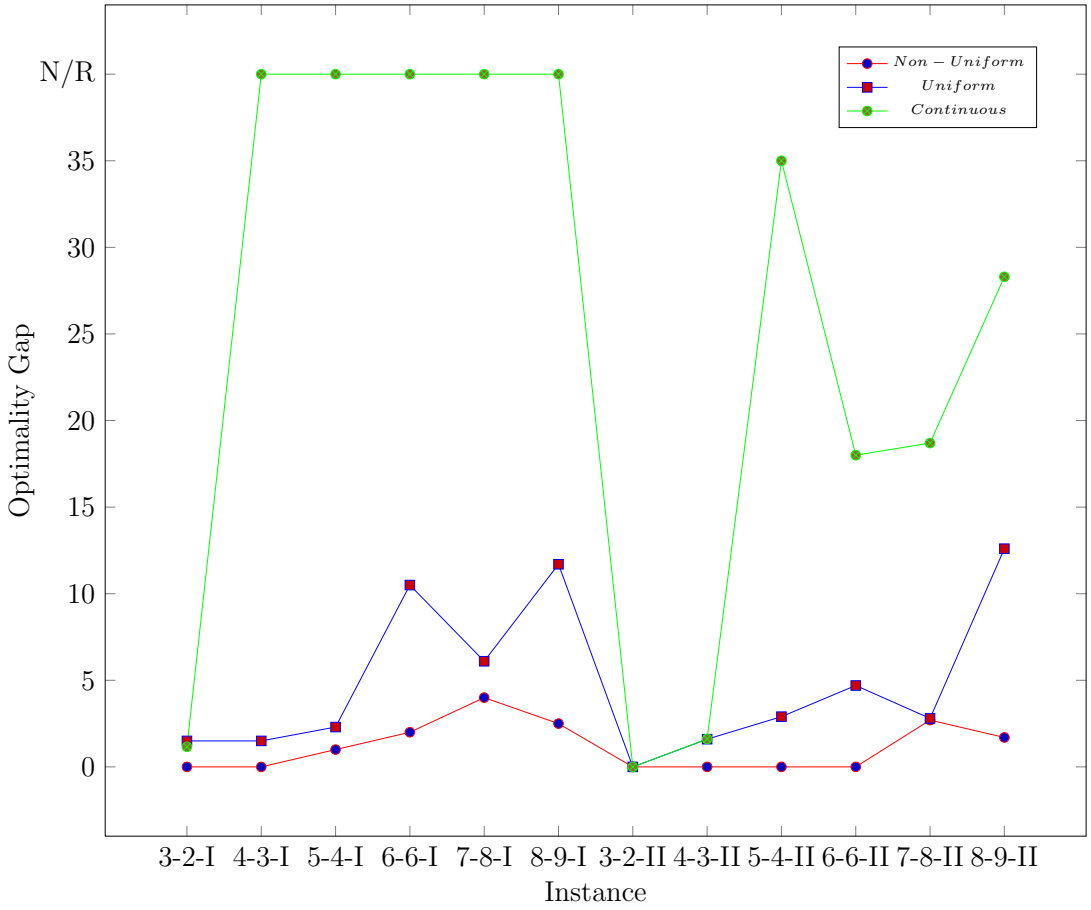

Figure 4.3: Optimality Gap Comparison

A few observations can be made through the results from Figure 4.3. First, when dealing with instances that require a large number of time points, the continuous time formulation results in very large number of binary variables and struggles with closing the optimality gap. This should not be attributed to the inefficiency of the particular continuous time formulation used in this analysis, since other continuous time formulations, when dealing with such large instances, have also struggled with closing the optimality gap [38, 22]. This struggle should not be interpreted as a direct relationship between the number of binary variables and optimality gap in general. To further support this point, one can look at the optimality gap for the non-uniform discrete formulation. Figure 4.2 shows that the non-uniform discrete time formulation has more binary variables than its uniform discrete counterpart, when dealing with *type I* instances. However, Figure 4.3 shows that the non-

uniform discrete formulation ends up with considerably smaller optimality gap, at the end of the 20 minutes, in all *type I* instances.

To further investigate these observations, Table 4.5 provides a closer look into the value of the LP relaxation at the root node of the branch-and-bound tree for each of the three formulations as well as the final branch-and-bound upper bound on the objective function value for each formulation after the time limit has been reached. The columns of the table are as follows: *RLP* represents the objective function value of the root LP relaxation, while *IMP* represents the improvement in the upper bound between the two terms and is calculated as follows:

$$\frac{RLP - UB}{UB}$$

	Continuous			Uniform Discrete			Non-uniform Discrete		
Ins.	RLP	UB	IMP	RLP	UB	IMP	RLP	UB	IMP
3-2-I	2600	2591.3	0.3 %	2600	2597.8	.08%	2600	2548.8	2%
4-3-I	4099.3	4099.2	0.002%	3239.3	3138.3	3.21%	3664.2	3553.1	3.12%
5-4-I	5694.3	5694.2	0.001%	5686.9	5616.8	1.24%	5800	5727.8	1.27%
6-6-I	8712.8	8712.8	0.0%	8975.1	8872.5	1.15%	8984.5	8766	2.49%
7-8-I	10344.6	10344.6	0.0%	6861.8	6752.6	1.62%	10749	10684.8	0.6%
8-9-I	13017	13016.9	0.001%	11120.1	10875.4	2.25%	13346.8	13145.6	1.67%
3-2-II	2600	2564.5	1.38%	2600	2560.2	1.54%	2600	2550.3	1.95%
4-3-II	4100	4070.3	0.73%	4100	4066.4	0.83%	4100	3984.9	2.9%
5-4-II	5764	5759.3	0.08%	5025	4739.6	6.02%	4057.4	3872.5	4.77%
6-6-II	8095.9	8036.6	0.73%	7433.3	7373.5	0.81%	6496.3	6213.8	4.55%
7-8-II	8933	8887.1	0.51%	8672.9	8483.8	2.23%	8351.7	8172.5	2.2%
8-9-II	8107.66	8080.3	.33%	9162.8	9008.8	1.7%	7160.5	6844.7	4.61%

Table 4.5: LP Relaxation Results

The results from the *IMP* column show that, for both types of instances, the continuous time formulation consistently performs inferior to its discrete time counterparts in improving the LP relaxation upper bound. This phenomenon to some extent explains the struggle of the continuous time formulation with closing the optimality gap. Furthermore, it can be noticed that the ability of the continuous time formulation to reduce the LP upper bound decreases when the size of the instance, and subsequently the number of time points needed for solving the instance, increase.

4.3 Chapter Summary

This chapter delivered the desired comparison between flexible discrete, with both uniform and non-uniform time discretizations, and continuous time formulation in the context of a proper case study. Several instances that closely resembled the operations of an actual analytical services were solved through both formulations. The best results obtained for each instance through each formulation were fully studied and compared against each other in terms of the quality of the objective function, number of binary variables and computational complexity. At the end, meaningful insights were given on the reasons behind the behavior of each of the formulations.

Chapter 5

Conclusion and Recommendations

This thesis aims at studying the effects of the manner of time representation in scheduling of operations. Performing such a study can be properly done in the context of a cohesive case study. Thus, scheduling of operations in an analytical services facility was chosen as the venue for performing the case study. This thesis focuses on both developing novel and competitive discrete and continuous time formulations to model the operations of an analytical services facility as well as comparing the performance of the proposed formulations. The findings and the merits of this study is discussed in Section 5.1 while the scope of future work and possible research areas is discussed in section 5.2.

5.1 Conclusions

Since none of the existing continuous time formulations in the literature could be readily applied to model the operations of an analytical services facility, there was a need to develop one for the study to be carried out. This thesis presents a novel continuous time formulation for scheduling of operations at an analytical services facility. The formulation is able to incorporate and use multitasking capability of machines in an analytical services facility. The result of incorporating this capability is a more efficient use of the machines, which increases the throughput of the facility. This means that the multitasking capability of the machines in the facility is an important feature that, if overlooked, would heavily compromise the quality of the solutions. A potential drawback of incorporating such features would be an increase in the complexity and solution time of the formulation. However, the computational results presented in section 2.3 suggest that the presented formulation does not have such a drawback since the computational cost of the model in comparison to

one of the most widely used singletasking models is negligible. The proposed formulation is able to solve problems of size comparable to other continuous time formulations, to near optimality within a solution time budget of 20 minutes, which is promising. In conjunction with multitasking in the machines, the presented formulation is capable of incorporating other technical issues that arise from the situation where processing some of the samples does not finish within the predetermined scheduling horizon.

At the next step, this thesis presents a comparison between a flexible discrete time formulation, with both uniform and non-uniform time discretization, and a continuous time formulation in the context of short-term scheduling of operations in an analytical services facility. Because of the lack of multitasking flexible discrete time formulations, a flexible discrete time formulation that was capable of modeling the operations of an analytical services facility was developed.

The flexible discrete time formulation with non-uniform time discretization, generates the best results for instances where there is high variability in the processing time of the processing units. However, this approach struggles when encountering situations where the processing times of the processing units are in the same order of magnitude. The continuous time formulation results in better solutions for smaller instances, specifically the smaller instances where the processing times are more or less in the same order of magnitude (essentially requiring small number of binary variables), while the non-uniform discrete formulation with a uniform discretization of time results in better solutions for larger instances of this type. Hence, the computational results suggest that the flexible discrete time formulation is better equipped to deal with larger instances and when the size of the instances increase, it can consistently generate better solutions compared to the continuous time formulation, which is in contrast to some of the previous results reported in the literature on comparing the performance of discrete (without flexible time discretization capability) and continuous time formulations [37]. This is further proof of the effectiveness of flexible discrete time formulations, calling for more investigations to be done on such formulations. However, one must notice that this is limited to the scope of the study performed in the current work and different studies on other problems might yield different conclusions.

5.2 Recommendations

In terms of future research, there are multiple directions to take. One such direction would be to explore alternatives that would improve the solution time for the continuous time formulation. Continuous time formulations are by nature more accurate than discrete

time formulations. The continuous time formulation's optimality gap for larger instances, juxtaposed with the formulation's better performance for smaller instances, suggests that if the solution time of the formulations could be improved, the formulation could produce better solutions compared to discrete formulations in general. Such improvements could be achieved either by development of formulations with a smaller number of binary variables or through improvement in the solution algorithms for the current formulation, by tightening the LP relaxations of the current formulations. The literature of the continuous time formulations appear to be heavily invested in developing better formulations, in terms of the number of binary variables, but it appears that exploring the improvement of the current solution algorithms might also have a large impact on the performance of these formulations.

To shed more light on this issue, consider the following continuous time formulation's constraint, Constraint (2.22), which regulates the length of the time slots and as a result, determines the location of the time points and is present, in more or less similar formats, in many other continuous time formulations.

$$TR_{ij(n+1)} \geq TR_{ijn} + \tau(p)Y_{ijn} - SL_{n+1}; \quad \forall\, p \in P,\ j \in J_p,\ i \in I_j,\ n = 0, \ldots, N.$$

This is the main constraint responsible for regulating each individual time slot and it is heavily dependent on the value of a binary variable, Y_{ijn}. When the binary variable has a fractional value close to zero in the LP relaxation, and has value 1 in the optimal MILP solution, the location of the time points marking the beginning and the end of the time slot is widely different in the LP relaxation compared to the optimal MILP solution. This seems to result in larger solution times and making the formulation struggle in closing the optimality gap. Furthermore, many of the constraints in continuous time formulations are Big-M type constraints, which are notoriously bad for solving integer programs efficiently and should be strengthened as much as possible.

Another research direction is to explore other time discretization schemes for the non-uniform discrete formulations. In the context of this study, two different time discretization schemes were explored, i.e uniform and non-uniform time discretization, with the processing time of each processing unit as the time step of that processing unit for the non-uniform discretization approach. It was shown that each one of the schemes works better for particular type of instances. The flexible discrete time formulation provides a platform for many different time discretization schemes to be used. Thus, it would be worth exploring sophisticated time discretization schemes that take the specific situation of each instance more into account and compare their performance against the schemes used in this work.

One interesting scheme for discretization of time would be an iterative algorithm that would start with a relatively inaccurate time discretization that results in a small MILP problem and then, based on the solutions obtained from solving this small MILP problem, it could add extra time points wherever it is deemed to result in better solutions. Devising such a scheme could prove very useful, specifically if one could devise the algorithm in such a way that refining the discretization would be performed through solving several small MILP problems as opposed to solving a very large one, similar to the idea of dynamic time discretization proposed by Boland et al [3]. Furthermore, Merchan et al [28] developed several tightening methods for flexible discrete time formulations, and it would be interesting to combine their methods with dynamic time discretization methods used by Boland et al [3].

Furthermore, another research avenue to explore is the implementation of the schedules obtained through flexible discrete and continuous time formulations in an actual analytical services facility and monitor the challenges that lay ahead in implementing them. This is specifically important in the case of the flexible discrete formulation where it is possible to have multiple time discretization schemes that each one would have its own specific benefits and challenges.

References

[1] S. C. Aggarwal. A focussed review of scheduling in services. *European Journal of Operational Research*, 9(2):114–121, 1982.

[2] M. H. Bassett, J. F. Pekny, and G. V. Reklaitis. Decomposition techniques for the solution of large-scale scheduling problems. *American Institute of Chemical Engineers Journal*, 42(12):3373–3387, 1996.

[3] N. Boland, M. Hewitt, M. Duc Vu, and M. Savelsbergh. Solving the traveling salesman problem with time windows using time-expanded networks, 2016. Talk presented at TRISTAN. http://tristan-symposium.org/wp-content/uploads/2016/06/TRISTAN_2016_Program_Book.pdf.

[4] E. H. Bowman. The schedule-sequencing problem. *Operations Research*, 7(5):621–624, 1959.

[5] P. Castro, A. P. F. D. Barbosa-Povoa, and H. Matos. An improved RTN continuous-time formulation for the short-term scheduling of multipurpose batch plants. *Industrial & Engineering Chemistry Research*, 40(9):2059–2068, 2001.

[6] J. Cerdá, G. P. Henning, and I. E. Grossmann. A mixed-integer linear programming model for short-term scheduling of single-stage multiproduct batch plants with parallel lines. *Industrial & Engineering Chemistry Research*, 36(5):1695–1707, 1997.

[7] M. Conforti, G. Cornuéjols, and G. Zambelli. *Integer Programming*. Berlin:Springer, 2014.

[8] I. T. Dedopoulos and N. Shah. Optimal short-term scheduling of maintenance and production for multipurpose plants. *Industrial & Engineering Chemistry Research*, 34(1):192–201, 1995.

[9] A. Elkamel, M. Zentner, F. Pekny, and G. V. Reklaitis. A decomposition heuristic for scheduling the general batch chemical plant. *Engineering Optimization Journal*, 28(4):299–330, 1997.

[10] C. A. Floudas and X. Lin. Continuous-time versus discrete-time approaches for scheduling of chemical processes: a review. *Computers & Chemical Engineering*, 28(11):2109–2129, 2004.

[11] S. Gupta and I. A. Karimi. An improved MILP formulation for scheduling multiproduct, multistage batch plants. *Industrial & Engineering Chemistry Research*, 42(11):2365–2380, 2003.

[12] C. Hui and A. Gupta. A bi-index continuous-time mixed-integer linear programming model for single-stage batch scheduling with parallel units. *Industrial & Engineering Chemistry Research*, 40(25):5960–5967, 2001.

[13] C. Hui, A. Gupta, and H. A. J. van der Meulen. 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, 2000.

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

[15] ILOG, Inc. ILOG CPLEX: High-performance software for mathematical programming and optimization, 2014. See http://www.ilog.com/products/cplex/.

[16] I. A. Karimi and C. M. McDonald. Planning and scheduling of parallel semicontinuous processes. 2. Short-term scheduling. *Industrial & Engineering Chemistry Research*, 36(7):2701–2714, 1997.

[17] E. Kondili, C.C. Pantelides, and R. W. H. Sargent. A general algorithm for short-term scheduling of batch operations-I. MILP formulation. *Computers and Chemical Engineering*, 17:211–227, 1993.

[18] S. Lagzi, R. Fukasawa, and L. A. Ricardez-Sandoval. A computational study of continuous and discrete time formulations for short-term scheduling of operations in multipurpose plants. Submitted to European Journal of Operational Research, 2016.

[19] S. Lagzi, R. Fukasawa, and L. A. Ricardez-Sandoval. A novel continuous time formulation for short-term scheduling of operations in multipurpose plants. Submitted to Computers & Chemical Engineering, 2016.

[20] K. Lee, S. Heo, H. Lee, and I. Lee. Scheduling of single-stage and continuous processes on parallel lines with intermediate due dates. *Industrial & Engineering Chemistry Research*, 41(1):58–66, 2002.

[21] K. Lee, H. Park, and I. Lee. A novel nonuniform discrete time formulation for short-term scheduling of batch and continuous processes. *Industrial & Engineering Chemistry Research*, 40(22):4902–4911, 2001.

[22] J. Li, X. Xiao, Q. Tang, and C. A. Floudas. 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*, 51(21):7300–7319, 2012.

[23] X. Lin and C. A. Floudas. Design, synthesis and scheduling of multipurpose batch plants via an effective continuous-time formulation. *Computers & Chemical Engineering*, 25(4):665–674, 2001.

[24] A. S. Manne. On the job-shop scheduling problem. *Operations Research*, 8(2):219–223, 1960.

[25] C. A. Méndez and J. Cerdá. An MILP-based approach to the short-term scheduling of make-and-pack continuous production plants. *OR Spectrum*, 24(4):403–429, 2002.

[26] C. A. Méndez, G. P. Henning, and J. Cerdá. Optimal scheduling of batch plants satisfying multiple product orders with different due-dates. *Computers & Chemical Engineering*, 24(9):2223–2245, 2000.

[27] C. A. Méndez, G. P. Henning, and J. Cerdá. An MILP continuous-time approach to short-term scheduling of resource-constrained multistage flowshop batch facilities. *Computers & Chemical Engineering*, 25(4):701–711, 2001.

[28] A. F. Merchan, H. Lee, and C. T. Maravelias. Discrete-time mixed-integer programming models and solution methods for production scheduling in multistage facilities. *Computers & Chemical Engineering*, 2016. http://dx.doi.org/10.1016/j.compchemeng.2016.04.034.

[29] L. Mockus and G. V. Reklaitis. Mathematical programming formulation for scheduling of batch operations based on nonuniform time discretization. *Computers & Chemical Engineering*, 21(10):1147–1156, 1997.

[30] S. Moon, S. Park, and W. K. Lee. New MILP models for scheduling of multiproduct batch plants under zero-wait policy. *Industrial & Engineering Chemistry Research*, 35(10):3458–3469, 1996.

[31] S. Mouret, I. E. Grossmann, and P. Pestiaux. A novel priority-slot based continuous-time formulation for crude-oil scheduling problems. *Industrial & Engineering Chemistry Research*, 48(18):8515–8528, 2009.

[32] S. Orcun, I. K. Altinel, and Ö. Hortaçsu. General continuous time models for production planning and scheduling of batch processing plants: mixed integer linear program formulations and computational issues. *Computers & Chemical Engineering*, 25(2):371–389, 2001.

[33] C. C. Pantelides. Unified frameworks for optimal process planning and scheduling. In *Proceedings on the second conference on foundations of computer aided operations*, pages 253–274. Cache Publications, New York, 1994.

[34] B. P. Patil, R. Fukasawa, and L. A. Ricardez-Sandoval. Scheduling of operations in a large-scale scientific services facility via multicommodity flow and an optimization-based algorithm. *Industrial & Engineering Chemistry Research*, 54(5):1628–1639, 2015.

[35] J. F. Pekny and M. G. Zentner. Learning to solve process scheduling problems: The role of rigorous knowledge acquisition frameworks. In *Proceedings of the Second International Conference on Foundations of Computer-Aided Process Operations, Crested Butte, Colorado*, pages 275–309, 1993.

[36] N. Shah, C. C. Pantelides, and R. W. H. Sargent. A general algorithm for short-term scheduling of batch operations-II. Computational issues. *Computers & Chemical Engineering*, 17(2):229–244, 1993.

[37] H. Stefansson, S. Sigmarsdottir, P. Jensson, and N. Shah. Discrete and continuous time representations and mathematical models for large production scheduling problems: A case study from the pharmaceutical industry. *European Journal of Operational Research*, 215(2):383–392, 2011.

[38] A. Sundaramoorthy and I. A. Karimi. A simpler better slot-based continuous-time formulation for short-term scheduling in multipurpose batch plants. *Chemical Engineering Science*, 60(10):2679–2702, 2005.

[39] A. Sundaramoorthy and C. T. Maravelias. Computational study of network-based mixed-integer programming approaches for chemical production scheduling. *Industrial & Engineering Chemistry Research*, 50(9):5023–5040, 2011.

[40] S. Velez and C. T. Maravelias. Multiple and nonuniform time grids in discrete-time MIP models for chemical production scheduling. *Computers & Chemical Engineering*, 53:70–85, 2013.

[41] S. Wang and M. Guignard. Redefining event variables for efficient modeling of continuous-time batch processing. *Annals of Operations Research*, 116:113–126, 2002.

[42] K. L. Yee and N. Shah. Improving the efficiency of discrete time scheduling formulation. *Computers & Chemical Engineering*, 22:403–410, 1998.

[43] M. G. Zentner, J. F. Pekny, G. V. Reklaitis, and J. N. D. Gupta. Practical considerations in using model-based optimization for the scheduling and planning of batch/semicontinuous processes. *Journal of Process Control*, 4(4):259–280, 1994.

[44] X. Zhang and R. W. H. Sargent. The optimal operation of mixed production facilities. part a. general formulation and some solution approaches for the solution. *Computers and Chemical Engineering*, 20:897–904, 1996.