A multitasking continuous time formulation for short-term scheduling of operations in multipurpose plants

Saman Lagzi, Ricardo Fukasawa

Combinatorics & Optimization Department, University of Waterloo, Waterloo, Canada

Luis Ricardez-Sandoval*

Chemical Engineering Department, University of Waterloo, Waterloo, Canada

Abstract

Short-term scheduling in multipurpose batch plants has received significant attention in the past two decades. Both discrete-time and continuous-time formulations have been proposed to model the problem; however, multipurpose plants that have machines with the ability to process multiple tasks at the same time, i.e. multitasking, have been overlooked by the available continuous-time formulations in the literature. This paper presents a novel MILP formulation that is capable of accommodating the multitasking feature in the machines of a facility. The performance of the presented formulation is studied in comparison with a single-tasking 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. The presented formulation takes into account several other operational constraints of multipurpose facilities and can be readily applied to facilities that have machines capable of multitasking, including plants in the analytical services sector.

Keywords: Scheduling, Continuous-Time Formulation, MILP, Analytical Services Sector, Multitasking, Multi-Purpose Batch Plants.

1. Introduction

The problem of scheduling operations can be generally described as a set of tasks that need to be performed on a set of machines, under capacity and demand constraints. In this work, we address scheduling of operations in multipurpose facilities with the following operational constraints: Each task, with a specific quantity of materials, needs to visit a specific *sequence* of processing units, where some processing units may appear more than once in the sequence of a task. Different tasks may have different processing sequences. Each processing unit consists of a set of machines in parallel that are not necessarily identical in terms of capacity and processing time; however, materials in the tasks need to be processed only at one of the machines in each processing unit before they can proceed to the next processing unit

*Corresponding author

Email addresses: slagzi@uwaterloo.ca (Saman Lagzi), rfukasawa@uwaterloo.ca (Ricardo Fukasawa), laricardezsandoval@uwaterloo.ca (Luis Ricardez-Sandoval)

in the sequence of the task. Machines are assumed to run continuously, i.e., without disruptions, and accordingly, materials processed in the machines cannot be removed from the machines while in operation. 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 materials from multiple tasks simultaneously, provided the materials in the tasks are available and the machine has enough capacity to process them. Therefore, flow conservation of materials in each task through the network is necessary, meaning that it is essential at each time point, to keep track of the amount of materials 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 amount of materials in a task that starts being processed in a machine, leaves the machine, after completing its processing. Another point to notice is the fact that the amount of materials from each task that will be processed at each processing unit, at each point in time, is determined from the optimization, making it a simultaneous batching and scheduling problem (Harjunkoski et al., 2014).

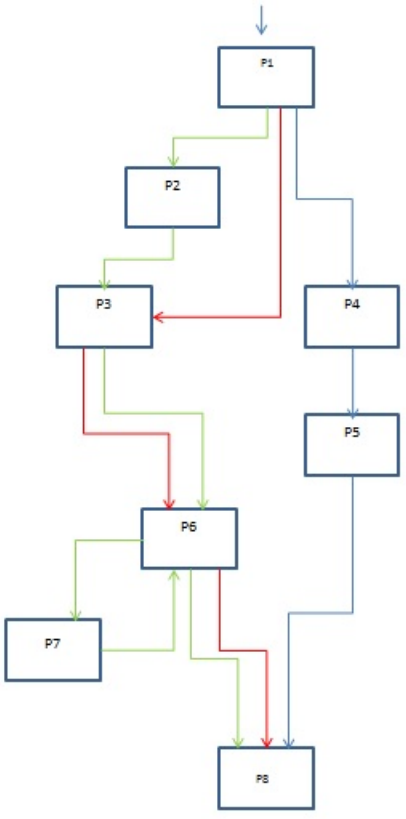

Figure 1: Illustration of a possible plant layout.

An additional operational condition is that some machines or materials may only become available sometime in the near future. In other words, there are release dates for both the machines and the tasks. This is because, in some cases, processing subsets of materials at a machine may not be completed within the scheduling horizon and therefore, their processing must be completed in a later scheduling horizon.

Figure 1 presents an illustration of the layout of a plant with characteristics that are similar to what was discussed above. The processing units are depicted through rectangles, while different possible sequences for different tasks are shown through arrows. As it is apparent, different tasks may have different sequences, while some processing units might appear more than once in the sequence of a task. Figure 2 shows a more detailed look at a portion of the facility in Figure 1 showing in addition two tasks: Task 1, with 130 units of material in it, presented in red, and Task 2, with 110 units of material in it, depicted in green. The machines in each processing unit are depicted through circles. The maximum capacity of the machines is shown inside the circles. The machines are assumed to have no minimum working capacity and all the tasks and the machines in the facility are assumed to be available from the beginning of the scheduling horizon. As it is evident from the figure, each task has its own specific sequence, where the sequence of task 1 is $(P1, P3, P6, P8)$ and the sequence of task 2 is $(P1, P2, P3, P6, P7, P6, P8)$. Note that $P6$ appears twice in the sequence of task 2. The machines can process multiple tasks at the same time, provided that their capacity is not violated. To illustrate a possible sequence of operations, consider machine M1 in processing unit P1. The machine has a capacity of 140, therefore, it can start processing 120 units of material from task 1 and 20 from task 2. After such processing is done and the corresponding materials have proceeded to the next processing unit in their sequence, still 10 units of materials from task 1 and 90 units of material in task 2 will be left, which can be loaded into machine M1 when it starts operating for the second time.

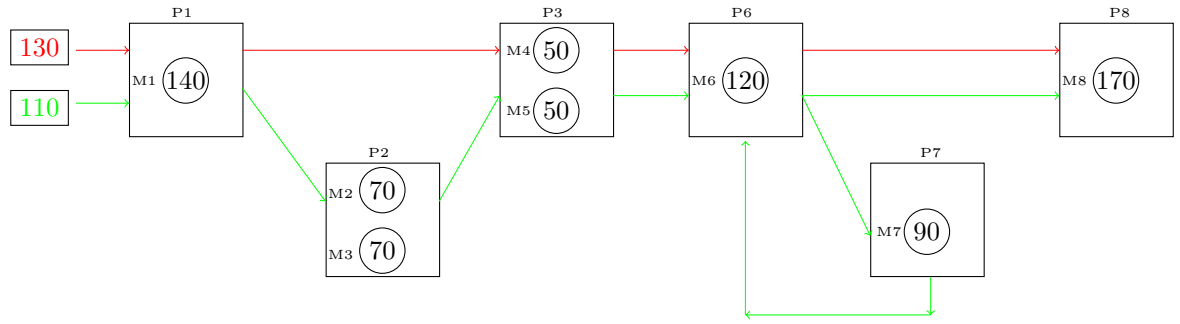

Figure 2: Operations Figure

The key motivation to consider this type of scheduling problem is that it incorporates several operational details present at analytical services facilities. 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 important. Furthermore, scheduling of operations for such facilities has not been widely studied. Indeed, although the scheduling of operations for such facilities could

be categorized as a simultaneous batching and scheduling problem, the existing works on simultaneous batching and scheduling problems (Sundaramoorthy and Maravelias, 2008) have not considered facilities that have machines with multitasking capabilities. To the best of the authors' knowledge, Patil et al. (2015) is the only work reported that has presented an optimal scheduling framework for this type of facilities.

In that study, time is represented as a series of discrete events, i.e, the scheduling horizon is partitioned into several time intervals with predetermined lengths. Hence, the length of the time intervals is determined prior to solving the scheduling algorithm; accordingly, the quality of the solution is highly dependent on the predetermined length of the time intervals, which may lead to large-sized formulations depending on the accuracy desired (Sundaramoorthy and Maravelias, 2011).

To overcome this difficulty, continuous time formulations have been proposed to address several scheduling problems. Furthermore, continuous time formulations are capable of modeling some operational features, e.g., variable machine processing times, 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. 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. Nonetheless, since several characteristics of the present problem are still shared by these continuous time formulations, we chose to adapt and extend their application to multitasking facilities. The aim of this work is to devise a continuous time mixed-integer programming formulation for scheduling of operations subject to the aforementioned operational constraints that is capable of maximizing the throughput of the network.

Since several continuous time formulations have been proposed to address different operational conditions, an analysis of which would be the best type of formulation to extend was required. Such analysis is the focus of the next subsection.

1.1. Review of continuous-time formulations

One special type of multipurpose plants for which continuous-time formulations were developed is called *sequential processes* plants. In these plants, a specific processing unit (referred as a stage) can only be used once in the sequence of a task and all tasks follow the same sequence of stages (with some possible skips). Furthermore, 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, (Moon et al., 1996; Cerdá et al., 1997; Méndez et al., 2000; Hui et al., 2000; Méndez et al., 2001; Hui and Gupta, 2001; Orcun et al., 2001; Lee et al., 2002) 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 sequences, prior to performing any optimization, which is suboptimal and very time consuming. In addition, as illustrated in Figures 1 and 2 the assumptions of sequential stages is not applicable, making such formulations not suitable for the problem of interest.

Another type of facilities for which continuous time formulations have been applied are *network represented* multipurpose facilities. According to Floudas and Lin (2004), such continuous time formulations can be classified into two sub-classes, namely *Unit-specific event based formulations* and *Global event based formulations*.

Unit-specific event based formulations (Ierapetritou and Floudas, 1998; Lin and Floudas, 2001; Mouret et al., 2009), partition the planning horizon into a sequence of time instances called event points. The location of each event point along the time axis is not pre-specified *a priori*. Each event point specifies the beginning of a task or utilization of a unit (alternatively a machine). The event points are defined on a unit basis which allows tasks assigned to the same event point at different units to occur at different times. Although Unit-specific event based formulations have a relatively flexible nature, according to Sundaramoorthy and Karimi (2005), flow conservation of materials can not 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 of when materials processed in the previous processing unit would be available for the next processing unit. For example, event 2 for machine 1 can happen at hour 1 and it can happen at hour 2 for machine 2. So if there exists a task whose sequence is machine 1 followed by machine 2, it becomes 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. Therefore such attempts are omitted from this article.

Global event based formulations (Sargent, 1996; Karimi and McDonald, 1997; Lee et al., 2001; Castro et al., 2001; Wang and Guignard, 2002; Gupta and Karimi, 2003; Sundaramoorthy and Karimi, 2005; Li et al., 2012) 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. Although flow conservation of materials is relatively easier to achieve in these approaches, the extension 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*. Nonetheless, this type of formulations were the most suitable to be adapted for this work. In particular, this work extends the Global event based continuous time formulation for multipurpose plants previously proposed by Sundaramoorthy and Karimi (2005), adding features that allow multitasking in the facility and, thus, enable it to produce better schedules in terms of resource utilization.

It is worth noticing that multitasking is similar to variable recipe tasks, where the amount of different materials for each task that would be processed in each machine at each time point is decided by the optimization algorithm. However, the current work can be easily distinguished from the existing works in the literature on variable recipe tasks (Blömer and Günther, 1998; Castro et al., 2009). The work by Blömer and Günther (1998) is a discrete time formulation whereas, the current work is a continuous time formulation. On the other hand, although Castro et al. (2009) proposed a Global event-based formulation for a variable recipe task problem, their formulation had not considered the case where each task needs to go through a specific sequence of processing units before leaving the facility.

As is evidenced by the above discussion, there is no continuous time formulation in the literature that is appropriate for the operational conditions required in this work, particularly for analytical services facilities. The formulation proposed in this work can be readily used to model such a problem, thus eliminating such gap in the literature.

The rest of the paper is organized as follows: In Section 2, we present a detailed description of the problem along with a novel continuous time formulation. Computational results are presented in Section 3; the conclusion and possible future developments are discussed in Section 4.

2. Continuous Time Formulation

A multipurpose facility receives a set of tasks I that needs to be processed within a pre-specified scheduling horizon, H, using a set P of processing units. Each processing unit consists of a set of machines that perform its corresponding process. The set of machines in processing unit p is denoted as J_p. The set of all machines in the facility is denoted by J, where J is the disjoint union of the J_p's.

Task $i \in I$ consists of an amount a_i of materials that need to visit a specific sequence of processing units. The sequence or recipe for task i is denoted by S_i, where S_i represents 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 materials in task i must visit every processing unit in S_i in the pre-specified sequence, that is, p^i_k in S_i can only be visited if the previous processing unit, p^i_{k-1}, in S_i has already been visited. A material is considered to have visited a processing unit, p, if it has been processed by one of the machines in J_p. Depending on their nature, materials can be processed in a machine either in fractional amounts (e.g., fractions of a kilo or a liter of a certain substance) or in integer amounts (e.g., pallets, boxes or samples). It is essential to mention that all the materials in a task have the same identity and unit size and will not change through being processed in the machines along their sequence. This means that the identity of the task would stay the same throughout its sequence.

Each machine $j \in J$ has a capacity, denoted as β^{max}_j, and an associated processing time, denoted as τ_j. This means that machine j can be loaded with at most β^{max}_j amount of materials from potentially different tasks. Once a machine has been turned on to process the materials, it will run without interruption for a time τ_j. Note that the processing time of machine j is fixed and does not depend on the amount of materials from different tasks being processed by the machine. After this time, the machine

is considered to be available; also, the materials are considered to have visited the corresponding processing unit and they are ready to visit the next processing unit in their sequence. In addition, machine j is assumed to have a minimum working capacity, denoted as β_j^{min}, which is the minimum quantity of materials from one or more tasks that must be loaded into the machine before this machine can be turned on; hence, β_j^{min} may have a value of 0 if there is no minimum working capacity for machine j.

Materials in task $i \in I$ will be available to start visiting the first processing unit in their sequence, p_1^i, at time TA_i; machine $j \in J$ will become available to start processing materials 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 materials in task i were part of another task in the previous scheduling horizon and did not visit all the processing units in their sequence. In such cases, S_i is the sequence of the remaining processing units that have not been visited. If machine j is processing materials 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*. The objective is to maximize the total weighted task throughputs of the machines in the facility. The task throughput of machine j for task i, is the total amount of materials from task i processed at machine j, throughout the scheduling horizon, where c_{ij} is the weight of the task throughput of machine j for task i.

To model the above problem, a continuous-time formulation developed from the global event based formulation is presented. The proposed formulation extends the Global event based continuous time formulation proposed by Sundaramoorthy and Karimi (2005) to be able to account for the multitasking feature of the machines in the facility.

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-1} and T_n, for all $n = 1, \ldots, N$.

Constraints (1)-(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).

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

$$T_n - T_{n-1} = SL_n; \quad \forall\ n = 1, \ldots, N, \tag{2}$$

$$T_0 = 0. \tag{3}$$

A time point specifies an alternative to start or finish processing subsets of materials from a set of tasks at a machine. At each time point, multiple machines from different processing units could either start or finish processing materials. That is, if a subset of materials 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:

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

Accordingly, constraint (4) is introduced to ensure that a subset of materials in a task will be assigned to a machine for processing, only after the task is accepted at the facility. Furthermore, it will ensure that a subset of materials 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 $p \in S_i$.

$$T_n \geq \max\{TA_i, TM_j\} Y_{ijn}; \quad \forall\ j \in J,\ i \in I_j,\ n = 0, \ldots, N. \tag{4}$$

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 Sundaramoorthy and Karimi (2005), an idle task is introduced. The idle task is a task that does not have a specific processing sequence. If a machine is not processing materials from a real task, then it is processing materials from the idle task. Furthermore, the idle task does not have a processing time, i.e., a machine can finish processing materials in the idle task at any time and begin processing subsets of materials from a set of real tasks. In addition, a machine can potentially stay idle throughout the scheduling horizon. Moreover, the materials 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 materials from other tasks. Furthermore, materials from the idle

task cannot be processed in a machine together with materials from the real tasks at any time. In the present formulation, the idle task is denoted as task number 0, whereas the set I of real tasks is assumed to be composed of tasks $1, \ldots, |I|$. Accordingly, the following variables are introduced in relation to the concept of an idle task:

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

Since a time point specifies an alternative to start processing materials 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:

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

Constraint (5) ensures that, if materials 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 (6) ensures that machine j will not start processing at time point n if no subset of materials are scheduled to start being processed in machine j at that time point.

$$Z_{jn} \geq Y_{ijn}; \ \ \forall\, j \in J, \ i \in I_j \cup \{0\}, \ n = 0, \ldots, N. \tag{5}$$

$$Z_{jn} \leq \sum_{i \in I_j \cup \{0\}} Y_{ijn}; \ \ \forall\, j \in J, \ n = 0, \ldots, N. \tag{6}$$

After a machine starts processing materials 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 materials that continue to be processed at each machine, as well as the materials that complete their processing. To deal with these conditions the following binary variables are introduced:

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

$$YR_{ijn} = \begin{cases} 1, & \text{If a subset of materials from task } i \text{ continues to be processed at machine } j \text{ at time point } n;\ \forall\ j \in J,\ i \in I \cup \{0\},\ n = 0, \ldots, N. \\ 0, & \text{Otherwise.} \end{cases}$$

Constraint (7) assures that, if a subset of materials in task i started or continued to be processed at machine j at time point n, then, at the next time point the materials will either complete their processing or continue being processed in machine j.

$$YR_{ijn} = YR_{ij(n-1)} + Y_{ij(n-1)} - YE_{ijn};\ \ \forall\ j \in J,\ i \in I_j \cup \{0\},\ n = 0, \ldots, N. \tag{7}$$

Constraint (8) considers that, if a subset of materials in task i completes its processing at machine j at time point n, then machine j will immediately start processing a new set of materials, either from the same tasks or from a new set of tasks (including the idle task).

$$Z_{jn} \geq YE_{ijn};\ \ \forall\ j \in J,\ i \in I_j \cup \{0\},\ n = 0 \ldots, N. \tag{8}$$

Constraints (9)-(10) ensure that machine j starts processing a new subset of materials from a set of tasks at time point n, if and only if it is not continuing processing materials from any set of tasks at that time point.

$$Z_{jn} \leq 1 - YR_{ijn};\ \ \forall\ j \in J,\ i \in I_j \cup \{0\},\ n = 0, \ldots, N. \tag{9}$$

$$(1 - \sum_{i \in I_j \cup \{0\}} YR_{ijn}) \leq Z_{jn};\ \ \forall\ j \in J,\ n = 0, \ldots, N. \tag{10}$$

The materials can not be removed from a machine while the machine is running. Therefore, it is required to ensure that a subset of materials 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 subset of materials at that time point. This is achieved through constraint (8) along with constraint (9). Also, since a machine cannot both continue processing a subset of materials and start processing a new subset of materials at the same time point, constraint (5) and constraint (9) ensure that if a subset of materials from a set of tasks is scheduled to continue being processed at machine j at time point n, then machine j cannot start processing a new subset of materials at that time point.

Constraint (11) ensures that a machine cannot process materials from a real task while the machine is processing the idle task.

$$Y_{ijn} \leq 1 - Y_{0jn}; \quad \forall\ j \in J,\ i \in I_j, n = 0, \ldots, N. \tag{11}$$

Capacity constraints are required to determine how much of the materials from each task will be processed in each machine at each time point. Accordingly, the following variables are introduced:

B_{ijn}: Is a nonnegative variable, representing the amount of materials from task i that begins processing at machine j at time point n; $\forall\ j \in J,\ i \in I_j \cup \{0\},\ n = 0, \ldots, N$.

BE_{ijn}: Is a nonnegative variable, representing the amount of materials from task i that completes its processing at machine j at time point n; $\forall\ j \in J,\ i \in I_j \cup \{0\},\ n = 0, \ldots, N$.

BR_{ijn}: Is a nonnegative variable, representing the amount of materials from task i that continues its processing at machine j at time point n; $\forall\ j \in J,\ i \in I_j \cup \{0\},\ n = 0, \ldots, N$.

Note that in the case where the materials in the tasks are of an integral nature (e.g., pallets, boxes or samples), these variables need to be defined to take integral values only.

Constraint (12) considers that, if machine j is not assigned to process materials from task i at time point n, then the subset of materials in task i assigned to machine j to start being processed at that time point, is empty. Constraint (13) is placed to ensure that the total amount of all the subsets of materials from different tasks, scheduled to start being processed at machine j at time point n, does not exceed the capacity of the machine. Since it is possible for some machine, $j \in J$, to have a non zero β_j^{min}, in order to avoid infeasible solutions (materials being scheduled while the machine is scheduled to continue processing other materials), constraint (13) is designed to be enforced only when the machine is scheduled to start processing materials.

$$\beta_j^{min} Y_{ijn} \leq B_{ijn} \leq \beta_j^{max} Y_{ijn}; \quad \forall\ j \in J,\ i \in I_j \cup \{0\},\ n = 0, \ldots, N. \tag{12}$$

$$\beta_j^{min} Z_{jn} \leq \sum_{i \in I_j \cup \{0\}} B_{ijn} \leq \beta_j^{max} Z_{jn}; \quad \forall\ j \in J,\ n = 0, \ldots, N. \tag{13}$$

To extend the capacity constraints to the situation where processing materials from task i is set to be continued or completed at machine j at time point n, constraints (14)-(17) are introduced.

$$\beta_j^{min} YR_{ijn} \leq BR_{ijn} \leq \beta_j^{max} YR_{ijn}; \quad \forall\ j \in J,\ i \in I_j \cup \{0\},\ n = 0, \ldots, N. \tag{14}$$

$$\beta_j^{min}(1 - Z_{jn}) \leq \sum_{i \in I_j \cup \{0\}} BR_{ijn} \leq \beta_j^{max}(1 - Z_{jn}); \quad \forall\ j \in J,\ n = 0, \ldots, N. \tag{15}$$

$$\beta_j^{min} YE_{ijn} \leq BE_{ijn} \leq \beta_j^{max} YE_{ijn}; \quad \forall\ j \in J,\ i \in I_j \cup \{0\},\ n = 0, \ldots, N. \tag{16}$$

$$\beta_j^{min} Z_{jn} \leq \sum_{i \in I_j \cup \{0\}} BE_{ijn} \leq \beta_j^{max} Z_{jn}; \quad \forall\ j \in J,\ n = 0, \ldots, N. \tag{17}$$

Constraint (18) is considered to ensure that the total amount of materials in task i that is processed

in the facility, throughout the scheduling horizon, does not exceed the amount of materials in task i, i.e. a_i.

$$\sum_{1\leq n\leq N}\sum_{j\in J_p:\ p=p_1^i} B_{ijn} \leq a_i;\quad \forall\ i\in I. \tag{18}$$

Most facilities require conservation of materials for feasible operation of the plant. Therefore, constraint (19) is considered to ensure that the amount of materials from task i that starts being processed or continues to be processed at machine j at time point n, is equal to the amount of materials from task i that completes or continues processing at machine j at the next time point.

$$BR_{ijn} + BE_{ijn} = BR_{ij(n-1)} + B_{ij(n-1)};\quad \forall\ j\in J,\ i\in I_j,\ n=0,\ldots,N. \tag{19}$$

The next constraint is introduced to ensure that a subset of materials 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 subset of materials 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 this condition, the following notation is introduced (If materials are of an integral nature, the variable should be defined such that it can only take integer values):

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

Constraints (20) ensures that a subset of materials 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.

$$\sum_{j\in J_p:\ p=p_k^i} B_{ijn} + W_{ikn} = W_{ik(n-1)} + \sum_{j\in J_p:\ p=p_{k-1}^i} BE_{ijn};\quad \forall\ i\in I,\quad k=2,\ldots,n(i),\ n=1,\ldots,N. \tag{20}$$

Constraints (19) and (20) are referred to as flow conservation constraints, since they can be interpreted as corresponding materials to a flow that either must be put in a machine, remain in a machine or be waiting i.e. no materials are lost or created. Also, recall that, as previously mentioned, the identity of the task stays the same throughout its sequence, i.e. all the materials in a task have the same identity and unit size and will not change through being processed in the machines along their sequence.

The next step in the formulation is to ensure that, if a subset of materials starts to be processed at a machine, it takes an amount of time equal to the processing time of the machine, before the materials complete their processing at that machine.

TR_{ijn}: is a nonnegative continuous variable, representing the amount of time remaining to complete processing materials 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 (21) ensures that, the amount of time remaining to complete processing materials 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 materials at time point n. Constraint (22) is considered to relate the remaining time to complete processing materials 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 this subset of materials at machine j prior to time point n.

$$TR_{ijn} \leq \tau_j YR_{ijn}; \quad \forall\ j \in J,\ i \in I_j \cup \{0\},\ n = 0, \ldots, N. \tag{21}$$

$$TR_{ij(n+1)} \geq TR_{ijn} + \tau_j Y_{ijn} - SL_{n+1}; \quad \forall\ j \in J,\ i \in I_j,\ n = 0, \ldots, N. \tag{22}$$

Constraint (22) is not considered for the idle task, since a machine can stop being idle at any time if materials from a real task are assigned to start being processed at the machine at that time.

The task throughput of machine j for task i, is defined as follows:

$$\sum_{n=0}^{N} BE_{ijn} \quad \forall\ j \in J,\ i \in I_j.$$

Hence, the objective function is:

$$\max \sum_{n=0}^{N} \sum_{j \in J} \sum_{i \in I_j} c_{ij} BE_{ijn}. \tag{23}$$

The presented formulation aims at maximizing (23), subject to constraints (1)-(22). The weights of the task throughputs, c_{ij}, enable the objective function to be flexible and useful for many industrial purposes. For instance, in many situations, tasks can have different ranks or levels of significance for various reasons such as having a closer due date, more economic gain, etc. Furthermore, in some cases, due to the nature of the operation, materials of a particular task are preferred to be processed in particular machines. These aspects of operations can be reflected through the weight of the task throughput. It is worth mentioning that with slight modifications, the presented formulation is capable of accommodating other objective functions such as minimizing the turnaround time of the facility, maximizing the profit or minimizing the operational cost.

This section is ended by commenting on which parts of the model were already present in the original formulation by Sundaramoorthy and Karimi (2005) (on which this model is based on) and which parts needed to be added or modified (and why) to account for multitasking and other operational conditions. Constraint (4) was added to the original formulation to account for the release dates of tasks and machines. The original formulation had no counterpart for this constraint since it had not considered such a situation. Constraints (5)-(6) 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 (8)-(10) 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 (11) 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.

Some of the capacity constraints, namely constraints (13), (15) and (17), 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 (12), (14) and (16), 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 (18) 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 (21)-(22) 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 single tasking 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. The rest of the constraints, constraints (1)-(3), (7), and (19)-(20) were already present in the original formulation.

3. Case Study and Computational Results

The performance of the proposed model has been evaluated using instances inspired by the operations of an analytical services facility. The goal is to evaluate the performance of the model through instances that share many features with real instances from an actual facility.

In the facility considered in this work, the materials in the tasks have an integral nature (e.g, pallets, boxes, samples). Also, for the materials in a task, visiting the last processing unit in the sequence of the task has priority over visiting the rest of the processing units in the sequence. To better illustrate this, assume that the materials 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 materials in task 1 can finish their processing in the facility by visiting processing unit 6, while the materials in task 2 need to visit other processing units after visiting processing unit 6. Then, processing the materials in task 1 at the machines of processing unit 6, $j \in J_6$, has priority over processing the materials in task 2. Moreover, most of the processing units have at most two machines and all the machines have a minimum working capacity of 0.

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. The scheduling horizon for all the studied instances is chosen to be 8 hours. The rest of the section is organized as follows. First, an instance with full details is presented along with the schedule produced through the continuous time model to better illustrate the results that can be obtained from the proposed model. Afterwards, the performance of the model on 8 different instances is evaluated and is compared against a version of the model that prohibits multitasking (i.e. each machine can process at most one task at a time). 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 ILOG, Inc (2014), with a specified relative optimality tolerance of 0.1%.

3.1. 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. Since both the computational time and the quality of the solution depend on the number of time points, the problem was solved for multiple number of time points, and the number of time points that resulted in the largest objective function value is reported.

Table 1 shows the information related to the processing units and their corresponding machines.

Processing Unit	J_p	β_j^{max}	β_j^{min}	τ_j	TM_j
1	1	140	0	50	0.0
2	2,3	70,70	0,0	30,30	0.0
3	4,5	50,50	0,0	60,60	0.0
4	6	120	0	185	0.0

Table 1: Processing Units Information

Likewise, Table 2, lists the information related to the tasks that need to be performed in the facility within an 8 hours scheduling horizon.

The number of time points is 6 and the objective function value is 1360; the solution time is 0.1 seconds

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

Table 2: Tasks Information

and does not include the time spent on tuning the number of time points. Figure 3, 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 amount of materials from the task being processed in the corresponding machine.

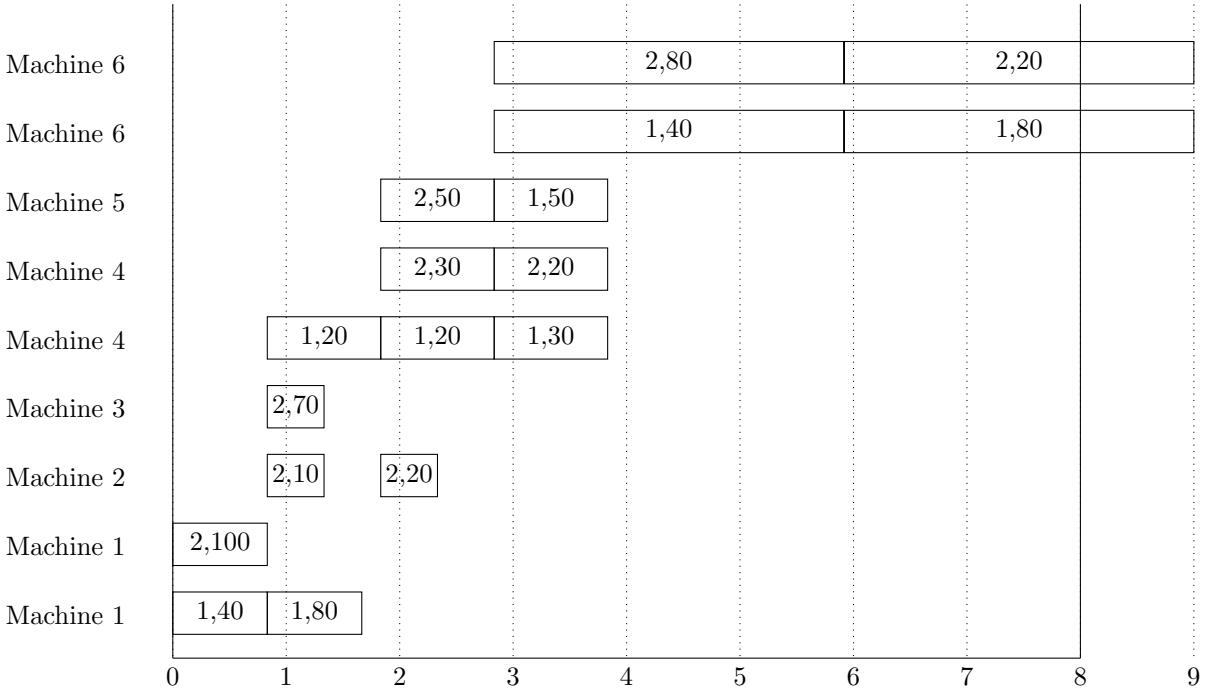

Figure 3: Illustrative Example, Objective Value =1360, N=5

The Gantt chart shows that the model behaves as expected, by scheduling as many materials as possible on the machines and enforcing multitasking in the machines at points were it is beneficial to process materials from both of the actual tasks on a machine.

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 materials will only be available sometime in the near future. To illustrate this, note that some of the materials 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 materials in task 1 need to visit another processing unit, e.g., processing unit 4, after visiting processing unit 3. The staff of the facility are allowed to work overtime to finish processing the materials in task 1. To accommodate processing the materials in task 1, another scheduling horizon is required. To circumvent this condition, the rolling-horizon technique can be employed where the 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 materials in task 1 in the next schedule, or they need to wait until the end of hour 9, which is the time that part of the materials 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 materials 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, and accommodate the fact that the materials 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.

3.2. 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 materials from multiple tasks at the same time. Overlooking this feature may result in suboptimal solutions. To cast more light onto this issue, 8 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 (the instances are available upon request). 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 amount of materials in each task, the number of machines in each processing unit and the machines specifications such as processing times and capacities are different). 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. This is a heuristic used in Castro et al. (2001), 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. Note that the number of time points are increased in both formulations until none of the two formulations show any improvement in terms of the quality of the objective function. For instance, problem *P-4-3-1* does not show any improvement for the multitasking formulation while N ranges between 8 to 12 but since the single tasking formulation has shown improvement for this range, the number of time points has been increased in both formulations, for the purpose of consistency.

Table 3 presents the data related to the size and complexity of each instance; Table 4 presents the computational results. The columns in Table 4 represent the following: *O.V.* stands for the objective function value of the best feasible solution found within the time limit, *B.Var.* and *Gen.Var.* stand for the number of binary and general variables, respectively, while *T/G.* stands for either the solution time, if the problem is solved to optimality, or the optimality gap after 20 minutes of running time. Optimality gap is calculated using the following formula:

$$\frac{UB - O.V.}{O.V.}$$

where UB denotes the smallest upper bound on the objective function value obtained from the branch-and-bound tree search. 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. In addition, column *I.G.* reports the integrality gap, calculated through the following formula:

$$\frac{LPV - O.V.}{O.V.}$$

where LPV stands for the optimal objective function value from the initial LP relaxation of the problem. The integrality gap is reported in terms of percentage. Note that the largest objective function value found for each instance through each of the two formulations is marked with a $*$ sign.

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
P7-12	7	12
P8-16	8	16

Table 3: Instances Information

The computational results show that, if solved to optimality, the multitasking model outperforms the single tasking 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 single tasking model; therefore, it can produce equal or better optimal objective function under any circumstances. This is because, if it is optimal that each machine processes at most on task at a time throughout the scheduling horizon, then the multitasking model and the single tasking model would both result in a similar schedule, while, if a better solution can be obtained by processing materials from more than one task in a machine together, then the multitasking model will be able to generate better solutions compared to the single his point, the optimal schedule of the actual tasks obtained for instance P3-2 through both single tasking and multitasking formulations with $N = 5$ is depicted through Gantt charts in Figure 4 and Figure 5. Table 5 represents the information related to the processing units and their corresponding machines, while Table 6 shows the tasks information. As it can be observed from the figures, the multitasking feature allows machines 1 and 4 to process materials of different tasks simultaneously, which results in a better optimal solution for the multitasking formulation compared to the single tasking formulation.

The single tasking 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 (or at least to get the best solutions within the time limit). However, that would not be a

		Multitasking					Single tasking				
Instance	N	O.V.	B.Var.	Gen.Var.	T/G.	I.G.	O.V.	B.Var.	Gen.Var.	T/G.	I.G.
P3-2	4	1400	86	66	.23	188%	1150	116	54	.29	180%
P3-2	5	2100*	126	100	.14	207%	1900*	152	82	.84	197%
P3-2	6	2100	166	125	1.07	322%	1900	188	110	1.15	325%
P3-2	7	2100	206	153	2.71	438%	1900	224	138	1.77	451%
P4-3-1	7	3900	293	231	.23	207%	3400	318	211	.24	220%
P4-3-1	8	4500*	349	275	.91	224%	4100	369	254	.81	229%
P4-3-1	9	4500	405	318	4.7	282%	4110	420	297	2.69	292%
P4-3-1	10	4500	461	363	91.37	340%	4400*	471	340	52.69	325%
P4-3-1	11	4500	517	406	178	398%	4400	522	383	90.91	384%
P4-3-1	12	4500	571	476	458	409%	4400	568	426	309.91	393%
P4-3-2	19	4300	1588	1169	3.02%	287%	3360*	1506	1123	25.26%	383%
P4-3-2	20	4390	1680	1237	6.59%	302%	3090	1590	1190	40.97%	458%
P4-3-2	21	4340	1772	1304	13.51%	330%	3065	1674	1257	46.19%	495%
P4-3-2	22	4479	1864	1372	15.69%	339%	3215	1578	1324	41.81%	499%
P4-3-2	23	4435	1956	1440	22.29%	366%	2900	1842	1391	60.90%	599%
P4-3-2	24	4539*	2046	1511	25.09%	378%	2750	1938	1467	72.32%	674%
P4-3-2	25	4393	2140	1580	34.37%	416%	3120	2022	1534	54.23%	614%
P4-3-2	26	4579	2232	1647	34.29%	418%	2847	2106	1601	70.00%	718%
P4-3-2	27	4264	2324	1715	49.55%	479%	2935	2190	1668	66.77%	728%
P4-3-2	28	4308	2416	1783	53.78%	497%	2750	2274	1735	78.78%	821%
P5-3	6	2450	285	226	48.75	483%	2000	377	203	6.51	499%
P5-3	7	2600	364	288	147.95	672%	2100	449	264	34.76	745%
P5-3	8	2850	443	350	759.72	807%	2150*	521	325	489	994%
P5-3	9	3100*	522	412	765.40	920%	2150	593	386	7.45%	1263%
P5-3	10	3100	601	474	22.56%	1107%	2150	665	447	27.71%	1532%
P5-3	11	3100	680	536	35.48%	1293%	2150	737	508	44.52%	1801%
P5-4	9	4990	646	559	493.28	389%	4160*	768	525	64.34	442%
P5-4	10	5150	746	644	7.70%	462%	4160	861	610	525	539%
P5-4	11	5144	846	730	7.88%	529%	4160	954	695	1.72%	636%
P5-4	12	5345*	946	816	3.84%	581%	4160	1047	780	20.37%	731%
P5-4	13	5050	1046	902	9.90%	702%	4160	1140	865	25.58%	829%
P5-4	14	5345	1146	988	3.84%	730%	3980	1233	950	27.86%	971%
P6-10	10	4063	2002	1998	21.83%	1668%	1500*	2420	1910	22.95%	4394%
P6-10	11	4099*	2276	2272	20.76%	1931%	1500	2684	2183	39.87%	5196%
P6-10	12	3677	2550	2546	34.62%	2475%	1300	2948	2456	69.46%	6930%
P6-10	13	3364	2824	2820	47.15%	3055%	1500	3212	2729	51.11%	6795%

Table 4: Computational Results

		Multitasking					Single tasking				
Instance	N	O.V.	B.Var.	Gen.Var.	T/G.	I.G.	O.V.	B.Var.	Gen.Var.	T/G.	I.G.
P7-12	10	5660	1552	1536	14.32%	1254%	2255	1823	1463	12.29%	3371%
P7-12	11	5795	1781	1763	18.21%	1475%	2320*	2046	1689	12.34%	3821%
P7-12	12	5820*	2010	1990	25.63%	1715%	2320	2267	1915	24.15%	4473%
P7-12	13	5780	2239	2217	32.71%	1977%	2240	2488	2141	43.77%	5287%
P7-12	14	5223	2468	2444	45.71%	2465%	2240	2707	2367	46.77%	5931%
P8-16	9	6160	1804	1839	16.82%	1936%	1600*	2314	1734	10.95%	7245%
P8-16	10	6218	2130	2170	21.56%	2412%	1600	2632	2066	19.87%	9174%
P8-16	11	6580*	2456	2503	29.19%	2751%	1600	2952	2398	25.46%	11000%
P8-16	12	6519	2782	2836	36.73%	3251%	1520	3270	2730	29.11%	14311%
P8-16	13	6500	3108	3169	46.95%	3752%	1520	3590	3062	39.61%	16491%

Table 4: Computational Results

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 single tasking model.

Processing Unit	J_p	β_j^{max}	β_j^{min}	τ_j	TM_j
1	1	150	0	110	0.0
2	2	100	0,0	120	0.0
3	3,4	80,80	0,0	80,80	0.0

Table 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 6: P3-2: Tasks Information

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 single tasking counterpart, then, within a fixed amount of solution time, the multitasking model might not reach optimality and therefore the single tasking model might be able to produce better quality solutions. Therefore, it is essential to monitor the solution time for both model as well as the quality of the objective function. The results presented in Table 4 suggest that, for the same instance with the same number of time points, there is not a considerable difference in the solution time of the two models. Furthermore, by looking at column *I.G.*

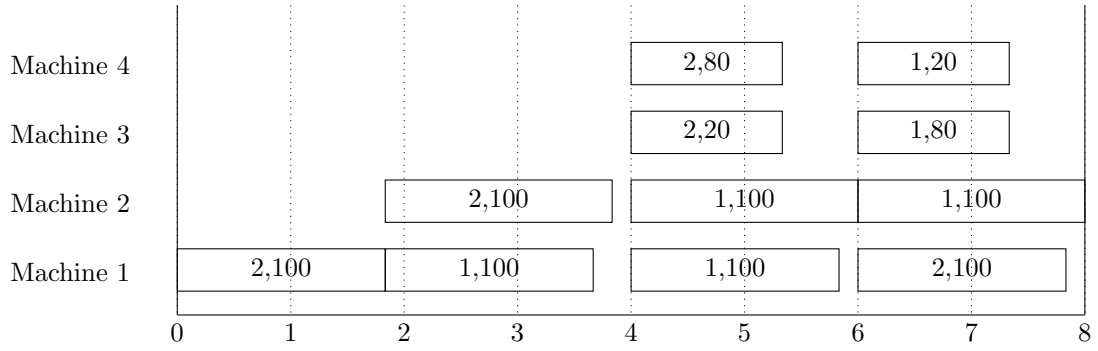

Figure 4: Instance P3-2 with single tasking, Objective Value =1900, N=5

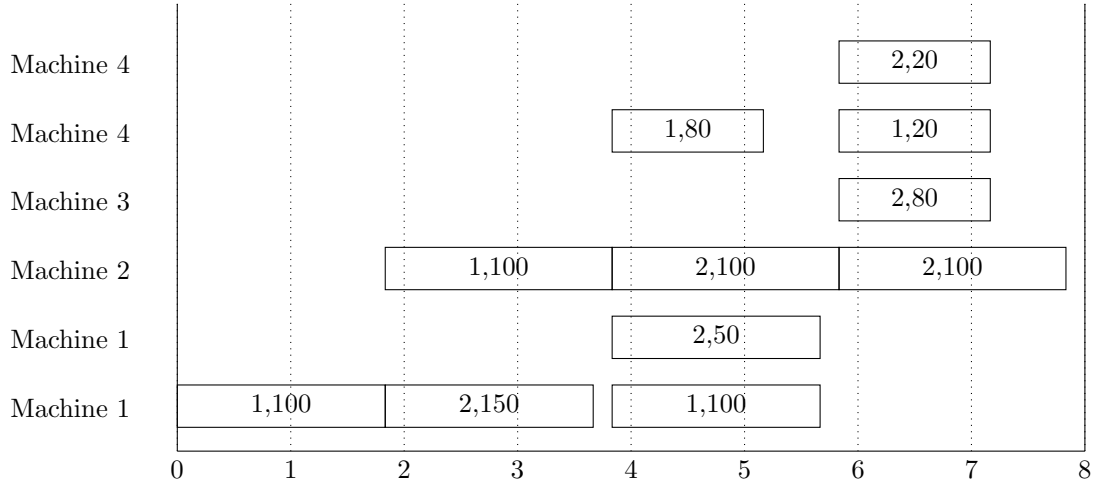

Figure 5: Instance P3-2 with multitasking, Objective Value =2100, N=5

in Table 4, there does not seem to be a considerable difference between the two models in regards with the integrality gap of the initial LP solution for the smaller instances. However, the multitasking model results in slightly smaller integrality gaps, specifically in dealing with larger instances (P6-10, P7-12 and P8-16).

It is worth noticing that some of the instances studied, e.g., P6-10, P7-12 and P8-16, are as large as some of the large instances solved in the continuous time formulation studies reported in the literature (Sundaramoorthy and Karimi, 2005; Li et al., 2012). An interesting observation to make here is the fact that for these larger instances (P6-10, P7-12 and P8-16), the difference between the values of the best solutions found by the single tasking and the multitasking model is quite large. In fact, for all of these instances, the multitasking model yields more than twice larger objective function values, compared to its single tasking counterpart. This observation is important since most of the facilities with multitasking machines, including the ones active in the analytical services sector, are dealing with relatively large demand, further emboldening the need to integrate the multitasking capability of the machines in such facilities, using multitasking models.

A final comment for this section is about the size of the problems that are presented in these computational tests as compared to an actual analytical services facility. The analytical services facility that inspired this work receives around 40 to 60 tasks per day, with each task usually having around 50 to 80 units of material in it, to a total of anywhere from 3000 to 5000 units of material per day. The size of the sequences of the tasks varies between 4 to 9 processing units. In contrast, for example, P8-16 has 16 tasks, with more than 1200 units of material, while the size of the sequences of the tasks vary from 4 to 6 processing units along the sequence. This means that in terms of the order of magnitude of the number of tasks, or alternatively the number of total samples in the facility, the presented multitasking model can handle about 25% to 30% the size that a real facility would face.

4. Conclusion

This paper presents a novel continuous time formulation for scheduling of operations at a multipurpose facility. The formulation is able to incorporate and use multitasking capability of machines in a multipurpose facility. The result of incorporating this capability is a more efficient use of the machines in the facility, which increases its throughput. 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 suggest that the current formulation does not have such a drawback since the computational complexity of the model in comparison to a single tasking model is negligible. The proposed formulation is able to solve large problems (in the context of 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 materials does not finish within the predetermined scheduling horizon.

In terms of future research, there are multiple directions that may be explored. The first and foremost is a thorough comparison between the presented formulation and discrete time formulations. One could argue that the answer to this question highly depends on the context that the two formulation are going to be used, and while that is valid, it still calls for comparison in specific cases, e.g., the context of analytical services industries or industrial applications with similar characteristics, may provide a suitable venue for such comparisons.

Another direction worth exploring is the fact that most of the continuous time formulations, including the presented formulation, have limited ability in solving most real world scheduling problems to near optimality in a timely manner. One of the reasons for that might be the fact that most of this formulations are solved through conventional solvers without any systematic study of the solution strategies for them. Thus, another research avenue can focus on finding new approaches that reduces the solution time of continuous time formulations, e.g., recognizing the facet defining inequalities and adding effective cuts.

If the continuous time formulations could be solved faster, there is also a potential for studies targeting the challenges laying in the implementation of schedules obtained from these formulations in real world

applications. Since it can be argued that these challenges would be case specific, it would be interesting to implement the schedules obtained from those formulations in an analytical services facility.

5. Acknowledgment

The financial supports provided by Natural Sciences and Engineering Research Council of Canada (NSERC), Ontario Centers for Excellence (OCE) and the industrial partner in analytical services sector are gratefully acknowledged.

6. References

References

Blömer, F., Günther, H.-O., 1998. Scheduling of a multi-product batch process in the chemical industry. Computers in Industry 36 (3), 245–259.

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

Castro, P. M., Westerlund, J., Forssell, S., 2009. Scheduling of a continuous plant with recycling of byproducts: A case study from a tissue paper mill. Computers & Chemical Engineering 33 (1), 347–358.

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

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

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

Harjunkoski, I., Maravelias, C. T., Bongers, P., Castro, P. M., Engell, S., Grossmann, I. E., Hooker, J., Méndez, C., Sand, G., Wassick, J., 2014. Scope for industrial applications of production scheduling models and solution methods. Computers & Chemical Engineering 62, 161–193.

Hui, C., Gupta, A., 2001. 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.

Hui, C.-W., Gupta, A., van der Meulen, H. A., 2000. A novel MILP formulation for short-term scheduling of multi-stage multi-product batch plants with sequence-dependent constraints. Computers & Chemical Engineering 24 (12), 2705–2717.

Ierapetritou, M., Floudas, C., 1998. Effective continuous-time formulation for short-term scheduling. 1. multipurpose batch processes. Industrial & Engineering Chemistry Research 37 (11), 4341–4359.

ILOG, Inc, 2014. ILOG CPLEX 12.6.1: High-performance software for mathematical programming and optimization. See `http://www.ilog.com/products/cplex/`.

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

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

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

Li, J., Xiao, X., Tang, Q., Floudas, C. A., 2012. Production scheduling of a large-scale steelmaking continuous casting process via unit-specific event-based continuous-time models: Short-term and medium-term scheduling. Industrial & Engineering Chemistry Research 51 (21), 7300–7319.

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

Méndez, C., Henning, G., Cerdá, J., 2001. An milp continuous-time approach to short-term scheduling of resource-constrained multistage flowshop batch facilities. Computers & Chemical Engineering 25 (4), 701–711.

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

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

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

Orcun, S., Altinel, I. K., Hortaçsu, Ö., 2001. 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.

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

Sargent, X. Z. . R. W. H., 1996. 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.

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

Sundaramoorthy, A., Maravelias, C. T., 2008. Simultaneous batching and scheduling in multistage multiproduct processes. Industrial & Engineering Chemistry Research 47 (5), 1546–1555.

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

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